KAPAK

Yazılım Uzmanı

Yazan Yazan

Editör: Selçuk Tüzel

Künye sayfası:

Yazılım Uzmanı 1

Yazan Yazan

Editör: Tuncer Karaarslan

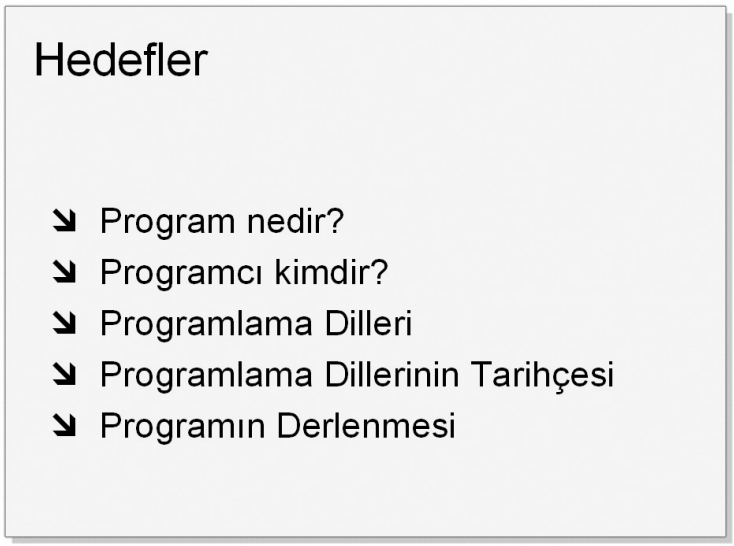
Düzelti: Selçuk Tüzel

Teknik Editör: Zeynep Çömlekçi

Şef Editör: Mehmet Çömlekçi

İçindekiler

Modül 1: Programlamaya Giriş



Bu modülde, bir programcının bilmesi gereken temel programlama kavramlarına giriş yapılacaktır. Bir programı oluşturan öğeler ve programın çalışma süreci tamamlanana kadar geçirdiği aşamalar ayrı ayrı işlenecektir. Bu kavramlar programcının ve programlama dillerinin tanımlanmasına yardımcı olacaktır.

Bu modülün sonunda;

- Bir programın çalışma prensibini açıklayabilecek,
- Programcı kavramını tanımlayabilecek,
- Değişik programlama dillerinin gelişimini açıklayabilecek,
- Derleme işlemini tanımlayabileceksiniz.

Konu 1: Program Nedir?

- Bilgisayarın, bir işi yapması için tasarlanan komutlar zinciri
- Program Türleri
  - Sistem Programlar
  - Sürücüler (Driver)
  - Uygulamalar

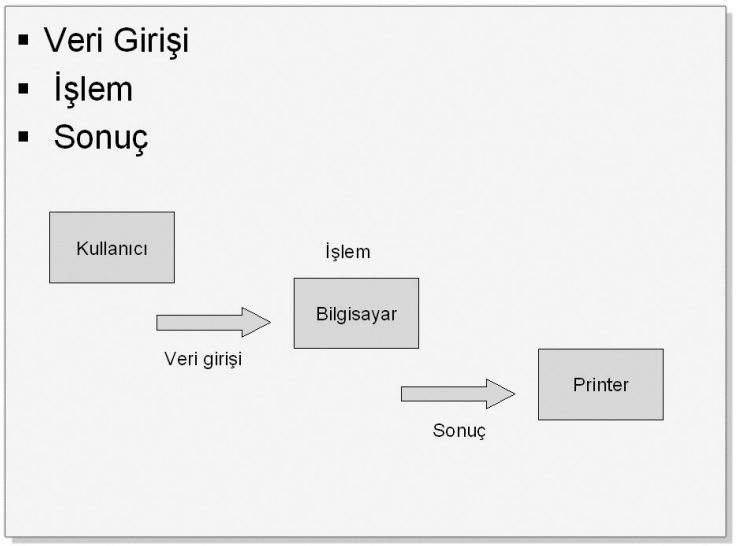
Günümüzde bilgisayarların kullanım alanları büyük ölçüde artmıştır. Dolayısıyla işlerimizi daha hızlı ve düzenli bir şekilde yapmamız, bilgisayarları ne kadar iyi kullandığımıza bağlıdır. Bunun için de çeşitli amaçlara göre yazılan programları kullanırız.

Program, bilgisayarın belli bir işi yapması için tasarlanan komutların tümüdür. Kullanım amaçları ve yerlerine göre birçok değişik program türü vardır:

- **Sistem programları.** Her program, bir işletim sistemi üzerinde çalışır. İşletim sistemi, diğer programların çalışması için gerekli olan kaynakları ve ortamı sağlar.
- **Sürücüler (Driver).** İşletim sistemi ile donanım aygıtları arasında iletişim sağlayan programlardır. Klavye ile yazılan yazıların algılanması için, klavyenin sürücü programı kullanılır.
- **Uygulamalar.** İşletim sistemi üzerinde çalışan, kullanıcıların ihtiyaç duyduğu işlevleri sağlayan programlardır.

Bir Internet sitesini gezmek istediğimizde, Internet Explorer tarayıcısı kullanılabilir. Bu uygulama, işletim sisteminden sitenin istenen sayfasındaki yazı ve resimleri almasını ister. İşletim sistemi, ağ kartıyla (Ethernet) sürücü programı sayesinde Internet sitesinin sunucusuna isteği gönderir.

Programların Çalışma Modeli



Programların kullanılmasındaki amaç, girilen bilgilerin işlenip sonuçların istenen şekilde üretilmesidir.

- **Veri girişi.** Program, kullanıcıların veri girmesi ile başlar. Girilen veriler daha sonra işlenmek üzere hafızada saklanır.
- **İşlem.** Veriler, programın yazılma şekline göre bir dizi işlemden geçirilir.
- **Sonuç.** İşlenen veriler kullanıcıya aktarılır.

Programlar, belli kurallar çerçevesinde yazılır. Bu yazım kuralları sayesinde bilgisayar, programın işleyişini anlar ve gerekli sonuçları çıkartır. Yazılan programlar, belirtilen yazım kuralları kontrol edilerek derlenir. Bu derleme işlemi sonunda, yazılan kaynak kodlar bilgisayarın anlayacağı tek dile çevrilir. Makine dili denilen bu dil, sadece 1 ve 0 sayılarından oluşur.

Örnek: ATM makinesinden para çekmek.

1. Kullanıcı ATM makinesine kartını yerleştirir.
2. Şifresini girer.
3. ATM cihazında çalışan uygulama kartta yazan bilgileri okur.
4. Şifre kontrolü işlemi yapılır.
5. Şifre doğru girilmişse kullanıcı çekmek istediği miktarı girer.
6. Bankadaki hesap kontrol edilir.
7. Uygunsa kullanıcıya ödeme yapar.

Konu 2: Programcı Kimdir?

- Belirli işlevlere sahip programlar geliştirir.
- Kullanılan teknolojiyi, platformu iyi tanıması gerekir.
- Programcı türleri
  - Mimar
  - Geliştirici
  - Test Mühendisi

Programcı, belirli işlevlere sahip programlar geliştirebilen bir uzmandır. Bir programcının, üzerinde çalıştığı platformu, kullandığı teknolojileri iyi tanıması ve bilgisayarın anlayacağı mantıksal dilde düşünebilmesi gerekir. Programcıların çoğu genellikle aynı işi gerçekleştirse de, üstlendikleri görevlere göre programcılar üç gruba ayrılabilir:

- **Mimar.** Programların yazılması için gerekli teknolojileri belirleyen, gerekli durumlarda programın daha kolay yönetilmesi için küçük parçalara ayıran programcıdır.
- **Geliştirici.** Programı yazan kişidir.
- **Test mühendisi.** Programın geliştirilmesi aşamasında, hataların kaynaklarını bulan ve geliştiricilere raporlayan programcıdır.

Konu 3: Programlama Dilleri

- Programcı ile bilgisayarın haberleşmesini sağlar.
- Programlar 1 ve 0 sayılarından oluşan makine diline çevrildikten sonra çalıştırılır.
- Programlama Dilinin özellikleri:
  - Sözdizimi (Syntax)
  - Gramer
  - Semantik
- 2500'den fazla programlama dili mevcuttur.

Dünyada konuşulan her dilin amacı iletişim sağlamaktır. Farklı kültürlerden insanların anlaşabilmesi için ortak konuştukları bir dil gerekir. Programlama dillerinin amacı da bilgisayar ile programcının haberleşmesidir. Programcı, bilgisayara hangi komutların çalıştırması gerektiğini bilgisayarın anlayacağı dilden konuşarak söyler.

Bilgisayarda, programlar makine diline çevrildikten sonra çalışır. 1 ve 0 sayılarından oluşan bu makine dili, en alt seviye dildir. Dolayısıyla programların bu dilde yazılması oldukça zordur. Programcılar konuşma diline daha yakın, kolay anlaşılabilecek diller kullanır. Bu dillere yüksek seviye programla dilleri denir. Programlama dillerinin seviyeleri, makine diline yakın olup olmamaları ile ölçülür.

Bir programlama dili şu unsurlardan oluşur:

- **Söz dizimi (Syntax).** Bir dil, kendine ait kelimeler ile konuşulur. Programlama dillerinin de benzer bir davranışı vardır. Programlama dillerindeki bu kelimeler, programlama dilinin anahtar kelimeleridir (komutlarıdır).
- **Gramer.** Programlama dillerini kullanmak için sadece kelimeleri bilmek yeterli değildir. Eğer anlamlı bir şekilde bir araya getirilemiyorsa, bu kelimeler hiçbir anlam ifade etmez.
- **Semantik (Anlamsal).** Bir dili, kelimeleri doğru bir gramer kullanımı ile bir araya getirerek kullanabiliriz. Ancak konuşulan kelimelerin ne için kullanıldığı da önemlidir. Bir programlama dilinin özelliklerinin nasıl ve ne için kullanıldığı da, bu dilin semantiğidir.

Örneğin bir finans programı, Yeni Türk Lirası cinsinden bir miktarı dolara çevirecektir. Yapılacak işlem, o andaki parite değerini merkez bankasından aldıktan sonra, girilen miktarı bu değerle çarpıp kullanıcıya göstermektir. Kullanılan programlama dili **ÇARP**, **GÖSTER**, **EŞİTLE** komutları ile bu işlemi gerçekleştirecektir.

```
ÇARP EŞİTLE GÖSTER miktar parite sonuç
```

Bu şekilde yazılan program söz dizimi açısından doğrudur. Girilen veriler ve komutlar dışında, programlama dilinin anlamayacağı bir kelime kullanılmamıştır. Ancak komutlar yanlış sırada kullanılmıştır. **ÇARP** komutu hangi sayıları çarpması gerektiğini bilemeyecektir.

```
parite EŞİTLE sonuç ÇARP miktar
GÖSTER parite
```

Komutları ve değişkenleri, programlama dilinin gramerine göre doğru yerlerde kullanmamız gerekir. Bu şekilde kullanılan komutlar doğru bir şekilde çalışır. Fakat **GÖSTER** komutunun ne için kullanıldığı yani semantiği de önemlidir. İstenilen, miktar ile pariteyi çarpmak, sonuca eşitlemek ve sonucu göstermektir.

```
sonuç EŞİTLE miktar ÇARP parite
GÖSTER sonuç
```

Şu ana kadar 2500'den fazla programlama dili yazılmıştır. Bunlardan bazıları Pascal, Basic, C, C++, Java, JavaScript, Cobol, Perl, Python, Ada, Fortran, Visual Basic .NET, Microsoft Visual C# programlama dilleridir.

Yüksek seviye programlama dillerine Visual Basic .NET ve Microsoft Visual C++ dillerini örnek verebiliriz. C ile işletim sistemi yazılabildiğinden, daha alt seviye bir dil olarak değerlendirilir.

Programlama Dillerinin Tarihçesi

- Makine dili 10110110, 11011110
- Yordamların (Subroutine) ve Kütüphanelerin (Library) oluşması
- 1957 FORTRAN
- 1959 COBOL
- 1968 Pascal
- 1972 C
- Nesne Yönelimli Programlama Dilleri:
  C++, JAVA
- 2000 .NET
  Visual Basic .NET, Visual C#

Bilgisayarlar, icat edilmeleriyle birlikte belli bir işi yapmak için bir dizi komuta ihtiyaç duymuşlardır. En başta çok basit işlemler yapan bu komutlar, zamanla nesne yönelimlilik (object orientation) gibi ileri seviyede özellikler kazanmıştır.

İlk programlama dilleri, bilgisayarların üzerinde bazı araçların yerlerini değiştirerek veya yeni bileşenler eklenerek yapılıyordu. Programın işlemesi için bir devinime ihtiyaç vardı. Eskiden programlar fiziksel olarak yazılıyordu. Daha sonra, fiziksel programlama yerini elektrik sinyallerine bıraktı. Artık, kurulan elektronik devrelere düşük ya da yüksek voltajda akım gönderilerek bilgisayarın davranışı belirlenmeye başlanmıştı. Yüksek voltaj 1, düşük voltaj 0 sayılarını ifade ediyordu. Böylelikle bugün de kullanılan makine dilinin ortaya çıkması için ilk adımlar atılmış oldu.

Ancak bu şekilde programlar yazmak, sistemi oluşturan elektronik devrelerin her program için baştan kurulmasını gerektiriyordu. Böylelikle programlar bazı kavramlar çerçevesinde yazılmaya başlandı. Öncelikle bilgisayar donanımı her program için baştan kurulmamalı, bunun yerine basit bir donanımın üzerine yazılan komutlar kullanılmalıdır. Daha sonra, programlar tek bir komutlar zinciri yerine, küçük parçalar halinde yazılmalıdır. Bu parçaların programın içinde defalarca kullanılabilmesi yordam (subroutine) kavramını ortaya çıkarmıştır. Bu modelin kullanılması ise mantıksal karşılaştırmaları, döngülerin kullanılmasını ve yazılan kodlar tekrar kullanıldığı için kütüphane (library) mantığını ortaya çıkarmıştır.

1957 yılında IBM, düşük seviye (makine diline yakın) bir programlama dili olan FORTRAN dilini ortaya çıkardı. FORTRAN ile beraber basit mantıksal karşılaştırmalar, döngüler, lojik (true-false) ve sayısal (integer, double) değişkenler kullanılmaya başlandı.

1959 yılında, bu programlama dilinin özelliklerini alıp, giriş çıkış (Input/Output – IO) gibi yeni işlevler sağlayan COBOL dili ortaya çıktı. Daha sonra 1968 yılında, COBOL ve FORTRAN dillerinin en iyi özelliklerini alarak Pascal ortaya çıktı. Ayrıca Pascal dili, hafızadaki adresler üzerinde işlem yapmaya olanak veren işaretçi (pointer) kavramını da beraberinde getirdi.

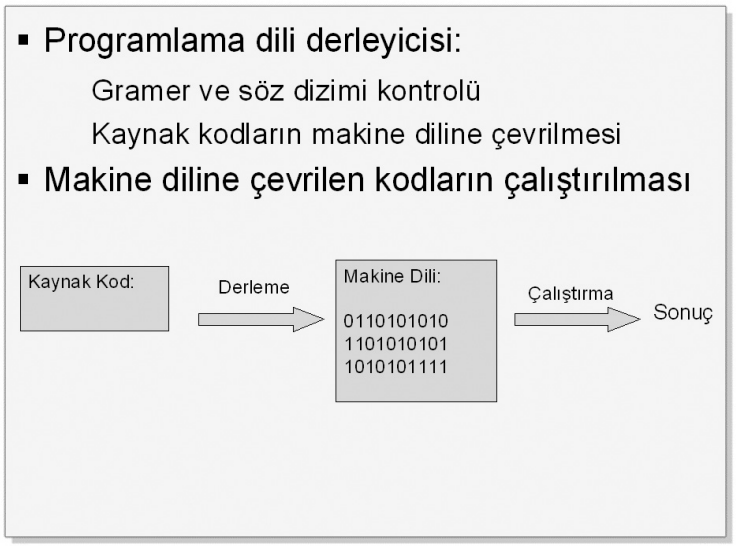
1972 yılında C, Pascal dilindeki birçok hatayı gidererek ortaya çıktı. C dili ilk defa Unix işletim sistemini yazmak için kullanılmaya başlanmıştır. C, düşük seviye bir dil olması, kuvvetli giriş çıkış işlemleri sağlaması gibi birçok özelliği ile işletim sistemlerinin yazılmasında tercih edilmiştir.

Bütün programlama dilleri birçok özelliğe sahip olmasına rağmen, modüler programlamanın birçok eksiğini gidermek amacıyla, yeni bir programlama modeli olan nesne yönelimli programlama (Object Oriented Programming – OOP) ortaya çıkarıldı. C dilinin ve OOP modelinin tüm özellikleriyle C++ dili oluşturuldu.

C++ dilini, Sun Microsystems tarafından çıkartılan Java takip etti. Java dilinin kullanım alanları, nesneye yönelimli bir programlama dili olması ve beraberinde getirdiği artık toplama (Garbage Collection – GC) gibi performans artırıcı özellikleri ile büyük ölçüde genişledi.

Microsoft, 2000 yılında .NET platformunu sunarak, otuzdan fazla programlama dilini aynı çatı altına topladı. VisualBasic.NET ve Visual C# günümüzde .NET platformunu kullanan en güçlü yüksek seviyeli programlama dilleri arasında yer alır. .NET platformu hakkında daha detaylı bilgi için Modül 2'ye bakın.

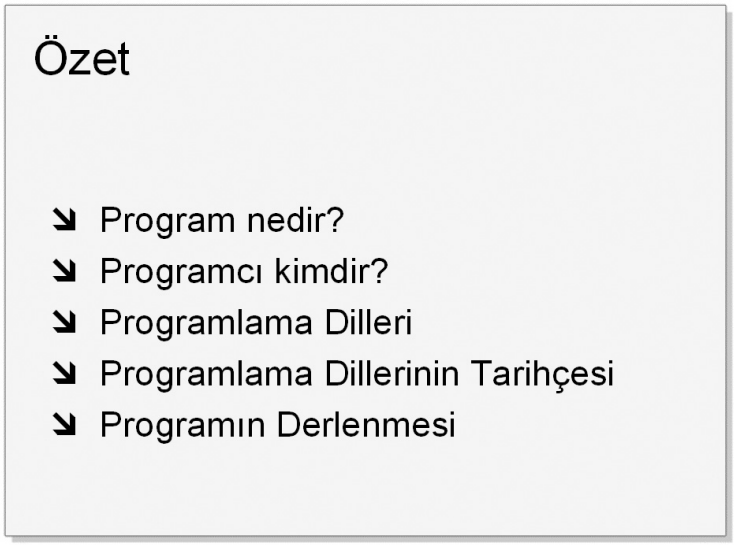
Konu 4: Programın Derlenmesi



Programlar yazıldıktan sonra, çalışmaya uygun hale getirilene kadar bir dizi işlemden geçer. Bu işlemi gerçekleştiren, programlama dilinin derleyicisidir. (Compiler)

- Programlar bir programlama dilinin gramer ve söz dizimi yapısına uygun bir şekilde yazılır.
- Yazılan kodlar o dilin derleyicisi tarafından kontrol edilir.
- Kontrol işleminden sonra, bu kodlar bilgisayarın anlayacağı makine diline çevrilir. Ancak bir yazım veya mantık hatası varsa, programcıya gerekli hata mesajı verilerek derleme işlemi iptal edilir.
- Makine diline çevrilen kodlar çalıştırılır.

Modül Sonu Soruları & Alıştırmalar



1. Varolan bir metin dosyasını (`.txt`) görüntülemek için Notepad programını kullanabiliriz. İşletim sistemi, dosyayı kullanıcılara göstermek için monitör ile iletişim kurar. Monitör, işletim sisteminden gelen verilerle gerekli görüntüleme işlemlerini yapar.

   Bu senaryodaki program çeşitlerini belirtin.

2. Bir arkadaşımıza e-posta yollamak istediğimizde, e-posta adresi, konu ve mesaj bilgilerini gireriz. Daha sonra e-posta uygulaması mesajımızı verilen adrese yollar.

   Programın çalışma modelinin aşamalarını belirtin.

3. C dilini kullanarak yazdığınız kodların bilgisayar tarafından çalıştırılabilir hale gelmesi için hangi aşamaların gerçekleşmesi gerekir?

Modül 2: Microsoft .NET Platformu

Microsoft .NET, uygulama geliştiricilerin yazılım geliştirme sürecinde altyapı işlemleri için harcadığı eforu en aza indirgemek ve daha güvenli, güvenilir ve sağlıklı uygulamalar geliştirebilmelerini sağlamak için geliştirilmiş bir altyapıdır.

Bu modülü tamamladıktan sonra;

- Microsof.NET platformu hakkında genel bilgi sahibi olacak,
- .NET Framework ve bileşenlerini açıklayabilecek,
- Microsoft .NET platformunun yazılım geliştiricilere sunduğu avantajları tanımlayabileceksiniz.

Konu 1: Yazılım Geliştirme Dünyası

Microsoft 1975 yılında Bill Gates ve Paul Allen tarafından kurulduğunda, vizyonu "Her eve, her masaya bir PC" idi. Donanım ve yazılım alanlarındaki gelişmelerin hızı ve birbirlerini sürekli tetiklemesinin sonucunda bilgisayar kullanıcılarının sayısı hızla arttı. Artan kullanıcı sayısı beraberinde yeni gereksinim ve talepleri getirdi. Bu taleplerin doğal sonucu olarak da farklı platformlar ve farklı servis sağlayıcıları ortaya çıktı. İletişim, finansal hizmetler, ticaret ve eğlence kullanıcıların (özellikle Internet'in yaygınlaşmasıyla birlikte) en yoğun talep gösterdiği hizmetler halini aldı. Günümüze baktığımızda, Microsoft'un çıkış noktasındaki hedefine büyük oranda ulaştığını görebiliyoruz. Ancak geldiğimiz noktada hızla artan bilgisayar ve Internet kullanıcısı sayısı, beraberinde güvenlik, iletişim ve entegrasyon gibi alanlarda çeşitli engelleri de getirdi.

Gelişmelere kendi açımızdan, yani yazılım geliştiriciler açısından baktığımızda işlerin çok daha zor ve zahmetli durumda olduğunu görürüz. Kurumsal uygulamaların geliştirilmesinde performans, güvenlik ve süreklilik gibi konularda belirli bir seviyeyi yakalamak için oldukça fazla efor sarf etmemiz gerekiyor. Örneğin, elektronik cihazlarla soket iletişimi kuracak uygulamaları geliştirebilmek için iki alternatifimiz var. Birincisi, 3. parti firmalar tarafından geliştirilmiş olan bileşenler satın almak ve uygulamamıza entegre etmektir. Diğer alternatifimiz ise, oldukça uzun sürecek bir kodlama ile benzer bir iletişim katmanını geliştirmektir. Her ikisi de firmaların birinci tercihi olmayacaktır. Sorunumuz, sadece soket iletişimi noktasında değil elbette. Bölümün başında da belirttiğimiz gibi uygulama geliştiriciler, güvenlik, performans ve yetkilendirme gibi pek çok konuda oldukça zahmetli altyapı kodlarını geliştirmekle uğraşmak zorunda kalıyor. İşin kötü yanı, geliştirilen bu altyapı kodlarının çoğu zaman istenen verimliliği sunmaktan oldukça uzak kalmasıdır. Kabul etmemiz gereken şey, bu altyapı kodlarını geliştirecek bilgiye sahip olmadığımız; sahip olsak bile, altyapı kodlarını yazacak zamana ve işgücüne sahip olmadığımız; zaman ve işgücü konusundaki ihtiyaçlarımızı karşılayabilsek bile, bu kodların testi, güvenliği, güvenilirliği, performansı ve uygulamalara entegrasyonu konusunda hiçbir zaman istenen düzeye ulaşamayacağımızdır. Keşke ihtiyaç duyduğumuz tüm altyapı işlemleri için hazır, kullanımı kolay ve esnek bir platform olsaydı.

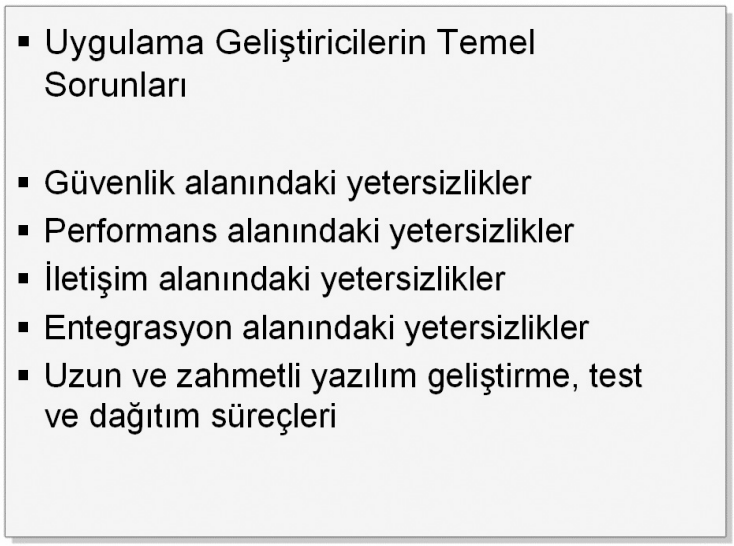
Hayalini kurduğumuz, aslında şöyle bir sistem:

> "Bir sanal mağazada cep telefonlarından sorumlu departmanda satış müdürü olarak çalışıyorsunuz. İş dışındasınız ve akıllı cihazınıza bir mesaj geliyor: 'Piyasaya henüz çıkmış olan telefonumuz inanılmaz satışlar yapıyor, telefon çok popüler ve stoklarımız da oldukça azalmış durumda.' Bu mesajın hemen ardından, akıllı cihazınız üzerinden, şirketiniz için fiyat ve teslim zamanı açısından en uygun olan tedarikçiyi bulup, ihtiyacınız kadar telefonu sipariş edebiliyorsunuz. Peki ya bu koşullar altında çalışmıyor olsaydınız? Şirketiniz, sizi cep telefonunuzdan arayacak ve problemi iletecekti. Sonra da siz ancak şirketinize dönebildiğiniz zaman tedarikçilerle teker teker irtibata geçerek hangisi-

nin şirketiniz için en yararlı olduğuna karar verecektiniz. Sipariş ve teslimat bilgileri üzerinde anlaştıktan sonra işleminizi tamamlamış olacaktınız. Yani sadece birkaç dakikada yapabileceğiniz basit bir işlem için, belki de bütün bir gününüzü kaybedecektiniz. Verimliliğiniz düşerken, zamanınızı etkili şekilde kullanamayacaktınız. Oysa akıllı cihazınız üzerinden tüm bu işlemleri kısa bir şekilde çözebildiğinizden, işe gitmenize bile gerek kalmadan çok kısa bir zamanda şirketiniz için en iyi olan seçimi yapabilirsiniz."

İşler kesinlikle çok daha verimli ve kolay ilerlerdi. Elbette bu, kurulabilecek hayallerin sadece mobil platforma yönelik bölümünden bir kesit.

Konu 2: Sorunun Temeli



Microsoft, vizyonu doğrultusunda attığı adımların yazılım geliştiricilere yansıyan sonuçlarını sürekli izliyordu ve yazılım geliştiricilerin sorunlarını şu başlıklar altında ele alıyordu.

- Uygulamalar, sistemler ve kurumdaki birimler arasındaki ve farklı kurumlar arasındaki iletişim sorunu.
- Çalışanların ihtiyaç duydukları verilere, ihtiyaç duydukları anda, kesintisiz, hatasız ve güvenli bir şekilde ve istedikleri platformdan erişebilmeleri gereksinimi.
- Uygulama geliştirme sürecinde, geliştiricilerin altyapı kodları ile uğraşması ve bunun sonucunda, uygulama geliştirme ve test süresinin uzaması.
- Bir uygulamanın farklı platformlarda çalıştırılabilmesi için, aynı işlemleri gerçekleştirecek kodların tekrar tekrar yazılması gereksinimi.

Konu 3: Çözüm Platformu

Microsoft 1990 yılında, yaşanacak 10 yılı da öngörerek, bu ve benzeri sorunlara çözüm sunacak, uygulama geliştiricilerin ve son kullanıcıların işlerini kolaylaştıracak bir platform geliştirmeye başladı. Microsoft bu platforma öylesine inanıyordu ki, kaynaklarının %80'inden daha fazlasını, yani kaderini bu platforma bağlamıştı. Çok geniş bir analiz ve geliştirme ekibinin çalışmaları sonucunda ortaya çıkan ürün 2000 yılında dünyaya sunulduğuna, insanların karşılarında gördükleri yapı karşısında hissettiklerini tanımlamak için kullanılabilecek en uygun kelime "hayranlık" idi.

Microsoft.NET platformu, her türlü yazılım geliştirme ihtiyacına yönelik hazır bir altyapı sunarak, uygulama geliştiricilerin Windows, Web ve mobil platformlara yönelik uygulamaları çok daha hızlı, kolay ve güçlü bir şekilde geliştirebilmelerine olanak tanıyordu. Uygulama geliştiriciler şifreleme, kimlik doğrulama, yetkilendirme, soket iletişimi, her türlü veri kaynağına yönelik veritabanı işlemleri, XML ve Web servisi teknolojilerine kadar burada saymadığımız (editörler bir modülün 100 sayfayı geçmesine pek sıcak bakmıyorlar) pek çok teknolojiyi ve hatta milyonlarca hazır sınıf ve fonksiyonu karşılarında gördüler. Bugüne kadar günler, haftalar ve hatta aylar harcayarak geliştirmeye çalıştıkları bu yapıların hepsini, karşılarında kullanıma hazır bir şekilde görmekten de son derece memnunlardı.

Modül 3: Microsoft Visual Studio Arayüzü

Bu modül, Microsoft Visual Studio arayüzünü tanımayı sağlar ve etkili bir biçimde kullanmayı gösterir. Ev ve işyerindeki çalışma ortamını düzenlemek, daha verimli çalışmayı sağlar. Yazılım geliştirirken de çalışılan ortamı tanımak ve kişiselleştirmek, rahat çalışmak açısından önemlidir.

Bu modülü tamamladıktan sonra;

- Microsoft Visual Studio çalışma ortamını tanıyacak,
- Menülerin işlevlerini açıklayabilecek,
- Başlangıç sayfasının özelliklerini kullanabilecek,
- Solution Explorer, Toolbox, Properties panellerini tanıyacak,
- Microsoft Visual Studio yardımını etkili bir şekilde kullanabileceksiniz.

Konu 1: Visual Studio Çalışma Ortamı

- Visual Studio bir dosya editörüdür.
- Çalışma Sayfaları
  - Sekmeler halinde gösterilir.
- Araç Çubukları
  - Menü komutlarına görsel arayüz
  - Özel araç çubukları tanımlanabilir.
- Menüler
- Paneller
  - Sabitlenebilir, Kayan, Gizlenebilir pencereler

Visual Studio, çok gelişmiş özelliklere ve yardımcı araçlara sahip bir dosya editörüdür. .NET platformu üzerinde geliştirilen proje dosyaları dışında, metin dosyaları, `*.sql` ve `*.rtf` uzantılı dosyalar da düzenlenebilir. Visual Studio ortamını oluşturan ve kullanımını kolaylaştıran dört ana bileşen vardır:

Çalışma Sayfaları

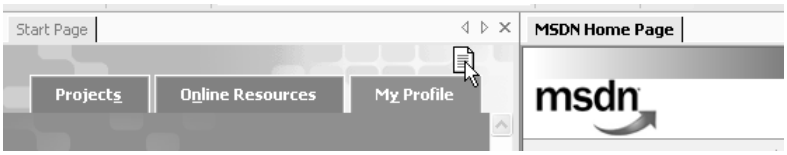
Visual Studio ortamında dosyalar, birer çalışma sayfası (Tab Pages) olarak açılır. Bu dosyalar sekmeler halinde sıralanır. Sayfalar arasında **CTRL+TAB** kısayolu ile geçiş yapılır.

Bu çalışma modelinde, sadece bir sayfa görünür ve üzerinde çalışma yapılır. Ancak Visual Studio bize, çalışma ortamını parçalara bölme imkanı verir.

Örnek:

1. Visual Studio çalışma ortamını açın. Başlangıç sayfası karşınıza çıkar. (Eğer başlangıç sayfasını göremiyorsanız, Help menüsünden Show Start Page komutunu seçin.)
2. View menüsünden, Web Browser alt menüsünü işaretleyin ve Show Browser komutunu seçin. Visual Studio, açmak istediğimiz Internet tarayıcısı için yeni bir sayfa oluşturur.
3. **CTRL** tuşunu basılı tutarak **TAB** tuşuna basın. Açtığınız Internet tarayıcısından başlangıç sayfasına dönülür.
4. Başlangıç sayfasını sağ tıklayın ve açılan menüden New Vertical Tab Group komutunu seçin. Visual Studio, sayfaları “sekme gruplarına” ayırarak birden fazla sayfa üzerinde çalışma imkanını sağlar.

5. Başlangıç sayfasını, sayfa başlığını tıklayıp fare düğmesini basılı tutarak Internet tarayıcısının bulunduğu sekme grubuna taşıyın.



RESİM 3.1.

İPUCU **Visual Studio ortamını bir Web tarayıcısı olarak kullanabilirsiniz.**

Araç Çubukları

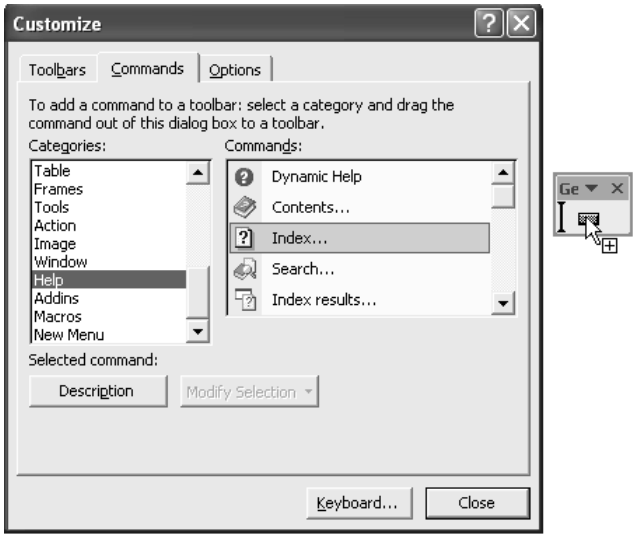
Visual Studio, menü komutları için görsel kısayolları araç çubukları (Toolbars) ile sunar. Benzer işlemler için kullanılan komutlar bir araç çubuğunda gruplanır. Örneğin Standard araç çubuğu, yeni dosya oluşturmak, bir dosyayı açmak ve kaydetmek gibi genel dosya işlemleri için kullanılır.

Araç çubukları, varsayılan durumda menülerin altında bulunur. Ancak araç çubukları taşınarak yerleri değiştirilebilir veya kayan duruma getirilebilir. Ayrıca istenen araç çubukları saklanabilir veya gösterilebilir. Araç çubuklarının listesini görmek için View menüsünden Toolbars alt menüsünü işaretleyin.

Visual Studio bize kendi araç çubuklarımızı oluşturma imkanı da verir. Farklı işlevlere sahip komutlar gruplanıp, kişisel bir araç çubuğu oluşturulabilir.

Örnek:

1. Başlangıç sayfasının üstündeki bir araç çubuğunu sağ tıklayın. Açılan menüde, varolan tüm araç çubukları listelenir. İşaretli olan araç çubukları eklenmiş araç çubuklarıdır. Bu listeden Web araç çubuğunu seçin.
2. Web araç çubuğunu çift tıklayın. Bu işlem, araç çubuğunu "Floating" (kayan menü) durumuna getirir. Tekrar çift tıklandığında, araç çubuğu "Dockable" (sabit) durumuna gelir.
3. Araç çubuğunu sağ tıklayın. Açılan menünün en altındaki Customize (özelleştir) komutunu seçin.
4. Toolbars sekmesinde New (yeni) komutunu tıklayın. Açılan pencerede araç çubuğunun ismi için "Genel İşlemlerim" yazın. **OK** düğmesini tıklayın. Visual Studio, verilen isimde bir araç çubuğu oluşturur ve kayan durumda görüntüler.
5. Commands (komutlar) sekmesinde, Categories (kategoriler) listesinden Help kategorisini seçin. Bu listenin yan tarafında bulunan Commands listesinden Index komutunu bulun. Bu komutu taşıyıp, oluşturduğumuz "Genel İşlemlerim" araç çubuğuna bırakın (Resim 3.2).



RESİM 3.2: Araç çubuğu oluşturmak.

Bu şekilde şu komutları da ekleyin:

Categories	Commands
Tools	Options
File	Exit
View	Show Web Browser
Window	Close All Documents

6. Araç çubuğunu çalışma ortamının altına taşıyarak sabitleyin.
7. Araç çubuğunu sağ tıklayın ve listeden "Genel İşlemlerim" araç çubuğunu seçerek çalışma ortamından kaldırın.

Menüler

Birçok çalışma ortamının yaptığı gibi, Visual Studio da benzer öğeler üzerinde işlevleri olan komutları menüler halinde gruplar. Menülerin araç çubuklarından farkı, sabit olmaları ve özelleştirmeye açık olmamalarıdır. Menüler bu modülde detaylı olarak ele alınacaktır.

Paneller

Paneller, Visual Studio içindeki pencerelerdir. Çalışma ortamında birçok panel bulunur. Bunlar arasında Solution Explorer, Toolbox, Object Browser, Properties, Watch, Output, Search Result, Task List gibi sıkça kullandığımız paneller sayılabilir.

İPUCU **Görmek istenen paneller View menüsünden seçilebilir.**

Paneller, Visual Studio ortamı içersinde istenen yere taşınabilir veya sabitlenebilir. Panellerin birkaç genel özelliği vardır:

- **Auto Hide (Otomatik gizle).** Panelin, fare imleci üzerindeyken gözükmesi ve imleç çekildikten sonra gizlenmesidir.
- **Dockable (Sabitlenebilir).** Panelin, Visual Studio ortamı içersinde bir yerde sabitlenebilme özelliğidir.
- **Floating (Kayan).** Kayan paneller herhangi bir yere sabitlenemez. Ancak her sayfanın üstünde durur ve böylece sürekli görünür.

Panellerin bu özelliklerine Window menüsünden erişilebilir.

Örnek:

1. View menüsünden Other Windows alt menüsünü işaretleyin ve Favorites panelini seçin. Panelin başlığında, biri Auto Hide, diğeri Close olmak üzere iki düğme görülür.
2. Auto Hide düğmesini tıklayarak paneli gizleyin.
3. Paneli tekrar seçin, Window menüsünden Auto Hide özelliğini seçin. Daha sonra aynı menüden Floating özelliğini seçin. Panelin taşınabildiği, ancak sabitlenemediği görülür.
4. Panel seçiliyken, Window menüsünden Dockable özelliğini seçin. Bu sefer, panelin taşındığında çalışma ortamının herhangi bir yerine sabitlenebildiği görülür.
5. Panel seçiliyken, Window menüsünden Hide komutunu seçin. Paneli tekrar açmak için bu adımları tekrarlayın.

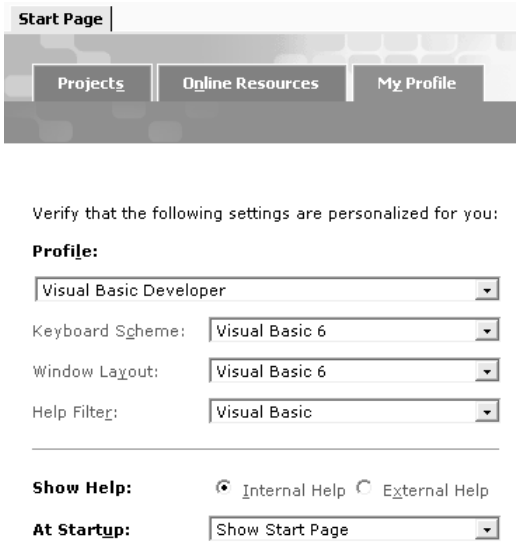
Konu 2: Start Page

- Visual Studio ortamının başlangıç sayfasıdır.
- Projects
  - Oluşturulan Visual Studio projeleri listesi
- Online Resources
  - Internet üzerindeki kaynaklar
  - Kod örnekleri, güncellemeler, makaleler
- My Profile
  - Çalışma şekline göre özel ayarlar

Visual Studio çalışma ortamını açtığımız zaman karşımıza ilk gelen başlangıç sayfasıdır. Bu sayfa üç bölümden oluşur (Resim 3.3).

- **Projects.** O ana kadar çalıştığınız projeleri gösterir. Bu menüden son projelerinizi açabilirsiniz. Son projelerde gözükmeyen bir proje (Open Project) veya yeni bir proje (New Project) de açabilirsiniz.
- **Online Resources.** Bu bölümde örnek uygulamalar (Find Samples), ipuçları bulabilir, en yeni teknolojileri, güncellemeleri veya en son eklenen haberleri takip edebilir, MSDN kütüphanelerinde kod örnekleri ve makaleler araştırabilirsiniz.
- **My Profile.** Bu bölümde çalışma şeklinize göre bir profil seçebilirsiniz. Profiller; kullanılan kısayollara, panellerin yerlerine ve görünümlerine, Visual Studio yardımını kullanırken yapılan filtrelemeye göre değişir.

  Örneğin, profili Visual Basic Developer olarak ayarlarsak Toolbox, sayfaların sol tarafında çivili olarak durur. Yardım panelinde bir arama yapmak istediğimizde ise, sonuçlar Visual Basic filtresine göre çıkar. Ayrıca Solution Explorer paneli **CTRL+R** kısayolu ile açılır.



RESİM 3.3: Start Page.

Görünüm, kısayollar ve yardım filtresi birbirinden bağımsız olarak da ayarlanabilir. Bu durumda seçilen profil, custom (özel) olarak gözükecektir.

At Startup seçeneklerinden, Visual Studio açılırken hangi pencerenin gözükeceğini belirleyebilirsiniz. Örneğin, başlangıçta en son çalıştığınız projenin açılmasını istiyorsanız, "Load last loaded solution" seçeneğini tercih etmelisiniz.

İPUCU **Giriş sayfasını kapattıktan sonra, Help menüsünden Show Start Page seçeneğini tıklayarak açabilirsiniz.**

Konu 3: Menüler

- Birçok uygulamada kullanılan benzer menü görünümü
- File, Edit
  - Dosya, metin düzeni işlemleri
- View, Window
  - Paneller, çalışma sayfaları görünümleri
- Project, Build, Debug
  - Proje, derleme ve hata ayıklama işlemleri
- Tools, Help
  - Yardımcı araçlar, yardım seçenekleri

Visual Studio menüleri birçok uygulamanın menülerine benzer niteliktedir. Menü isimlerinde, belirli bir harfinin altı çizilmiştir. Belirtilen harflere **ALT** tuşu ile birlikte basıldığında, o menülere kısayolla ulaşılır. Menü komutlarının bazılarında ise, sadece o komuta özel bir kısayol tanımlıdır. Bu kısayollar **CTRL** veya **SHIFT** gibi birkaç tuş kombinasyonu ile gerçekleşir.

- **File (Dosya).** Tüm dosya işlemleri bu menü altındadır. Standard araç çubuğu ile bu menüdeki bazı komutlara ulaşılır. File menüsündeki komutlar ile:
  - Yeni bir proje, bir dosya veya boş bir çözüm (solution) oluşturmak,
  - Oluşturulmuş bir projeyi veya varolan bir dosyayı açmak,
  - Web üzerinde paylaştırılmış dosya veya projeler açmak,
  - Açık olan dosya veya projeleri kapatmak,
  - En son kullanılan dosya veya projeleri (Recent Files) açmak,
  - Dosyaları kaydetmek, yazdırmak mümkündür.
- **Edit (Düzenle).** Tüm yazı düzenleme işlemleri için, bu menüdeki komutlar kullanılır. Text Editor araç çubuğu da bu menünün komutlarına kısayoldur. Edit menüsündeki komutlar ile:
  - Copy, Cut, Paste, Delete, Select All gibi temel işlemleri
  - Find And Replace, Go, Bookmark gibi navigasyon işlemleri
  - Outlining ile metinleri gruplama işlemleri
  - Satırları yorum satırı yapma, yorum satırlarını kaldırma, büyük-küçük harf çevrimi gibi ileri seviye işlemler gerçekleştirilir.

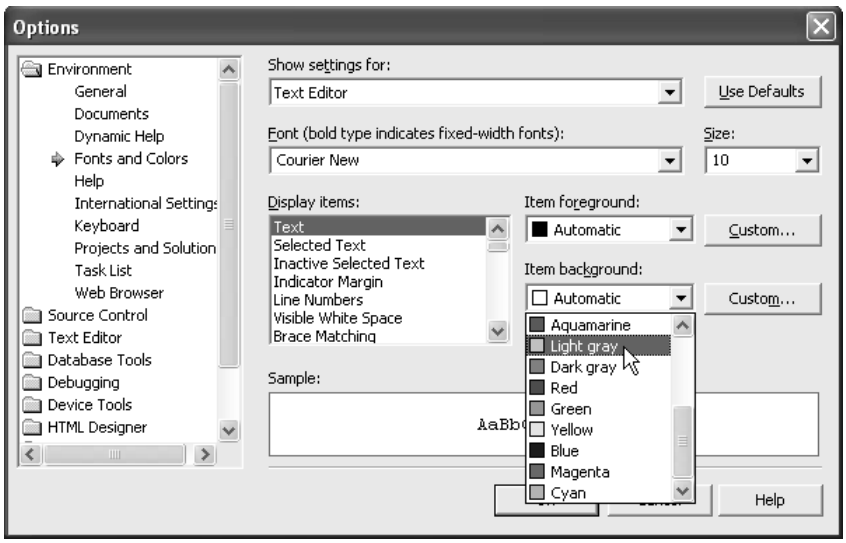
- **View (Görünüm).** Visual Studio çalışma ortamındaki tüm paneller bu menü komutlarıyla gösterilir. Ayrıca Navigate Backward ve Navigate Forward komutlarıyla en son çalışılan satıra geri dönülür.
- **Project (Proje).** Projeye dosya eklemek, çıkarmak, proje özelliklerini göstermek için bu menü kullanılır.
- **Build (Derleme).** Projelerin çalışmak üzere derlenmesi için gereken komutlar, bu menü altındadır.
- **Debug (Hata Ayıklama).** Projede hata ayıklarken gereken komutlar Debug menüsü altındadır. Projeyi Debug durumunda başlatmak, BreakPoints (hata ayıklarken durulması gereken satırları ayarlamak) gibi işlemler yapılır.
- **Tools (Araçlar).** Visual Studio ile beraber yüklenen yardımcı araçların listelendiği menüdür. Araç çubuklarını özelleştirmek için kullanılan Customize seçeneği gibi Options seçeneği de en sık kullanılan özelliklerden biridir.

  Visual Studio çalışma ortamının tüm ayarları Options menüsünden yapılır. Environment ve Text Editor en sık kullanılan seçeneklerdir.

  - **Environment (Ortam).** Sayfa düzeni ve görünüm ayarları, yazı tipi (font) ve renk ayarları, komutlar için kısayol ayarları, Internet tarayıcısı ayarları, yardım ve dinamik yardım ayarları buradan yapılır.
  - **Text Editor (Metin düzenleyicisi).** Farklı programlama dillerine özgü yazı düzeni ayarları buradan yapılır.

  Örnek:

  1. Tools menüsünden Options komutunu seçin.
  2. Sol panelde bulunan Environment menüsünden Fonts and Colors (Yazı düzeni ve renkler) sekmesine gelin.
  3. Sağ panelde bulunan Display items (Öğeleri listele) menüsünden Text alanını seçin ve Item background (Öğe arka planı) özelliğini Light Grey (Açık gri) olarak belirleyin. Tüm sayfaların arka plan rengi açık gri olacaktır (Resim 3.4).



RESİM 3.4: Arka plan renginin değiştirilmesi.

4. Sol panelde Environment menüsünden Web Browser sekmesine gelin. Home Page (ana sayfa) özelliğinin altındaki Use Default seçeneğini kaldırın ve metin kutusuna `www.bilgeadam.com` yazın.
5. Sol panelde Text Editor menüsünden Basic alt menüsünü seçin. Burada Visual Basic diline özel metin düzenleme seçenekleri bulunur. Sağ panelde, Display sekmesinin altında Line Numbers (Satır numaraları) seçeneğini işaretleyin. Bu seçenek, Visual Basic projelerinde çalışırken satır numaralarını gösterir.

- **Window (Pencere).** Sayfaların ve panellerin görünümlerini ve özelliklerini değiştirmek için kullanılan komutlar bu menü altında bulunur. Tüm açık çalışma sayfaları bu menü altında görüldüğü gibi, istenen sayfa seçilerek ön plana getirilir. Ayrıca, Close All Documents (Tüm sayfaları kapat) komutu ile açık olan bütün sayfalar kapatılır. Auto Hide All (Tümünü otomatik gizle) komutu ile, sabit hale getirilmiş tüm paneller gizlenir.
- **Help (Yardım).** Visual Studio çalışma ortamında çok sık kullanılan yardım panellerinin görünümü bu menü ile sağlanır. Bu menü ile ayrıca, kullanılan Visual Studio çalışma ortamının sürümü hakkında bilgi alınır, son güncellemeler kontrol edilir, teknik destek için gereken e-posta adreslerine veya telefonlara ulaşılır.

  Yardım kullanımı bu modülde detaylı olarak ele alınacaktır.

Konu 4: Solution Explorer Paneli

- Visual Studio projeleri, bir “Çözüm” altında toplar.
- Çözüm içinde bulunan tüm dosyalar ve klasörler görüntülenir.
- Panele ait araç çubuğu basit işlemler gerçekleştirir.
  - Refresh, Show All Files, Properties
- Visual Basic profilinde, **CTRL+ALT+L** ile ulaşılır.

Visual Studio çalışma ortamında projeler bir çözüm (solution) altında açılır. Bir çözüm içine farklı dilde ve tipte projeler dahil edilebilir. Visual Studio ile bir çözüm açıldığında, Solution Explorer panelinde (Resim 3.5) çözüm içinde bulunan tüm projelerle, ilgili dosya ve klasörler görüntülenir. Panelde koyu yazı tipinde gözüken proje, çözüm içindeki başlangıç projesidir.

Bu panelden, öğeler üzerinde silme, kopyalama, taşıma ve ismini değiştirme işlemleri yapılabilir. Ayrıca panelin üst kısmında, seçilen öğe üzerinde basit işlemler gerçekleştirmek için bir araç çubuğu bulunur.

- **Refresh (Yenile).** Proje dosyaları üzerindeki değişikliklerin gözükmesini sağlar.
- **Show All Files (Bütün dosyaları göster).** Seçilen projenin bulunduğu klasördeki tüm dosyaları ve alt klasörleri gösterir. Panelde gözüken beyaz öğeler proje içine dahil edilmemiş öğelerdir. Proje kapsamında kullanılmak istenen öğeler (örneğin, arka plan resmi), sağ tıklanıp Include In Project komutu ile projeye dahil edilmelidir.
- **Properties (Özellikler).** Paneldeki tüm öğelerin özellikleri, Properties komutu ile görülebilir. Bu komut seçildiğinde, öğenin özellikleri Properties paneli ile görüntülenir. (Properties paneli bu modülde detaylı olarak ele alınacaktır.)

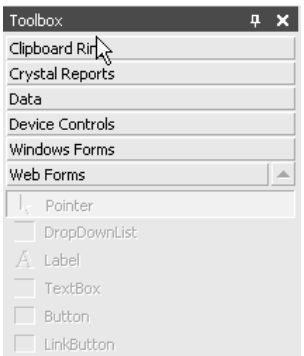
RESİM 3.5: Solution Explorer paneli.

Solution Explorer paneli, View menüsünden görülebildiği gibi, varsayılan klavye seçeneklerinde **CTRL+ALT+L** kısayolu ile de görülebilir.

Konu 5: Toolbox Paneli

- Projelerde kullanılan çeşitli bileşenler listelenir.
- Nesneler, sekmeler halinde gruplanır.
  - Windows Forms, Web Forms, ClipBoard Ring
- Visual Basic profilinde, **CTRL+ALT+X** ile ulaşılır.

Toolbox (Araç kutusu) paneli, projelerde kullanılan çeşitli bileşenlerin listelendiği paneldir. Buradaki öğeler, sekmeler içinde gruplanmıştır. Her sekme, ortak platformlarda çalışan veya benzer işlevleri olan nesnelere sahiptir. Örneğin, Data sekmesinde veritabanı işlemlerinde kullanılan bileşenler vardır. Windows Forms bileşenleri Windows platformunda çalışan projelerde, Web Forms bileşenleri ise Web tabanlı projelerde kullanılan nesnelerdir. ClipBoard Ring sekmesinde ise kopyalanan metinler bulunur. Nesnenin silik gözükmesi, o anda çalışılan sayfada kullanılamayacağı anlamına gelir (Resim 3.6).



RESİM 3.6: Toolbox paneli.

Toolbox panelinde nesneler, en sık kullanılandan en az kullanılana göre sıralanır. Örneğin, Windows Forms sekmesinde en üstte **`Label`**, **`Link Label`**, **`Button`**, **`TextBox`** nesneleri bulunur. Nesneler, yerleri ve sıraları taşınarak

değiştirilebilir, ayrıca başka bir sekmeye de taşınılabilir. Varsayılan sıralama dışında, alfabetik olarak da sıralama yapılabilir.

Visual Studio çalışma ortamı, Toolbox panelindeki nesnelere yeni isim verme, nesneleri silme veya panele yeni sekmeler ve nesneler ekleme imkanlarını da sağlar.

Örnek:

1. View menüsünden Toolbox panelini seçin.
2. Panelde herhangi bir yeri sağ tıklayın ve Show All Tabs (Bütün sekmeleri göster) komutunu seçin.
3. Windows Forms sekmesinde **`TextBox`** nesnesini sağ tıklayın. Çıkan menüden Rename Item (Ad Değiştir) komutunu seçin ve "**`Metin Kutusu`**" yazın.
4. "**`Metin Kutusu`**" nesnesini taşıyarak sekmenin en üstüne getirin.
5. Paneli sağ tıklayın ve Sort Items Alphabetically (Nesneleri alfabetik olarak sırala) komutunu seçin. **`Metin Kutusu`** nesnesinin, alfabetik sırada yerini aldığı görülür.
6. Paneli sağ tıklayın ve Add Tab (Sekme ekle) komutunu seçin. Sekmeye "**`Medya`**" ismini verin.
7. Sekmeyi sağ tıklayın ve Add/Remove Items (Nesne Ekle/Kaldır) komutunu seçin. Customize Toolbox iletişim kutusu açılır. Burada Toolbox paneline eklenebilecek tüm bileşenler listelenir. Com Components sekmesine gelin ve listeden **`Windows Media Player`** nesnesini işaretleyin. OK düğmesini tıklayarak iletişim kutusunu kapatın. **`Windows Media Player`** nesnesinin, oluşturulan Medya sekmesine eklendiği görülür.

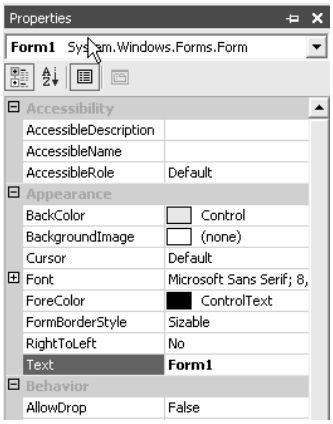
Toolbox paneline varsayılan klavye seçeneklerinde **CTRL+ALT+X** kısayolu ile ulaşılır.

Konu 6: Properties Paneli

- Visual Studio ortamındaki nesnelerin özelliklerini listeler.
- Özellik adı – Değeri
- Özellikler kategorilere göre gruplanmıştır; alfabetik olarak da sıralanabilir.
- **F4** ile her yerden ulaşılır.

Properties (Özellikler) paneli (Resim 3.7), seçilen bir nesnenin özelliklerini görüntüler. Paneldeki görünüm, "Özellik adı – değeri" şeklindedir. Silik olarak gözüken özellikler salt okunurdur ve değiştirilemez. Panelin üzerindeki açılır liste, çalışma sayfasındaki nesneleri listeler. Buradan istenen nesne seçilerek özellikleri görüntülenir.

Paneldeki özellikler kategorilere göre gruplanmıştır, ancak alfabetik olarak da dizilebilir. Panelin üstünde bulunan araç kutusundan Categorized (Kategorileştirilmiş) veya Alphabetic (Alfabetik) seçenekleri işaretlenerek özelliklerin görünümleri değiştirilebilir.



RESİM 3.7: Properties paneli.

Panelin en altında bulunan bölümde, her özelliğin açıklaması bulunur.

İPUCU **Bir nesnenin üzerindeyken** F4 **tuşuna basınca, Properties paneli görüntülenir.**

Konu 7: Help Kullanımı

- En sık kullanılan kaynaktır.
- MSDN (Microsoft Developer Network) kütüphaneleri
- Dynamic Help
  - İçeriği, seçilen nesnelere göre değişir.
  - **F1** ile dinamik yardım
- Search
  - Zengin arama seçenekleri
- Index
  - Alfabetik konu dizini
- Contents
  - MSDN kütüphanelerinin hiyerarşik görünümü

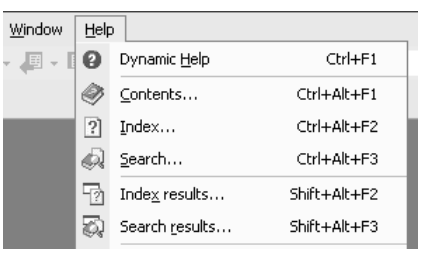
Yazılım geliştirirken en çok kullanacağımız kaynaklar yardım dosyaları olacaktır. Bir programlama dilinin çok çeşitli özellikleri ve kullanım farklılıkları olabilir. İyi bir programcı bütün bu özellikleri ezbere bilen değil, bu özellikleri en kısa sürede bulan, öğrenen ve kullanan programcıdır. Yardım dosyalarının kullanımını bilmek, programcılığın temelini oluşturan önemli unsurlardan biridir.

DİKKAT **Visual Studio yardımını kullanmak için, MSDN (Microsoft Developer Network) yardım kütüphanelerinin yüklenmiş olması gerekir.**

Visual Studio yardımı programcıya, gelişmiş özelliklere sahip paneller ve yardım dosyaları ile geniş bir kullanım kolaylığı sağlar.

Visual Studio yardım dosyalarının yapısı, başlık, içerik ve ilişkili konular (See Also) bölümlerinden oluşur. Ayrıca her yardım dosyasının altında bulunan Send Comments bağlantısı ile konu hakkında yorum gönderilebilir.

Visual Studio yardım panelleri Dynamic Help, Search, Index ve Contents olarak dörde ayrılır. Bu panellere, Help menüsünden ulaşılabilir (Resim 3.8).



RESİM 3.8: Help menüsü.

Dynamic Help

Dynamic Help (Dinamik yardım) paneli, içeriği otomatik olarak değişen bir araçtır. Kod yazarken, panellerde veya sayfalarda nesneler seçildiğinde, kullanıcının başka bir işlem yapmasına gerek kalmadan, o nesne hakkındaki yardım konularını listeler. **F1** tuşuna bastığımız zaman ise, seçilen nesneye ait, Dynamic Help panelindeki ilk yardım konusu yeni bir sayfada görüntülenir.

Paneldeki yardım konuları Help, Samples ve Getting Started olarak üç bölüme ayrılmıştır. Help bölümü, seçilen nesneyle ilişkili olan kavramların listelendiği bölümdür. Samples, konuyla ilgili kod örnekleri bulunan yardım dosyalarını gösterir. Getting Started, çalışılan sayfalara göre değişen, temel işlemleri içeren başlangıç yazılarını gösterir.

Search

Search (Arama) paneli, MSDN kütüphanelerinde arama yapılan paneldir. Look for metin kutusuna, aranacak anahtar kelimeler girilir. Filtered by (Filtreleme) ile arama sonuçları belli konulara göre sınırlanır ve istenmeyen seçeneklerin gösterilmesi engellenir.

Search panelinde, Search in titles only, Match related words, Search in previous results, Highlight search hits arama seçenekleri bulunur:

- **Search in titles only.** Sadece konu başlıklarında arama yapar; içerik kısmına bakmaz.
- **Match related words.** Kelimeleri yazıldığı gibi arar; benzer yazımlı kelimeleri aramaz.
- **Search in previous results.** İlk aramadan sonra aktif olan bu seçenek ile kelimeler, bir önceki aramada bulunan sonuçlar arasında aranır.
- **Highlight search hits.** Bulunan yardım sayfalarında, aranan kelimelerin seçili olmasını sağlar.

Bulunan sonuçlar Search Results (Arama Sonuçları) panelinde gösterilir. Bu panelde;

- Title, konunun başlığını
- Location, MSDN kütüphanelerinde hangi başlık altında bulunduğunu
- Rank, konunun aranan kelimeye olan yakınlık derecesini ifade eder.

Index

Index (Dizin) paneli, yardım dosyalarındaki bütün konuları alfabetik sırada dizer. Filtreleme işlevi, arama panelinde olduğu gibidir. Bu panelin özelliği, aranacak kelime yazılırken, bu kelime ile başlayan tüm konuların alfabetik sırada gösterilmesidir. Bu şekilde, aranan konulara çok hızlı bir şekilde ulaşılabilir.

Eğer bir konu ile ilgili birden fazla yardım dosyası varsa, Index Results (Dizin Sonuçları) panelinde bu seçenekler gösterilir.

Contents

Contents (İçerik) panelinde, tüm MSDN içeriği konulara göre hiyerarşik yapıda, kategorilere ayrılmış olarak gösterilir. Bu panelde de aynı şekilde filtreleme yapılarak istenmeyen içerikler çıkartılabilir.

Bir yardım dosyası açıkken, Help menüsünden Sync Contents (İçerik senkronizasyonu) komutu seçilerek o yardım dosyasının Contents panelindeki yeri bulunabilir.

LAB 3.1: Help Kullanımı

Bu lab tamamlandıktan sonra:

- Dynamic Help kullanımını öğrenecek,
- Search paneli ile arama yapabilecek,
- Contents paneli ile MSDN kütüphanelerinin hiyerarşik yapısını öğrenecek,
- Index paneli ile içeriğe hızlı bir şekilde ulaşabilecek,
- Yardım dosyalarını yorumlayabileceksiniz.

Bu labı tamamlamak için, MSDN yardım kütüphaneleri yüklenmiş olmalıdır.

Dynamic Help

1. Help menüsünden Show Start Page komutunu seçin.
2. Help menüsünden Dynamic Help komutunu seçerek Dynamic Help panelini açın. Panelde gösterilen ilk konunun ismi nedir?
3. **CTRL+ALT+X** tuşlarına basarak Toolbox panelini açın. Dynamic Help menüsünde ne değişti?
4. Toolbox panelinde, Windows Forms sekmesindeki `Button` nesnesini seçin. Dynamic Help panelindeki ilk konunun ismi ne olarak değişti?
5. `Button` seçiliyken **F1** tuşuna basın. Açılan sayfanın ismi nedir?

Contents

1. Help menüsünden Sync Contents komutunu seçin. Button Members konulu yardım dosyası hangi konuların altında bulunuyor?
2. Contents panelinin ilk başlığı olan Visual Studio .NET altında, Getting Assistance → "Using Help in Visual Studio .NET" → "Tips for Using the Help Keyword Index" konulu yardımı açın. File menüsünden Print komutunu seçin ve sayfayı yazdırın.

DİKKAT **Sayfayı yazdırmak için bilgisayarınıza bağlı bir yazıcı bulunması gerekir.**

İPUCU **Yardım dosyalarını yazdırmak, özellikle uzun metinlerde kolay çalışma imkanı sağlar.**

3. Contents panelini kapatın.

Search

1. Help menüsünden Search komutunu seçin. Look for metin kutusuna Visual Studio .NET yazın. Search in titles only ve Match related words seçeneklerini işaretleyin. Search düğmesini tıklayın.

   Kaç tane konu bulundu? En üst dereceli konu nedir?

2. Search in previous results seçeneğini işaretleyin. MSDN kelimesini aratın. Kaç konu bulundu?
3. Search in previous results seçeneğinin işaretini kaldırın. MSDN kelimesini tekrar arattığınız zaman kaç konu bulundu? Search in titles only seçeneğinin işaretini kaldırınca kaç konu bulundu?
4. Search Results ve Search panellerini kapatın.

Index

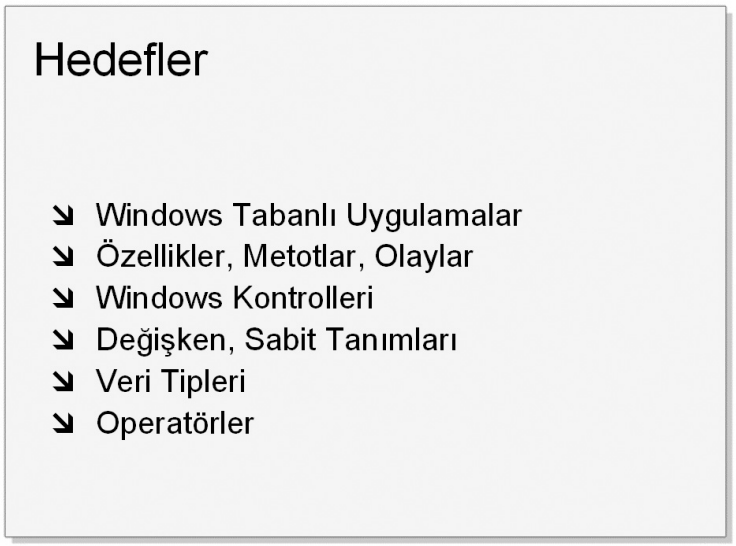
1. Help menüsünden Index komutunu seçin. Look for metin kutusuna "file types" yazın. İlk çıkan konu nedir?
2. Filtre olarak Visual Basic seçeneğini işaretleyin. İlk olarak hangi konu gösterilir?
3. “File Types” konusunu tıklayın. Açılan sayfada Solution Files (.sln and .suo) adlı bölümü inceleyin.
4. Project Files başlığında, Visual Basic and Visual C# alt başlığının altındaki “File Types and File Extensions in Visual Basic and Visual C#” konusunu tıklayın.
5. Açılan yardım dosyasını inceledikten sonra, sayfanın See Also başlığı altındaki “What's New in Projects” konusunu sağ tıklayın. Açılan menüden “Open Link in New Window” komutunu seçin. Window menüsünden “New Vertical Tab Group” komutunu seçin.

   Bir önceki yardım dosyasıyla arasındaki benzerlikleri inceleyin.
6. Window menüsünden “Close All Documents” seçeneği ile bütün sayfaları kapatın ve Visual Studio ortamından çıkın.

Modül Sonu Soruları & Alıştırmalar

1. Visual Basic profili için, Object Browser paneline hangi kısayolla ulaşılır?
2. Visual Studio ortamında tüm sabitlenmiş panelleri gizlemek için hangi menü komutu kullanılır?
3. Properties panelindeki özellikler alfabetik olarak nasıl sıralanır?

Modül 4: Visual Basic.NET ile Windows Tabanlı Programlama

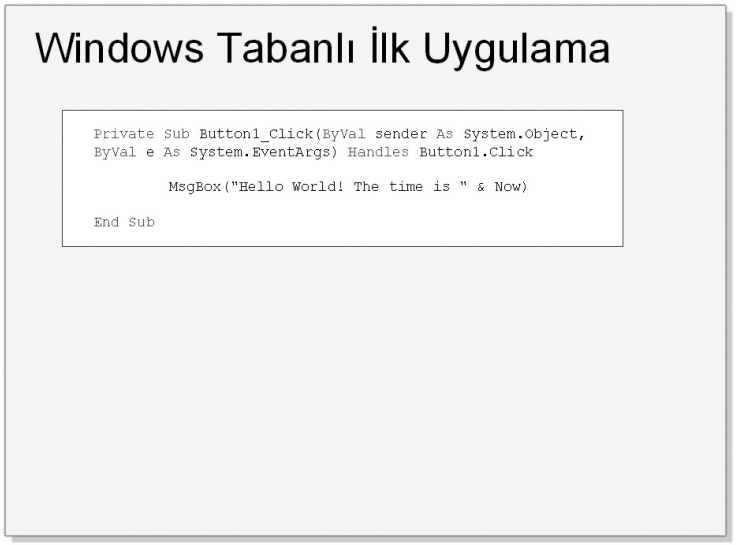


Windows tabanlı uygulamalar, Windows işletim sistemi üzerinde çalışan uygulamalardır. Windows uygulamaları Windows formları ve kontrollerinden oluşur. Visual Studio bu formların ve üzerindeki kontrollerin tasarımını ve kodların yazılmasını büyük ölçüde kolaylaştırarak, uygulama geliştirme sürecini daha hızlı ve kolay hale getirir.

Bu modülü tamamladıktan sonra;

- Windows tabanlı programlamada kullanılan kontrolleri tanıyacak,
- Kontrollerin özellik, metot ve olay kavramlarını öğrenecek,
- Visual Basic .NET dilinde değişken ve sabit tanımlamayı öğrenecek,
- Veri tiplerini tanıyacak,
- Operatörleri kullanabileceksiniz.

Konu 1: İlk Uygulama (Hello World, The Time Is...)



Visual Basic .NET ile yazacağımız Windows uygulaması ekrana, "**Hello World!**" yazısını ve o anki zamanı gösteren bir bilgi mesajını çıkartır.

1. Visual Studio çalışma ortamını açın.
2. File menüsünden, New alt menüsünü ve Project komutunu seçin. "New Project" iletişim kutusu, yazılacağı dile, çalışacağı ortama göre değişen proje tiplerini listeler.
3. Proje tiplerinden Visual Basic Project ve Windows Application tipinin seçili olduğunu kontrol edin.
4. Name özelliğine **HelloWorld** yazın ve OK düğmesini tıklayın. Açılan Windows projesinde başlangıç olarak bir adet Windows Form, tasarım görünümünde açılır.
5. Toolbox panelinden **Button** kontrolünü formun üzerine sürükleyip bırakın. Properties panelini açarak **Button** kontrolünün **Text** özelliğine "**Hello World!**" yazın.
6. Eklenen **Button** kontrolünü çift tıklayarak kod sayfasına geçin. **Button** kontrolü tıklandığında çalıştırılacak kodu yazın:

```
MsgBox("Hello World! The time is " & Now)
```

NOT **Yazdığınız kodun ne anlama geldiğini belirtmek için yorum satırları kullanmak, kodları okumayı kolaylaştırır. Yorum satırları tek tırnak ' ile başlayarak yazılmalıdır.**

7. `MsgBox` metodunun yazıldığı kodun üstüne, yapılmak isteneni belirten bir yorum satırı yazın.

```
' MsgBox metodu ile kullanıcıya Merhaba diyoruz.
' Now özelliği ile o andaki saat ve gün
' değerlerini de kullanıcıya gösteriyoruz.
```

8. **F5** tuşuna basarak projeyi çalıştırın.

İPUCU **Çalışma sayfaların isimlerinin yanında yıldız işaretinin gözükmesi, o sayfada değişiklik yapıldığını, ancak henüz kaydedilmediğini belirtir. Proje dosyalarınızı** CTRL+S **tuşlarına basarak sıkça kaydedin.**

Konu 2: Özellikler, Metotlar ve Olaylar

- Özellikler
  - Görünüm, yerleşim, davranışlara özgüdür.
  - Properties paneli
  - Text, Name, Size, BackColor
- Metotlar
  - Yapılan işlemler
  - Parametre ile ya da parametresiz çağrılırlar.
  - Focus, Select, Hide, Show
- Olaylar
  - Başlarına gelen işlemlerdir.
  - Click, MouseDown, Enter

.NET kontrolleri üç temel kavramdan oluşur.

Özellikler

Özellikler, kontrollerin görünümü, yerleşimi veya davranışlarına özgü niteliklerdir. Örneğin, bir **`Button`** kontrolünün **`Text`** özelliği, üzerinde yazan yazıya erişmemizi sağlar.

Kontrollerin özellikleri, tasarım anında Properties panelinden ulaşılabileceği gibi, kod tarafında da okunup değiştirilebilir.

Kontrollerin birçok özelliği hem okunabilir, hem de değiştirilebilir. Ancak bazı özellikler salt okunur (ReadOnly) ve salt yazılır (WriteOnly) olabilir. Bu tip özellikler Properties panelinde gözükmezler.

Kontrollerin birçok ortak özellikleri vardır.

- **`Text` (Yazı).** Kontrollerin **`Text`** özelliği, üzerinde görüntülenen yazıdır. Bu özellik, çalışma anında sıkça okunup değiştirilerek kullanıcıyla iletişim sağlanır.

  `TextBox` kontrolüne girilen bir değerin okunup **`Label`** kontrolüne yazılması için, kontrollerin **`Text`** özellikleri kullanılır.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Label1.Text = TextBox1.Text
End Sub
```

RESİM 4.1.

- **`Name` (İsim).** `Name` özelliği kontrollere ulaşmak için kullanılan özelliktir. Birçok kontrolün `Text` özelliği aynı olabilir. Ancak her biri ayrı birer nesne olduğu için, `Name` özelliklerinin benzersiz olması gerekir.

```
TextBox2.Text = TextBox1.Text
```

İki `TextBox` kontrolünün yazıları aynı, fakat isimleri farklıdır.

- **`Size` (Büyüklük).** Kontrollerin büyüklük özelliğidir. `Height` (yükseklik) ve `Width` (genişlik) özelliklerinden oluşur. Genellikle tasarım anında belirlenen bu özellik, çalışma anında da değiştirilebilir.

```
Label1.Height = 10
Label1.Width = 20
```

- **`BackColor` (Arka plan rengi).** Kontrollerin arka plan renginin ayarlandığı özelliktir. Bu özelliğin değeri, `Color` (renk) nesnesinde tanımlı değerlerle belirlenir.
- **`ForeColor` (Önalan rengi).** Kontrollerin üzerindeki yazıların rengini belirler.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Button1.BackColor = Color.Black
        Button1.ForeColor = Color.White
End Sub
```

RESİM 4.2.

- **`Visible` (Görünür).** Kontrollerin ekranda görünüp görünmediklerini belirleyen özelliktir. `True` ve `False` olmak üzere iki değer alabilir. `Boolean` veri tiplerinden bu modülde bahsedilecektir.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
```

```
        'Label kontrolünü gizle
        Label1.Visible = False

        'Label kontrolünü göster
        Label1.Visible = True
    End Sub
End Class
```

Metotlar

Metotlar kontrollerin yaptığı işlemlerdir. Metotlar parametreyle veya parametresiz çağrılabilir. Parametreyle çağırmak, metodun girilen değere göre işlem yapacağını belirtir. Örneğin **Focus** (Odaklan) metodu, parametre beklemeden çalışır ve kontrolün seçilmesini sağlar.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        ' İşlem yapıldıktan sonra
        ' TextBox kontrolüne odaklan
        TextBox1.Focus()
End Sub
```

Kontrollerin bazı ortak metotları vardır.

- **Select (Seç).** **Select** metodu **Focus** ile aynıdır, ama **TextBox** kontrolünün **Select** metodunun diğerlerinden bir farkı vardır. **TextBox** içindeki yazının belli bir kısmını ya da tamamını, verilen parametrelere göre seçer (Resim 4.3).

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        TextBox1.Text = "Yazılım Uzmanı"
        TextBox1.Focus()

        ' Sekizinci karakterden sonra, beş karakter seç
        TextBox1.Select(8, 5)
End Sub
```

RESİM 4.3.

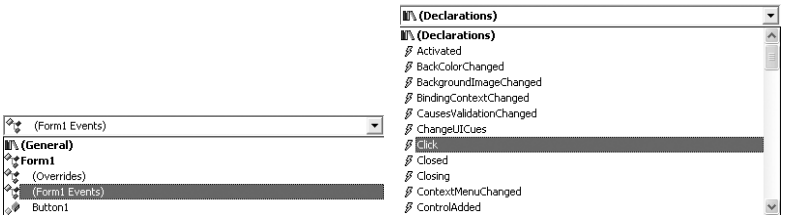
- **BringToFront (Öne Getir).** Kontrolü, üst üste duran kontroller arasından en öne getirir.

- **`SendToBack` (Arkaya Gönder).** Kontrolü, üst üste duran kontrollerin en arkasına gönderir.
- **`Hide` (Sakla).** Kontrolün gözükmesini engeller.
- **`Show` (Göster).** Kontrolün gözükmesini sağlar.

Olaylar

Olaylar kontrollerin başına gelen işlemlerdir. Olayların metotlardan farkı, bu işlemlerin kontrollerin elinde olmadan gerçekleşmesidir. Örneğin bir **`Button`** kontrolünün tıklanması, o kontrolün isteği dışında yapılmıştır. Bu olayın tetiklenmesinde kontrolün bir rolü yoktur. Bu olaylar gerçekleştiği zaman yapılması gereken işlemler, ilgili olayın yordamına yazılır. **`Button1`** isimli kontrol tıklandığı zaman gerçekleştirmek istenen eylemler **`Button1_Click`** yordamına yazılır.

Visual Studio ortamı, kontrollerin olaylarını kolay bir şekilde seçmeyi sağlar. Kod sayfalarında kontrollerin bulunduğu listeden, istenen kontrol seçilir. Kontrolün olaylarının listelendiği diğer listeden de istenen olay seçilir (Resim 4.4).



RESİM 4.4: Kontrollerin olaylarının seçilmesi.

```
Private Sub Form1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Click
        MsgBox("Form üzerine tıklandı")
End Sub
```

Visual Studio, olayların yordam isimlerini **`KontrolİsmiOlayİsmi`** biçiminde yazar.

Kontrollerle çalışırken benzer olaylar kullanılır.

- **`Click` (Tıklandığında).** Kontrol tıklandığı zaman tetiklenen olaydır. Windows tabanlı programlamada en sık kullanılan olaylardan biridir.
- **`MouseDown` (Fare düğmesi basıldığında).** Fare, kontrolün üzerindeyken herhangi bir düğmesine basıldığı zaman gerçekleşen olaydır. Bu olay, **`Click`** olayından önce çalışır.
- **`MouseUp` (Fare düğmesi bırakıldığında).** Fare, kontrolün üzerindeyken basılan düğme bırakıldığı zaman çalışır.
- **`Enter` (Girildiğinde).** Kontrol seçildiği veya üzerine odaklanıldığı zaman gerçekleşen olaydır.

- **`Leave` (Çıkıldığında).** Başka bir kontrol seçilmek üzere çıkıldığında, bu kontrolün **`Leave`** olayı tetiklenir.
- **`VisibleChanged` (Görünürlüğü değiştiğinde).** Kontrolün görünüp görünmediğini belirten **`Visible`** özelliği değiştiği zaman tetiklenir.

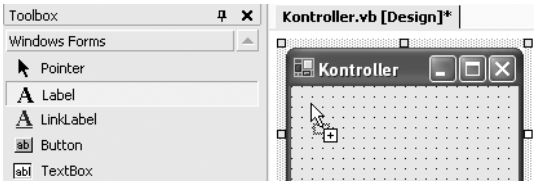
İPUCU **Olayların çalışma sıralarını test etmek için tüm olay yordamlarına, mesaj kutusu çıkaran (`MsgBox`) kod yazın. Daha sonra projeyi çalıştırıp kontroller üzerinde yapılan değişikliklere göre olayların çalışma sıralarına bakın.**

Konu 3: Visual Basic.NET'e Kontrollerin Eklenmesi

Visual Studio'ya Kontrol Eklenmesi

- Toolbox panelinden kontrollerin eklenmesi
- Form
- Button
- TextBox
- Label
- ComboBox
- ListBox
- Timer

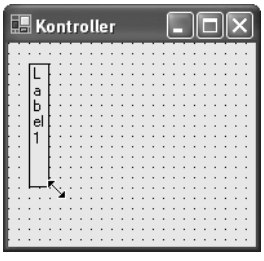
Windows tabanlı uygulamalar geliştirirken sıkça kullanacağımız bir grup kontrol vardır. **`Form`** kontrolü hariç diğer bütün kontroller Toolbox panelinden seçilir. Bu kontroller sürüklenip form üzerinde istenen pozisyona bırakılır (Resim 4.5).



RESİM 4.5: Kontrollerin eklenmesi.

Kontroller, Toolbox panelinde çift tıklanarak da eklenebilir.

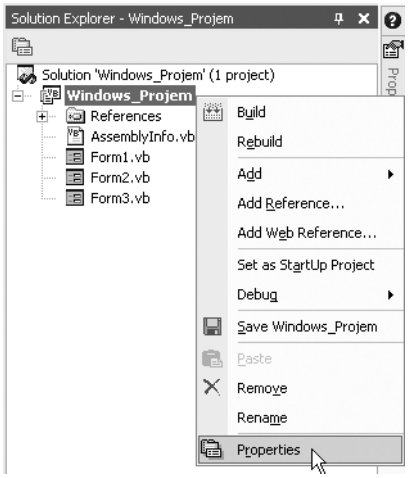
Kontrollerin tasarım anında büyüklükleri ve yerleri **`Size`** ve **`Location`** özellikleri ile değiştirilebileceği gibi, fare ile de istenen şekilde ayarlanabilir (Resim 4.6).



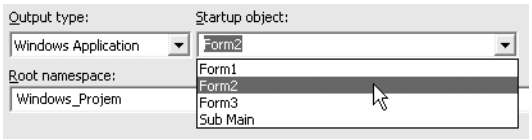
RESİM 4.6: Kontrollerin ayarlanması.

Form

Windows uygulamaları, Windows kontrollerinin tutulduğu pencereler olan formlardan oluşur. Bir Windows projesi açıldığı zaman **`Form`** kontrolü otomatik olarak eklenir. İkinci bir form eklemek için Project menüsünden Add Windows Form komutunu seçilir. Proje çalıştığı zaman başlangıç formu görüntülenir. Başlangıç formu projenin özelliklerinden değiştirilir (Resim 4.7 ve Resim 4.8).



RESİM 4.7: Proje özelliklerinin açılması.



RESİM 4.8: Başlangıç formunun değiştirilmesi.

Visual Studio ortamında formlar, tasarım sayfası ve kod sayfası olmak üzere iki farklı sayfada görüntülenir. Tasarım sayfası, formun ve üzerindeki kontrollerin görünümlerini kolay bir şekilde değiştirmeyi sağlar. Visual Studio bu sayfada yapılan değişiklikleri kod sayfasında eşzamanlı olarak günceller. Örneğin, bir **`Button`** kontrolünün genişliğini fare ile değiştirdiğimiz zaman, kod sayfasında bu kontrolün **`Width`** özelliği yapılan değişikliğe göre güncellenir. Aynı değişiklikler Properties panelinde de görülebilir.

Formların, diğer kontrollerin özelliklerinden farklı bazı özellikleri vardır.

- **`ControlBox` (Denetim kutusu).** Form üzerindeki simge durumuna küçültme, ekranı kaplama ve formu kapatma (Minimize / Maximize / Close) kutularının görünümünü ve erişilebilirliğini kontrol eder.

NOT **Formun `ControlBox` özelliği `False` iken uygulama, Debug menüsünden Stop Debugging komutu seçilerek kapatılabilir.**

- **`StartPosition` (Başlangıç pozisyonu).** Form açıldığı zaman nerede gözükeceğini belirler. **`CenterScreen`** seçeneği formu ekranın ortasında gösterir.

  Formlar açıldığı zaman **`Load`** olayı gerçekleşir. Eğer form, başlangıç formu olarak seçilmişse, proje başladığı zaman çalıştırılmak istenen kodlar bu olayın yordamına yazılır.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        Label1.Text = "Proje başlatıldı. Kayıt zamanı: " &
Now
End Sub
```

Button

Bir Windows düğmesini temsil eder. **`Button`** kontrolü tıklandığında **`Click`** olayı tetiklenir. Bu olay gerçekleştiği zaman yapılacak işlemler, **`ButtonİsmiClick`** yordamında yazılır.

```
Private Sub btnRenkDegistir_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnRenkDegistir.Click
        btnRenkDegistir.ForeColor = Color.Gray
End Sub
```

TextBox

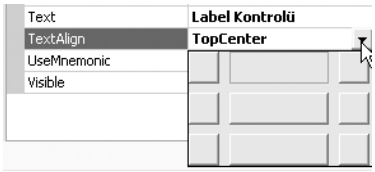
Bir Windows metin kutusunu temsil eder. Kullanıcıların değer girerek programla haberleşmesini sağlamak amacıyla kullanılır. **`TextBox`** kontrolündeki yazı değiştiği zaman **`TextChanged`** olayı gerçekleşir.

```
Private Sub TextBox1_TextChanged(ByVal sender As Object,
ByVal e As System.EventArgs) Handles TextBox1.TextChanged
        ' TextBox içindeki yazı değiştiği zaman
        ' aşağıdaki kod çalışır.
        MsgBox("Yazı değiştirildi: " & TextBox1.Text)
End Sub
```

Label

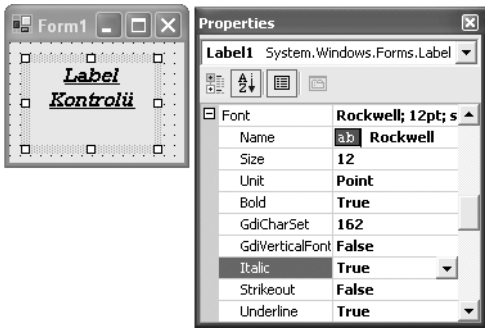
Bir Windows etiketini temsil eder. Kullanıcıya, form üzerinde bir yazıyı göstermek amacıyla kullanılır. Bu yazının görünümü, **`Label`** kontrolünün bazı özellikleri ile değiştirilir.

- **`TextAlign` (Yazı hizalama).** Yazının **`Label`** kontrolü üzerinde nerede duracağını belirler (Resim 4.9).



RESİM 4.9: `TextAlign` özelliği.

- `Font` **(Yazı Tipi).** `Font` özelliği birçok alt özellik taşır. Bunlardan bazıları en sık kullanılan özelliklerdir (Resim 4.10).
  - `Name`. Yazı tipinin ismini belirler. Varsayılan durumda `Microsoft Sans Serif` seçilidir.
  - `Size`. Karakterlerin büyüklüğünü belirler. Varsayılan büyüklük `8,5` değerindedir.
  - `Bold` **(Kalın).** Yazının kalın tipte olmasını belirler.
  - `Italic` **(İtalik).** Yazının italik tipte olmasını belirler.
  - `UnderLine` **(Altı çizgili).** Yazının altı çizgili olmasını belirler.



RESİM 4.10: `Font` özelliği.

ComboBox

Bir Windows açılan kutusunu temsil eder. `ComboBox` kontrolü, kullanıcıların bazı değerleri açılan bir listeden seçmesini sağlar. Listeye tasarım anında veya çalışma anında öğe eklenebilir. Listeye öğe eklemek için kontrolün `Items` özelliğinden faydalanılır.

Tasarım anında öğe eklemek için Properties panelinden `Items` özelliği seçilir. String Collection Editor penceresinde, her öğenin değeri tek bir satırda yazılır (Resim 4.11).

RESİM 4.11: String Collection Editor penceresi.

Çalışma anında öğe eklemek için kod sayfasında, kontrolün **Items** özelliğinin **Add** metodu kullanılır.

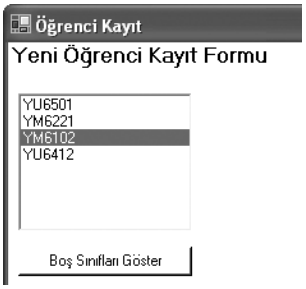
```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
        ComboBox1.Items.Add("Lise")
        ComboBox1.Items.Add("Üniversite")
        ComboBox1.Items.Add("Yüksek Lisans")
        ComboBox1.Items.Add("Doktora")
End Sub
```

RESİM 4.12: ComboBox kontrolünün çalışması.

ListBox

Bir Windows liste kutusunu temsil eder. Kontroldeki öğeler sabit bir liste olarak görüntülenir. **ListBox** kontrolüne öğe ekleme işlemi, **ComboBox** kontrolündeki işlemlerle aynıdır. **ListBox** kontrolünün **ComboBox** kontrolünden farkı, birden fazla öğenin seçilebilir olmasıdır (Resim 4.13).

```
Private Sub btnBosSiniflar_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnBosSiniflar.Click
        ListBox1.Items.Add("YU6501")
        ListBox1.Items.Add("YM6221")
        ListBox1.Items.Add("YM6102")
        ListBox1.Items.Add("YU6412")
End Sub
```



RESİM 4.13: ListBox kontrolünün çalışması.

Timer

Bir Windows sayacını temsil eder. Sayaç çalışmaya başladığı zaman, belirli zaman aralıklarında **Tick** olayı gerçekleşir. **Timer** kontrolünün **Interval** değeri, **Tick** olayının kaç milisaniyede bir gerçekleşeceğini belirler. Örneğin, **Interval** değeri **2000** olan bir sayaç, **Tick** olayında yazılan kodları iki saniyede bir çalıştırır.

Sayacı başlatmak için kontrolün **Start** metodu, durdurmak için ise **Stop** metodu kullanılır. **Enabled** özelliği, sayacın aktif olup olmadığını belirler.

```
Private Sub btnBasla_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnBasla.Click
        ' Sayaç 5 saniyede bir çalışacak
        Timer1.Interval = 5000
        Timer1.Start()
End Sub

Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        MsgBox("Sayaç çalışıyor...")
End Sub

Private Sub btnDur_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnDur.Click
        Timer1.Stop()
End Sub
```

LAB 4.1: Kronometre Uygulaması

Bu labı tamamladıktan sonra;

- Form ve üzerindeki kontrollerin görünüm özelliklerini öğrenecek,
- **ComboBox** ve **ListBox** kontrollerine öğe ekleyebilecek,
- **TextBox** kontrolünden değer okuyabilecek,
- **Timer** kontrolünün çalışma şeklini öğreneceksiniz.

Form Üzerine Kontrollerin Eklenmesi, Biçimlendirmelerin Yapılması

1. "**Kronometre**" isminde yeni bir Windows projesi açın.
2. Properties panelinden, **Form1** nesnesinin **BackColor** özelliğini "**Menu**" olarak seçin. **Font** özelliğini, yanındaki **+** simgesini tıklayarak genişletin. **Font** özelliğinin alt özellikleri listelenir.
   - **Name** özelliğini **Tahoma**,
   - **Text** özelliğini "**Yazılım Uzmanlığı Kronometre Uygulaması**",
   - **Size** özelliğini **10** olarak ayarlayın.

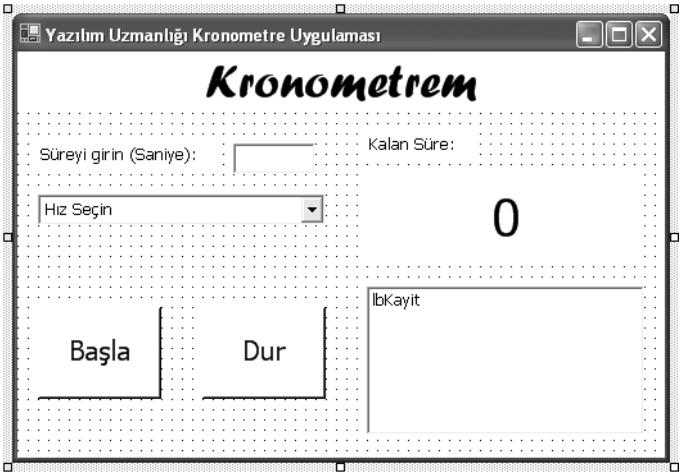
   Form görünüm özellikleri, eklenecek kontrollerin (değiştirilmedikleri sürece) görünümlerini de etkiler.
3. Toolbox panelinden form üzerine bir **Label** ekleyin. Özelliklerini atayın:
   - **Text: Kronometrem**
   - **Font – Name: Forte**, **Font – Size: 28**
   - **Dock: Top**
   - **TextAlign: BottomCenter**
4. Bir **Label** kontrolü ekleyin. Özelliklerini atayın:
   - **Text: 0**
   - **Font – Size: 30**
   - **TextAlign: MiddleCenter**
   - **Name: lblSure**
5. Forma bir **Timer** kontrolü ekleyin. **Name** özelliğini **tmrKronometre** olarak değiştirin.

İPUCU **Kod tarafında kullanacağınız kontrollerin isimlerini değiştirmek, daha sonra ulaşmak istediğinizde zaman kazandıracaktır.**

6. Bir **ComboBox** ekleyin. **Text** özelliğini "**Hız Seçin**" olarak, **Name** özelliğini de **cmbInterval** olarak değiştirin. **Items Collection** içine sırayla **1000**, **2000**, **3000**, **4000** değerlerini girin.

Bu kontrol, çalışma anında **Timer** kontrolünün **Interval** özelliğini değiştirmeyi, dolayısıyla kronometrenin hızını ayarlamayı sağlayacaktır.

7. **Text** özellikleri birinde "**Dur**", diğerinde "**Başla**" olan iki **Button** ekleyin. Kontrollerin **Name** özelliklerini sırayla **btnDur** ve **btnBasla** olarak değiştirin.
8. Bir **ListBox** kontrolü ekleyin ve **Name** özelliğini **lbKayit** olarak değiştirin. Bu kontrol kronometrenin başlama ve durma zamanlarını kaydetmeyi sağlayacaktır.
9. Bir **TextBox** kontrolü ekleyin. **Name** özelliğini **txtSure** olarak değiştirin ve **Text** özelliğinde yazan yazıyı silin.
10. Eklenen kontrolleri, Resim 4.14'te görünen şekilde düzenleyin.



RESİM 4.14: Kronometre form görünümü.

Kodların yazılması

1. Form üzerinde sağ tıklayın ve View Code komutunu seçin.
2. Açılan kod sayfasında, **KalanSure** isimli bir değişken tanımlayın.

```
Dim KalanSure As Integer
```

3. Formun tasarım görünümüne dönün ve **Başla** isimli **Button** kontrolünü çift tıklayın. **btnBasla_Click** yordamı içine **Timer** kontrolünü ayarlayıp başlatan, **ListBox** kontrolüne kayıtları giren, kalan süreyi **Label** kontrolünde görüntüleyen kodları yazın.

```
Private Sub btnBasla_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnBasla.Click
        ' Başlangıç zamanı "KalanSure" değişkenine atanır.
        KalanSure = txtSure.Text

        ' Kalan süre kullanıcıya gösterilir.
```

```
        lblSure.Text = KalanSure

        ' ListBox kontrolüne kayıt girilir.
        lbKayit.Items.Add("Kronometre başladı: " &
Now.TimeOfDay.ToString)

        ' ComboBox kontrolünden seçilen değer,
        ' Timer kontrolünün çalışma hızını belirler.
        tmrKronometre.Interval = cmbInterval.Text

        ' Timer kontrolünü çalıştırır.
        tmrKronometre.Start()
End Sub
```

4. **Dur** isimli **Button** kontrolünü çift tıklayın. **btnDur_Click** yordamı içine **Timer** kontrolünü durduracak ve **ListBox** kontrolüne kayıtları ekleyecek kodları yazın.

```
Private Sub btnDur_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnDur.Click
        ' Timer kontrolünü durdurur.
        tmrKronometre.Stop()

        ' ListBox kontrolüne kayıt girilir.
        lbKayit.Items.Add("Kronometre durduruldu: " &
Now.TimeOfDay.ToString)
End Sub
```

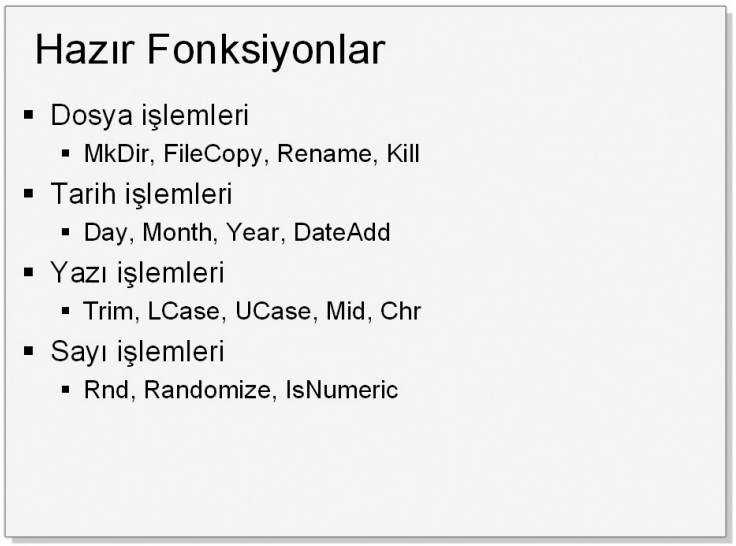
5. Tasarım görünümünde **tmrKronometre** isimli **Timer** kontrolünü çift tıklayın. **tmrKronometre_Tick** yordamı içine kalan süreyi azaltacak ve süre sıfırlandığında kronometreyi durduracak kodları yazın.

```
Private Sub tmrKronometre_Tick(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
tmrKronometre.Tick
        ' Her saniye geçtiğinde sure değeri 1 azalacaktır.
        KalanSure = KalanSure - 1

        'KalanSure değeri kullanıcıya gösterilir
        lblSure.Text = KalanSure

        ' KalanSure değeri sıfıra ulaşmışsa kronometre
durdurulur.
        If KalanSure = 0 Then
            tmrKronometre.Stop()
            lbKayit.Items.Add("Süre Doldu: " &
Now.TimeOfDay.ToString)
```

```
            MsgBox("Süre doldu")
        End If
End Sub
```

6. Projeyi başlatın, metin kutusuna **5** değerini girin. **Hız Seçin** açılan kutusundan **1000** değerini seçin ve **Başla** düğmesini tıklayın.
   - Süre başladıktan ve bittikten sonra **ListBox** kontrolündeki değişiklikler nelerdir?
   - Hız **3000** olarak seçildiğinde başlama ve bitiş zamanları arasındaki süre ne kadardır?

Konu 4: Hazır Fonksiyonlar



Visual Basic.NET dilindeki hazır fonksiyonlar, program yazarken en çok kullanılan işlemleri programcılara sunar. Bu fonksiyonlar **`Microsoft.VisualBasic`** altında gruplanmıştır. Örneğin, metin kutusuna girilen bir yazının soldan ilk üç karakterini almak için **`Microsoft.VisualBasic.Left`** hazır fonksiyonu kullanılır.

```
Label1.Text = Microsoft.VisualBasic.Left(TextBox1.Text, 3)
```

Farklı işlemler için hazırlanmış birçok fonksiyon vardır.

Dosya işlemleri:

- Yeni bir klasör açmak:

```
MkDir("C:\Kayitlar")
```

- Bulunduğu yer verilen bir dosyayı başka bir yere kopyalamak:

```
FileCopy("C:\LogDosyasi.txt ", " C:\Kayitlar\LogDosyasi.txt
")
```

- Bir dosyanın ismini değiştirmek:

```
Rename("C:\LogDosyasi.txt", "C:\Log_Dosyasi.txt")
```

- Bir dosyayı silmek:

```
Kill("C:\Log_Dosyasi.txt")
```

Tarih işlemleri:

- Verilen tarihin gününü almak:

```
MsgBox(Day(Now))
```

- Verilen tarihin ayını almak:

```
MsgBox(Month(Now))
```

- Verilen ay numarasının ismini almak:

```
MsgBox(MonthName(7))
```

- Verilen tarihin saatini almak:

```
MsgBox(Hour(Now))
```

- Verilen tarihe, ilk parametrede belirtilen zaman cinsinden bir değer eklemek:

```
MsgBox(DateAdd(DateInterval.Day, 20, Now))
```

- Günün saat, dakika, saniyesini almak:

```
MsgBox(TimeOfDay)
```

Yazı işlemleri:

- Yazının başındaki ve sonundaki boşlukları atmak:

```
Label1.Text = Trim("      BilgeAdam        ")
' Sonuç: BilgeAdam
```

- Tüm yazıyı küçük harfe çevirmek:

```
Label1.Text = LCase("BILGEAdam")
' Sonuç: bilgeadam
```

- Tüm yazıyı büyük harfe çevirmek:

```
Label1.Text = UCase("bilgeADAM")
' Sonuç: BİLGEADAM
```

- Yazının belirli bir bölümünü almak. İkinci parametrede verilen pozisyondan başlayarak, üçüncü parametredeki değer kadar karakter alınır:

```
Label1.Text = Mid("BilgeAdam", 6, 4)
' Sonuç: Adam
```

- Yazının parametrede belirtilen sıradaki karakteri almak:

```
Label1.Text = GetChar("BilgeAdam",9)
' Sonuç: m
```

- Verilen karakter koduna karşılık gelen karakteri almak:

```
Label1.Text = Chr(65)
' Sonuç: A
```

Sayı işlemleri:

- Rasgele sayı üretmek:

```
' Maximimum 400 değerini alan bir sayı üretir
Rnd() * 400
```

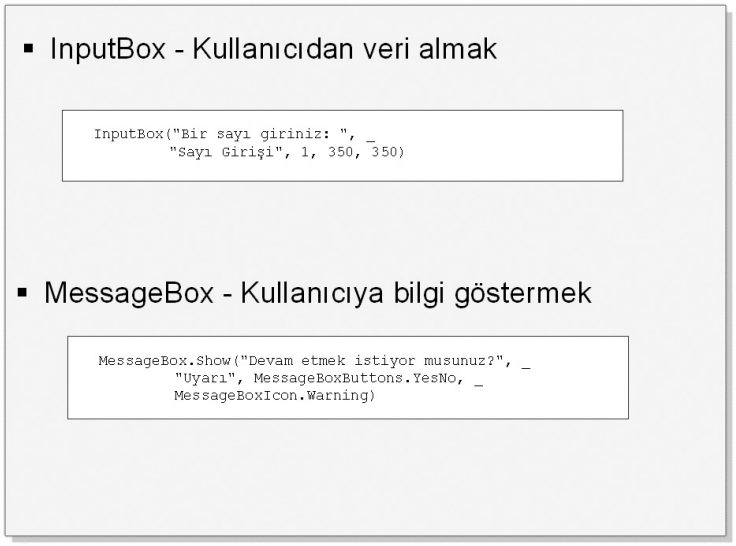
- **Rnd** fonksiyonu, uygulama baştan başladığında aynı değerleri üretir. Her çalışmada farklı değer üretmek için, **Rnd** fonksiyonundan önce **Randomize** fonksiyonunun çağırılması gerekir:

```
Randomize()
Rnd() * 400
```

- Parametrede verilen bir değerin sayı olup olmadığını kontrol etmek. Geriye dönen değer **True** ya da **False** mantıksal değeridir:

```
IsNumeric(TextBox1.Text)
```

Konu 5: InputBox



InputBox, kullanıcının veri girmesi için açılan bir mesaj kutusudur. Formlarda **TextBox** kontrolüne ihtiyaç duymadan veri almayı sağlar (Resim 4.15).

```
InputBox("Bir sayı giriniz: ", "Sayı Girişi", 1, 350, 350)
```

InputBox metodunun ilk parametresi, mesaj kutusunun gövdesinde gözükecek yazıdır (**Prompt**). Diğer tüm parametreler isteğe bağlıdır.

- **Title (Başlık).** Mesaj kutusunun başlığıdır
- **DefaultResponse (Varsayılan cevap)**. Kullanıcı veri girmediğinde varsayılan değerdir.
- **XPos (X pozisyonu).** Mesaj kutusunun sol kenarının, ekranın sol kenarına olan uzaklığıdır.
- **YPos (Y pozisyonu).** Mesaj kutusunun üst kenarının, ekranın üst kenarına olan uzaklığıdır.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        ListBox1.Items.Add(InputBox("İsim soyad giriniz:",
"Bilgi Girişi", "Bilinmiyor"))
End Sub
```

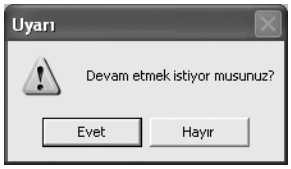
RESİM 4.15: InputBox.

Konu 6: MessageBox

MessageBox, kullanıcıya bilgi göstermek için açılan mesaj kutusudur. Bu mesaj kutusu dört öğeden oluşur (Resim 4.16).

- **Text (Yazı).** Mesaj kutusunda verilmek istenen bilgiyi tutan yazıdır.
- **Caption (Başlık).** Mesaj kutusunun başlığıdır.
- **Buttons (Düğmeler).** Mesaj kutusunda hangi düğmelerin gösterileceğini belirler.
- **Icon (Simge).** Mesaj kutusunda gösterilecek olan simgeyi ve açıldığı zaman çıkartılacak sesi belirler.

```
MessageBox.Show("Devam etmek istiyor musunuz?", "Uyarı",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
```



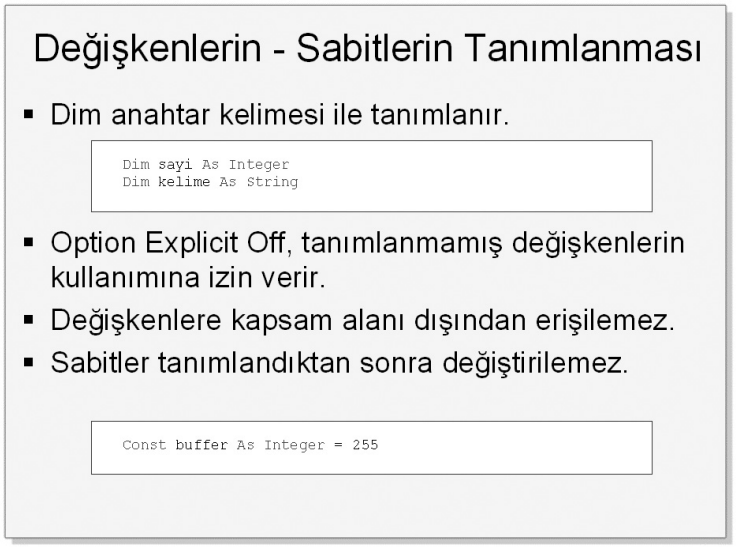
RESİM 4.16: **MessageBox**.

Mesaj kutusu, kapanırken hangi düğmenin tıklandığını **DialogResult** nesnesi ile programcıya bildirir.

```
If MessageBox.Show("Değişiklikler kaydedilsin mi?", "Kayıt",
MessageBoxButtons.YesNoCancel) = DialogResult.Cancel Then
        ' İptal tuşuna basıldığı zaman
        ' buraya girilir.
End If
```

NOT MsgBox **hazır fonksiyonu** MessageBox.Show **metodu ile aynı işleve sahiptir.**

Konu 7: Değişken – Sabit Nedir? Değişkenlerin ve Sabitlerin Tanımlanması



Değişken Nedir, Nasıl Tanımlanır?

Program yazarken, bazı verilerin daha sonra kullanılmak üzere bir yerde tutulması gerekebilir. Örneğin, bir hesaplama yapılırken, önceden hesaplanmış verilerin kullanılması istenirse, bu verileri tekrar hesaplamak yerine hafızada tutmak performansı artırır. Veya veritabanından alınan bir kullanıcı isminin hafızada tutulması, bu değer her istendiğinde veritabanına bağlanıp alınmasına tercih edilmelidir. Verilerin bu şekilde hafızada tutulması değişkenlerle sağlanır.

Değişkenler farklı türlerde verileri tuttukları için, farklı tiplere sahip olabilir. Bir negatif veya pozitif sayıyı tutan değişken ile yazı tutan bir değişken farklı tiplere sahiptir.

Değişkenler **`Dim`** anahtar kelimesi ile tanımlanır.

```
Dim sayi
```

NOT **Dim sözcüğü, "boyut" anlamına gelen Dimension kelimesinin kısaltmasıdır.**

Bu şekilde tanımlanan değişkenler **`Object`** (nesne) tipindedir, yani her türden veriyi tutabilirler. Nesne tipindeki değişkenler, tanımlanmalarının kolay olması ve istenen tipte değer tutabilmeleri açısından avantajlı olsa da, performansı önemli ölçüde düşürürler. Tuttukları değerlerin tipleri biliniyorsa, değişkenleri tanımlarken tiplerini belirlemek gerekir.

```
Dim sayi As Integer
Dim kelime As String
```

Tanımlanan değişkenlerin tipleri **As** anahtar kelimesinden sonra belirtilir.

Değişken isimlerini belirlerken bazı noktalara dikkat etmek gerekir:

- Boşluk, nokta, soru işareti, noktalı virgül, çift tırnak, tek tırnak, aritmetik operatörler, karşılaştırma operatörleri ve atama operatörleri ile parantezler kullanılamaz.
- Sayı ile başlayamaz.
- Visual Basic.NET dilinde tanımlı anahtar kelimeler kullanılamaz.
- **Dim DeğişkenAdı As VeriTipi** kuralına uyulmalıdır.

İPUCU **Değişken isimlerinde Türkçe karakter kullanılırsa, farklı dil seçenekli işletim sistemlerinde çalışma anında hata üretecektir.**

Hatalı bazı değişken tanımları:

```
Dim dim As Double
Dim (sayi) As Short
Dim 333sayisi As Integer
Dim "kelime" As String
Dim <isim> As String
```

Aynı tipteki değişkenler tek bir satır içinde tanımlanabilir.

```
Dim sayi1, sayi2 As Integer
```

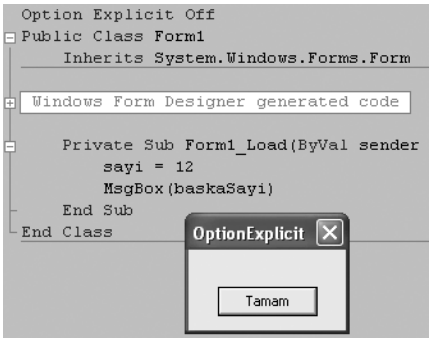
Değişkenlere değer atamak = operatörü ile yapılır. Eşitliğin sağ tarafındaki değer, sol tarafta bulunan değişkene atanır. Dolayısıyla sağ taraftaki ifadenin değeri değişmez.

```
sayi1 = 10
sayi2 = sayi1
```

Değişkenler tanımlandıkları sırada başlangıç değeri alabilirler.

```
Dim isim As String = "Enis Günesen"
```

Uygulama geliştirirken, değişkenlerin tanımlanarak kullanılması, yazım yanlışlarından kaynaklanan karışıklıkları engeller. Değişkenlerin tanımlanmadan kullanılması için **Option Explicit** seçeneğinin kapalı olması gerekir. **Option Explicit** seçeneğini projenin özelliklerindeki Build sekmesinden değiştirilebileceği gibi, kod sayfalarının en üstünde de değiştirilebilir.



RESİM 4.17: `Option Explicit` seçeneği kapalı.

`Option Explicit` seçeneğinin `On` olması, tanımlanmamış değişkenlerin kullanılmasına izin vermez, tasarım anında hata üreterek programcıya bildirir. Varsayılan durumda `On` değeri seçilidir.

Değişkenler program içinde, tuttukları verilere ulaşmak için kullanılır. Ancak değişkenlere ulaşmak, tanımlandıkları yerde veya alt bloklarda mümkündür. Bu kavrama değişkenlerin kapsam alanı (Scope) denir.

Kapsam alanı dışındaki bir yerden değişkene ulaşılamaz.

```
Namespace NameSpace1
    Module Module1
        Dim ModulDegiskeni As Integer

        Class Class1
            Dim SinifDegiskeni As Integer

            Sub Sub1()
                Dim YordamDegiskeni As Integer

                Do
                    Dim DonguDegiskeni As Integer
                Loop
            End Sub
            Sub Sub2()
                Dim YordamDegiskeni2 As Integer
            End Sub
        End Class
    End Module
End Namespace
```

Tablo 4.1'de kod bloklarından hangi değişkenlere ulaşılabildiği görülüyor.

TABLO 4.1.

	Module1	Class1	Sub1	Sub2	Loop
ModulDegiskeni	Evet	Evet	Evet	Evet	Evet
SinifDegiskeni		Evet	Evet	Evet	Evet
YordamDegiskeni			Evet		Evet
YordamDegiskeni2				Evet	Evet
DonguDegiskeni					Evet

Uygulamanın çalışması değişkenlerin kapsam alanlarındayken, bu değişkenler bellekte tutulur. Dolayısıyla değişkenlerin tanımlandıkları yer, kullanılacağı amaca göre seçilmelidir. Örneğin bir değişken birden fazla yordamda kullanılacaksa, bir üst düzeyde (sınıf düzeyinde) tanımlanmaları gerekir. Ancak sadece bir yordam içinde kullanılan değişkenler sınıf düzeyinde tanımlanırsa, bellekte fazladan yer tutar ve performans düşer. Sınıf düzeyindeki değişkenler, aynı sınıf içindeki fonksiyonlar ile değiştirilebilir ve sınıf örneğinin yaşam süresinde ilgili özelliklerine erişim sağlanabilir.

Sabit Nedir, Nasıl Tanımlanır?

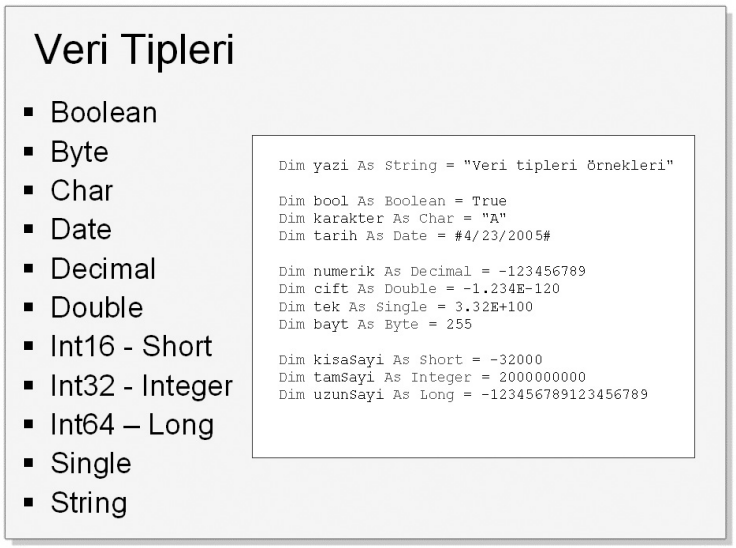
Sabit, sürekli aynı değeri tutan değişkendir. Uygulamanın çalışması boyunca değişmeyen bir değer kullanılıyorsa, sabit kullanılması kodun kolay okunmasını sağlayacaktır.

Sabitler tanımlandıktan sonra değiştirilemeyeceği için, tanımlandıkları anda değerlerinin verilmesi gerekir.

```
Const PI As Double = 3.14
```

Sabitlerin kapsam alanları değişkenler ile aynıdır.

Veri Tipleri



Veri tipi, değişkenlerin tuttukları değerlerin türünü ve bellekte tutulacak büyüklüğünü tanımlar. Değişkenleri veri tipleri ile tanımlarken verinin büyüklüğüne göre bir veri tipi seçilmelidir.

Visual Basic.NET veri tipleri Tablo 4.2'de listelenmiştir.

TABLO 4.2: Visual Basic.NET Veri Tipleri

Veri Tipi	Büyüklük	Değer
Boolean	2 Bayt	**True** – **False**
Byte	1 Bayt	0 – 255
Char	2 Bayt	Tek bir Unicode karakteri tutar
Date	8 Bayt	01.01.0001 tarihi 00:00:00 saati – 31.12.9999 tarihi 23:59:59 saati
Decimal	16 Bayt	Maksimum 29 haneli sayı tutar. +/-79,228,162,514,264,337,593,543,950,335 arasında değer alır
Double	8 Bayt	Negatif sayı aralığı: -1.79769E+308 ile -4.94065E-324 Pozitif sayı aralığı: 4.94065E-324 ile 1.79769E+308
Int32	4 Bayt	-2,147,483,648 – 2,147,483,647
Int16	2 Bayt	32,768 – 32,767
Int64	8 Bayt	-9,223,372,036,854,775,808 – 9,223,372,036,854,775,807.
Single	4 Bayt	Negatif sayı aralığı: -3.4028235E+38 ile -1.401298E-45 Pozitif sayı aralığı: 1.401298E-45 ile 3.4028235E+38
String		Maksimum 2,147,483,647 Unicode karakter tutar

Double ve **Single** veri tiplerinin aralığında belirtilen "E + sayı" ifadesi, 10 ^ sayı ile çarpılacağını belirtir. Örneğin 12 E-3 ifadesi, 12 * 0.001 anlamına gelir.

```
-1.7E-5 = -0.000017
-1.7E+10 = -17000000000.0
0.7432E+2 = 74.32
7432E-3 = 7.432
```

NOT `Int16`, `Int32`, `Int64` **.NET veri tipleridir. Visual Basic dilindeki karşılıkları** `Short`, `Integer`, `Long` **veri tipleridir.**

```
Dim yazi As String = "Veri tipleri örnekleri"

Dim bool As Boolean = True
Dim karakter As Char = "A"
Dim tarih As Date = #4/23/2005#

Dim numerik As Decimal = -123456789
Dim cift As Double = -1.234E-120
Dim tek As Single = 3.32E+100
Dim bayt As Byte = 255

Dim kisaSayi As Short = -32000
Dim tamSayi As Integer = 2000000000
Dim uzunSayi As Long = -123456789123456789
```

Uygulamalar çalışırken, çoğu zaman veri tipleri birbirlerine otomatik olarak dönüştürülür. Örneğin **InputBox** geriye **String** tipinde bir değer döndürür. Ancak kullanıcı mesaj kutusuna sayı girerse ve biz bu değeri **Integer** tipinde bir değişkene atarsak, veri dönüştürme işlemi yapılır. Buna Implicit Conversion (kapalı dönüştürme) denir.

```
Dim rakam As Integer
rakam = InputBox("Rakam giriniz")
```

Büyük veri tiplerinden küçük veri tiplerine dönüşüm sırasında, değer kayıpları meydana gelebilir. Örneğin **Single** tipinden **Short** tipine yapılacak bir dönüşümde virgülden sonraki sayılar kaybedilecektir.

```
Dim virgullu As Single = 1.12
Dim kisaSayi As Short = virgullu
' kisaSayi değişkenin son değeri 1 olur
```

NOT `Option Strict` **seçeneği** `On` **olarak ayarlanırsa, Implicit Conversion işlemine izin verilmez.**

Veri dönüştürme işlemlerinin kapalı olarak yapılması sisteme bırakılmıştır. Bu işlemde kodların tekrar okunması sırasında dönüştürme işlemleri gözden

kaçabilir ve değer kayıpları fark edilmez. Bu durumda dönüştürme işlemleri hazır fonksiyonlar ile açık olarak yapılmalıdır. Buna Explicit Conversion (açık dönüştürme) denir.

```
Dim rastgeleSayi As Double = Rnd() * 30
MsgBox(rastgeleSayi)
Dim rastgeleTamSayi As Integer = CInt(rastgeleSayi)
MsgBox(rastgeleTamSayi)
```

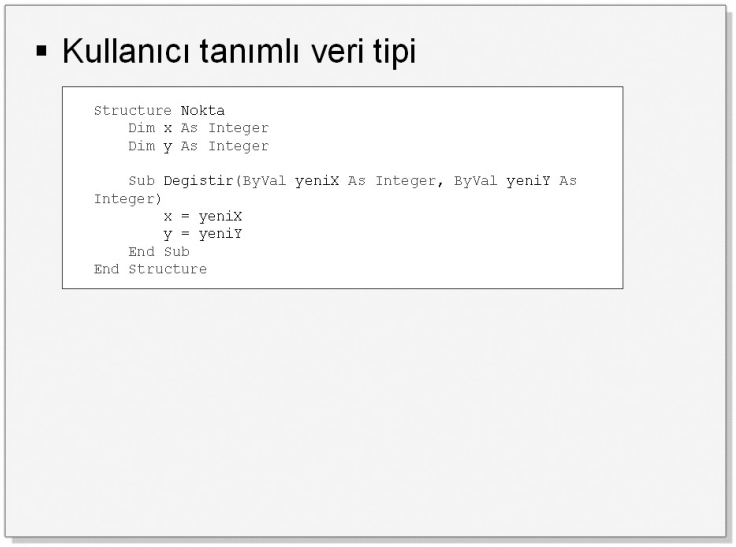
Explicit Conversion fonksiyonları:

- **CStr.** Verilen değeri **String** tipine dönüştürür.
- **CInt. Integer** veri tipinin alabileceği değerler arasında girilen sayıları **Integer** tipine dönüştürür.
- **CDbl. Double** veri tipinin alabileceği değerler arasında girilen sayıları **Double** tipine dönüştürür.
- **CDate.** Doğru tarih ve saat biçimde yazılmış herhangi bir ifadeyi **Date** tipine dönüştürür.
- **CLng. Long** veri tipinin alabileceği değerler arasında girilen sayıları **Long** tipine dönüştürür.
- **CSng. Single** veri tipinin alabileceği değerler arasında girilen sayıları **Single** tipine dönüştürür.
- **CDec. Decimal** veri tipinin alabileceği değerler arasında girilen sayıları **Decimal** tipine dönüştürür.

Sayı dönüştürme fonksiyonları, **True Boolean** tipini -1, **False Boolean** tipini 0 olarak dönüştürürler. Ayrıca tüm fonksiyonlar, düzgün biçimde verilmiş **String** tipinde değişkenleri de ilgili veri tipine değiştirirler.

```
Dim sayı As Integer = CInt("12345abc")
' InvalidCastException hatası fırlatılır.
```

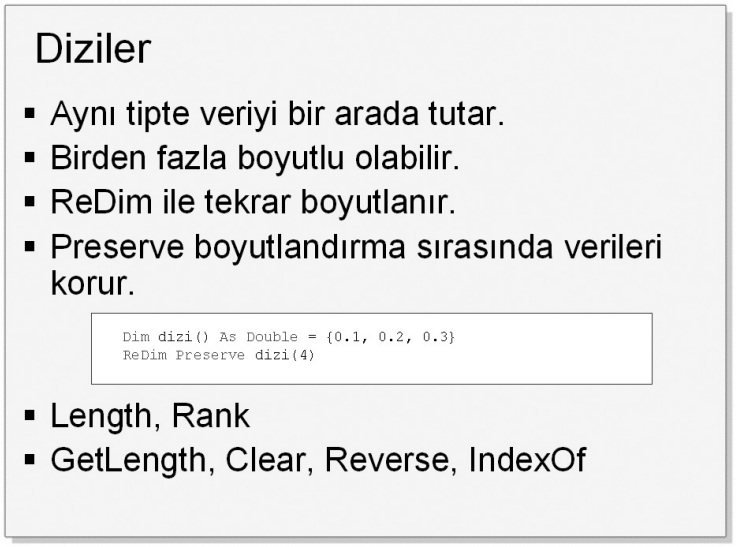
Structure



Structure veri tipleri, programcıların kendilerinin tanımladığı veri tipleridir. **Structure**, birkaç veri tipinin bir araya getirilmesiyle oluşturulan bileşik bir tiptir. **Structure** veri tiplerinde yordam tanımları da yapılabilir.

```
Structure Nokta
    Dim x As Integer
    Dim y As Integer

    Sub Degistir(ByVal yeniX As Integer, ByVal yeniY As
Integer)
        x = yeniX
        y = yeniY
    End Sub
End Structure

Structure Ucgen
    Dim n1 As Nokta
    Dim n2 As Nokta
    Dim n3 As Nokta
End Structure
```

Dizilerle Çalışmak



Dizi değişkenleri, aynı tipte birçok veriyi bir arada tutmayı sağlar. Benzer işlemlerde kullanılan değişkenler bir dizi altında listelenebilir. Örneğin kullanıcıdan alınan isimler **`String`** tipinde bir dizi içinde toplanabilir.

```
Dim isimler(3) As String
```

Diziler tanımlanırken, ismi verildikten sonra parantez içinde kaç eleman içereceğini belirtmek gerekir. Dizilerin indisleri sıfırdan başlar. Örnekteki isimler dizisinin **`String`** tipinde 4 tane elemanı vardır.

Dizilerin elemanlarına ulaşmak için, istenen elemanın indisinin verilmesi gerekir.

```
isimler(0) = "Ali"
isimler(1) = "Ahmet"
isimler(2) = "Mehmet"
isimler(3) = "Ayşe"

MsgBox(isimler(3))
```

Dizilere tek tek değer atanabildiği gibi, tanımlarken de başlangıç değerleri atanabilir.

```
Dim isimler() As String = {"Ali","Ahmet","Mehmet","Ayşe"}
```

Diziler tek boyutlu olduğu gibi, birkaç boyutlu diziler de tanımlanabilir.

```
' İlk boyutunda 5, İkinci boyutunda 6 Integer değeri olan
' 2 boyutlu dizi
Dim matris(4,5) As Integer
```

Burada, dizinin ilk boyutunda 5 tane eleman vardır. İlk boyuttaki her eleman için İkinci boyutta 6 eleman bulunur. Dolayısıyla dizinin toplam 30 elemanı vardır. Bu dizide bir boyut daha olsaydı, o boyutun her elemanı için diğer boyutlardaki 30 eleman bulunacaktı.

Çok boyutlu dizilerin eleman sayıları boyutlarındaki eleman sayıları çarpılarak hesaplanabilir.

```
Dim dizi(boyut1,boyut2,boyut3,... ,boyutn) As VeriTipi
' Eleman sayısı:
' (boyut1 + 1) * (boyut2 + 1) * ... * (boyutN + 1)
```

Çok boyutlu dizilere başlangıç değerleri, dizinin boyutu dikkate alınarak verilmelidir. Boyutlardaki elemanlar küme parantezleri ile gruplanmalıdır.

```
' İlk boyutunda 2, ikinci boyutunda 4 eleman olan
' 2 boyutlu dizi
Dim matris(,) As Integer = {{1, 2, 3, 4}, {5, 6, 7, 8}}
```

Çok boyutlu dizilerin elemanlarına ulaşmak için, her boyut için indis göstermek gerekir.

```
matris(0, 0) = 1
```

Dizileri tanımladıktan sonra, eğer boyutun büyüklüğü (eleman sayısı) yetmiyorsa tekrar boyutlandırmaya ihtiyaç duyarız. Aynı şekilde boyutun büyüklüğünü azaltmak için de tekrar boyutlandırma kullanılır.

```
Dim dizi() As Double = {0.1, 0.2, 0.3}
ReDim dizi(4)
```

ReDim, yeniden boyutlandır anlamına gelir. Burada dizininin boyutu beş eleman alacak şekilde ayarlanır. Ancak **ReDim**, dizileri boyutlandırırken değerleri korumaz. **Preserve** anahtar kelimesi kullanılmadan tekrar boyutlandırılan diziler, içerdiği verileri kaybeder.

```
' (3,0) boyutlu olan bir dizi,
' değerlerini koruyarak (3,1) boyutlu yapılır
Dim dizi(,) As Double = {{1.0},{2.0},{3.0},{4.0}}
ReDim Preserve dizi(3, 1)
```

Bazı Dizi Özellikleri ve Metotları

Diziler, .NET Framework içinde tanımlı **Array** sınıfı ile temsil edilir. Tüm diziler **Array** sınıfında tanımlı özellikleri ve metotları kullanırlar.

- **Length.** Dizinin bütün boyutlarındaki toplam eleman sayısını veren özelliktir.

```
Dim ComboBoxDizisi(19) As ComboBox
MsgBox(ComboBoxDizisi.Length)
'Sonuç = 20

Dim dizi(1, 4, 4, 5, 6) As Integer
MsgBox(dizi.Length())
'Sonuç = 2 * 5 * 5 * 6 * 7 = 2100
```

- **Rank.** Dizinin boyut sayısını veren özelliktir.

```
MsgBox(dizi.Rank)
'Sonuç = 5
```

- **GetLength.** İndisi verilen boyutun kaç elemanlı olduğunu gösterir. Burada indisin sıfırdan başladığına dikkat edilmelidir.

```
Dim dizi(10, 40, 50, 80, 90) As Integer
MsgBox(dizi2.GetLength(4))
'Sonuç = 91
```

Clear, **Reverse** ve **IndexOf** metotları **Array** sınıfında **Shared** (paylaştırılmış) olarak tanımlı metotlardır. İşlemin yapılacağı dizi parametre olarak verilmelidir.

- **Clear.** Parametre olarak verilen dizinin, belirtilen indis aralığındaki tüm değerlerini temizler. Temizleme işleminde atanan değer, dizi elemanlarının tiplerine göre değişir. Örneğin **Integer** tipinde tanımlı bir dizinin elemanları temizlenirse **0** değerini alacaktır. Buna karşın **String** tipindeki elemanlar "" (boş yazı) değerini alır.

```
Dim dizi() As Integer = {12, 13, 14, 15}
' 1. indisten başlayarak, 3 elemanı temizle
Array.Clear(dizi, 1, 3)
MsgBox(dizi(2))
'Sonuç = 0

' Dizinin tüm elemanlarını temizler
Array.Clear(dizi, 0, dizi.Length)
```

- **Reverse.** Parametre olarak verilen dizinin eleman sırasını tersine çevirir. Dizinin tüm elemanlarının veya belirli indis aralığındaki elemanlarının sırası tersine çevrilebilir.

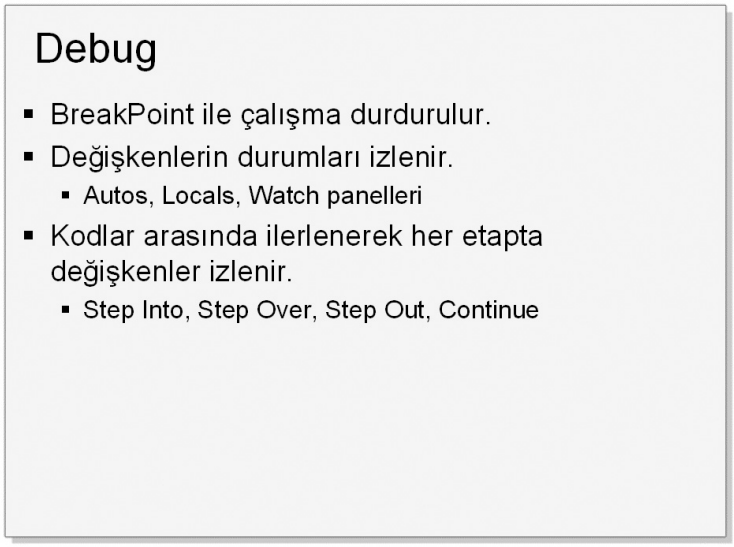
```
Dim harfler() As String = {"A", "B", "C"}
Array.Reverse(harfler)
```

```
MsgBox(harfler(2))
'Sonuç = A

Dim harfler() As String = {"A", "B", "C"}
Array.Reverse(harfler, 0, 1)
MsgBox(harfler(2))
'Sonuç = C
```

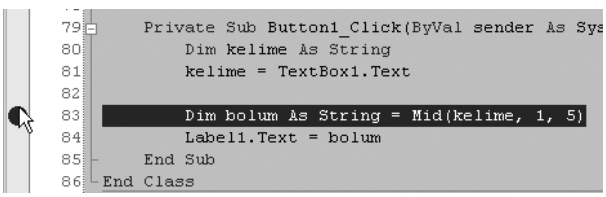
- **`IndexOf`.** İlk parametrede verilen dizide, ikinci parametrede verilen değeri arar. Aranan değer dizide bulunursa indisi, bulunamazsa -1 döndürür.

```
Dim notlar() As Single = {78.1, 99.9, 100, 12.2}
Dim maxNot As Single = 100
MsgBox(Array.IndexOf(notlar, maxNot))
' Aranan maxNot değerinin indisi = 2
```

Debug

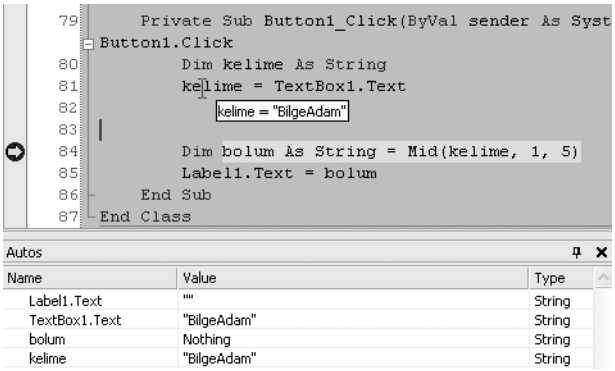


Visual Studio Debug aracı, çalışma anında kodlar arasında satır satır ilerleyerek hataları bulmayı sağlar. İncelemeye başlamak istenen kod satırı üzerinde bir **`BreakPoint`** (durma noktası) konarak, hata ayıklayıcının bu satır çalıştırılmadan önce orada durması sağlanır (Şekil 4.18).



RESİM 4.18: Durma noktası.

Uygulama çalıştırıldığında, **`BreakPoint`** konulan kod satırına kadar durmaz. Belirtilen satıra sıra gelindiğinde, kod sayfasında, o an üzerinde bulunan satır ok ile gösterilir. Visual Studio ile hata ayıklarken, tanımlanan değişkenlerin o andaki değerleri incelenerek mantıksal hatalar bulunabilir.



RESİM 4.19: Autos paneli.

Hata ayıklama sırasında bazı Visual Studio panelleri, değişkenlerin, kontrollerin ve nesnelerin değerlerini listelemek için kullanılabilir. Bu paneller Debug menüsünde Windows alt menüsünden gösterilir.

- **Autos Paneli.** Çalışmakta olan satırdaki ifade ile bir önceki ifadede bulunan değişken ve kontrollerin değerlerini listeler (Resim 4.19).
- **Locals Paneli.** İçinde bulunulan kapsam alanındaki tüm değişkenlerin değerlerini listeler.
- **Watch Paneli.** Değeri incelenmek istenen değişken veya kontroller bu panele elle yazılmalıdır.

Kodlar arasında ilerlemek ve hata ayıklamaya devam etmek için dört yol vardır. Bu komutlara Debug menüsünden veya Debug araç çubuğundan ulaşılabilir.

- **Step Into.** Kod satırında bir yordam çalıştırılacaksa, bu yordamın içine girer. Bu yordam farklı bir yerde ise, ilgili sayfa açılır ve hata ayıklamaya devam edilir.
- **Step Over.** Herhangi bir yordam içine girmeden, içindeki kapsam alanında çalışmaya devam eder.
- **Step Out.** Bulunan yordamdan çıkarak hata ayıklamaya devam eder.
- **Continue.** Birden fazla durma noktası yerleştirilmişse, bir sonraki noktaya kadar çalışmaya devam eder.

Hata ayıklama, çalıştırılacak hiçbir satır kalmadığında durur ve uygulama normal çalışmasına devam eder. Durma noktaları kaldırılarak ya da pasif hale getirilerek uygulamanın durması engellenebilir.

Bütün durma noktalarını kaldırmak için Debug menüsünden Clear All BreakPoints komutu, pasif hale getirmek için Disable All BreakPoints komutu verilmelidir. Durma noktalarını aktif hale getirmek için tekrar aynı komut seçilmelidir.

Alıştırma

Bu uygulamada veri tiplerinin kullanım yerlerine ve diziler ile çalışma örneklerine bakılacaktır.

Structure Veri Tipi

1. Sınıf isminde bir Windows projesi açın.
2. Açılan form üzerine sağ tıklayarak View Code komutunu seçin. Kod sayfasında sınıf düzeyinde bir **`Structure`** tanımlayın.

```
Structure Ogrenci
    Dim Isim As String
    Dim Soyad As String
    Dim Sube As Char
    Dim OrtalamaNotu As Single
    Dim DevamEdiyor As Boolean
End Structure
```

3. **`Ogrenci`** tipindeki değerleri tutmak için, sınıf düzeyinde iki elemanlı bir dizi tanımlayın.

```
Dim ogrenciler(1) As Ogrenci
```

4. Formun **`Load`** olayına, uygulama açılırken yeni öğrenci ekleme kodlarını yazın.

```
Dim ogrenci1 As Ogrenci
ogrenci1.Isim = "Ali"
ogrenci1.Soyad = "Veli"
ogrenci1.Sube = "C"
ogrenci1.OrtalamaNotu = 67.1
ogrenci1.DevamEdiyor = True

Dim ogrenci2 As Ogrenci
With ogrenci2
    .Isim = "Ahmet"
    .Soyad = "Veli"
    .Sube = "C"
    .OrtalamaNotu = 72.9
    .DevamEdiyor = True
End With

ogrenciler(0) = ogrenci1
ogrenciler(1) = ogrenci2
```

İPUCU `With` **anahtar kelimesi, belirtilen değişkeni tekrar yazmadan kullanmayı sağlar.**

5. Forma **btnOgrenciEkle** isminde bir **Button** kontrolü yerleştirin. Bu kontrolün **Click** olayına, diziye yeni bir öğrenci kaydı ekleyen kodu ekleyin.

```
' ogrenciler dizinde boş yer kalmadığı için
' diziyi, eski değerleri kaybetmeden tekrar
' boyutlandırmak gerekir.
ReDim Preserve ogrenciler(2)

Dim ogrenci As Ogrenci
With ogrenci
    .Isim = "Veli"
    .Soyad = "Veli"
    .Sube = "D"
    .OrtalamaNotu = 92.1
    .DevamEdiyor = False
End With

ogrenciler(2) = ogrenci
```

ogrenciler dizisine başka bir yordamdan nasıl erişildi?

Formun **Load** olayında **ogrenci** isimli bir değişken tanımlandığı halde, **Button** kontrolünün **Click** olayında aynı isimde bir değişken nasıl tanımlanabiliyor?

Dizi işlemleri

1. Forma **bntOzellikleriGoruntule** isminde bir **Button** kontrolü ekleyin ve **Click** olayında, diziden indisi verilen öğrenciyi alan kodları yazın.

```
Dim indis As Integer = InputBox("Ogrenci sırası numarasını
girin")
Dim secilenOgrenci As Ogrenci
secilenOgrenci = ogrenciler(indis)

Dim bilgiler As String

With secilenOgrenci
    bilgiler &= .Isim & " " & .Soyad & vbCrLf
    bilgiler &= "Notu: " & .OrtalamaNotu & vbCrLf
    bilgiler &= "Şubesi: " & .Sube & vbCrLf
    bilgiler &= "Devam ediyor mu: " & .DevamEdiyor
End With
MsgBox(bilgiler, MsgBoxStyle.Information, "Öğrenci
Bilgileri")
```

İPUCU `VbCrLf` **(Visual Basic Carriage Return-Linefeed) sabiti,** `String` **değişkenlerinde yeni satıra geçilmesini sağlar.**

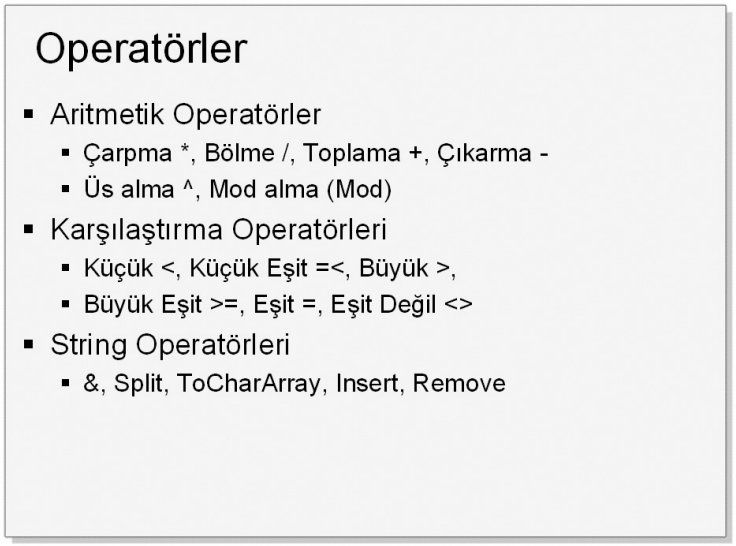
Aritmetik işlemler

1. Forma **`btnOrtalamaHesapla`** isminde bir **`Button`** kontrolü ekleyin ve **`Click`** olayında sınıfın ortalamasını hesaplayan kodu yazın.

```
Dim not1 As Double = ogrenciler(0).OrtalamaNotu
Dim not2 As Double = ogrenciler(1).OrtalamaNotu
Dim not3 As Double = ogrenciler(2).OrtalamaNotu

Dim ortalama As Integer = CInt((not1 + not2 + not3) / 3)
MsgBox(ortalama)
```

2. **`not3`** değişkeninin tanımlandığı yere **`BreakPoint`** koyun ve projeyi çalıştırın.
3. Form açıldığında **`btnOrtalamaHesapla`** düğmesini tıklayın. Uygulamanın çalışması durma noktası konulan yerde duracaktır.
4. Debug | Windows | Autos komutunu seçin. Autos panelinde **`not1`** ve **`not2`** değişkenlerinin değerlerini inceleyin.
5. Debug | Windows | Watch komutunu seçin. Watch panelinde Name sütununa "**`ogrenciler`**" yazın. **+** simgesini tıklayarak **`ogrenciler`** dizisini genişletin ve dizi elemanlarının değerlerini inceleyin.
6. Debug menüsünden Step Into komutunu seçin. Bu işlemi Debug araç çubuğu ile ya da **F11** tuşuna basarak da yapabilirsiniz.
7. Gösterilen hata mesajını inceleyin. Continue düğmesini tıklayarak uygulamayı sonlandırın.
8. Uygulamayı tekrar çalıştırın ve önce **`btnOgrenciEkle`** düğmesini daha sonra da **`btnOrtalamaHesapla`** düğmesini tıklayın.

Konu 8: Operatörler



Visual Basic .NET dilinde çalışırken, değişkenler üzerinde birçok işlem yapılır. Hesaplamalarda aritmetik işlemler, kontrollerde karşılaştırma işlemleri veya mantıksal işlemler yapılır. Bu işlemler için Visual Basic .NET dilinde tanımlı operatörler kullanılır.

Aritmetiksel Operatörler

Bu operatörler aritmetik işlemlerinde, sayılarla veya sayı tutan ifadelerle kullanılır.

- Çarpma

```
Dim sayi As Integer = 100
sayi = 200 * 2
```

- Bölme

```
Dim bolum As Double
bolum = sayi / 23
```

- Çıkarma

```
Dim sonuc As Integer = bolum – 100
```

- Toplama

```
Dim toplam As Integer
toplam += sonuc
' Bu ifade, "toplam = toplam + sonuc" ile aynı anlama gelir
```

İPUCU **Aritmetik operatörleri, eşittir ifadesi ile beraber kullanılırsa, işlem değişkenin kendisi ile yapılır.**

- Üs alma

```
toplam ^= 2S
```

- Mod alma

```
Dim kalan As Integer = toplam Mod 42
' Sonuç, toplam değişkenindeki değerin 42 ile
' bölümünden kalan sayıdır.
```

Karşılaştırma Operatörleri

Bu operatörler veri tiplerini birbirleriyle karşılaştırmak için kullanılır. Bu operatörler ile yapılan işlemlerin sonucunda **`True`** ya da **`False`** değeri döner. Karşılaştırma operatörleri yalnızca sayı tipleri üzerinde yapılmaz. **`String`** tipleri birbirleriyle alfabetik sıraya göre karşılaştırılabilir.

- Küçük

```
"A ile başlayan yazı" < "B ile başlayan yazı"
' Sonuç: True
```

- Küçük Eşit

```
Dim sayi As Double = 1.5
Dim sayi2 As Single = 1.3

sayi2 <= sayi
' Sonuç: True
```

- Büyük

```
sayi2 > sayi1
' Sonuç: False
```

- Büyük Eşit

```
sayi2 >= sayi1
' Sonuç: False
```

- Eşit

```
"Yazı" = "yazı"
' Sonuç: False
```

- Eşit Değil

```
"Yazı" <> "yazı"
' Sonuç: True
```

String Operatörleri

String tipleri üzerinde gerçekleştirilen işlemler için tanımlı operatörlerdir.

- **String** tipindeki değişkenleri birbirine bağlama işlemi **&** operatörü ile gerçekleşir.

```
Dim isim As String
Dim soyad As String

Dim IsimSoyad As String = isim & " " & soyad
```

- **Split.** Belirtilen ayraca göre yazıyı böler, çıkan sonuç **String** dizisinde tutulur. Ayraç karakterleri sonuç dizisinde yer almaz.

```
Dim Kelime As String = "Kelime1:Kelime2:Kelime3"
Dim parcalar() As String
parcalar = Kelime.Split(":")
' parcalar dizisinin üç elemanı olur:
' Kelime1
' Kelime2
' Kelime3

Dim parcalar2() As String
parcalar2 = Kelime.Split("m")
' parcalar2 dizisinin dört elemanı olur:
' Keli
' e1:Keli
' e2:Keli
' e3
```

- **ToCharArray**. **String** değerinin belli bir bölümündeki karakterleri ya da tüm karakterlerini, **Char** dizisi olarak döndürür.

```
Dim harfler() As Char = "Kelime".ToCharArray()

' Dizinin 1. elemanından başlayarak 4 karakter oku
Dim harfler() As Char = "Kelime".ToCharArray(1,4)
```

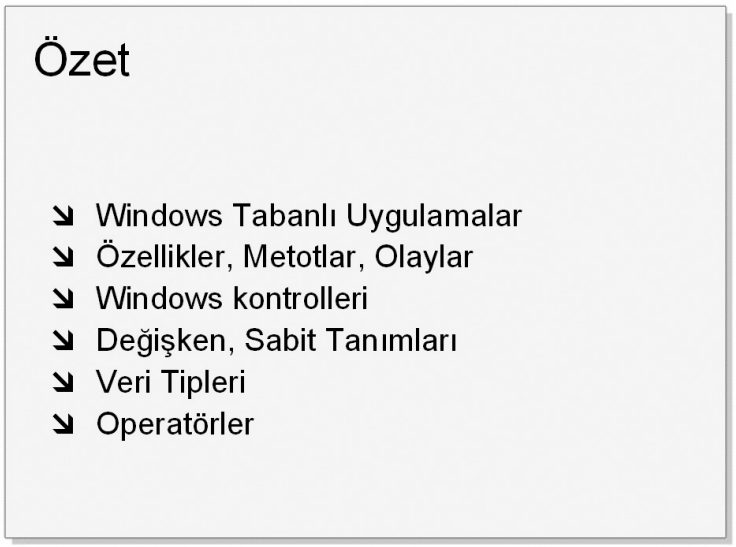
- **Insert**. **String** tipinde bir değişkenin değerine, ilk parametrede belirtilen yerden başlayarak ikinci parametredeki değeri ekler. Ancak bu değişkenin değerini değiştirmez. Yeni oluşturulan **String** ifadesini döndürür.

```
Dim sayilar As String = "0123456789"
Dim yeniSayilar As String
yeniSayilar = sayilar.Insert(5, "--- Rakamlar ---")
MsgBox(yeniSayilar)
' Sonuç: 01234--- Rakamlar ---56789
```

- **`Remove`**. İlk parametrede verilen değerden başlayarak, ikinci parametredeki değer kadar karakter, değişkenden çıkarılır.

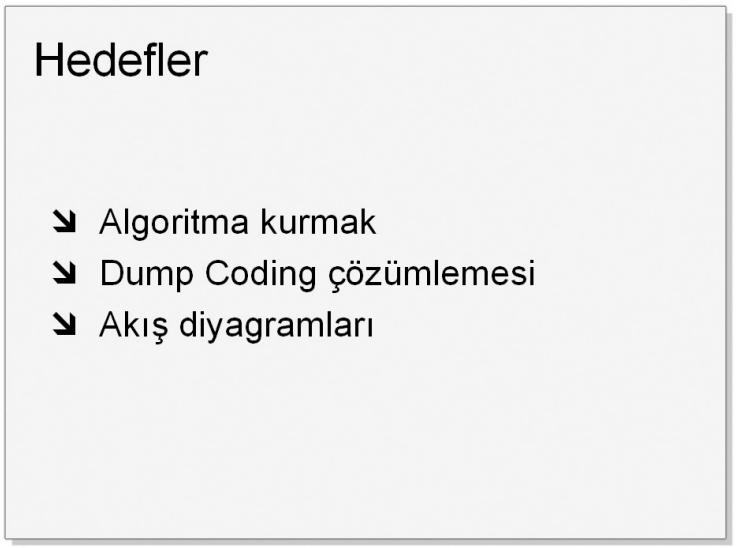
```
yeniSayilar = yeniSayilar.Remove(4, yeniSayilar.Length - 4)
MsgBox(yeniSayilar)
' Sonuç: 0123
```

Modül Sonu Soruları & Alıştırmalar



1. Arabanın fren yapması ve arabaya çarpılması, .NET nesnelerinin hangi kavramlarına girer?
2. Bir **`ListBox`** kontrolüne 10 saniyede bir kullanıcıdan alınan değerleri ekleyen kodları yazın.
3. Değişkenler ile sabitlerin farkı nedir?
4. 5 < 6 = -1 ifadesi hangi **`Boolean`** değerini döndürür, neden? **`Option Strict On`** seçildiğinde çıkan hata mesajını inceleyin.

Modül 5: Algoritma ve Dump Coding



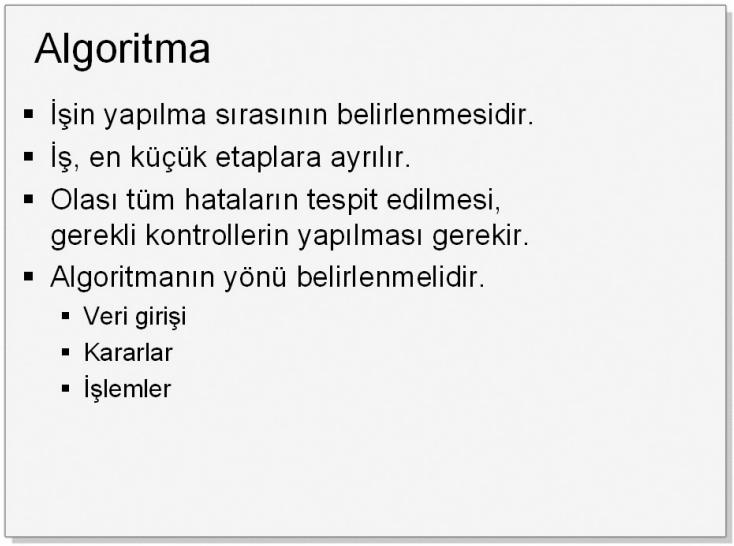
Programlamanın temelinde, çalışma akışını ve izlenecek yolları belirleyen algoritmalar vardır. Bir iş yapılmaya başlanmadan önce nasıl planlanıyorsa, kodlamaya geçilmeden önce de bir çalışma planı belirlenmelidir. Programlar, bu planda yazılan kodları belli bir sıra ile okur ve işler. Dolayısıyla algoritma yapısını çok iyi kurmak gerekir. Kurulan algoritmalar akış diyagramları ile görsel zenginlik kazanırlar.

Dump Coding yöntemi algoritmaları çözmenin uzun, fakat etkili bir yoludur. Bu yöntem, adımları tek tek inceleyerek algoritma akışını çözer.

Bu modül tamamlandıktan sonra;

- Algoritma kurmayı öğrenecek,
- Dump Coding ile algoritmaları çözümleyecek,
- Akış diyagramları ile algoritmaları görsel olarak ifade edebileceksiniz.

Konu 1: Algoritma Nedir?



Algoritma, bir işin hangi etaplardan geçilerek yapılacağını gösteren çalışma planıdır. Algoritma bir programlama dili değildir. Programlama dillerine yol gösteren bir yöntem dizisidir. Her dilde algoritma yazılıp uygulanabilir. Örneğin bir cep telefonunun el kitapçığında yazan, rehber kaydı girmek için izlenecek yollar, o işin algoritmasıdır.

Algoritma yazarken, programın çalışması için kullanılan kaynakların, yapılması gereken kontrollerin veya işlemlerin açıkça ifade edilmesi gerekir. Ayrıca iyi bir algoritmanın, tüm ihtimalleri kontrol edip istenmeyen durumlarda ne yapılması gerektiğini de belirtmesi gerekir.

Örneğin, bir e-ticaret uygulamasında ürün satış algoritması çıkarılır. Satın alınacak ürün seçildikten sonra, kullanıcıdan adet miktarı bilgisi alınır. Uygulama yazılırken, bu değerin `Int16` veri tipinde olacağına karar verildiği düşünülürse; kullanıcının girdiği adet miktarı bu değişkene atanmadan önce kontrol edilmelidir. Eğer `Int16` veri tipinin tutamayacağı bir değer girilmişse, çalışma anında uygulamanın beklenmedik şekilde durduğu ya da istenmeyen sonuçların üretildiği gözlemlenir. Ayrıca sistemin verdiği hata, kullanıcının anlamayacağı bir mesaj içereceği için, uygulamanın imajını da kötü yönde etkiler.

- **Veri girişi.** Çalışma zamanında çoğu kez, işleyişin tamamlanması için dışarıdan bir bilgi girilmesi gerekir. Algoritmanın çalışması için ihtiyaç duyduğu veriler, işlemi başlatan kişiden veya belirtilen bir kaynaktan alınabilir. Bu bilgiler sağlanmadan işlem devam etmez.
- **Kararlar.** Karar ve kontrol yapıları algoritmanın akışını yönlendiren en önemli kavramlardır. Girilen veya işlem sonucunda elde edilen veriler,

işlemin amacına göre kontrol edilir ve sonuca göre algoritma akışı istenen yere yönlendirilir.

- **İşlemler.** Algoritmanın akışı boyunca veriler üzerinde değişiklikler, yeni değer atamaları gibi işlemlere ihtiyaç duyulur. Algoritmalar kurulurken, yapılan işlemlerin yalın halde, tek tek yazılması okunabilirliği artırır.

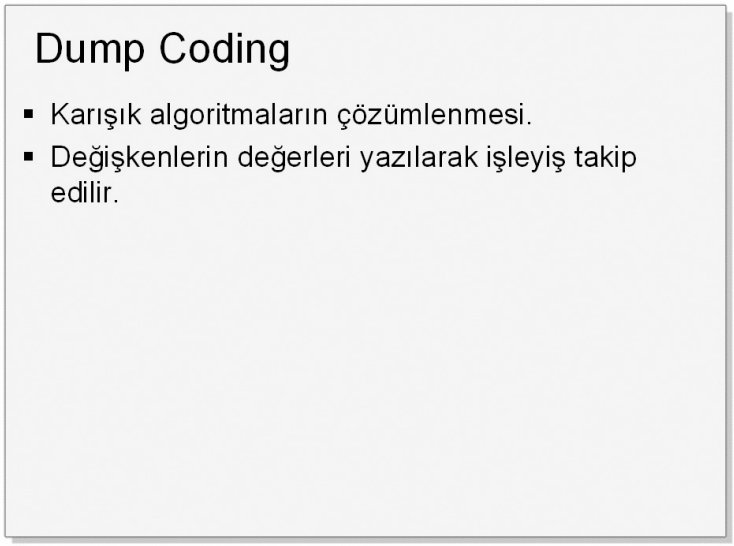
Algoritmalar adım sırası ile çalışır ve karar yapıları sonucunda farklı bir yere yönlendirilmediği müddetçe, bir sonraki adım ile işlemeye devam eder.

Örnek: Telefon kulübesinden telefon açmak için örnek bir algoritma

1. Telefon kulübesine git.
2. Telefon kartı al.
3. Telefon sırasında kaç kişi olduğuna bak.
4. Kişi sayısı sıfırdan fazlaysa 3'e dön.
5. Kapı kapalıysa kapıyı aç.
6. İçeri gir, kapıyı kapat.
7. Telefon kartını telefona yerleştir.
8. Ahizeyi kaldır.
9. Numarayı çevir.
10. Konuşmanın bitip bitmediğine bak.
11. Konuşma bittiyse kartı al, bitmediyse 10'a dön.
12. Bir daha konuşma yapılacaksa 7'ye dön.
13. Kapıyı aç, dışarı çık.

Bu algoritmanın işlemesi için, her ihtimal gözden geçirilerek, algoritma akışı gerekli yerlere yönlendirilir. Örneğin, kapının kapalı olması durumunda kapıyı açmak için gerekli komutlar verilmelidir. Bu algoritmanın ihtiyaç duyduğu veriler ya kullanıcı tarafından verilir ya da işlem başlamadan önce belirlidir. Sıradaki kişi sayısı, telefon kartı gibi veriler kullanıcı tarafından sağlanmış; çevrilecek numara, algoritma başlamadan önce belirlenmiştir.

Konu 2: Dump Coding Nedir?



Dump Coding yöntemi birçok karışık algoritmayı çözümlememizi sağlar.

Dump Coding yöntemi, algoritmanın her adımında, değişkenlerin tek tek değerlerini yazıp işleyişi takip etmektir.

Örnek: İki sayının OBEB'ini (ortak bölenlerin en büyüğünü) alan algoritmalardan bir tanesi Euclid tarafından geliştirilmiştir.

1. İki sayı gir. Büyük A, küçük B.
2. A sayısını B sayısına böl. Tam bölünüyorsa, OBEB B sayısıdır. Çıkış.
3. A sayısının değerini, kalan sayının değeri yap.
4. A ile B sayılarının yerini değiştir. İkinci etaba dön.

Bu algoritmanın çalışma mantığı, Dump Coding yöntemi ile adım adım incelenir.

1. İki sayı girilir. A = 12 ve B = 8.
2. A sayısı, B sayısına tam bölünmüyor. Algoritma diğer etaptan devam eder.
3. Kalan sayı = 4. Dolayısıyla A = 4 olur.
4. A sayısı ile B sayısının yerleri değiştirilir. A = 8 ve B = 4 olur. İkinci etaba dönülür.
5. A sayısı B sayısına tam bölünüyor. OBEB = 4.

Konu 3: Akış Diyagramlarında Kullanılan Semboller

Madde madde yazılan algoritmaların okunması kolaydır, ancak işleyişin bütününü görmek, çoğu zaman mümkün değildir. Akış diyagramları, algoritmaları görsel biçimde göstermeyi, dolayısıyla daha anlaşılır hale getirmeyi sağlar. Algoritmada yapılacak işlemlerin çeşitlerine göre çeşitli semboller kullanılır.

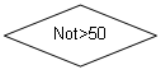
- **Başla – Bitir.** Algoritmanın hangi aşamadan başlayacağını ve ne zaman biteceğini gösteren semboldür. Bir algoritmayı temsil eden akış diyagramında, bir tane Başla ve bir tane Bitir sembolü olmalıdır (Resim 5.1).

RESİM 5.1: Başla-Bitir sembolleri.

- **Veri Girişi.** Kullanıcıdan ve başka bir kaynaktan alınan verilerin isimlerini tutar (Resim 5.2).

RESİM 5.2: Veri girişi sembolü.

- **Karar Verme.** Karar yapısını belirten semboldür. Üstünde koşul ifadesi belirtilir (Resim 5.3).

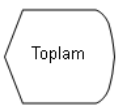


RESİM 5.3: Karar verme sembolü.

- **Veritabanı.** Veritabanında okuma veya yazma işlemi yapıldığını gösterir (Resim 5.4).

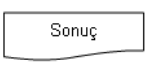
RESİM 5.4: Veritabanı sembolü.

- **Ekran.** Üzerinde yazılan yazının bilgi olarak ekranda gözükeceğini belirtir (Resim 5.5).



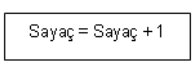
RESİM 5.5: Ekran sembolü.

- **Printer.** Üzerinde yazılan yazının yazıcıdan çıkarılacağını belirtir (Resim 5.6).



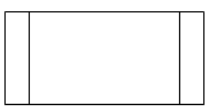
RESİM 5.6: Printer sembolü.

- **İşlem.** Bir işlem yapılacağını belirten semboldür. Her işlem için ayrı bir fonksiyon sembolü kullanılması, akış diyagramını daha anlaşılır kılar (Resim 5.7).



RESİM 5.7: İşlem sembolü.

- **Fonksiyon.** İşlem sembolüne yazılamayacak büyüklükte işlemler, alt işlem olarak bu sembolle belirtilir (Resim 5.8).

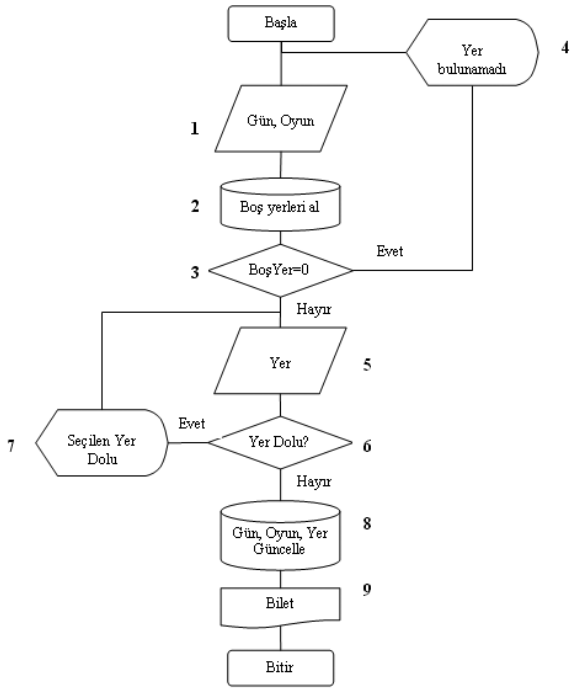


RESİM 5.8: Fonksiyon sembolü.

Konu 4: Algoritma Uygulamaları

Bilet Satma

Bir tiyatro uygulamasının sürekli gerçekleştireceği temel işlem, bilet satmaktır. Bu işlemi gerçekleştirmek için gerekli kodlar yazılmadan önce, algoritma kurulmalıdır (Resim 5.9).



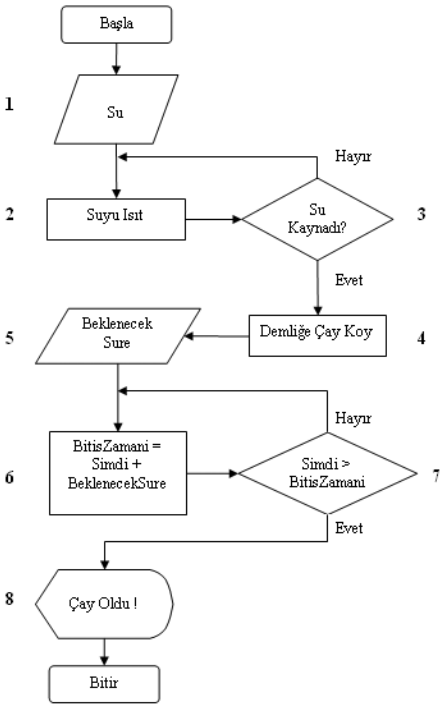
RESİM 5.9: Bilet satma algoritması.

1. Kullanıcının istediği oyun, gün ve yer bilgileri alınır.
2. Veritabanı sorgulanarak, belirtilen günde oynayan oyunun boş yerleri çıkartılır.
3. Boş yer sayısı sıfırsa, o günde belirtilen oyun oynanmıyordur ya da oyundaki bütün yerler satılmıştır.
4. Her iki durumda da bilet kesilemediği için ekranda hata mesajı gösterilir. Gün ve oyun bilgilerini baştan almak için ilk etaba dönülür.
5. Kullanıcıdan oturmak istediği yer bilgisi alınır.
6. İstediği yerin dolu olup olmadığı kontrol edilir.
7. Yer dolu ise ekrana hata mesajı gösterilir ve yer bilgisi tekrar alınmak üzere 5. etaba dönülür.
8. Yer boşsa, veritabanında oyunun yer kayıtları güncellenir.
9. İstenilen gün, oyun ve yer bilgilerini içeren bilet yazıcıdan çıkartılır.

Çay Demleme

Bu örnekte, bir çay demleme işleminde yapılması gereken işlemleri, kontrol edilmesi gereken olayları içeren algoritma kurulur (Resim 5.10).

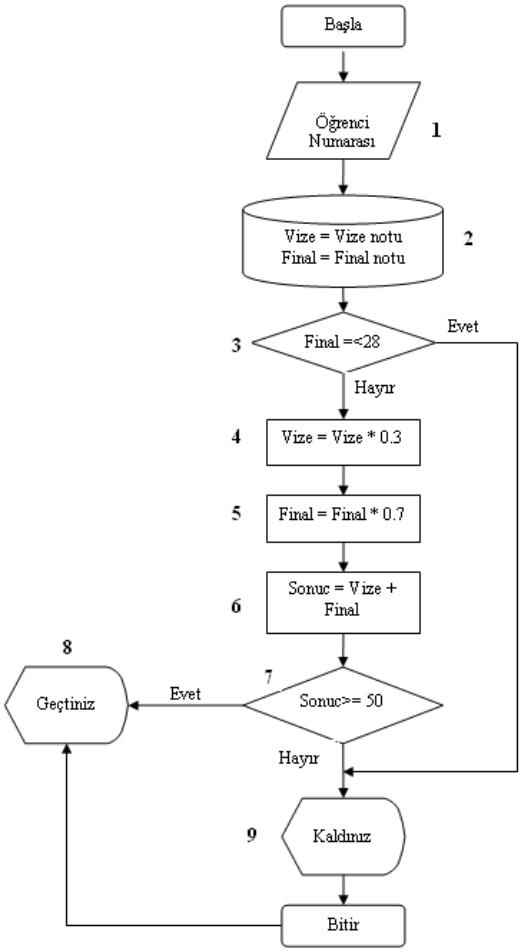
1. Kullanıcıdan su vermesi beklenir.
2. Suyu ısıtma işlemi yapılır.
3. Suyun kaynayıp kaynamadığı kontrol edilir. Kaynamamışsa 2. etaba dönülür.
4. Çay daha önceden hazır olduğu için, kullanıcıdan beklenmez. Demliğe çay koyma işlemi yapılır.
5. Kullanıcıdan, demleme işleminin ne kadar süreceği bilgisi alınır.
6. Kullanıcıdan alınan demleme süresi ile şimdiki zaman (çayın demlenmeye başladığı zaman) toplanır. Çıkan değer, **`BitisZamani`** isimli değişkene atılır. Bu değişken demleme işleminin ne zaman biteceği bilgisini tutar.
7. Şimdiki zaman, bitiş zamanından küçükse çayın demlenmesi için ayrılan süre daha dolmamış demektir. Bu süre dolana kadar 7. etap tekrarlanır.
8. Çayın demlendiğini, kullanıcıya ekran üzerinde bildiren bir mesaj çıkartılır.



RESİM 5.10: Çay demleme algoritması.

Üniversite Eğitim Notunu Hesaplama

Üniversitede bir dersin başarı notu, genelde bir vize ve bir final notu ile hesaplanır. Vize notunun katsayısı finalden daha düşüktür. Sonuçta çıkan not 50 ve üstündeyse öğrenci geçer, 50'nin altındaysa kalır. Bu örnek, vizenin %30 ve finalin %70 ağırlıklı olduğu başarı notunun hesaplanmasını akış diyagramı ile gösterir (Resim 5.11).

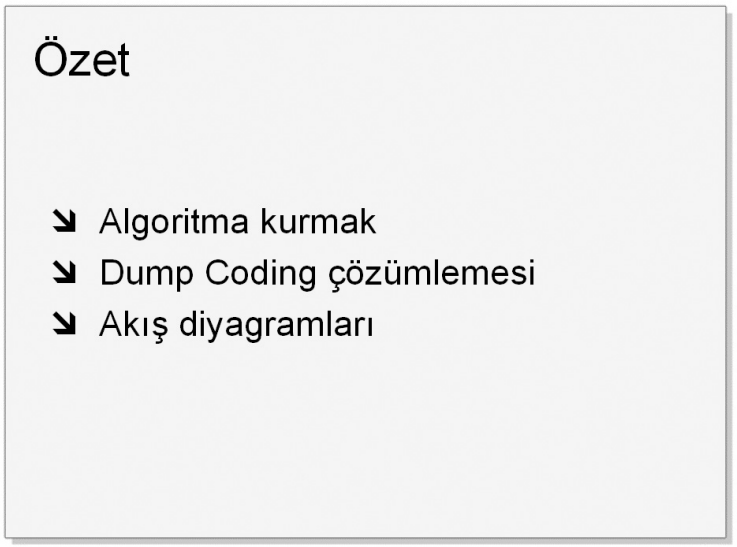


RESİM 5.11: Üniversite notu algoritması.

1. Notu hesaplanacak öğrencinin numarası kullanıcıdan alınır.
2. Veritabanından öğrencinin vize ve final notları çekilir.
3. Eğer final notu 28 veya daha düşükse öğrenci kalır ve 9. etaba gidilir. Bu durumda vize notu 100 olsa dahi, sonuç olarak toplanan not 50'nin altında olur. Dolayısıyla öğrencinin kalması kesinleşir. Böyle bir kontrol yapılması, gereksiz işlemlerin yapılmasını engeller.
4. `Vize` değişkenine, veritabanından alınan vize notunun %30'u atanır.
5. `Final` değişkenine, veritabanından alınan final notunun %70'i atanır.

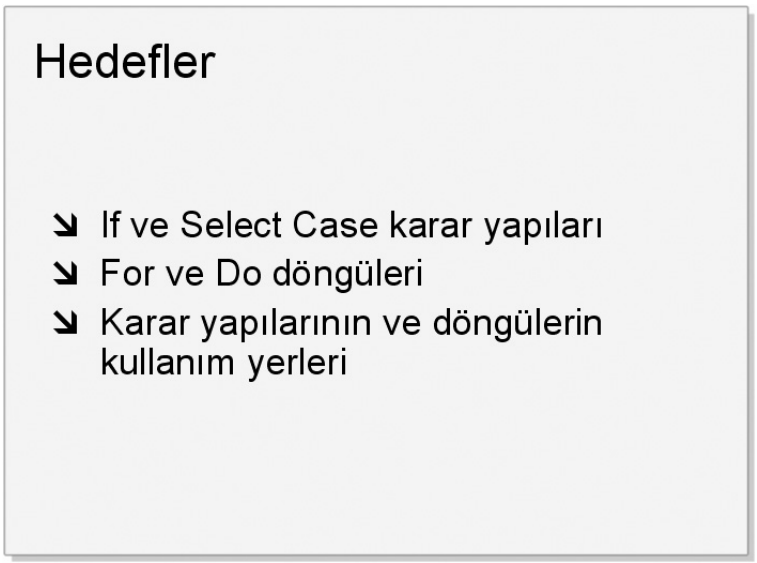
6. **Sonuc** değişkenine, **Vize** ve **Final** değerlerinin toplamı atanır.
7. **Sonuc** değerinin 50'den büyük olup olmadığı kontrol edilir.
8. **Sonuc** 50'den büyükse ekrana "**Geçtiniz**" yazan bir mesaj çıkartılır. Algoritmadan çıkılır.
9. **Sonuc** 50'den küçükse ekrana "**Kaldınız**" yazan bir mesaj çıkartılır.

Modül Sonu Soruları & Alıştırmalar



1. Algoritma kurulurken esas alınacak noktalar nelerdir?
2. Dump Coding ile algoritmanın farkı nedir?
3. Bir ürünün bilgilerinin, veritabanından çekilerek kullanıcıya görüntüleme işleminin algoritmasını kurun.
4. Bu algoritmayı akış diyagramı ile gösterin.

Modül 6: Karar Yapıları ve Döngüler

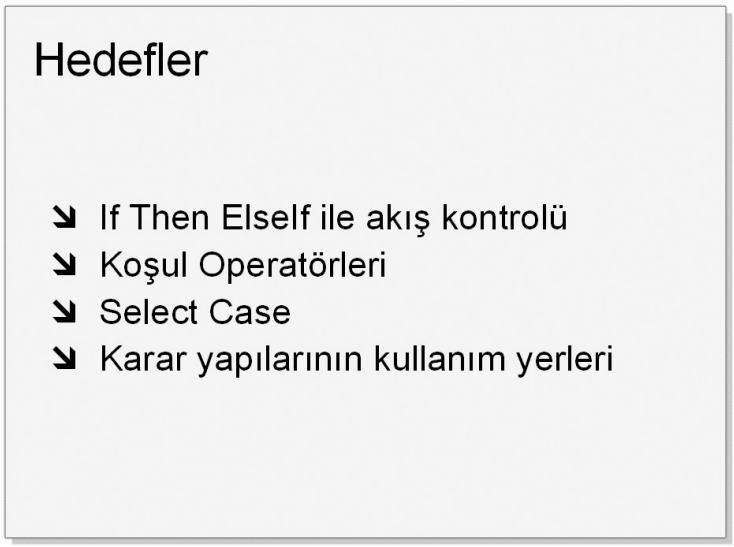


Karar yapıları ve döngüler, algoritmaların akışını yönlendirir. If ve Select Case karar yapıları ile gerekli kontroller yapılarak, uygulama istenen şekilde yönlendirilir. Döngüler ile, belli bir yol izelenerek birçok kez tekrarlanacak işlemler bir defa yazılır. Bu işlem, döngü sonlanana kadar gerçekleştirilir.

Bu modül tamamlandıktan sonra:

- **If** ve **Select Case** karar yapılarını öğrenecek,
- **For** ve **Do** döngülerini tanıyacak,
- Hangi karar yapısının ve döngünün nerede kullanılacağını öğreneceksiniz.

Konu 1: Karar Yapıları



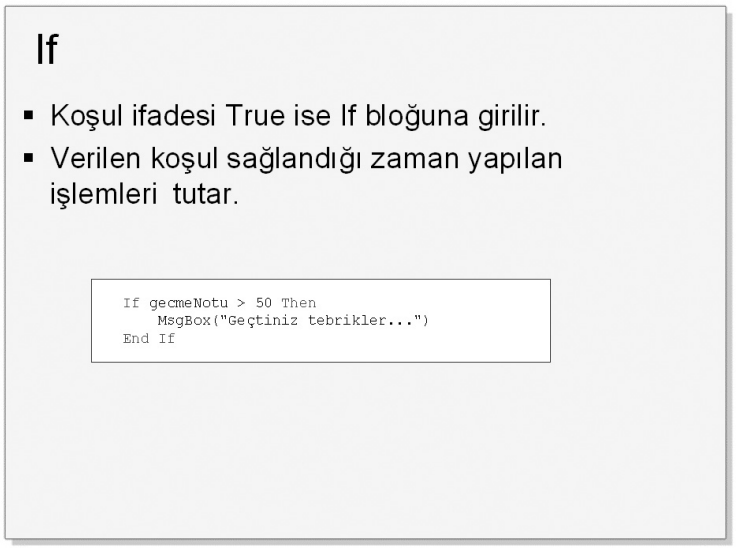
Uygulamalar çalıştırılırken, yazılan kodların çalışma sırası, satırların teker teker işlenmesi ile gerçekleşir. Ancak çoğu zaman, bazı kodların sadece belli durumlarda çalışması istenir. Örneğin uygulama açılırken kullanıcı adı ve parola sorulması, kullanıcıların seviyelerine göre erişim izinlerinin tanımlanması gibi durumlarda kontrol işlemleri yapılmalıdır. Bu kontroller de karar yapıları ile gerçekleştirilir.

Algoritmaların akışını kontrol etmekte en büyük rol, karar yapılarınındır. Visual Basic .NET dilinde farklı şekillerde kullanılan, ancak benzer görevlere sahip karar yapıları tanımlıdır.

Bu bölüm tamamlandıktan sonra;

- **`If Then ElseIf`** karar yapıları ile akış kontrolü yapabilecek,
- Kontrollerde kullanılan koşul operatörlerini tanıyacak,
- **`Select Case`** karar yapısını kullanabilecek,
- Hangi karar yapısının nerede kullanılacağını öğreneceksiniz.

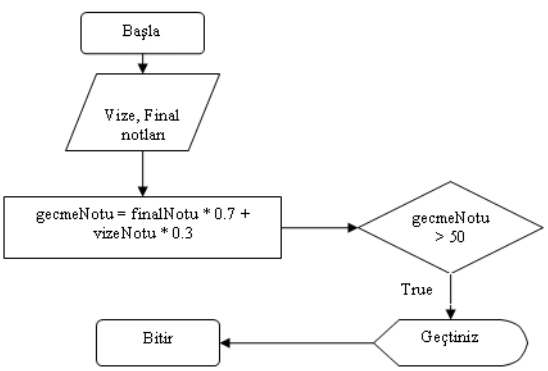
If



If karar yapısı, bir koşul sağlandığı zaman yapılacak işlemleri kapsar. Kontrol edilecek koşul ifadesinin sonucu **True** değerini alırsa, **If EndIf** bloğu arasındaki kodlar çalıştırılır.

```
If Koşul Then

End If
```

Örnek: Vize ve final notunu kullanıcıdan aldıktan sonra, geçme notunu hesaplayan ve notun 50'den büyük olması durumunda ekrana "geçtiniz" mesajını çıkartan algoritma (Resim 6.1).



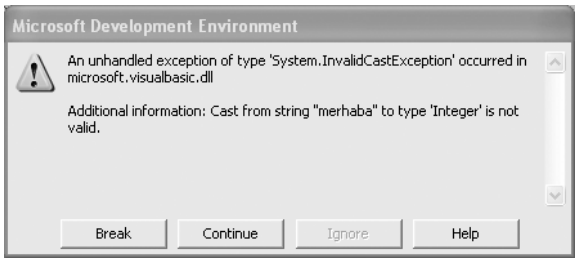
RESİM 6.1.

```
Dim gecmeNotu As Single
Dim finalNotu As Short = InputBox("Final Notunu girin:")
Dim vizeNotu As Short = InputBox("Vize Notunu girin:")

gecmeNotu = finalNotu * 0.7 + vizeNotu * 0.3

If gecmeNotu > 50 Then
    MsgBox("Geçtiniz tebrikler...")
End If
```

Örnek: **InputBox** ile çoğunlukla sayı tipinde bir değer almak istenir. **InputBox** metodundan dönen değer her zaman **String** tipinde olacağı için, bu değer Implicit Conversion ile istenen sayı tipine çevrilir.

```
Dim sayi As Integer = InputBox("Sayi giriniz")
```

Ancak **InputBox** kutusunda **Cancel** düğmesi tıklandığında ya da sayı tipinde bir şey girilmediğinde Resim 6.2'de görülen hata mesajı alınır.



RESİM 6.2: Hata mesajı.

Bu durumda, girilen değerin sayı tipinde olup olmadığı kontrol edilmek zorundadır.

```
Dim sayi As Integer
Dim gecici As String = InputBox("Sayi giriniz:")

If IsNumeric(gecici) Then
    sayi = gecici
End If
```

If yapısında geçen koşul ifadelerinin sonucu **Boolean** tipinde bir değerdir. Karşılaştırma operatörlerinin dönüş tipinin **Boolean** olduğundan Modül 4'teki Karşılaştırma Operatörleri başlıklı ayrımda bahsedildi. Eğer **If** deyimindeki bir karşılaştırma ifadesi kullanılmazsa, buradaki değer Implicit Conversion ile **Boolean** tipine çevrilir.

```
Dim sayi As Integer = InputBox("Sayi girin:")
```

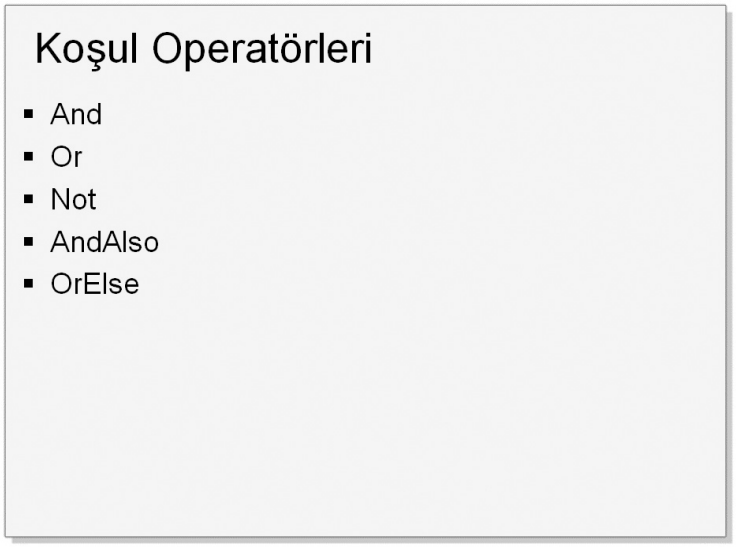
```
If sayi Mod 2 Then
    MsgBox(sayi & " tek sayıdır.")
End If
```

sayi Mod 2 ifadesinin değeri, girilen sayıya göre **1** ya da **0** olabilir. Bu değerler **If** kontrolünde **Boolean** tipine çevrilir. **1** değeri **True** olarak çevrileceği için, girilen sayı tek ise **If** bloğunun içine girilir.

If blokları içinde iç içe **If** kontrolleri yapılabilir.

```
Dim SinifKodu As String = InputBox("Açılacak Sınıf Kodunu
Girin:")

If SinifKodu.Length = 6 Then
    ' Sınıf kodlarının son 4 harfi ise sayı tipindedir
    ' ve sınıf numarasını belirtir
    Dim sinifNumarasi As String = Mid(SinifKodu, 3, 4)

    If IsNumeric(sinifNumarasi) Then
        ' Sınıf kodlarının ilk iki harfi,
        ' sınıfın türünü belirler
        Dim sinifTuru As String = Mid(SinifKodu, 1, 2)

        If sinifTuru = "YM" Then
            Label1.Text = "YM sınıfı açılıyor"
        End If

        If sinifTuru = "YU" Then
            Label1.Text = "YU sınıfı açılıyor"
        End If
    End If
End If
```

Koşul Operatörleri



Veri tipleri ve değişkenler üzerinde kontrol yapılırken birden fazla koşula ihtiyaç duyulabilir. Bu durumda, koşulları birbirleriyle karşılaştıracak operatörler kullanılır. Bu kontrollerden dönen değerler **`Boolean`** tipinde olduğu için, koşul operatörleri de bu değerler üzerinde işlem yaparlar.

And

Bu operatör, verilen koşulların kesişimini alır. Eğer tüm koşulların değeri **`True`** ise sonuç da **`True`** olur. En az bir tane **`False`** değeri olan koşul varsa, sonuç **`False`** olur (Tablo 6.1).

TABLO 6.1: And Operatörü

Koşul 1	Koşul 2	Koşul 1 And Koşul 2
`True`	`True`	`True`
`True`	`False`	`False`
`False`	`True`	`False`
`False`	`False`	`False`

Or

`Or` operatörü, verilen koşulların birleşimini alır. Eğer tüm koşulların değeri **`False`** ise sonuç **`False`** olur. En az bir tane **`True`** değeri varsa sonuç **`True`** olur (Tablo 6.2).

TABLO 6.2: Or Operatörü

Koşul 1	Koşul 2	Koşul 1 Or Koşul 2
True	True	True
True	False	True
False	True	True
False	False	False

XOr

XOr operatörü, verilen koşulların farklarını kıyaslar. Eğer her iki data aynı ise **False** değerini üretir. Eğer iki data birbirinden farklı ise **True** değerini üretir (Tablo 6.3).

TABLO 6.3: XOr Operatörü

Koşul 1	Koşul 2	Koşul 1 XOr Koşul 2
True	True	False
True	False	True
False	True	True
False	False	False

```
Dim x As String = "1111111"
Dim y As String = "111111O"

MessageBox.Show(x Xor y) ' Sonuç: 1 ya da True
```

Not

Bir koşulun değerini tersine çevirir. Koşul **False** ise **True**, **True** ise **False** olur (Tablo 6.4).

TABLO 6.4: Not Operatörü

Koşul	Not Koşul
True	False
False	True

AndAlso

Koşullardan biri **False** ise, diğerleri kontrol edilmeden **False** değeri döndürülür. Bu tip bir kullanım, birçok koşulun kontrol edilmesi gerektiğinde performansı artırır.

```
Dim dizi() As String = {}
' Diziye eleman ekleme işlemleri
' ...
```

```
If dizi.Length > 0 AndAlso dizi(1).EndsWith(".") Then
      Label1.Text = "Cümle sonundaki kelime: " & dizi(1)
End If
```

Bu örnekte, dizinin ilk elemanı üzerinde bir kontrol yapılmak isteniyor. Ancak diziye eleman eklenmemişse, ilk elemana ulaşırken hata üretilecektir. Dolayısıyla dizinin uzunluğunu da kontrol etmek gerekir. Kontrol **And** ifadesi ile yapılsaydı, dizi elemanın noktayla bitip bitmediği ve dizinin uzunluğu kontrol edilecekti. Bu durumda iç içe **If** ifadeleri ile uzun bir kod yazılacaktı. Pek çok kıyaslama gerekecek ve performans düşecekti. Ancak burada, dizi uzunluğu koşulu sağlanmazsa, diğer koşula geçilmeden **If** kontrolünden çıkılır.

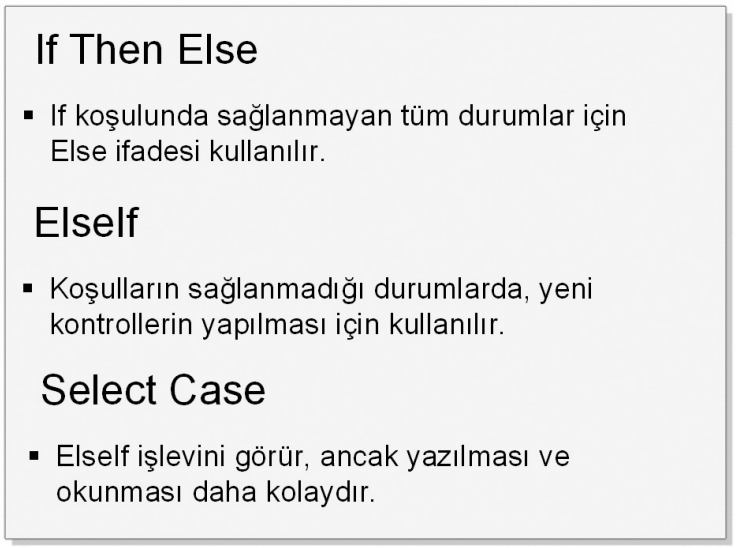
OrElse

Koşullardan biri **True** ise, diğerleri kontrol edilmeden **True** değeri döndürülür.

```
Dim Rol As String
' Veritabanından, kullanıcının rolü alınır.
' ...

' Sadece Administrator, Moderator ve Power User rolündeki
' kullanıcılar dosya silme işlemi yapabilirler.
If Rol = "Administrator" OrElse Rol = "Moderator" _
    OrElse Rol = "Power User" Then

    Kill("C:\BilgeAdamVeriTabani.mdb")
End If
```

Dosya silme işlemi için, kullanıcının rolü veritabanından alındıktan sonra, kontrol işlemi yapılır. Eğer bir kullanıcın rolü Administrator, Moderator veya Power User rollerinden biriyse, diğer kontrollerin yapılması gerekmez. Bu örnekte Rol değişkeni Administrator değerine eşitse, diğer iki koşul kontrol edilmeden **True** ifadesi döner.

If Then Else



`Else` ifadesi, **`If`** yapısındaki koşulun sağlanmadığı bütün durumlarda devreye girer.

```
If Koşul Then
   'Diğer kodlar
Else
   'Diğer kodlar
End If
```

Koşul **`True`** ise **`If – Else`** arasındaki kodlar, koşul **`False`** ise **`Else – End If`** arasındaki kodlar çalışır.

Örnek: Her 100 milisaniyede bir, formun renginin siyahken beyaz olması, beyazken de siyah olması için, formun renginin kontrolü yapılması gerekiyor.

```
Dim Beyaz As Boolean = True

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Timer1.Tick
    If Beyaz Then
        Me.BackColor = Color.Black
        Beyaz = False
    Else
        Me.BackColor = Color.White
        Beyaz = True
    End If
End Sub
```

If kontrolünde formun beyaz olup olmadığı **Boolean** tipindeki bir değişkende tutulur. Koşulda eğer **Beyaz** adlı değişken **True** ise, formun arka planı siyah yapılır. Bu koşulun sağlanmadığı durumda, yani **Beyaz** değişkeninin **False** olduğu durumda, **Else** içindeki kodlar çalışır ve formun arka planı beyaz yapılır. Her kontrolden sonra **Beyaz** değişkeninin değiştirilmesinin nedeni, formun bir siyah, bir beyaz olmasının istenmesidir.

ElseIf

If deyimindeki koşul sağlanmadıysa **Else** deyimindeki kodlar çalışıyordu. Ancak bazı durumlarda **Else** içinde de kontrol yapmak gerebilir.

```
If Koşul Then

ElseIf Koşul2 Then

ElseIf Koşul3 Then

End If
```

Örnek: Günün saatine göre karşılama mesajı çıkartmak için, **saat** değişkeninin birçok kez kontrol edilmesi gerekir. Sadece bir **If** kontrolü yapılsaydı, sadece iki karşılama mesajı çıkartılabilirdi.

```
Dim karsilamaMesaji As String = " BilgeAdama hoşgeldiniz!"
Dim saat As Byte = Hour(Now)

If 9 <= saat < 12 Then
    karsilamaMesaji = karsilamaMesaji.Insert(0, "Günaydın,")
ElseIf 12 <= saat < 16 Then
    karsilamaMesaji = karsilamaMesaji.Insert(0, "İyi
günler,")
ElseIf 16 <= saat < 18 Then
    karsilamaMesaji = karsilamaMesaji.Insert(0, "İyi
akşamlar,")
End If

' Formun başlığı karşılama mesajını
' gösterecek şekilde ayarlanır
Me.Text = karsilamaMesaji
```

Select Case

Select deyimi **ElseIf** ile benzer işlevi görür, ancak okunması daha kolaydır. **Select** ile seçilen bir değerin kontrol edilmesi **Case** ifadelerinde yapılır.

```
Dim dosya As String = TextBox1.Text

Select Case ComboBox1.SelectedText
    Case "Kopyala"
        Dim yeniYer As String
        yeniYer = InputBox("Kopyalanacak yeri girin:")
        FileCopy(dosya, yeniYer)

    Case "Ad Değiştir"
        Dim yeniAd As String
        yeniAd = InputBox("Dosyanın yeni adını girin:")
        Rename(dosya, yeniAd)

    Case "Sil"
        Kill(dosya)

    Case Else
        MessageBox.Show("Hatalı seçim")
End Select
```

Buradaki **Select Case** kullanımı, **ComboBox** kontrolünden seçilen öğeye göre bir işlemin gerçekleştirilmesini sağlar.. Seçilen öğenin yazısı **Case** ifadelerinde verilen değerlere eşitse, ilgili kodlar çalıştırılır. **Case Else** ise, diğer koşulların sağlanmadığı tüm durumlarda devreye girer.

Case ifadelerindeki kontroller, sadece bir tek değere "eşitlik" ile sınırlı değildir. **Select** ile kontrol edilecek değerin birden fazla olması durumunda, aynı kodların çalışması istenebilir veya değerlerin belirli aralıklarda olması gibi durumlarla da karşılaşılabilir.

To Kullanımı

To ile değerlerin belirli aralıklarda olup olmadığı kontrol edilir. **String** tipindeki değerlerin kontrolü alfabetik olarak yapılır.

```
Dim durum As String

Select Case UrunStokSayisi
    Case 0
        durum = "Ürün Tükenmiş"
    Case 1 To 10
        durum = "Çok Az"
    Case 10 To 25
        durum = "Az"
    Case 25 To 50
        durum = "Yeterli"
    Case 50 To 75
```

```
        durum = "Fazla"
    Case Else
        durum = "Çok Fazla"
End Select
```

Is Kullanımı

Is ifadesi karşılaştırma operatörleri ile kullanılır.

```
Randomize()
Dim sayi As Integer = Rnd() * 1000

Select Case sayi1
    Case Is < 10
        MsgBox(sayi & " sayısı tek basamaklıdır")
    Case Is < 100
        MsgBox(sayi & " sayısı iki basamaklıdır")
    Case Is < 1000
        MsgBox(sayi & " sayısı üç basamaklıdır")
End Select
```

Burada **sayi** değişkeninin kontrolü, küçüktür karşılaştırma operatörü ile yapılıyor.

Burada dikkat edilmesi gereken bir durum da, **sayi** değişkenini tek basamaklı olduğu zamanki durumdur. Örneğin, sayı değişkeni 6 değerini aldığında, **Case** ifadelerindeki bütün koşulların sağlandığı görülür. Birden fazla **Case** içine girilemediğinden, koşulun sağlandığı ilk **Case** içindeki kodlar çalıştırılır. Bu örnekte **Case Is < 1000** ifadesi en başa alınsaydı, girilen bütün sayıların üç basamaklı olduğu gösterilirdi.

Birden Fazla Koşulun Kontrolü

Bir **Case** ifadesine birden fazla koşul kullanılmak isteniyorsa, bu koşullar virgülle ayrılır. Bu koşullardan herhangi biri sağlandığı zaman **Case** içine girilir. Dolayısıyla virgül, **OrElse** koşul operatörü görevini görür.

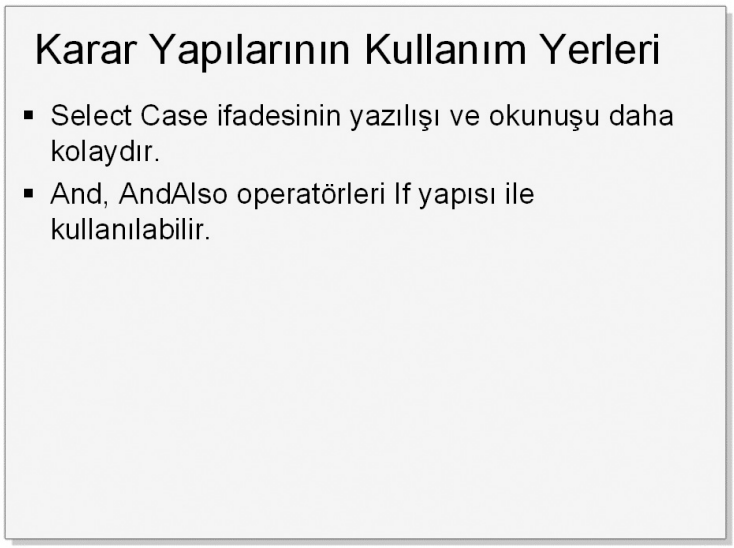
```
Select Case ifade
    Case Koşul1, Koşul2, Koşul3

    Case Koşul4, Koşul5
End Select
```

Örnek:

```
Select Case MsgBox("Devam etmek istiyor musunuz?",
MsgBoxStyle.YesNoCancel)
    Case MsgBoxResult.No, MsgBoxResult.Cancel
        Label1.Text = "İşlem iptal edildi"
```

```
    Case MsgBoxResult.Yes
        Label1.Text = "Kayıt işlemi gerçekleştirildi."
End Select

Dim UzmanlikAlani As String

Select Case ProgramlamaDili
    Case "PHP", "ASP"
        UzmanlikAlani = "Web"

    Case "T-SQL"
        UzmanlikAlani = "Veri Tabanı"
End Select
```

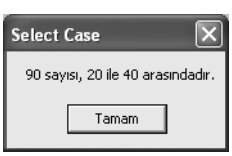
Hangi Karar Cümlesi Nerede Kullanılır?



If ve **Select** karar yapıları benzer işlevler görseler de, kullanım yerlerine ve birbirlerine göre değişik avantajları vardır. **If Else If** karar yapılarında, kontrol edilen değişkenlerin ya da değerlerin her seferinde tekrar yazılması gerekir. Bu durumda **Select** karar yapısı, kodların yazılışını ve okunuşunu kolaylaştırması açısından tercih edilmelidir.

Ayrıca, bir **Case** ifadesinde kontrol edilen koşullar virgülle ayrıldığında **OrElse** işlemi yapılır. **Select Case** karar yapısında **And** kullanımı yapılamaz.

```
Dim sayi As Integer = 90
Select Case sayi
    Case Is > 20, Is < 40
        MsgBox(sayi & " sayısı, 20 ile 40 arasındadır.")
    Case Is > 40
        MsgBox(sayi & " sayısı, 40 tan büyüktür.")
End Select
```



RESİM 6.3.

Bu örnekte ilk **Case** ifadesindeki ilk koşul gerçekleştiği için diğer koşullar göz ardı edilir. Verilen sayının 20 ile 40 arasında olmasının kontrolü, **Case 20 To 40** ifadesi ile ya da **If** karar yapısı kullanılarak yapılması gerekir (Resim 6.3).

```
If sayi > 20 And sayi < 40 Then
    MsgBox(sayi & " sayısı, 20 ile 40 arasındadır.")
ElseIf sayi > 40 Then
    MsgBox(sayi & " 40 tan büyüktür.")
End If
```

Bir grup **RadioButton** kontrolü içinden sadece bir tanesi seçilebildiği için, seçilen kontrolü bulmak için **ElseIf** yapısının kullanımı yeterli olacaktır.

```
If RadioButton1.Checked Then

ElseIf RadioButton2.Checked Then

ElseIf RadioButton3.Checked Then

End If
```

Ancak bu kontroller, **CheckBox** kontrolünün kullanım yapısına uymaz. Formlarda birden fazla **CheckBox** kontrolü seçilebildiği için, seçilen kontrolleri bulmak için **If EndIf** blokları kullanılmalıdır.

```
If CheckBox1.Checked Then

End If

If CheckBox2.Checked Then

End If

If CheckBox3.Checked Then

End If
```

Uygulama

Bu uygulama kullanıcıya, stok durumuna veya tarihe göre değişen görüntüleme seçenekleri sunarak, ürün katalogunu tanıtır. Ürünler kategorilere göre ayrılmıştır ve **ComboBox** kontrolleri ile filtrelerden biri seçilmediği takdirde işlem gerçekleşmez.

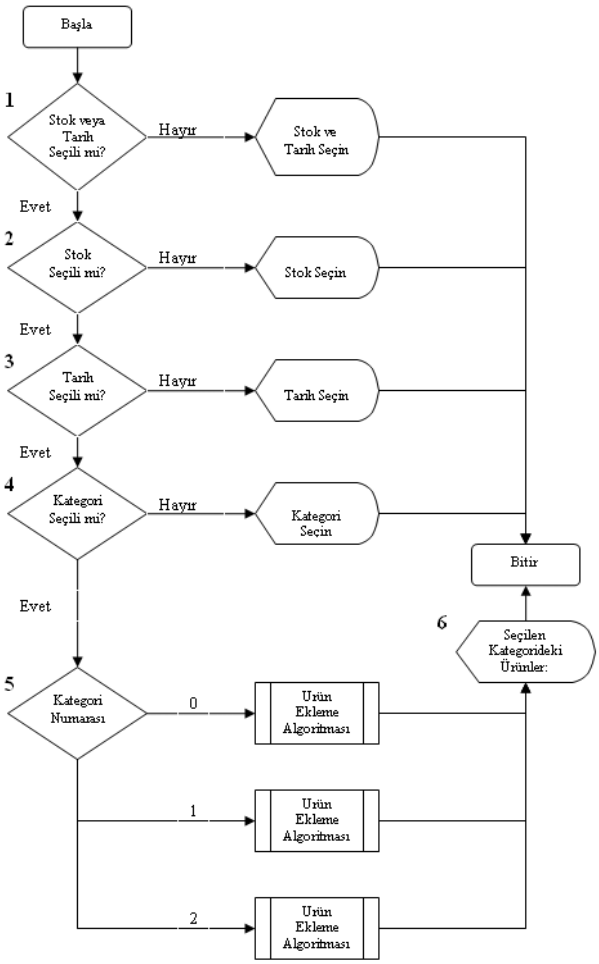
Stok durumu filtresi ile satılmakta olan ürünlerden sadece stokta bulunan ya da stokta kalmamış olanlar listelenebilir. Tarihe göre filtreleme ile yeni çıkan ürünler ya da tüm ürünler gözlenebilir.

Uygulamada, akış diyagramından koda geçiş aşaması rahat bir şekilde görülecektir.

Algoritmanın İncelenmesi

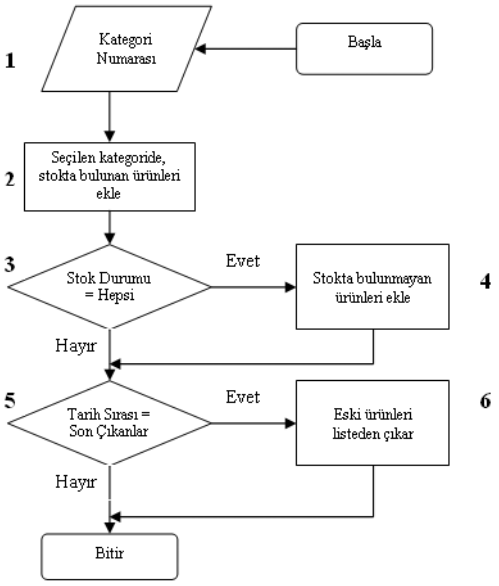
Uygulamanın algoritması başlangıç ve ürün ekleme algoritması olarak ikiye ayrılmıştır.

Başlangıç algoritmasında ürün kategorisinin, stok durumunun ve tarih filtrelerinin seçilip seçilmediği kontrol edilir. Herhangi biri seçilmediği zaman kullanıcıya ilgili mesaj gösterilir ve algoritmadan çıkılır. Tüm kontroller yapıldıktan sonra, ilgili kategorideki ürünlerin listeye eklenmesi için diğer algoritma devreye girer (Resim 6.4).



RESİM 6.4: Başlangıç algoritması.

Ürün ekleme algoritması, ilk algoritmada seçilen kriterlere göre, kullanıcıya gösterilecek ürün listesini doldurur. Bu algoritma başlangıç olarak kategori numarasını alır. Bu kategorideki stokta bulunan ürünleri listeye ekler. Stok durumu filtresinde “Hepsi” değeri seçiliyse, stokta o an bulunmayan ürünler de listeye eklenir. Tarih filtresinde “Son çıkanlar” değeri seçiliyse, eski ürünler listeden çıkartılır.



RESİM 6.5: Ürün ekleme algoritması.

Forma Kontrollerin Eklenmesi

1. **`UrunYelpazesi`** isminde bir Visual Basic Windows projesi açın.
2. Forma biri **`lbUrunler`**, diğeri **`lbKategoriler`** isminde iki **`ListBox`** kontrolü ekleyin. **`lbUrunler`** liste kutusu tüm filtreler uygulandıktan sonra çıkan ürünleri listeler. **`lbKategoriler`** liste kutusuna kategori isimlerini ekleyin:
   - **`Video`**
   - **`Kitap`**
   - **`Yazılım`**
3. Forma biri **`cmbTarihSirasi`**, diğeri **`cmbStokDurumu`** isminde iki **`ComboBox`** kontrolü ekleyin. **`cmbTarihSirasi`** son ürünleri; **`cmbStokDurumu`** stoktaki ürünleri gösteren filtre olarak kullanılacaktır. **`cmbTarihSirasi`** elemanlarına "**`Son Çıkanlar`**" ve "**`Tüm Ürünler`**" değerlerini, **`cmbStokDurumu`** elemanlarına "**`Sadece Stoktakiler`**" ve "**`Hepsi`**" değerlerini ekleyin.
4. **`lblMesaj`** isminde bir **`Label`** kontrolü ekleyin. **`Dock`** özelliğini **`Buttom`** yapın. Bu kontrol, filtrelerin seçilmediği durumda hata mesajlarını gösterecektir.
5. **`lblSecilenUrunler`** isminde bir **`Label`** kontrolü ekleyin ve **`lbUrunler`** liste kutusunun üstüne yerleştirin. Bu kontrol, seçilen ürünlerin hangi kategoride olduğunu gösterecektir.
6. Forma **`btnListele`** isminde bir **`Button`** kontrolü ekleyin.

Kodların Yazılması

Bu uygulamadaki kodların tamamı **btnListele** düğmesinin **Click** olayına yazılacaktır. Kodlar arasındaki numaralar, akış diyagramında işlenen durumlara referans verir. Algoritma 1, başlangıç algoritmasındaki numaraları; Algoritma 2, ürün ekleme algoritmasındaki numaraları ifade eder.

1. **btnListele** düğmesini çift tıklayın ve **Click** olayına gelin. Düğme her tıklandığında liste kutusuna ardı ardına öğeler eklenmemesi ve hata mesajlarının temizlenmesi için gerekli kodları yazın.

```
lblMesaj.Text = ""
lbUrunler.Items.Clear()
```

2. Kategori listesinden, stok ve tarih filtreleri için açılan kutulardan öğelerin seçili olup olmadığının kontrolü yapılır. Eğer seçilmemiş bir değer varsa, ilgili hata mesajı **lblMesaj** etiketinde görüntülenir.

```
' Algoritma 1 - 1
If cmbStokDurumu.SelectedIndex = -1 And
cmbTarihSirasi.SelectedIndex = -1 Then
    lblMesaj.Text = "Stok Durumu ve Tarih Sırası seçiniz."
    ' Algoritma 1 - 2
ElseIf cmbStokDurumu.SelectedIndex = -1 Then
    lblMesaj.Text = "Stok Durumunu seçiniz."
    ' Algoritma 1 - 3
ElseIf cmbTarihSirasi.SelectedIndex = -1 Then
    lblMesaj.Text = "Tarih Sırasını seçiniz."
    ' Algoritma 1 - 4
ElseIf lbKategoriler.SelectedIndex = -1 Then
    lblMesaj.Text = "Kategori seçiniz."
Else
    ' Algoritma 1 – 5
```

3. **If ElseIf** deyimlerinde tüm kontroller yapıldıktan sonra **Else** ifadesine geçilir. Algoritmanın akışı bundan sonra ürün ekleme işlemiyle devam edecektir.

```
Select Case lbKategoriler.SelectedIndex
    ' Algoritma 2 - 1
    Case 0
        ' Sadece stokta bulunan ürünler eklenir.
        ' Algoritma 2 - 2
        lbUrunler.Items.Add("MSDN Tv Visual Basic 5")
        lbUrunler.Items.Add("MSDN Tv Visual Basic 4")

        ' Stokta bulunan veya bulunmayan ürünlerin Hepsi
```

```
        ' seçiliyse, kalan ürünler de listeye eklenir.
        ' Algoritma 2 - 3
        If cmbStokDurumu.SelectedIndex = 1 Then
            ' Algoritma 2 - 4
            lbUrunler.Items.Add("MSDN Tv Visual Basic")
            lbUrunler.Items.Add("MSDN Tv Visual Basic 2")
            lbUrunler.Items.Add("MSDN Tv Visual Basic 3")
        End If

         ' Eski ürünlerin gösterilmesi istenmiyorsa
         ' listeden çıkartılır.
         ' Algoritma 2 - 5
        If cmbTarihSirasi.SelectedIndex = 0 Then
            ' Algoritma 2 - 6
            lbUrunler.Items.Remove("MSDN Tv Visual Basic")
            lbUrunler.Items.Remove("MSDN Tv Visual Basic 2")
        End If
```

4. Diğer iki kategori için liste ekleme işlemleri aynıdır.

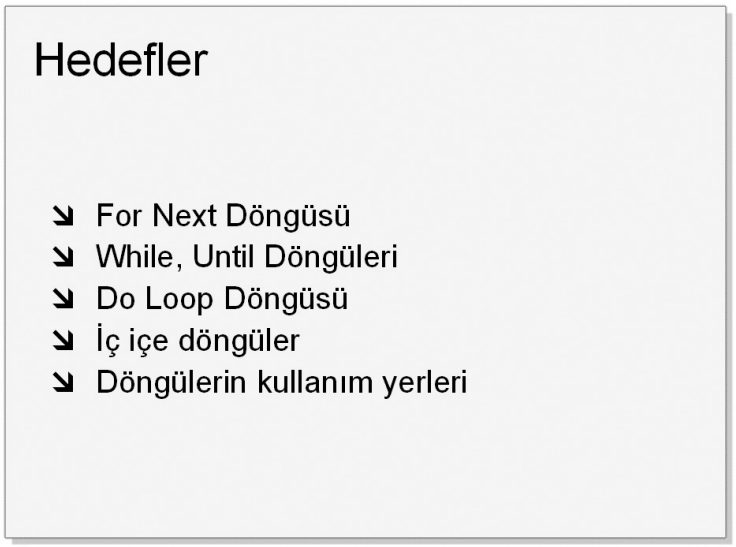
```
Case 1
    lbUrunler.Items.Add("Yazılım Uzmanlığı 1")
    lbUrunler.Items.Add("Yazılım Uzmanlığı 2")
    lbUrunler.Items.Add("Yazılım Mühendisliği Orta Düzey")
    lbUrunler.Items.Add("Yazılım Mühendisliği İleri Düzey")

    If cmbStokDurumu.SelectedIndex = 1 Then
        lbUrunler.Items.Add("Yazılım Mühendisliği Başlangıç
Düzey")
        lbUrunler.Items.Add("Access Giriş")
    End If

    If cmbTarihSirasi.SelectedIndex = 0 Then
        lbUrunler.Items.Remove("Yazılım Uzmanlığı 1")
    End If

Case 2
    lbUrunler.Items.Add("Visual Studio 6.0")
    lbUrunler.Items.Add("Visual Basic .NET Standard 2003")
    lbUrunler.Items.Add("Visual Basic  C# Standard 2003")

    If cmbStokDurumu.SelectedIndex = 1 Then
        lbUrunler.Items.Add("Visual Studio .NET 2005")
    End If

    If cmbTarihSirasi.SelectedIndex = 0 Then
```

```
            lbUrunler.Items.Remove("Visual Studio 6.0")
        End If
    End Select
```

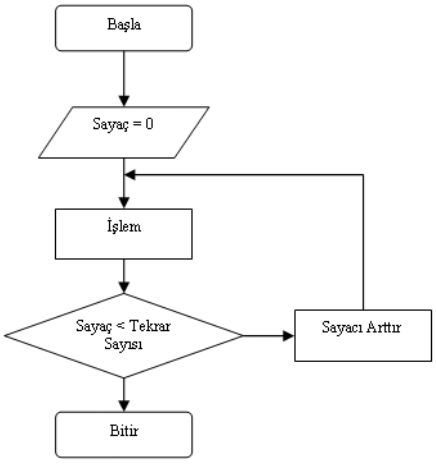
5. **`Select`** ifadesinde tüm eklemeler yapıldıktan sonra ikinci algoritma biter. İlk algoritmanın son aşaması olan, kullanıcıya hangi kategoride ürün seçildiğini gösteren mesaj yazılır ve **`If`** karar yapısı sonlanır.

```
    ' Algoritma 1 - 6
    lblSecilenUrunler.Text = lbKategoriler.Text & "
Kategorisindeki Ürünler"
End If
```

Konu 2: Döngüler



Algoritmalarda bazı işlemlerin tekrar çalışması için, bu işlemlerin her seferinde yazılması gerekir. Ancak bu çözüm, çok fazla tekrar için hem yazmayı, hem de okumayı zorlaştırır. Örneğin, yüz elemanlı bir diziye rasgele sayı atanması için işlemin yüz defa yazılması gerekir. Döngüler ile işlem sadece bir defa yazılır ve tekrar sayısına göre bu işleme geri dönülür (Resim 6.6).



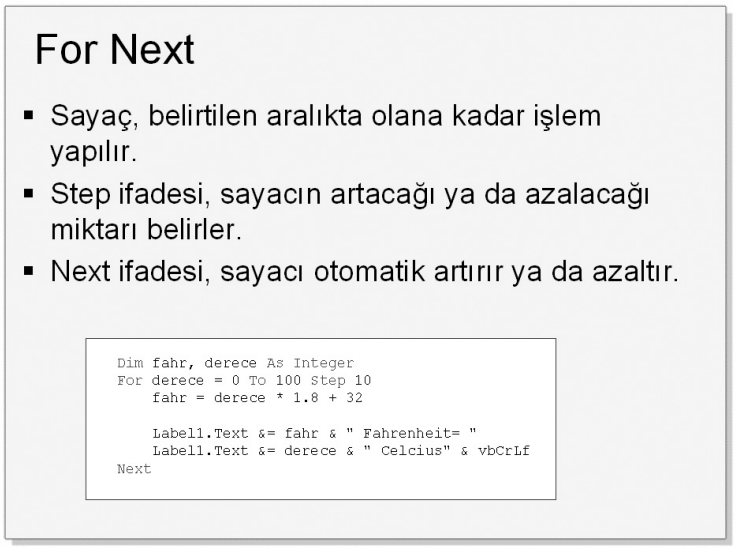
RESİM 6.6: Döngü.

Bu bölüm tamamlandıktan sonra;

- **`For Next`** döngüsünü tanıyacak,
- **`While`** ve **`Until`** döngülerinin farklarını ayırt edebilecek,

- **Do Loop** döngüsünü tanıyacak,
- İç içe döngüler kullanabilecek,
- Hangi döngünün nerede kullanıldığını öğreneceksiniz.

For Next



For döngüsü bir işlemin belirli sayıda yapılması için kullanılır.

```
Dim sayac As Byte
For sayac = 0 To 10
    MsgBox("Merhaba")
Next
```

Değişken tanımlamaları, **For** döngüsünün içinde de yapılabilir. Bu durumda, değişkenin kapsam alanı bu döngüyle sınırlı kalır.

```
For sayac As Byte = 0 To 10
    MsgBox("Merhaba")
Next
```

Bu döngünün avantajı, sayacın tekrar sayısı ile kontrolünü ve artırılmasını kendisi yapmasıdır. **Next** ifadesi **sayac** değişkenini varsayılan durumda bir artırır. **For** döngüsü içinde kullanılan **sayac** değişkeni sayısal bir değer olmalıdır. Döngülerde kullanılan sayaçlar, sadece belli bir sayıda işlem yapmayı sağlamaz. Sayaçların artma veya azalma adımları belirli olduğu için, kod içinde çoğu zaman bu avantajdan yararlanılır.

```
ListBox1.Items.Add("Karakter - ASCII kod karşılığı")
Dim i As Byte
For i = 0 To 255
    ListBox1.Items.Add(Chr(i) & "   - " & i)
Next
```

Örneğin dizi işlemlerinde, dizinin her elamanına ulaşmak için sayaç kullanılabilir. Sayacın artma hızı bir olduğu için **dizi(sayac)** ifadesi, sırayla dizinin elemanlarına ulaşmayı sağlar.

```
Dim i As Integer
Dim dizi(10) As Integer

For i = 0 To dizi.Length - 1
    dizi(i) = Rnd() * 100
Next
```

DİKKAT **Döngüler içinde dizi kullanılırken, sayaç sıfırdan başlamışsa döngünün biteceği nokta "dizi uzunluğu -1" olmalıdır.**

DİKKAT **Değişken tanımlamaları** For **döngüsünün içinde de yapılabilir. Bu durumda, değişkenin kapsam alanı bu döngüyle sınırlı kalır.**

Döngülerde sayaç değişkeninin artma veya azalma adımları **Step** ifadesi ile belirlenir.

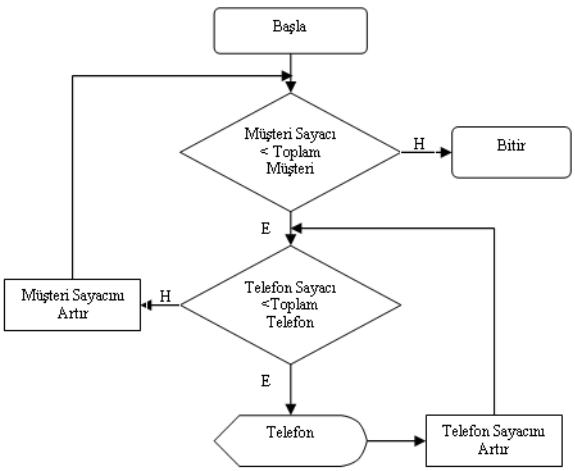
```
Dim fahr, derece As Integer
For derece = 0 To 100 Step 10
    fahr = derece * 1.8 + 32
    Label1.Text &= fahr & " Fahrenheit= "
    Label1.Text &= derece & " Celcius" & vbCrLf
Next
```

For Döngülerinin İç İçe Kullanımı

Çoğu zaman **For** döngülerindeki her etap için başka bir döngünün kurulması gerekir. Örneğin, bir müşterinin birden fazla telefon numarası bir dizi içinde tutuluyorsa, bütün müşterilerin telefonlarını listelemek için iki döngü kullanılması gerekir. İlk döngü tek tek müşterileri almak için, alt döngü ise her müşterinin telefonlarını almak için kullanılmalıdır (Resim 6.7).

```
Structure Musteri
    Dim Isim As String
    Dim Soyad As String
    Dim Telefonlari() As String
End Structure

Dim Musteriler() As Musteri
' Musteriler dizisi dolduruluyor
' ...

Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Dim i, j As Integer
```

```
    For i = 0 To Musteriler.Length – 1
        ' İlk müşteri seçiliyor
        Dim m As Musteri = Musteriler(i)

        Label1.Text &= m.Isim & " " & m.Soyad
        Label1.Text &= " müşterisinin telefonları:" & vbCrLf

        For j = 0 To m.Telefonlari.Length - 1
            Label1.Text &= m.Telefonlari(j)
            Label1.Text &= vbCrLf
        Next
    Next
End Sub
```



RESİM 6.7: İç içe döngüler.

Birden fazla boyutlu dizilerde işlem yaparken de **For** döngüsü iç içe kullanılabilir. Örneğin, iki boyutlu bir tabloda, ilk boyut için bir **For** döngüsü, diğer boyut için de başka bir **For** döngüsü kullanılarak dizinin tüm elemanlarına ulaşılabilir.

```
Randomize()
Dim tablo(4, 4) As String

Dim i As Byte
For i = 1 To 4
    tablo(0, i) = "Yazar " & i
    tablo(i, 0) = "Kitap " & i
    tablo(CInt(Rnd() * 3) + 1, CInt(Rnd() * 3) + 1) = "X"
Next
```

tablo isminde **String** değerleri tutan bir dizi oluşturulur ve dizinin ilk satırına yazar isimleri, ilk sütununa da kitap isimleri konur. **For** döngüsünün sayacı birden başladığı için dizinin 0,0 koordinatlı ilk elemanına değer atanmaz.

	Yazar 1	**Yazar 2**	**Yazar 3**	**Yazar 4**
Kitap 1				
Kitap 2				
Kitap 3				
Kitap 4				

Daha sonra, tablonun diğer elemanlarına rasgele X değerleri atanır. Bu değer hangi yazarın hangi kitabı yazdığını gösterir.

CInt(Rnd() * 3) + 1 ifadesi 1 ile 4 arasında rasgele bir sayı üretir. Bu sayı tablo dizisine indis olarak verildiğinde, kalan hücrelerde X değeri elde edilir.

	Yazar 1	**Yazar 2**	**Yazar 3**	**Yazar 4**
Kitap 1	X			
Kitap 2	X			
Kitap 3			X	
Kitap 4				X

Tablonun tüm elemanlarını listelemek için iç içe iki **For** döngüsü kullanılmalıdır.

```
For j As Integer = 0 To tablo.GetLength(0) - 1
    For h As Integer = 0 To tablo.GetLength(1) - 1
        Label1.Text &= tablo(j, h)
    Next
    Label1.Text &= vbCrLf
Next
```

Yazarlar ve Kitaplar tablosu hazırlandıktan sonra, hangi yazarın hangi kitabı yazdığını bulmak için yine tablo elemanları içinde dolaşıp X değerini aramak gerekir. İlk **For** döngüsü ile Kitaplar satırlarında, ikinci **For** döngüsü ile Yazarlar sütunlarında gezilir.

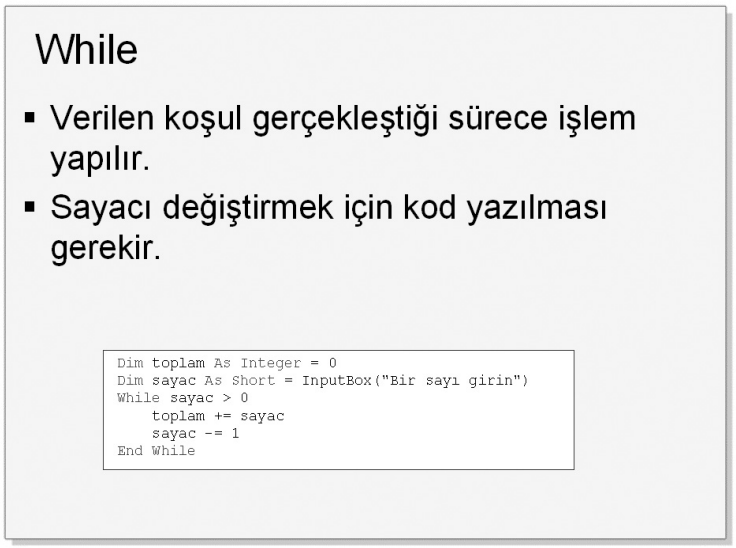
```
For j As Integer = 0 To tablo.GetLength(0) - 1
    For h As Integer = 0 To tablo.GetLength(1) – 1
        ' Tablonun her elemanının değeri
        ' X değeri ile karşılaştırılır.
        If tablo(j, h) = "X" Then
            Label2.Text = tablo(0, h) & ", "
            Label2.Text &= tablo(j, 0)
            Label2.Text &= " kitabını yazıyor"

            Exit For
        End If
```

```
    Next
    Label1.Text &= vbCrLf
Next
```

Tablonun her elemanı kontrol edilir ve X değeri bulunduğu zaman, yazar ismi ve kitap ismi ekrana yazdırılır. Yazar isimlerine, dizinin ikinci boyutunun ilk sırasında tutulduğu için **`tablo(0,h)`** kodu ile ulaşılır. Kitap isimlerine ise, dizinin ilk boyutunun ilk sırasında tutulduğu için **`tablo(j,0)`** kodu ile ulaşılır. Buradaki **`h`** ve **`j`** değişkenleri o anda kontrol edilen elemanın tablodaki indisleridir.

`Exit For` ifadesi, o anda bulunan **`For`** döngüsünden çıkmayı sağlar. Bu örnekte ikinci **`For`** döngüsü Yazarlar sütunu üzerinde döndüğü için, bu döngüden çıkıldığında, ilk **`For`** döngüsüne tekrar geçilir. Bu sefer yeni bir kitap için Yazarlar kontrol edilir. Sonuçta bir kitabı birden fazla yazar yazmasına rağmen, görüntülenecek olan sadece ilk yazardır.

While



While döngüsü bir koşul gerçekleştiği sürece çalışan döngüdür. **For** döngüsüne göre avantajı, sayı dışında herhangi bir veri tipi üzerinde karşılaştırma yapılabilir olmasıdır. Ancak **For** döngüsünde otomatik yapılan sayaçların artırılması ve kontrol edilmesi işlemleri bu döngüde yapılmaz. Bunun için kod yazılması gerekir.

```
While Koşullar

End While
```

While döngüsünde koşul kontrolleri döngünün içinde yapılır ve gerektiğinde **Exit While** ifadesi ile döngüden çıkılır.

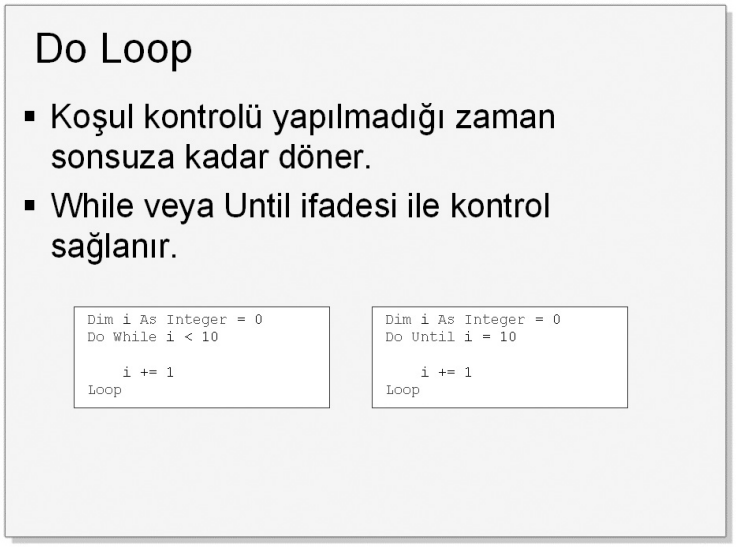
Birden ona kadar olan sayıların toplamını hesaplamak için, bir ve on arasındaki sayılar tek tek yazılıp toplanabilir. Bu, iyi bir yöntem olmasa da sonuç verir. Ancak kullanıcının girdiği bir sayıya kadar toplam almak için bir döngü gerekir.

```
Dim toplam As Integer = 0
Dim sayac As Short = InputBox("Bir sayı girin")
While sayac > 0
    toplam += sayac
    sayac -= 1
End While
```

Burada kullanıcının girdiği sayıdan itibaren sıfıra kadar giden bir döngü kurulur. Döngü sayacın sıfırdan büyük olduğu her durum için çalışır. Sayaç sıfırlandığında ise döngüden çıkılır.

```
While koşul1 and koşul2 or koşul3
End while
```

Do Loop



Do ifadesi ile başlayan döngülerin söz dizimi **Loop** ifadesi ile sonlanacak şekilde yazılır. **Loop** anahtar kelimesinin görevi, **Do** ifadesine geri dönmektir. Dolayısıyla koşul kontrolü yapılmayan bir **Do Loop** döngüsü sonsuza kadar çalışır.

```
Do
    Label1.Text = "Bu döngüden çıkılamaz."
Loop
```

Do Loop döngüsünde koşul kontrolleri döngünün içinde yapılır ve gerektiğinde **Exit Do** ifadesi ile döngüden çıkılır.

```
Do
    Dim kullanici, parola As String
    kullanici = InputBox("Kullanıcı Adı: ")
    parola = InputBox("Parola: ")

    If LCase(kullanici) = "ogrenci" And parola = "bilgeadam"
Then
        Exit Do
    End If
    MsgBox("Hatalı giriş!", MsgBoxStyle.Exclamation)
Loop

Label1.Text = "Giriş başarılı." & vbCrLf
```

Exit Do ifadesi ile karşılaşıldığı zaman döngüden çıkılacağı için, **If** içinde kontrol edilen koşul doğru ise mesaj kutusunun gösterildiği kod bölümüne geçilmez.

Do While

Koşullar, **Do Loop** döngüsünün içinde kontrol edilebildiği gibi, döngüye girmeden de kontrol edilebilir.

```
Dim i As Integer = 0
Do While i < 10

    i += 1
Loop
```

Döngünün bu şekilde kullanımının **While - End While** döngüsünden farkı yoktur. Ancak **Do While** döngüsünün yapısı daha esnektir ve kontrol **Loop** ifadesinde de yapılabilir.

```
Dim yanit As String
Do
    MsgBox("İşlem yapılıyor...")

    yanit = InputBox("Devam etmek istiyor musunuz?")
Loop While (yanit = "e" Or yanit = "E")
```

Buradaki fark, döngü içinde tanımlanan işlem bir defa yapıldıktan sonra koşulun kontrol edilmesidir. Yani en az bir defa yapılması istenen bir işlem **Do - Loop While** yapısı içersinde kullanılabilir.

Do Until

Do Loop döngüsünün, kontrol edilen koşul gerçekleşene kadar devam etmesi isteniyorsa, **Do Until** yapısı kullanılır. **Do While** döngüsü, koşul **True** olduğu sürece devam ederken; **Do Until** döngüsü, koşul **True** olduğu zaman sonlanır.

```
Dim dizi(9) As Integer
Dim i As Integer = 0

Do Until i = 10
    dizi(i) = i * i
    i += 1
Loop
```

Do While ile yazılan bir döngüyü **Do Until** ifadesine çevirmek için, **While** kontrolündeki koşulun tersinin alınması gerekir.

```
Dim yanit As String
Do
    MsgBox("İşlem yapılıyor...")

    yanit = InputBox("Devam etmek istiyor musunuz?")
Loop Until Not (yanit = "e" Or yanit = "E")

Do
    MsgBox("İşlem yapılıyor...")

    yanit = InputBox("Devam etmek istiyor musunuz?")
Loop Until (yanit <> "e" And yanit <> "E")
```

Sonsuz Döngüler

While döngüsü sayaç ile kullanılırken, sayacın değiştirilmesine dikkat edilmesi gerekir. Eğer sayaç değiştirilmezse, **While** ifadesindeki koşul hep **True** değerini alacağı için sonsuz döngüye girilir.

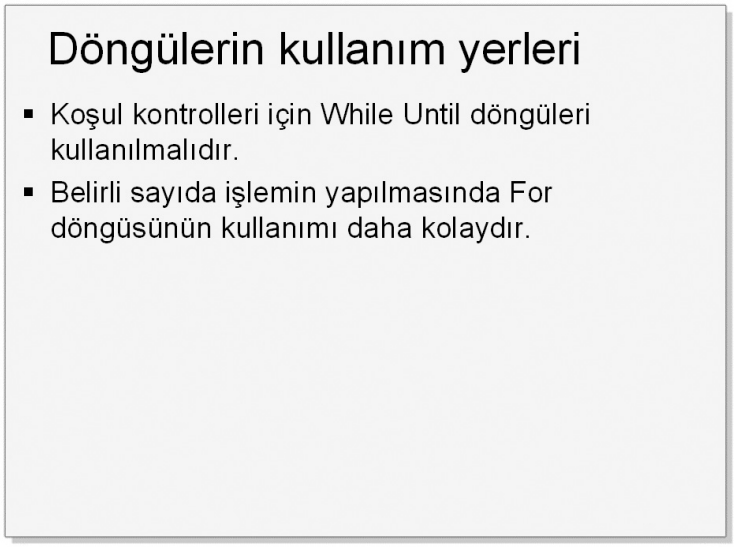
Sadece sayacın kontrol edilmediği durumlar değil, koşulların yazılmalarındaki mantık hataları da sonsuz döngüye sebebiyet verir.

```
Dim i As Short = 0

While i < 10 Or i > 5
    Label1.Text = "Sonsuz döngüye girildi"
    i += 1
End While
```

For döngüsünde sayacın değeri **Step** ifadesindeki değerden fazla bir sayıda azaltılırsa yine sonsuz döngüye girilir. Bu döngünün çalışması, **Integer** veri tipinin alabileceği minimum değere ulaşınca hata ile sonlanır.

```
For i As Integer = 0 To 9 Step 3
    MsgBox("j")
    i -= 4
Next
```

Hangi Döngü Nerede Kullanılır?



Visual Basic .NET dilinde **`While`** ve **`Until`** döngüleri, koşul kontrollerine izin verdiği için **`For`** döngüsüne göre daha esnek yapıdadır. **`For`** döngüsünde sayacın belli bir değere ulaşmış olup olmadığı kontrol edilir. Bu kontrol döngünün içinde otomatik olarak yapıldığı için yazılması daha kolaydır.

Örneğin, istenen bir işlemin sadece belli sayıda yapılması ise, sayaçlar ile uğraşmamak için **`For`** döngüsü tercih edilmelidir.

```
Dim i As Integer = 0
While i <= TekrarSayisi
    ' TekrarSayisi + 1 kadar işlem yapılır.

    i += 1
End While
```

```
For i As Integer = 0 To TekrarSayisi
    ' TekrarSayisi + 1 kadar işlem yapılır.
Next
```

Döngülerin belli koşullar sağlandığı sürece ya da sağlanana kadar çalışması, karar yapılarının kullanılmasını gerektirir. **`For`** döngüsünde koşul kontrolleri **`If`** veya **`Select`** karar yapıları ile yapılır. Ancak bu tip durumlarda **`While`** ve **`Until`** döngülerinin kullanılması kodun yazılımını kolaylaştırır.

```
For i As Integer = 0 To 0
    ' Yapılacak işlemler
```

```
    Dim cevap As String
    cevap = InputBox("Durmak için Cancel düğmesine basın")

    If cevap <> "" Then
        i -= 1
    End If
Next
```

```
Do
    ' Yapılacak işlemler
Loop Until InputBox("Durmak için Cancel düğmesine basın") =
""
```

Uygulama

Bu uygulamada, bir satranç tahtası üzerindeki bir filin hareket alanı hesaplanır. Satranç tahtası rasgele taşlarla doldurulur. Verilen bir koordinatta bulunan filin çapraz hareketlerine göre nereye ilerleyebileceği bulunur. Eğer filin önünde bir taş varsa, bu taşın bulunduğu yere ve daha gerisine ilerleyemeyecektir. Filin dört bir yanına çapraz olarak hareket edebileceği göz önünde bulundurulmalıdır.

Tahtanın Doldurulması

1. **Satranc** isminde bir Windows projesi açın.
2. Form üzerine **lbHareketAlani** isminde bir **ListBox**, **btnGoster** isminde bir **Button** ekleyin.
3. **btnGoster** düğmesinin **Click** olayına 8 x 8 boyutunda bir dizi tanımlayıp dolduran kodları yazın. Bu dizi **Boolean** tipinde değerler taşır. Verilen indisteki elemanın değeri **True** olması, o koordinatta bir taşın bulunduğunu belirtir.

```
Randomize()
Dim tahta(7, 7) As Boolean

For a As Integer = 0 To 7
    For b As Integer = 0 To 7
        tahta(a, b) = CInt(Rnd()) Mod 2
    Next
Next
```

Hareket Alanı

Tahta üzerindeki bir fil, dört çapraz yöne doğru ilerleyebilir. Dizide çapraz olarak ilerlemek **x** ve **y** koordinatlarının eşit oranda artması ve azalması demektir. Dizide ilerlerken **x** ve **y** koordinatlarının sıfırdan küçük ve dizinin boyutundan büyük olmamasına dikkat edilmelidir. Dört farklı yöne göre, koordinatlar artacak ya da azalacaktır.

1. Fili tahta üzerine yerleştirmek için kullanıcıdan koordinatları alın.

```
Dim x As Byte = InputBox("Filin x koordinatı:")
Dim y As Byte = InputBox("Filin y koordinatı:")
```

2. 0, 0 yönüne doğru olan yoldaki taşların kontrolünü yapın. Filin **x** ve **y** koordinatlarını birer düşürerek, koordinatlarda taş olup olmadığı kontrol edilir. Eğer taş yoksa bu kareye ilerlenebildiği, **lbHareketAlani** liste kutusuna koordinat eklenerek gösterilir. Yol üzerinde bir taş varsa, daha fazla ilerlenemeyeceği için **While** döngüsünden çıkılır.

```
Dim i As Integer = 1
While x - i >= 0 And y - i >= 0
    If Not tahta(x - i, y - i) Then
        lbHareketAlani.Items.Add(x - i & " - " & y - i)
        i += 1
    Else
        Exit While
    End If
End While
```

3. 7, 0 yönüne doğru ilerlenir ve olası hareketler liste kutusuna eklenir.

```
i = 1
Do While x + i < tahta.GetLength(0) And y - i >= 0
    If Not tahta(x + i, y - i) Then
        lbHareketAlani.Items.Add(x + i & " - " & y - i)
        i += 1
    Else
        Exit Do
    End If
Loop
```

4. 7, 7 yönüne doğru kontrol yapılır.

```
i = 1
Do Until x + i >= tahta.GetLength(0) Or y + i >=
tahta.GetLength(1)
    If Not tahta(x + i, y + i) Then
        lbHareketAlani.Items.Add(x + i & " - " & y + i)
        i += 1
    Else
        Exit Do
    End If
Loop
```

5. 0, 7 yönüne doğru taşlar kontrol edilir.

```
i = 1
Do While x - i >= 0 And y + i < tahta.GetLength(1)
    If Not tahta(x - i, y + i) Then
        lbHareketAlani.Items.Add(x - i & " - " & y + i)
        i += 1
    Else
        Exit Do
    End If
Loop
```

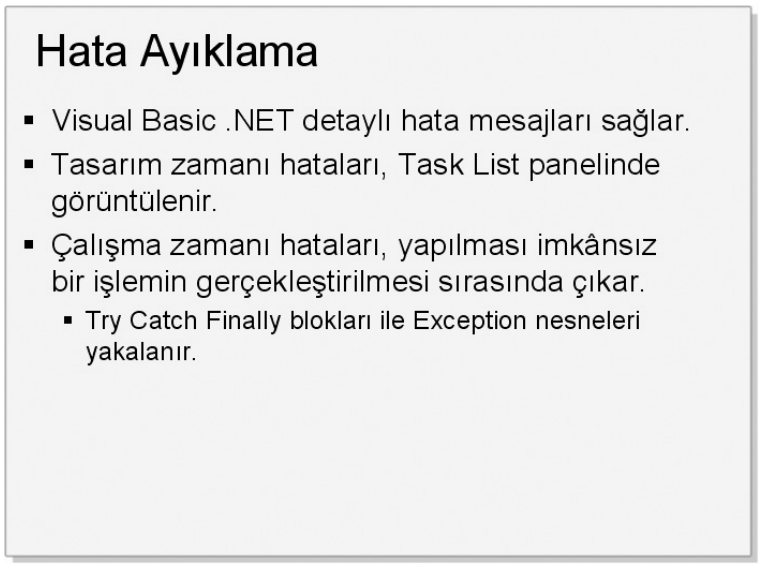
Debug

Kodlar yazıldıktan sonra indislerin doğru kullanılıp kullanılmadığını ve döngülerde mantıksal hatalar yapılıp yapılmadığını kontrol etmenin en kolay yolu, hata ayıklayıcı ile çalışmaktır. Bu örnekte birçok döngü kurulmuş ve filin dört hareket yönündeki engeller kontrol edilmiştir. Tüm hareketlerin işleyişine Debug ile kolayca bakılabilir.

1. **`InputBox`** ile filin koordinatlarının alındığı yere **`BreakPoint`** koyun ve projeyi çalıştırın.
2. **`x`** ve **`y`** koordinatlarına 0 değerini girin. Filin, tahtanın sol üst köşesinde olduğu varsayılır.

   Bu durumda fil hangi yönde ilerleyebilir? Step Into komutu ile kodlar arasında ilerleyin ve hangi döngü içine girdiğini bulun.
3. **`Locals`** panelinde **`i`** değişkeninin değerini izleyin. Aynı panelde tahta dizisinin, o anda kontrol edilen değerine bakın. **`x`** ve **`y`** değerlerinin bir fazlasını alarak, diğer değerin **`True`** ya da **`False`** olduğunu kontrol edin.

Konu 3: Hata Yakalama



Bir uygulama geliştiricisi program yazarken çok çeşitli hatalarla karşılaşabilir. Visual Basic .NET, ortaya çıkan hata durumlarında uygulama geliştiricisine çok detaylı hata mesajları verir. Bu hata mesajları, hataların nerede ve nasıl yapıldığını çok detaylı bir şekilde gösterir. Hataların en ince ayrıntısına kadar işlenmesi, uygulama geliştirmede büyük kolaylık sağlar.

Visual Basic .NET hata mesajları, çalışma zamanı (Run Time) ve tasarım zamanı (Design Time) hataları olarak ayrılabilir.

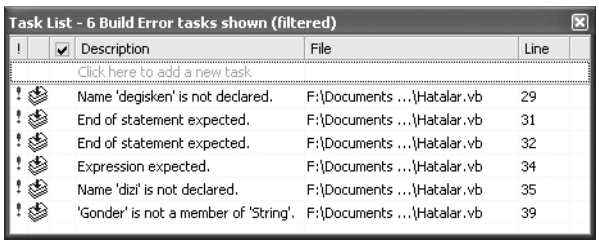
Tasarım zamanı hataları, kodların yazılması sırasında derleyici tarafından bulunan ve Task List panelinde gösterilen hatalardır (Resim 6.8). Task List panelinde hatanın açıklaması, hatanın projenin hangi dosyasında ve dosyanın kaçıncı satırında bulunduğu gösterilir (Resim 6.9). Tasarım zamanı hataları;

- Söz dizimi yanlış kullanıldığında,
- **`Option Explicit On`** durumunda, tanımlanmayan bir değişken kullanıldığında,
- **`Option Strict On`** durumunda, değişken tiplerini Implicit Conversion ile çevirirken meydana gelir.

Çalışma zamanı hataları, uygulama çalışırken yapılması imkansız bir işlemin gerçekleştirilmesi sırasında meydana gelir (Resim 6.10). Örneğin **`InputBox`** metodu ile bir sayının alınması sırasında, kullanıcı **`String`** tipinde bir değer girerse çalışma zamanında bir hata oluşur.

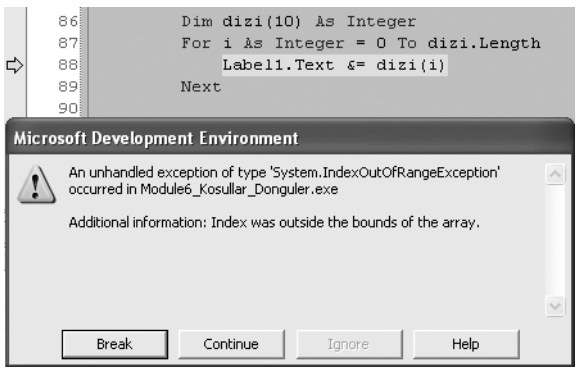
```
29          degisken = 1
30
31          Dim sayi As Integer 5
32          Dim Doğum Tarihi As Date
33
34          Dim dizi() As Integer = {12, 13, }
35          dizi(0) = 5
36
37          Dim mesaj As String
38          mesaj = InputBox("İletmek istediğiniz mesaj:")
39          mesaj.Gonder()
40
```

RESİM 6.8: Tasarım zamanı hataları.



RESİM 6.9: Task List paneli.

```
Dim dizi(10) As Integer
For i As Integer = 0 To dizi.Length
    Label1.Text &= dizi(i)
Next
```

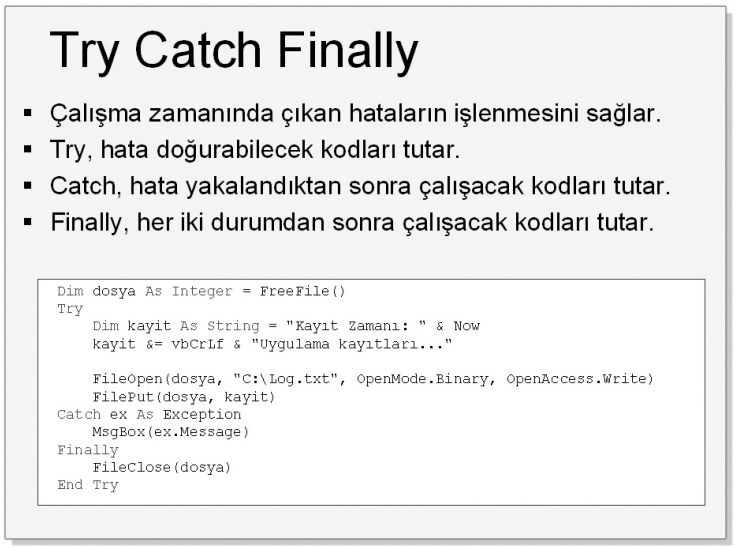


RESİM 6.10: Çalışma zamanı hatası.

Resim 6.10'daki hata mesajı, dizinin büyüklüğünün dışında bir indis verildiğini belirtir.

Visual Basic .NET dilinde uygulama geliştirirken oluşabilecek tüm hatalar .NET Framework altındaki **`Exception`** sınıfları halinde tanımlanır. Örneğin dizinin büyüklüğünden farklı bir indis verildiğinde **`IndexOutOfRangeException`** hatası ortaya çıkar. Tüm hatalar gibi bu hata da **`Exception`** taban sınıfından türetilmiştir.

Try Catch Finally



Çalışma zamanında ortaya çıkan hatalar uygulamanın beklenmedik bir şekilde sonlanmasına neden olur. Uygulamanın devam etmesi için bu hataların yakalanıp işlenmesi gerekir. **`Try Catch Finally`** blokları içinde, çalışma zamanı hataları meydana geldiği durumlarda çalışması istenen kodlar yazılır. **`Try`** bloğu içine, çalışırken hata üretebilecek kodlar yazılırken, **`Catch`** bloğu içine, hata oluştuğunda yapılması gereken işlemler yazılır.

```
Dim sayi As Byte
Dim sonuc As Integer
Try
    sayi = Rnd() * 3
    sonuc = 100 / sayi
    MsgBox("Bölme işlemi başarılı, sonuç: " & sonuc)

Catch ex As Exception
    MsgBox("Bölme işlemi başarısız. Hata Mesajı: " &
ex.Message)
End Try
```

Bu örnekte üretilen rasgele bir sayı ile bölme işlemi yapılıyor. Sayı sıfır değerini aldığında, bölme işlemi hata üretir. Dolayısıyla bu işlem **`Try`** bloğu içine yazılmalıdır. **`Catch`** bloğunda, işlemin başarısız olduğunu belirten bir mesaj yazılır. **`Exception`** nesnesinin **`Message`** özelliği, hatanın oluştuğu zaman üretilen mesajı tutar. **`Exception`** nesnesinin özellikleri **`Catch`** içinde kullanılmayacaksa, tanımlanmasına gerek yoktur.

```
Try

Catch
    Label1.Text = "Exception kullanılmıyor."
End Try
```

Finally bloğunda, **Try Catch** içinde yapılan tüm işlemlerden sonra çalıştırılacak kodlar yazılır. **Finally** bloğunda yazılan kodlar hata meydana gelse de, gelmese de çalıştırılacaktır.

```
' Dosyayı açmak için kullanılan dosya numarası
Dim dosya As Integer = FreeFile()
Try
    Dim kayit As String = "Kayıt Zamanı: " & Now
    kayit &= vbCrLf & "Uygulama kayıtları..."

    FileOpen(dosya, "C:\Log.txt", OpenMode.Binary,
OpenAccess.Write)
    FilePut(dosya, kayit)

Catch ex As Exception
    MsgBox(ex.Message)
Finally
    FileClose(dosya)
End Try
MsgBox("Uygulama akışı buradan devam edecek")
```

Finally bloğunda genellikle, kullanılan kaynaklar serbest bırakılır. Örnekte, bir dosya açılır. Dosya açma veya dosyaya veri yazma işlemlerinde bir hata meydana geldiğinde, **Catch** ifadesinde bu hata yakalanıp ilgili mesaj kullanıcıya gösterilir. **Finally** bloğu her durumda çalışacağı için, dosya kapatma işlemi burada yapılır. Uygulama **End Try** ifadesinden sonra işlemeye devam eder.

Try ve **Catch** içinde uygulamadan çıkılması belirtilse dahi **Finally** bloğu içinde yazılan kodlar çalıştırılır. Ancak **End Try** ifadesinden sonra uygulama sonlanır.

```
Try
    Dim kayit As String = "Kayıt Zamanı: " & Now
    kayit &= vbCrLf & "Uygulama kayıtları..."

    FileOpen(dosya, "C:\Log.txt", OpenMode.Binary,
OpenAccess.Write)
    FilePut(dosya, kayit)
    Exit Sub

Catch ex As Exception
```

```
    MsgBox(ex.Message)
    Exit Sub
Finally
    FileClose(dosya)
    MsgBox("Finally her durumda çalışır")
End Try
MsgBox("Uygulama akışı buraya gelmeyecek.")
```

Lab 1: Şifreleme Algoritması

Bu uygulamada, verilen bir yazı şifrelenerek bir dizi sayıya çevrilir. Bu sayılar, yazıda geçen karakterlerin ASCII kodlarının karışmış bir halidir. Şifreyi çözmek için, şifrelemede izlenen yolların tersi uygulanır.

İPUCU **Şifreleme algoritmalarında yazıyı şifrelerken izlenen yolların geri dönüşü olmalıdır. Örneğin rasgele sayılar kullanılarak şifrelenen bir yazıyı, tekrar rasgele sayılar kullanarak çözülemez.**

Şifreleme

Verilen yazının şifrelenmesi üç etaptan oluşur:

1. Yazının karakterleri ASCII kodlarına çevrilir.
2. Kodlar, gerekiyorsa başlarına 0 konarak, 4 haneli yapılır ve sıralı halde bir `String` değişkeninde tutulur.
3. Sıralı şekilde yazılan kodlar, bir baştan, bir sondan karakter alınarak tekrar düzenlenir.

Örnek: “acf” kelimesinin şifrelenmesi:

1. a c f karakterleri ASCII kodlarına çevrilir.

   a = 97

   c = 99

   f = 102

2. Kodlar başlarına sıfır konarak 4 haneli yapılır.

   0097

   0099

   0102

   Sıralı halde bir `String` değişkenine yazılır.

   009700990102

3. Sayının ortasına kadar, önce baştan, daha sonra sondan rakam alınarak tekrar yazılır. Siyah olarak gösterilen rakamlar, dizinin sonundan alınmıştır.

   0

   0 **2**

   0 **2** 0

   0 **2** 0 **0**

   0 **2** 0 **0** 9

   0 **2** 0 **0** 9 **1**

   0 **2** 0 **0** 9 **1** 7

   0 **2** 0 **0** 9 **1** 7 **0**

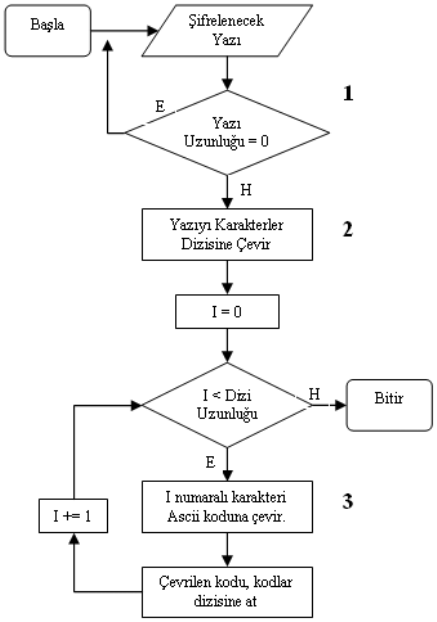
0**2**0**0**9**1**7**0**0
0**2**0**0**9**1**7**0**0**9**
0**2**0**0**9**1**7**0**0**9**0
0**2**0**0**9**1**7**0**0**9**0**9** = Şifre

Projenin Açılması

1. Visual Studio ortamında, **`Sifreleme`** isminde bir Windows projesi açın.
2. Açılan forma **`lblSifre`** isminde bir **`Label`** kontrolü, **`btnSifrele`** isminde bir **`Button`** kontrolü ekleyin. Bu kontroller kullanıcıdan alınan yazının şifrelenip görüntülenmesini sağlayacaktır.
3. Açılan forma **`lblDesifre`** isminde bir **`Label`** kontrolü ve **`btnSifreyiCoz`** isminde bir **`Button`** kontrolü ekleyin. Bu kontroller şifrelenmiş yazının **`lblSifre`** kontrolünden alınarak, şifrelenip görüntülenmesini sağlayacaktır.

DİKKAT **Şifreleme algoritmasının tüm kodları** `btnSifrele` **kontrolünün** `Click` **olayına yazılacaktır.**

ASCII Kodlarına Çevirme



RESİM 6.11: ASCII kodlarına çevirme algoritması.

1. Şifrelenecek yazının girilmesi için gerekli kodu yazın. Bir yazı girilene kadar kullanıcıdan yazı istemek için **`Do Loop Until`** döngüsünü kullanın.

```
Dim yazi As String

' Algoritma 1 - 1
Do
    yazi = InputBox("Şifrelenecek Kelimeyi girin")
Loop Until yazi.Length > 0
```

2. Girilen yazının karakterlerini bir dizide toplamak için **`String`** değişkeninin **`ToCharArray()`** metodunu kullanın.

```
' Algoritma 1 - 2
Dim karakterler() As Char = yazi.ToCharArray()
```

3. Karakterlerin ASCII kodu karşılığını tutmak için **`kodlar`** isminde bir dizi yaratın. Karakterler dizisindeki tüm elemanlar üzerinde işlem yapmak için bir döngü kurun. Karakterler dizisindeki her karakteri **`Asc`** hazır fonksiyonu ile ASCII koduna çevirin.

```
Dim uzunluk As Short = karakterler.Length - 1
Dim kodlar(uzunluk) As String

Dim i As Integer
' kelimedeki her karakterin ASCII kodu alınır
For i = 0 To uzunluk
    ' Algoritma 1 - 3
    kodlar(i) = Asc(karakterler(i))
Next
```

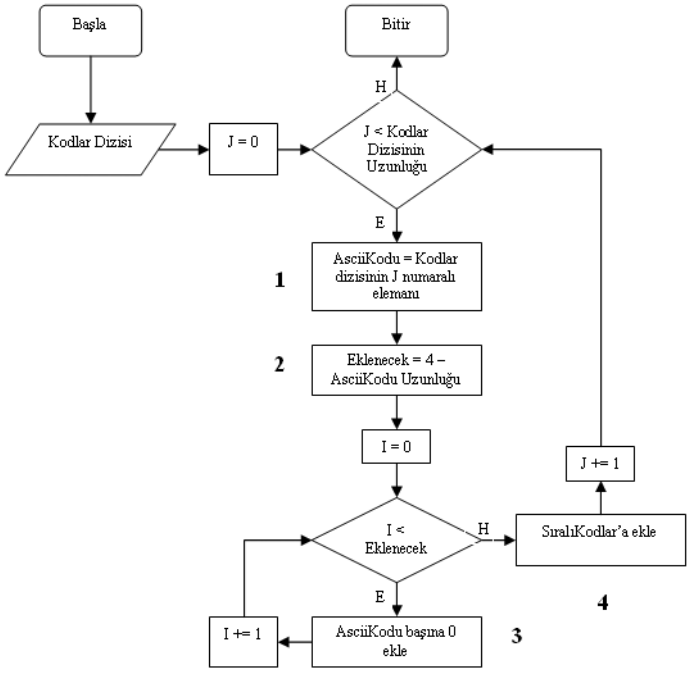
4. Bu algoritma sonunda elde edilen **`kodlar`** dizisi, şifrelenecek olan yazının her karakterinin ASCII kodunu tutar. Bu dizi diğer algoritmanın giriş değeri olarak kullanılacaktır.

Sıralı Kodlara Çevirme

ASCII karakter kodları 0 ile 255 arasında bir değer alır. Dolayısıyla, her kod maksimum üç haneli olacaktır. Şifre oluşturulurken yapılan son düzenlemede kolaylık sağlamak için, bu kodlar 4 haneli yapılır. Daha sonra bu kodlar diziden çekilerek **`SiraliKodlar`** adlı bir **`String`** değişkenine yazılır.

1. Dizideki kodları sıralı bir şekilde tutmak için **`SiraliKodlar`** adlı bir değişken tanımlayın. İlk algoritmadan alınan ASCII kodlarını tutan **`kodlar`** dizisi üzerinde bir döngü kurun.

```
Dim SiraliKodlar As String
Dim j As Short = 0
While j <= kodlar.Length – 1

    j += 1
End While
```



RESİM 6.12: Sıralı kodlara çevirme algoritması.

DİKKAT **2 – 4 etaplarında yazılacak tüm kodlar `While` döngüsünün içine yazılacaktır.**

Bu döngüde kullanılacak ASCII kodunu bir değişkene atan kodu yazın.

```
' Alogritma 2 - 1
Dim AsciiKodu As String = kodlar(j)
```

2. **`AsciiKodu`** değişkeninde tutulan kodun 4 haneli hale getirilmesi için kaç tane sıfır eklenmesi gerektiğini bulun. Eklenecek sıfırların sayısı, 4 – **`AsciiKodu`** değişkeninin uzunluğu kadardır. Örneğin 192 kodlu bir değişkene eklenmesi gerenken sıfır sayısı 4 – 3 = 1 tanedir.

```
Dim eklenecek As Byte = 4 - AsciiKodu.Length
```

3. Eklenecek sayı kadar çalışacak bir döngü içinde, sıfır ekleme işlemini yapın.

```
For i = 0 To eklenecek - 1
    ' Alogritma 2 - 3
    AsciiKodu = AsciiKodu.Insert(0, 0)
Next
```

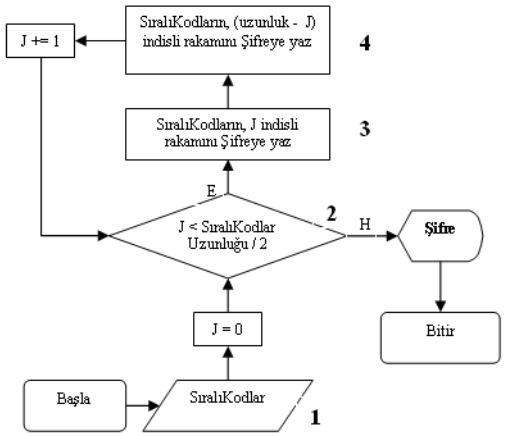
4. Düzenlenmiş **`AsciiKodu`** değerini **`SıralıKodlar`** değişkenine yazın ve sayacı bir artırarak diğer ASCII koduna geçin.

```
' Alogritma 2 - 4
SiraliKodlar &= AsciiKodu
j += 1
```

Algoritma sonunda ortaya çıkan değer, karakterlerin 4 haneli ASCII kodlarını tutan bir **`String`** değişkenidir. Bu değişken, diğer algoritmada tekrar düzenlenmek üzere kullanılacaktır.

Şifrenin Oluşturulması

Bir önceki algoritmada elde edilen **`SıralıKodlar`** değişkeni halen istenen şifreli yazı değildir. Çünkü 4 haneli kodlar sıralı bir şekilde durur ve kolayca çözülebilir. Şifrenin ilk bakışta anlaşılmasını daha da zorlaştırmak için, sıralanmış kodlar biraz daha karıştırılır.



RESİM 6.13: Şifrenin oluşturulması.

1. Döngüde kullanılacak **`j`** sayacını sıfırlayın ve şifrenin tutulacağı değişkeni tanımlayın.

```
'Algoritma 3 - 1
j = 0
Dim Sifre As String
```

2. **`SıralıKodlar`** değişkeni üzerinde yapılacak işlem sayısı, bir baştan, bir sondan ilerlendiği için, değişkenin uzunluğunun yarısı kadardır. Sayacın bu uzunluğa kadar tanımlı olan bir döngü oluşturun.

```
'Algoritma 3 - 2
Do While j < SiraliKodlar.Length / 2

    j += 1
Loop
```

DİKKAT **3 – 4 etaplarında yazılacak tüm kodlar** `Do While` **döngüsünün içine yazılacaktır.**

3. Şifreye, **`SıralıKodlar`**'ın **`j`** indisli karakterini ekleyin.

```
'Algoritma 3 - 3
Sifre &= Mid(SiraliKodlar, j + 1, 1)
```

4. Şifreye, **`SıralıKodlar`**'ın uzunluk – **`j`** indisli karakterini ekleyin.

```
'Algoritma 3 - 4
Sifre &= Mid(SiraliKodlar, SiraliKodlar.Length - j, 1)
```

5. Sonuç olarak çıkan şifre, girilen yazının ASCII kodlarının karışık düzende tutulması ile oluşturulur.

```
lblSifre.Text = "Girilen yazının şifrelenmiş hali: " & Sifre
```

Şifreyi Çözmek

Şifreleme algoritması kullanılarak oluşturulan şifrenin çözülmesi, izlenen yolların tersi uygulanarak gerçekleştirilir. Deşifre algoritması iki etaptan oluşur.

1. Bir baştan bir sondan karakter alınarak şifrelenen ASCII kodları, sıralı kodlar haline dönüştürülür.
2. 4 haneli olarak duran sıralı kodlar, karakterlere çevrilir. Karakterler ardı ardına konarak deşifre işlemi gerçekleştirilir.

Örnek: "acf" kelimesinin şifrelenmiş hali 020091700909 şeklindedir. Bu kelime şifrelenirken, karakterleri 4 haneli ASCII kodlarına çevrilmiş ve bu kodların rakamlarının sırası değiştirilmişti. Bu şifrenin önce 4 haneli sıralı kodlar haline getirilmesi için, şifrelenen yöntemin tersi işlenir. Sırayla okunan rakamlar önce başa, daha sonra sona yazılır.

```
Şifre: 0 2 0 0 9 1 7 0 0 9 0 9
```

Sıralı kodlara çevrim:

Sıranın ilk yarısı	Sıranın son yarısı
0	**2**
0 **0**	**0** 2
0 0 **9**	**1** 0 2
0 0 9 **7**	**0** 1 0 2
0 0 9 7 **0**	**9** 0 1 0 2
0 0 9 7 0 **0**	**9** 9 0 1 0 2

Sonuç olarak elde edilen sıralı ASCII kodları, sıranın ilk yarısı ve son yarısının birleşimi olur: 0097 0099 0102

DİKKAT **Sıranın ilk yarısı oluşturulurken, rakamlar sona eklenir. Ancak sıranın son yarısı oluşturulurken rakamlar başa eklenir.**

Bu 4 haneli kodlar **`String`** değerinden **`Integer`** değerine çevrilir ve bu değerlerin karşılığı olan karakterler yazılır.

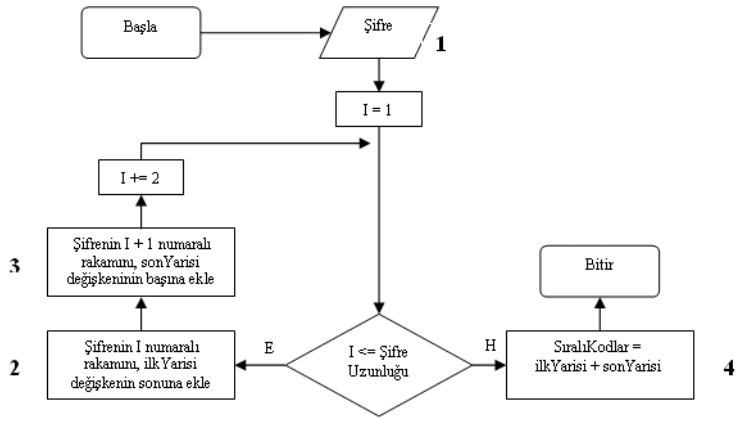
```
0097 → 97   → a
0099 → 99   → c
0102 → 102  → f
```

Elde edilen karakterler birleştirildiğinde şifre çözülmüş olur: "acf"

DİKKAT **Deşifre algoritmasının tüm kodları `btnSifreyiCoz` kontrolünün `Click` olayına yazılacaktır.**

Şifreyi Sıralı Kodlara Dönüştürmek

Bu algoritma verilen şifreyi sıralı ASCII kodlarına dönüştürür.



RESİM 6.14: Şifrenin sıralı kodlara dönüştürülmesi.

1. Şifreyi **`lblSifre`** etiketinden alın ve sıralı kodların oluşturulması için gereken değişkenleri tanımlayın.

```
' Algoritma 1 - 1
Dim Sifre As String = lblSifre.Text

Dim SiraliKodlar As String

Dim i As Short

' Başa ve sona rakam ekleneceği için
' değişkenlere başlangıç değerleri verilir
Dim ilkYarisi As String = ""
Dim sonYarisi As String = ""
```

2. Şifrenin tüm elemanları üzerinde bir döngü kurarak, sıralı kodların ilk ve son yarısını oluşturun. Kodların ilk yarısı, şifrenin tek haneli rakamları ile; kodların son yarısı, şifrenin çift haneli rakamları ile oluşturulur. Dolayısıyla döngünün sayacı ikişer ikişer artmalıdır. Şifrenin **i** indisli rakamını sıranın ilk yarısına, yanındaki rakamı (**i + 1** indisli rakamı) sıranın son yarısına ekleyen kodları yazın.

```
For i = 1 To Sifre.Length Step 2
    ' Algoritma 1 - 2
    ' Sıranın ilk yarısının sonuna rakam eklenir.
    ilkYarisi &= Mid(Sifre, i, 1)

    ' Algoritma 1 - 3
    ' Sıranın son yarısının başına rakam eklenir.
    sonYarisi = sonYarisi.Insert(0, Mid(Sifre, i + 1, 1))
Next
```

3. Sıralı kodların ilk yarısı ve son yarısı birleştirilir. Elde edilen değer, 4 haneli ASCII kodlarının sırayla tutulduğu bir **String** değeridir.

```
' Algoritma 1 - 4
SiraliKodlar = ilkYarisi & sonYarisi
```

Sıralı Kodların Okunması

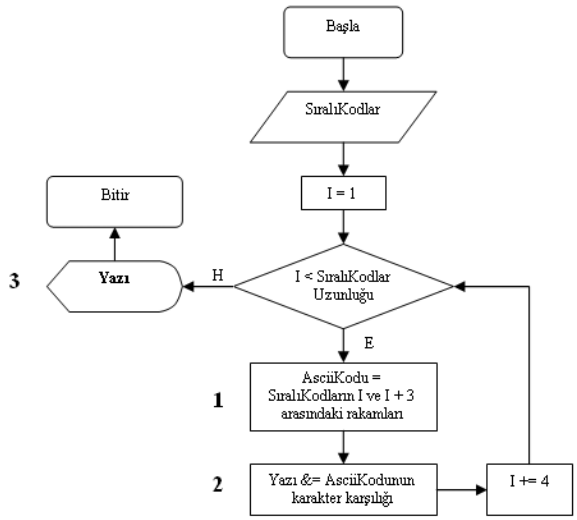
İlk algoritmada elde edilen sıralı ASCII kodları, bu algoritmada okunarak karakterlere çevrilir ve şifre çözülmüş olur.

1. Şifre çözüldüğü zamanki değerinin tutulacağı değişkeni tanımlayın ve sıralı kodlar üzerinde bir döngü kurun. Sıralı kodlar 4 haneli kodlardan oluştuğu için, döngüde bir seferde 4 rakam alınacaktır. Bunun için döngünün sayacı 4 artırılmalıdır.

```
i = 0
Dim yazi As String = ""

Do While i < SiraliKodlar.Length

    i += 4
Loop
```

DİKKAT **2 – 3 etaplarında yazılacak tüm kodlar** `Do While` **döngüsünün içine yazılacaktır.**

2. Döngü her seferinde bir ASCII kodu alır. Bu değeri tutan bir değişken tanımlanır ve sıralı kodlardan 4 haneli rakam bu değişkene atanır.



RESİM 6.15: Sıralı kodların okunması.

```
Dim AsciiKodu As Integer

' Algoritma 2 - 1
AsciiKodu = CInt(Mid(SiraliKodlar, i + 1, 4))
```

3. Alınan ASCII kodunun karakter karşılığı bulunur ve **`yazi`** değişkenine eklenir.

```
' Algoritma 2 - 2
yazi &= Chr(AsciiKodu)
```

4. Döngü sonunda elde edilen değer **`lblDesifre`** etiketine yazılır.

```
' Algoritma 2 - 3
lblDesifre.Text = yazi
```

Lab 2: Sıralama Algoritması

Bu algoritma, bir dizinin elemanlarını küçükten büyüğe sıralar.

Dizinin Doldurulması

1. **Siralama** isimli bir Windows projesi açın.
2. Form üzerine biri **lbSirasiz**, diğeri **lbSirali** isimli iki **ListBox** ekleyin. Bu kontroller dizinin sırasız ve sıralı halini listeler.
3. **btnListele** ve **btnSirala** isimli iki **Button** ekleyin.
4. Formun kod tarafına geçin ve bir dizi tanımlayın. Bu dizi bir çok yordamın içinde kullanılacağı için global olarak tanımlanır.

```
Dim dizi(4) As String
```

5. **btnListele** düğmesinin **Click** olayına, diziyi karışık bir şekilde isimlerle dolduran kodları yazın:

```
dizi(0) = "Enis"
dizi(1) = "Engin"
dizi(2) = "Tamer"
dizi(3) = "Kadir"
dizi(4) = "Fulya"

Dim i As Integer
For i = 0 To dizi.Length - 1
    lbSirasiz.Items.Add(dizi(i))
Next
```

Dizinin Sıralanması

Sıralama algoritması, dizi üzerinde bir döngü kurar ve sırayla dizinin bir elemanı seçilir. Bu eleman için bir başka döngü kurulur ve seçilen elemanın indisine kadar olan tüm elemanlarla bir karşılaştırma yapılır. Küçük olan sıranın başına konmak için büyük olan ile yer değiştirilir.

Örnek

1. Dizinin 2. elemanı seçilir: "**Engin**"

   Dizinin 2. indisine kadar olan elemanlarla karşılaştırılır. "**Engin**" değeri alfabetik sırada "**Enis**" değerinden küçük olduğu için, bu iki değer yer değiştirilir.

   Sıra, **Engin Enis Tamer Kadir Fulya** olur.
2. Dizinin 3. elemanı seçilir: "**Tamer**"

Dizinin 3. indisine kadar olan elemanlarla karşılaştırılır. "**Tamer**" değeri, "**Enis**" ve "**Engin**" değerlerinden büyük olduğu için sıralama değişmez.

3. Dizinin 4. elemanı seçilir: "**Kadir**"

   Dizinin 4. indisine kadar olan elemanlarla karşılaştırılır. "**Kadir**" < "**Tamer**" olduğu için bu iki değer yer değiştirilir.

   Sıra, **Engin Enis Kadir Tamer Fulya** olur.

   "**Kadir**" değeri, "**Enis**" ve "**Engin**" değerlerinden büyük olduğu için sıralama değişmez.

4. Dizinin 5. elemanı seçilir: "**Fulya**"

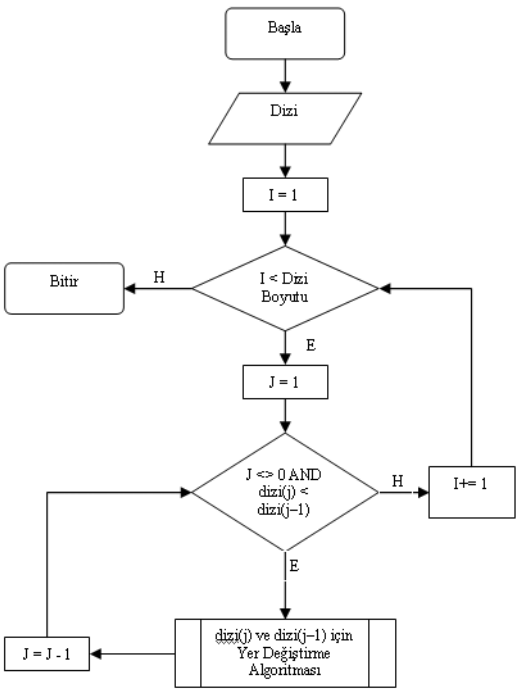
   Dizinin 5. indisine kadar olan elemanlarla karşılaştırılır. "**Fulya**" < "**Tamer**" olduğu için bu iki değer yer değiştirilir.

   Sıra, **Engin Enis Kadir Fulya Tamer** olur.

   "**Fulya**" < "**Kadir**" olduğu için bu iki değer yer değiştirilir.

   Sıra, **Engin Enis Fulya Kadir Tamer** olur.

Dizideki tüm değerler kontrol edildiği için algoritmadan çıkılır.



RESİM 6.16: Sıralama algoritması.

1. **btnSirala** düğmesinin **Click** olayına, dizi üzerinde bir döngü tanımlayın. Bu döngü dizinin (1 indisli) ikinci elemanından başlayarak dizi sonuna kadar devam edecektir. Daha sonra bu döngü içine başka bir döngü daha yazın. Bu döngü, ilk döngünün sayacından başlar ve sıfır

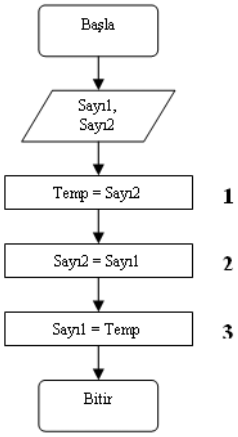
olana kadar devam eder. İkinci döngünün amacı, ilk döngüde seçilen elemanı, dizinin başına kadar olan elemanlarla karşılaştırmaktır.

```
Dim i As Integer

For i = 1 To dizi.Length - 1
    Dim j As Integer = i
    While j AndAlso dizi(j) < dizi(j - 1)
    ' Yer değiştirme Algoritması

        j -= 1
    End While
Next
```

While döngüsü, **j** değeri sıfır olana kadar ve dizinin kontrol edilen değeri bir önceki değerden küçük olana kadar devam eder. Burada, dizi elemanlarının kontrolünün sadece bir defa (bir önceki eleman ile) yapıldığı düşünülebilir. Ancak küçük eleman yer değiştirildiğinde **j** değeri bir düşürülür. Döngü tekrar çalıştığı zaman, aynı eleman bu sefer dizinin kalan elemanlarıyla karşılaştırılır.

AndAlso operatörü, **j** değerinin sıfır olması durumunda diğer kontrolün yapılmaması için kullanılır. Diğer kontrolde **dizi(j – 1)** ifadesi, negatif indisli değere ulaşılmak istendiği için hata mesajı verir.

2. Yer değiştirme algoritması, bir değişkenin değerinin geçici bir yerde tutulması ile gerçekleştirilir.



RESİM 6.17: Yer değiştirme algoritması.

Sıralama algoritmasında dizinin **j** ve **j - 1** indisli değerleri yer değiştirilir. **While** döngüsü içinde "**Yer değiştirme Algoritması**" yazan yorum satırını kaldırın ve yerine algoritma kodlarını yazın.

```
' Yer değiştirme
Dim temp As String = dizi(j - 1)
dizi(j - 1) = dizi(j)
dizi(j) = temp
```

3. **lbSirali** liste kutusunda dizinin yeni sırasını görüntüleyen kodları yazın.

```
For t As Integer = 0 To dizi.Length - 1
    lbSirali.Items.Add(dizi(t))
Next
```

Lab 3: Arama Algoritması

Arama algoritmaları, sıralı bir liste üzerinde bir değerin aranması için kullanılır. Karışık sırada olan bir listede yapılan arama, ancak listenin başından sonuna kadar tüm elemanlarının kontrol edilmesi ile gerçekleşir. Bu yöntem büyük dizilerde performansı düşürür. Belirli bir sırada olan dizilerde ise daha hızlı arama yöntemleri kullanılmalıdır. Bu labda ikili arama yöntemi (Binary Search) incelenecektir.

DİKKAT **İkili arama yöntemi sadece sıralı bir dizi üzerinde uygulanabilir. Ya da eldeki dizi öncelikle sıralanır.**

NOT **İkili arama yönteminde büyük-küçük kıyaslaması yapıldığından, dizinin sıralı olması gerekir.**

Dizinin Sıralanması

Arama algoritması sıralı bir dizi üzerinde çalışacağı için, dizi oluşturulduktan sonra sıralanması gerekir.

1. **`IkiliArama`** isminde bir Windows projesi açın.
2. Forma **`btnAra`** isimli bir **`Button`** ve **`lbDizi`** adlı bir **`ListBox`** ekleyin.
3. Kod sayfasına geçin ve global bir dizi tanımlayın.

```
Dim dizi(10) As Integer
```

4. Formun **`Load`** olayına diziyi rasgele sayılarla doldurmak için gereken kodları yazın.

```
For i As Integer = 0 To 10
    dizi(i) = Rnd() * 1000
Next
```

5. Diziyi sıralayın ve değerlerini **`lbDizi`** adlı listeye ekleyin.

```
Array.Sort(array)

For i As Integer = 0 To 10
    lbDizi.Items.Add(dizi(i))
Next
```

Arama Algoritması

İkili arama algoritması, dizi üzerinde aranacak değeri önce sıranın ortasındaki değerle karşılaştırır. Dizi küçükten büyüğe doğru sıralı olduğu için, aranan değer ortadaki değerden küçükse, arama dizinin ilk yarısında devam eder. Dizinin diğer yarısı aranan değerden büyük değerler içerdiği için, aramaya dahil edilmez.

Örnek

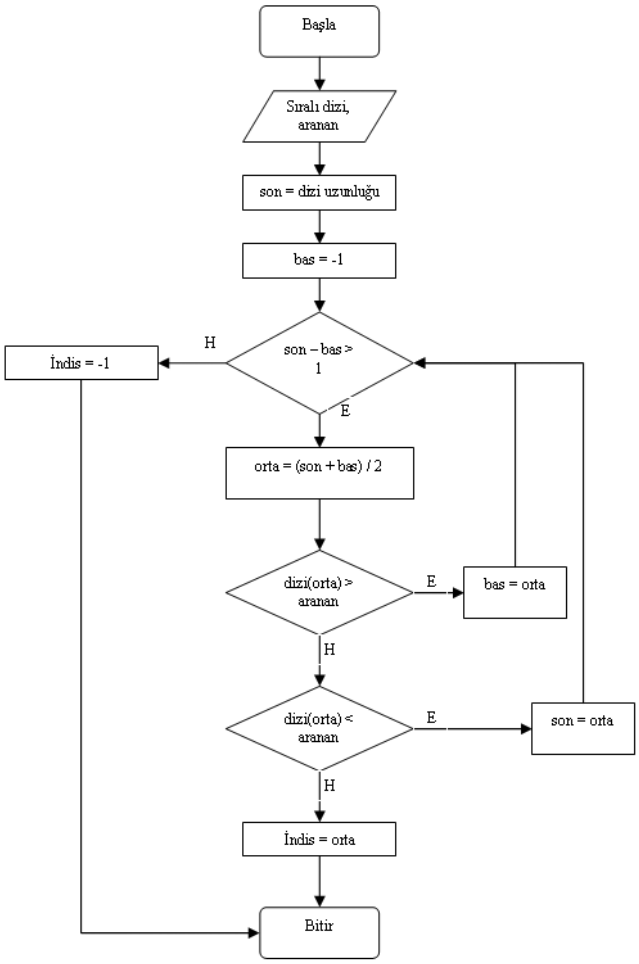
Küçükten büyüğe sıralı dizi üzerinde 9 değeri aranacaktır.

```
Dizi: 1 2 4 7 9 10 12 18
```

1. **Son**, **Baş** ve **Orta** değişkenleri tanımlanır: **Son** değeri dizinin son elemanının indisini, **Baş** değeri dizinin ilk elemanının indisini, **Orta** değeri ise **Son + Baş / 2** değerini alır. **Orta** değeri virgüllü bir değer alırsa tamsayıya çevrilir.
2. Başlangıç olarak **Baş -1**, son dizi uzunluğu değerini alır.

```
Baş = -1
Son =  8
Orta = (8 – 1) / 2 = 3
```

3. Dizinin **Orta** indisli değeri alınır. **Dizi(3) = 7**



RESİM 6.18: Arama algoritması.

4. Aranan 9 değeri, 7'den büyük olduğu için, dizinin son yarısında aranır. **Baş** değişkenine **Orta** değeri verilirse, dizinin başlangıç indisi değiştirilir, böylece aramalar dizinin son yarısında gerçekleşmiş olur.

```
Dizi: 9 10 12 18
Baş = 3
Son = 8
Orta = (8 + 3) / 2 = 5
```

5. Dizinin **Orta** indisli değeri alınır. Dizi(5) = 10
6. 9 değeri, 10'dan küçük olduğu için, kalan dizinin ilk yarısında aranır.

```
Dizi: 9 10
Baş = 3
Son = 5
Orta = (5 + 3) / 2 = 4
```

7. Dizinin ortasındaki değer alınır: 9
8. Böylece 9 değerinin indisi **Orta** değeri olur.

Kodlar

Arama algoritmasının kodları **btnAra** düğmesinin **Click** olayına yazılacaktır.

1. Algoritma için gerekli **bas**, **son** ve **orta** değişkenlerini tanımlayın ve başlangıç değerlerini verin. Aranan değerin indisini tutmak için de bir değişken tanımlayın.

```
Dim son As Integer = dizi.Length
Dim bas As Integer = -1
Dim orta As Integer
Dim indis As Integer
```

2. Kullanıcıdan aranacak değeri girmesini isteyin.

```
Dim hedef As Integer = InputBox("Aranan değeri girin:")
```

3. Dizide aranacak değer kalmadığı zaman çıkan bir döngü kurun. **son** ve **bas** değerleri arasındaki fark 1'e düştüğünde, dizide aranacak değer kalmamıştır.

```
While son - bas > 1

End While
```

4. **While** döngüsü içine, dizinin **orta** indisli değerini alan ve bu değeri aranan değerle karşılaştıran kodları yazın.

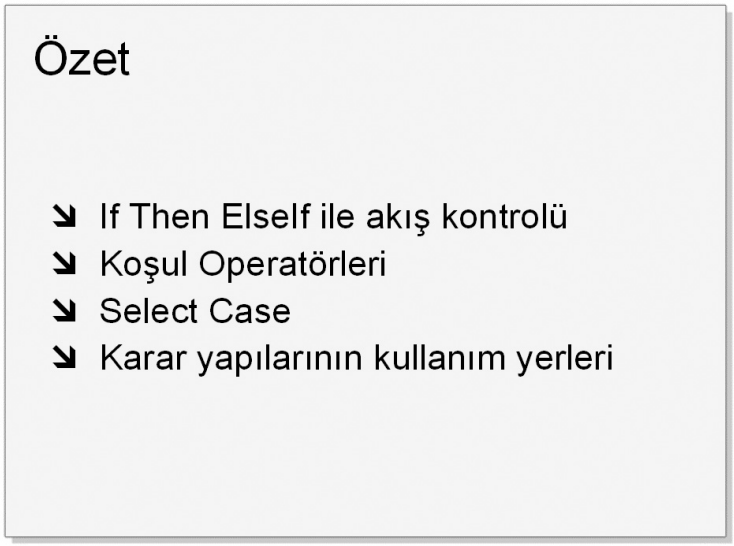
```
orta = (son + bas) / 2
If dizi(orta) > hedef Then
    son = orta
ElseIf dizi(orta) < hedef Then
    bas = orta
Else
    indis = orta
    MsgBox("İndis: " & indis)
    Exit Sub
End If
```

Eğer dizinin ortasındaki değer aranan değerse, indis bulunmuş demektir. **orta** değişkeni kontrolün yapıldığı değerin indisini tuttuğu için, sonuç **orta** değeri olur ve yordamdan çıkılır.

5. Eğer istenen değer bulunamadan döngüden çıkılırsa, **indis -1** değerini alır. **End While** ifadesinden sonra, aranan değerin bulunamadığını belirten kodu yazın.

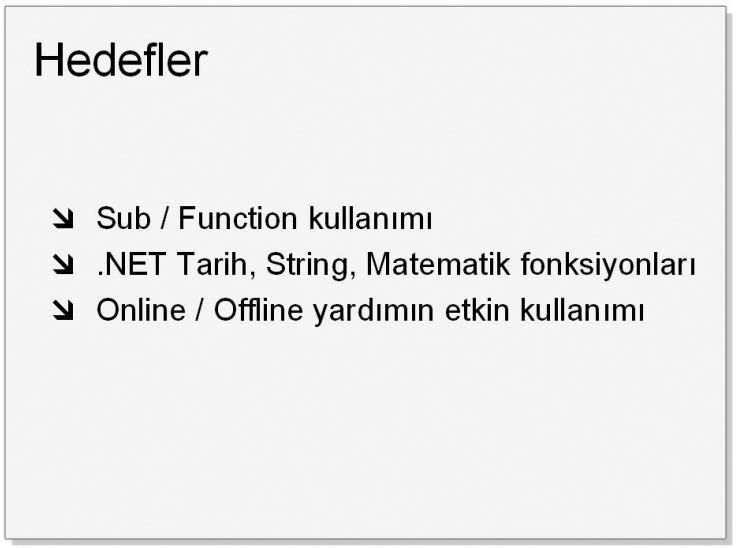
```
indis = -1
MsgBox("İndis: " & indis & " Aranan değer bulunamadı")
```

Modül Sonu Soruları & Alıştırmalar



1. `If` ifadesi hangi veri tipini kontrol eder?
2. `If - Select` karar yapılarının farkları nelerdir?
3. `True Or False And False`
   (`True Or False`) `And False`
   ifadeleri hangi değerleri döndürür?
4. `For Next` döngüsünün `Do` döngülerine göre avantajı nedir?
5. `While` ve `Until` arasındaki fark nedir?
6. 4 boyutlu bir dizinin tüm elemanlarını doldurmak için, dizi üzerinde kaç tane döngü kurulmalıdır?
7. Uygulamaları derledikten sonra hatalar hangi yollarla görülebilir?
8. `Finally` bloğundaki kodlar ne zaman çalışır?
9. Sıralı arama yönteminde (Linear Search) dizinin elemanlarının sıralı olması gerekli midir?
10. Sıralı arama yöntemini bir dizi üzerinde uygulayın.

Modül 7: Fonksiyonlar ve Yordamlar



Uygulama geliştirirken, bir işlemin birçok yerde kullanıldığı zamanlar olur. Bu gibi durumlarda bir kere yazılan kodlar, farklı yerlerde tekrar yazılır. Uygulama üzerinde bir değişiklik yapılmak istenirse, tekrar yazılan kodların tek tek bulunup değiştirilmesi gerekir. Böylece hem uygulamanın yazımı zorlaşır, hem de değişik yapmak giderek imkansız hale gelir. Bu problemler, birçok yerde yapılması istenen işlemlerin fonksiyonlar ve yordamlar içinde yazılması ile çözülür. Sadece fonksiyon ve yordamların isimleri kullanılarak, istenen yerlerde kodlar çalıştırılır.

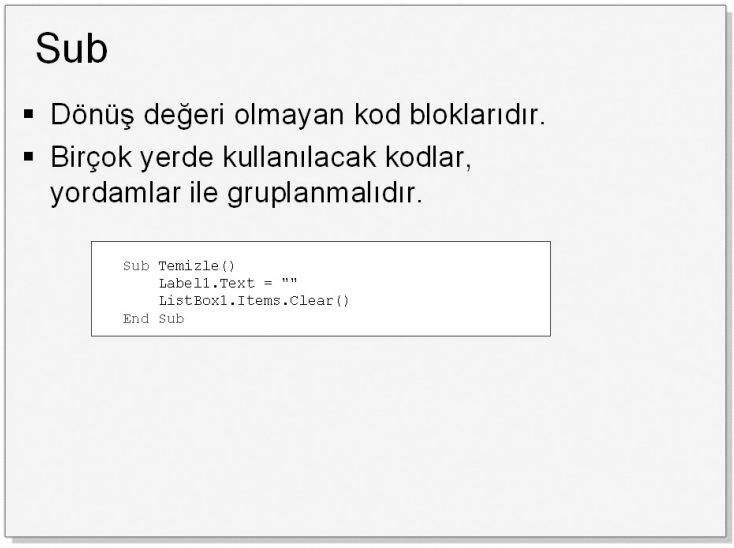
Yapılan işlemin sonucunda oluşan değer isteniyorsa fonksiyonlar kullanılır. Örneğin, veritabanına yeni bir kullanıcı ekledikten sonra, kullanıcının ID numarası isteniyorsa fonksiyon kullanılmalıdır. Eğer yapılan işlemlerin sonunda bir değer döndürülmüyorsa yordamlar kullanılır. Örneğin bir `ComboBox` kontrolüne öğe ekleme işlemi yordam içine yazılabilir.

.NET çatısındaki nesnelerin birçok fonksiyon ve yordamları vardır. Tüm fonksiyon ve yordamların kaç parametre aldığı, geriye dönüş değerinin ne olduğu, hangi nesneye ait oldukları ezberlenemez. Dolayısıyla Visual Studio yardımının kullanılması kaçınılmazdır.

Bu modül tamamlandıktan sonra;

- Yordam ve fonksiyon kullanarak kodlarınızın yönetilebilirliğini ve esnekliğini artıracak,
- Fonksiyon ve yordamların farklarını ayırt edebilecek,
- .NET çatısındaki tarih ve zaman, matematik, String fonksiyonlarını tanıyacak,
- Offline ve Online yardımı etkin bir şekilde kullanabileceksiniz.

Konu 1: Sub



Sub yordamları dönüş değeri olmayan kod bloklarıdır. Bu kodlar **Sub** ve **End Sub** ifadeleri arasına yazılır.

```
Sub YordamIsmi()

End Sub
```

Uygulama içinde birçok yerde çalışacak olan kodlar **Sub** yordamı içinde yazılır. Bu kodlar, içine yazıldıkları yordamın ismi ile çağırılarak, istenen yerde tekrar çalıştırılabilir. Örneğin, bir uygulama başlarken form üzerindeki kontrollerin temizlenmesi gerekiyorsa, bu kodları bir daha yazmamak için yordam kullanılabilir.

```
Sub Temizle()
    Label1.Text = ""
    ListBox1.Items.Clear()
End Sub
```

Yordamı tanımlarken parantezler içine, alabileceği parametreler yazılır. Eğer yordam parametre almıyorsa parantezlerin içi boş bırakılır.

```
Sub YazilimUrunleriEkle()
    ComboBox1.Items.Add("Yazılım Uzmanlığı")
    ComboBox1.Items.Add("Yazılım Mühendisliği")
    Label1.Text = "Yazılım paketleri eklendi..."
End Sub
```

Yordamlar, tanımlandıktan sonra başka bir yordam veya fonksiyon içinde kullanılır. Yordamı kullanmak için, gerekli yere isminin yazılması yeterlidir. Ayrıca **Call** ifadesi de tercihe bağlı olarak kullanılabilir.

```
Sub DersleriListele()
    Select Case ComboBox1.SelectedIndex
        Case 0
            Call Temizle()

            ListBox1.Items.Add("Access - İlişkisel
Veritabanları")
            ListBox1.Items.Add("Programlamaya Giriş Ve
Algoritma")
            ListBox1.Items.Add(".NET Framework")
            ListBox1.Items.Add("VB.NET ile Windows Tabanlı
Programlama")
            ListBox1.Items.Add("ASP.NET ile Web Tabanlı
Programlama")

            Label1.Text = "Yazılım Uzmanlığı dersleri
yüklendi."
        Case 1
            Call Temizle()

            ListBox1.Items.Add("SQL Server Veritabanı
Yönetimi")
            ListBox1.Items.Add("Visual Studio .NET ile
Uygulama Geliştirme")
            ListBox1.Items.Add("ADO.NET ile Veri Yönetimi ve
XML")
            ListBox1.Items.Add("XML Web Services, .NET
Remoting ve COM+")
            ListBox1.Items.Add("Proje Yönetimi")

            Label1.Text = "Yazılım Mühendisliği dersleri
yüklendi."
        Case Else
            Temizle()
            Label1.Text = "Yazılım paketi seçiniz."
    End Select
End Sub
```

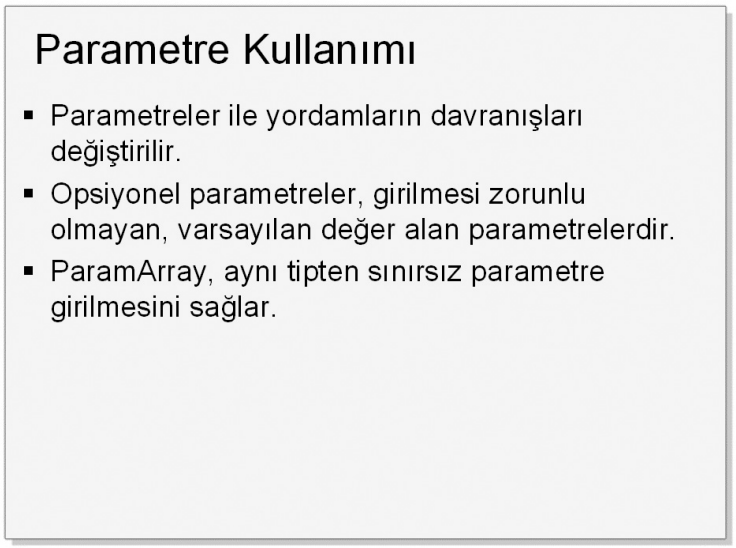
Burada **ComboBox** kontrolünden seçilen değerin kontrolün indisi üzerinden yapılması, **YazilimUrunleriEkle** yordamında eklenen elemanların sırası değişirse problem yaratır. Liste kutusuna eklenen dersler yanlış paketlerde gözükür. Ancak **ComboBox** kontrolünün seçili metni üzerinden kontrol yapılırsa da, eklenen isimler değiştiği zaman bir problem ortaya çıkar. Bu durumda iki

yordamın birbirine bağımlılığı görülür. Bu örnekte, bir yordamda değişiklik yapıldığı zaman diğer yordamın çalışma şekli de kontrol edilmelidir.

Label ve **ListBox** kontrollerini temizleyen kodlar sadece iki satır olduğu için **Temizle** yordamında yazılmayabilirdi. Ancak bu kodlar **DersleriListele** yordamında üç defa kullanıldığı için, her değişiklikte kodun yazıldığı üç yer bulunup gerekli düzeltmeler yapılacaktı. Örneğin, temizleme işlemi, liste kutusunda "**Dersler**" metni gözükecek şekilde değiştirebilir. Bu durumda, değişikliği sadece **Temizle** yordamında yapmak yeterli olur.

```
Sub Temizle()
    Label1.Text = ""
    ListBox1.Items.Clear()
    ListBox1.Items.Add("Dersler: ")
End Sub
```

Parametre Kullanımı



Yordamların bazı değerlere göre farklı işlem yapması istenebilir. İşlemin bağlı olduğu bu değerlere parametre veya argüman denir. Yordamlar parametre alacak şekilde tanımlanıp, çağırıldıkları sırada istedikleri parametreler verilerek kullanılır.

```
Sub YordamIsmi(Parametre1 As VeriTipi, Parametre2 As
VeriTipi, …)

End Sub
```

Örneğin, uygulamanın birçok yerinde kullanıcıya bilgi vermek amaçlı mesaj kutuları kullanılır. Eğer bu mesajlar bir yordam içine yazılırsa, daha sonra mesajlar bir **`Label`** üzerinde gösterilecek şekilde kodu değiştirmek kolay olacaktır. Yordamın göstereceği mesajların parametre olarak verilmesi gerekir.

```
Sub MesajGoster(ByVal mesaj As String)
    Label1.Text = mesaj
End Sub

Sub Yordam1()
    '...
    MesajGoster("1. Yordam içinden çağılır.")
End Sub

Sub Yordam2()
    '...
```

```
    MesajGoster("2. Yordam içinden çağılır.")
End Sub

Sub Yordam3()
    '...
    MesajGoster("3. Yordam içinden çağılır.")
End Sub
```

Yordamları çağırırken tüm parametrelerin belirtilen veri tipinde verilmesi gerekir. Farklı tipte verilen parametreler Implicit Conversion ile ilgili veri tipine çevrilir.

Yordamları tanımlarken parametrelerin isimleri ve veri tipleri belirtilmelidir. Ayrıca parametrelerin değer tipi olarak mı, yoksa referans tipi olarak mı geçileceği belirtilmelidir. **ByVal** olarak geçilen parametrelerin değerleri kullanılabilir, ancak değiştirilemez. **ByRef**, parametrelerin hafızadaki adreslerine ulaşmayı sağlar. Dolayısıyla parametrelerin değerleri değiştirilebilir. **ByVal** ve **ByRef** arasındaki farklar Modül 8'de detaylı olarak incelenecektir.

Yordamlar çağırıldıklarında, kodlar **End Sub** ifadesi görülene kadar çalıştırılır. Yordamın normal akışından çıkılmak istenirse **Exit Sub** veya **Return** ifadeleri kullanılır.

```
Sub MusteriBilgisi(ByVal MusteriId As Integer)
    If Not MusteriId > 0 Then
        Return
    End If

    ' MusteriId değerine göre
    ' müşteri bilgileri veritabanından çekilir.
End Sub
```

Return ifadesi **Exit Sub** ile aynı işlevi görür.

```
Sub MusteriBilgisi(ByVal MusteriId As Integer)
    If Not MusteriId > 0 Then
        Exit Sub
    End If
End Sub
```

Parametre olarak diziler kullanıldığında bu dizilerin büyüklükleri verilmez. Fakat parantezler kullanılarak, verilen parametrenin dizi olduğu belirtilmelidir.

```
Sub MatrisTopla(ByVal matris1(,) As Integer, ByVal
matris2(,) As Integer)
    Dim x As Integer = matris1.GetLength(0)
    Dim y As Integer = matris1.GetLength(1)

    If x <> matris2.GetLength(0) OrElse _
```

```
        y <> matris2.GetLength(1) Then
        MsgBox("Matris boyutlarının büyüklükleri birbiriyle
aynı olmalıdır.")
        Exit Sub
    End If

    Dim sonuc(x - 1, y - 1) As Integer

    For i As Integer = 0 To x - 1
        For j As Integer = 0 To y - 1
            sonuc(i, j) = matris1(i, j) + matris2(i, j)
        Next
    Next
End Sub
```

Diziler yordamlara parametre olarak geçilirken, sadece isimleri verilir.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Dim m1(,) As Integer = {{1, 3, 5}, {7, 9, 11}}
    Dim m2(,) As Integer = {{0, 2, 4}, {6, 8, 10}}

    MatrisTopla(m1, m2)
End Sub
```

Opsiyonel Parametreler

Yordamlara parametre verilmesi opsiyonel olabilir. Parametre tanımlarken **`Optional`** anahtar kelimesi kullanılırsa, yordam çağırıldığında bu parametrenin girilmesi zorunlu olmaz. Opsiyonel parametreler tanımlanırken başlangıç değerleri verilmelidir, çünkü bu alan boş bırakıldığı zaman hangi değerin işleneceği bilinmelidir.

Örneğin, **`MsgBox`** kullanımında bazı parametrelerin isteğe bağlı girilebildiği görülür. Girilmeyen parametreler için varsayılan değerler kullanılır.

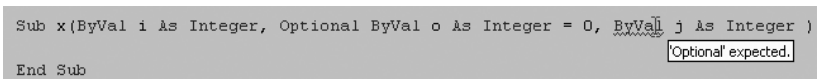
```
MsgBox("Mesaj")
MsgBox("Mesaj", MsgBoxStyle.YesNoCancel)
MsgBox("Mesaj", MsgBoxStyle.MsgBoxRight, "Uyarı")
MsgBox("Mesaj", , )
MsgBox("Mesaj", , "Dikkat")
```

Örnek:

```
Sub Yaz(ByVal mesaj As String, Optional ByVal SatirSayisi As
Integer = 1)
    Label1.Text &= mesaj
    ' Verilen satır sayısı kadar satır başı yapılır.
```

```
    For i As Integer = 0 To SatirSayisi - 1
        Label1.Text &= vbCrLf
    Next
End Sub

Sub DiziYazdir(ByVal Dizi1() As String, ByVal Dizi2() As
String, Optional ByVal Format As String = "Bitişik")

    If Dizi1.Length <> Dizi2.Length Then
        Yaz("Dizi uzunlukları eşit olmalı")
        Exit Sub
    End If

    If Format.ToLower() = "bitişik" Then
        For i As Integer = 0 To Dizi1.Length - 1
            Yaz(Dizi1(i) & Dizi2(i))
        Next
    ElseIf Format.ToLower() = "altalta" Then
        For i As Integer = 0 To Dizi1.Length - 1
            Yaz(Dizi1(i), 2)
            Yaz(Dizi2(i), 2)
        Next
    End If

End Sub
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Dim isimler() As String = {"Enis", "Kadir", "Ebru"}
    Dim Soyadlar() As String = {"Günesen", "Sümerkent",
"Gül"}

    DiziYazdir(isimler, Soyadlar)
    DiziYazdir(isimler, Soyadlar, "AltALTA")

End Sub
```

Opsiyonel parametreler, yordamların son argümanları olmalıdır (Resim 7.1). Bir opsiyonel parametreden sonra ancak başka bir opsiyonel parametre gelebilir.



RESİM 7.1: Opsiyonel parametre.

Bir yordamda veya fonksiyonda birçok opsiyonel parametre kullanılıyorsa, istenen parametreler boş bırakılabilir. Boş bırakılan parametrelerin sırası önemli değildir, ancak virgüller ile ayrılarak yerleri belirtilmelidir.

```
Sub Yordam(Optional ByVal param1 As String = "Merhaba", _
    Optional ByVal param2 As Date = #1/1/2005#, _
    Optional ByVal param3 As Boolean = True)

    ' Çalışacak kodlar...
End Sub

Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Yordam("Hoşgeldiniz", , )
    Yordam(, , )
    Yordam(, Now, )
End Sub
```

ParamArray

Yordamları ve fonksiyonları çağırırken parametrelerin mutlaka girilmesi gerekir. Ancak bazı durumlarda yordamlara ve fonksiyonlara girilecek parametrelerin sayısı tasarım zamanında belli olmaz. **ParamArray** anahtar kelimesi ile yordamlara, aynı veri tipinde bir parametre dizisi verilebilir. **ParamArray** ile verilen dizi, yordamın son parametresi olarak tanımlanmalıdır.

```
Sub YasOrtalamasi(ByVal sinif As String, ByVal ParamArray
Yaslar() As Byte)
    Dim toplam As Integer = 0
    Dim ortalama As Double = 0.0

    Dim i As Integer
    For i = 0 To Yaslar.Length - 1
        toplam += Yaslar(i)
    Next

    ' Parametre verilmezse i = 0 olur
    If i > 0 Then
        ortalama = toplam / i
    End If
    MsgBox(sinif & " sınıfının yaş ortalaması: " ortalama)
End Sub
```

Yaş ortalamasını hesaplayan bu yordamın ilk parametresi verilmek zorundadır. **ParamArray** ile tanımlı olan dizi, yordam çağırılırken girilen tüm parametreleri tutar. Fakat girilen bu parametrelerin veri tipleri aynı olmak zorundadır. Bu örnekte girilecek yaşlar **Byte** tipinde olacaktır.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    ' İlk parametre verildikten sonra,
    ' istenen sayıda parametre verilebilir
    YasOrtalamasi("YU6112", 45, 14, 25, 28)

    ' Yaşlar parametre olarak verilmeyebilir
    YasOrtalamasi("YU6112")
End Sub
```

Parametrelerin sınırlı olmaması, dizilere eleman ekleme işlemini kolaylaştırır. Örneğin, bir diziye birçok eleman eklemek için, bu elemanların bir dizi içinde parametre olarak verilmesi gerekir.

```
Dim Raf() As String
Sub KitapEkle(ByVal Kitap() As String)
    ' Raf dizisine, kitaplar dizisinin
    ' elemanları eklenir.
End Sub
```

Bu yordamın kullanımı için, eklenecek değerlerin önce bir diziye aktarılması gerekir. Yordamın yazılması kolay, ancak kullanımı zordur. Bu yordamı kullanacak olan programcının işi **`ParamArray`** ile kolaylaştırılır.

```
Sub Mesaj(ByVal msg As String)
    Label1.Text &= msg & vbCrLf
End Sub

Dim Raf() As String = {}

Sub KitapEkle(ByVal Kitap As String, Optional ByVal
Genisletme As Byte = 1)
    If Genisletme = 0 Then
        Mesaj("Dizi boyutu genişletilemedi...")
        Return
    End If

    ' Genişletme faktörü kullanıcıya bırakıldığı için
    ' dizide boş alanlar olabilir. İlk boş alan bulunup
    ' veri buraya aktarılır.
    Dim i As Integer
    While i < Raf.Length
        If Raf(i) = "" Then
            Raf(i) = Kitap
            Exit Sub
        End If
```

```
            i += 1
        End While

        ' Dizide boş yer yoksa yeniden boyutlandırılır.
        ReDim Preserve Raf(i + Genisletme)
        Raf(i + 1) = Kitap
    End Sub
```

Önce, diziye bir tek eleman ekleyen yordam yazılır. Dizinin tüm alanları doluysa, genişletme parametresinde verilen değer kadar tekrar boyutlandırılır. Genişletme değişkeni **Byte** veri tipinde tanımlı olduğu için negatif değer alamaz. Dolayısıyla, dizinin boyutunun küçültülmesi engellenmiş olur.

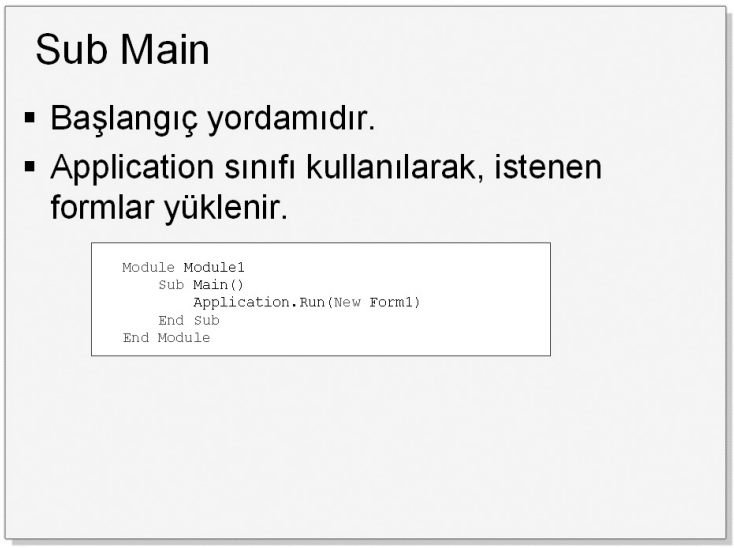
Bu yordam tek başına kullanılabildiği gibi, diziye birçok eleman ekleyecek yordama yardımcı olarak da kullanılabilir.

```
Sub Ekle(ByVal ParamArray Kitaplar() As String)
    For i As Integer = 0 To Kitaplar.Length - 1
        ' Genişletme faktörü 5 ile tek tek kitap eklenir.
        KitapEkle(Kitaplar(i), 5)
    Next
    Mesaj(Kitaplar.Length & " kitap rafa eklendi.")
End Sub
```

Bu yordam ise sınırsız parametre alarak, dizi işlemlerinde programcıya kolaylık sağlar.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Ekle("Kitap1", "Kitap2", "Kitap3")
End Sub
```

Sub Main

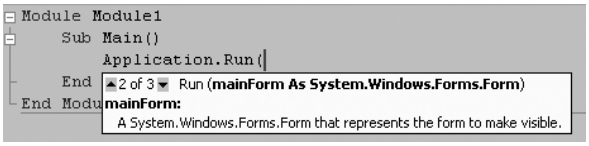


Yeni bir yordam tanımlarken **Main** yordamı hariç istenen isim verilebilir. **Main** yordamı bütün uygulamaların giriş noktasıdır. Windows uygulamalarında formlar yüklenmeden önce o form içinde tanımlı **Main** yordamı çalıştırılır. Bu **Main** yordamında **Application** sınıfı başlangıç formunu **Run** metodu ile yükler. **Application** sınıfı, .NET Framework'te, uygulamaları başlatmak, yönetmek ve sonlandırmak için kullanılır.

Projenin özelliklerinden başlangıç nesnesi **Sub Main** olarak ayarlanırsa, uygulama çalıştığı zaman tüm projede **Main** yordamını arar. Windows uygulamaları geliştirilirken **Main** yordamı yazılırsa başlangıç formunun da bu yordam içinde belirtilmesi gerekir. Bu yordam bir modülün içinde tanımlanabilir.

```
Module Module1
    Sub Main()
        Application.Run(New Form1)
    End Sub
End Module
```

Application sınıfının **Run** metodu, parametre olarak başlangıç formunu ister. Uygulama başladığı zaman hangi formun çalışması isteniyorsa, bu formdan oluşturulup parametre olarak verilir. **New** anahtar kelimesi, sınıfları oluşturmak için kullanılır.



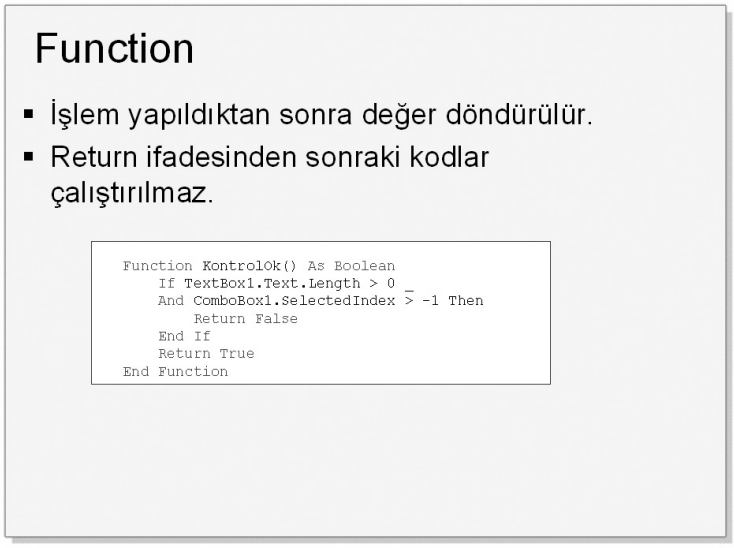
RESİM 7.2: **Run** metodu.

Başlangıç formu olarak seçilen bir Windows formunda **Main** yordamı tanımlanırsa, bu yordam **Shared** olarak tanımlanmalıdır. **Shared** metotlar uygulama genelinde paylaştırılan sabit metotlardır.

```
Shared Sub Main()
    MsgBox("Başlangıç formları kod ile yüklenmelidir.")
End Sub
```

Başlangıç formu olarak ayarlanmış bir formun içine bu **Main** yordamı tanımlanırsa, formu yüklemek için herhangi bir kod yazılmadığı için uygulama sadece mesaj kutusunu gösterecektir.

Konu 2: Function



Fonksiyonlar bir işlem yaptıktan sonra geriye değer döndürürler. Örneğin, bir çarpma fonksiyonunun dönüş değeri, parametre olarak verilen iki sayının çarpımı olacaktır. Fonksiyonların tanımları değişkenler gibidir.

```
Function Fonksiyon(ByVal Param1 As VeriTipi, ...) As
DönüşVeriTipi


End Function
```

Fonksiyonların geriye dönüş değerleri **`Return`** ifadesi ile ya da fonksiyonun ismi verilerek yapılır.

```
Function KontrolOk() As Boolean
    If TextBox1.Text.Length > 0 _
    And ComboBox1.SelectedIndex > -1 Then
        Return False
    End If
    Return True
End Function

Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    If Not KontrolOk() Then
        MsgBox("Seçiminizi yaptıktan sonra devam
edebilirsiniz.")
        Exit Sub
```

```
    End If

    ' Kontrol tamamlandıktan sonra yapılacak işlemler
End Sub
```

Bu fonksiyonun çalışması **Return** ifadesinden sonra yazılan değerin döndürülmesiyle sonlanır. Burada dikkat edilmesi gereken nokta, fonksiyon değer döndürdükten sonra sonlandığı için **Return** ifadesinden sonra gelen hiçbir kodun çalıştırılmamasıdır. Eğer dönüş değerini belirledikten sonra başka bir işlemin yapılması isteniyorsa, fonksiyonun ismi kullanılır. Fonksiyonun ismi bir değişken gibi gözükse de, temsil ettiği değer fonksiyonun dönüş değeridir.

```
Function GunlukKur(ByVal Cinsi As String) As Single
    Select Case Cinsi
        Case "d", "D"
            Return 1.43

        Case "e", "E"
            Return 1.81

        Case "s", "S"
            Return 2.91
    End Select
End Function

Function KurHesapla(ByVal Miktar As Single, Optional ByVal
Cinsi As String = "d") As Double
    KurHesapla = Miktar * GunlukKur(Cinsi)

    If KurHesapla < 0 Then
        ' Pozitife çevrilir.
        KurHesapla = 0 - KurHesapla
        MsgBox("Miktar negatif girilmiş.")
    End If
End Function

Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Label1.Text = KurHesapla(-1000)
End Sub
```

Bu örnekte, fonksiyonun ismi bir değişken olarak kullanılır ve hesaplanan değer bu değişkene atanır. Daha sonra bu değişkenin değeri, yani fonksiyonun dönüş değeri üzerinde işlem yapılabilir. Eğer **Return** ifadesi kullanılsaydı, kontrol yapılan kodlar çalıştırılmadan fonksiyondan çıkılacaktı.

```
Function KurHesapla(ByVal Miktar As Single, Optional ByVal
Cinsi As String = "d") As Double
    Return Miktar * GunlukKur(Cinsi)

    ' Bu satırdan sonra yazılan kodlar işlenmez.
End Function
```

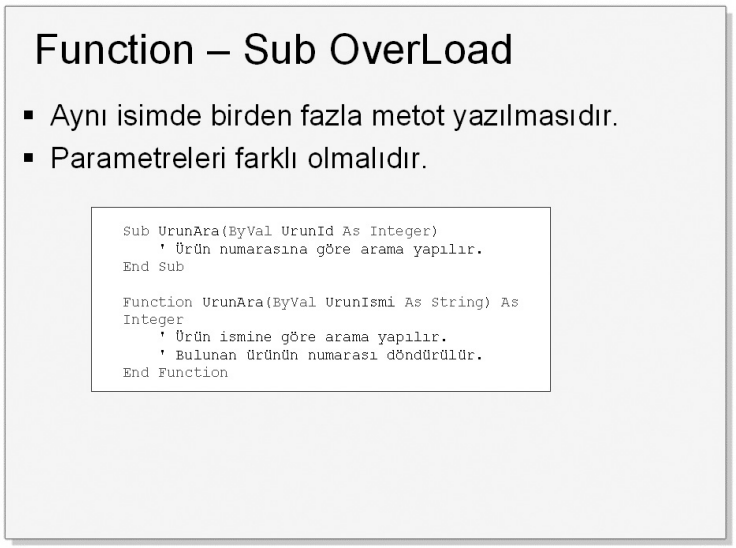
Örnek: Sınıf geçme notunun hesaplanması işleminin, geriye bir sonuç döndürüleceği için fonksiyon ile yazılması gereklidir. Parametre olarak final ve vize notları alınır ve bu değerlerle hesaplanan geçme notu sonuç olarak döndürülür. Vize notlarının girilmesi zorunlu değildir, dolayısıyla bu değerler **ParamArray** olarak verilebilir.

```
Function NotHesapla(ByVal Final As Integer, ByVal
VizeKatSayisi As Single, ByVal ParamArray vizeler() As
Integer) As Integer
    Dim vizeToplam As Integer = 0
    Dim vizeOrtalama As Double = 0.0

    Dim i As Integer
    For i = 0 To vizeler.Length - 1
        vizeToplam += vizeler(i)
    Next

    If i > 0 Then
        vizeOrtalama = vizeToplam / i
    End If

    Dim finalKatSayisi As Single = 1 - VizeKatSayisi

    Return finalKatSayisi * Final + VizeKatSayisi *
vizeOrtalama
End Function
```

Fonksiyonun ilk parametresi final notudur. Final notu bir tane olacağı için girilmesi zorunludur. Daha sonra vize notlarının ortalaması hesaplanarak final notu ile toplanır. Parametre olarak verilen vize katsayısı, vize notlarının ortalamadaki ağırlıklarını belirler.

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Dim gecmeNotu As Integer
    gecmeNotu = NotHesapla(70, 0.6, 90, 80, 86, 75, 90)
    MsgBox(gecmeNotu)
End Sub
```

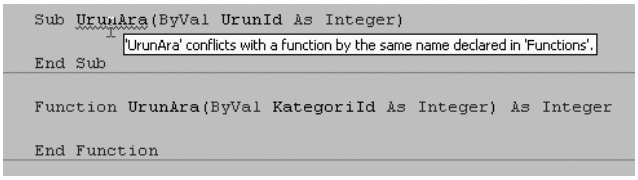
Fonksiyonların ve Yordamların Aşırı Yüklenmesi



Fonksiyon ve yordamları kullanırken, aynı isimde birden fazla kez tanımlanabildikleri görülür. Buna Aşırı Yükleme (OverLoad) denir. Bir yordamın ve fonksiyonun aşırı yüklenmesi kullanımını kolaylaştırır. Aynı isimde farklı seçenekler sunması, metotların kullanışlılığını artırır.

```
Sub UrunAra(ByVal UrunId As Integer)
    ' Ürün numarasına göre arama yapılır.
End Sub

Function UrunAra(ByVal UrunIsmi As String) As Integer
    ' Ürün ismine göre arama yapılır.
    ' Bulunan ürünün numarası döndürülür.
End Function

Function UrunAra(ByVal UrunIsmi As String, ByVal
UretimTarihi As Date) As Integer
    ' Ürün ismine ve üretim tarihine göre arama yapılır.
    ' Bulunan ürünün numarası döndürülür.
End Function

Function UrunAra(ByVal UretimTarihi As Date) As Integer
    ' Üretim tarihine göre arama yapılır.
    ' Bulunan ürünün numarası döndürülür.
End Function
```

Aynı isimdeki metotların ayrımı, parametrelerin veri tipi ve sayısına göre yapılır. Metotların isimleri, parametre sayısı ve parametrelerin veri tipleri metotların imzalarını (Method Signatures) oluşturur. Örneğin, ürün numarasına göre arama yapan yordamın imzası **`UrunAra(Integer)`** şeklindedir. **`Integer`** parametre alan, **`UrunAra`** isminde başka bir yordam veya fonksiyon tanımlanamaz (Resim 7.3). Fonksiyonların dönüş tipleri ile imzaları tanımlanmaz.



RESİM 7.3: Hatalı tanımlama.

Metotları aşırı yüklerken dikkat edilmesi gereken bazı noktalar vardır.

- İmzaları aynı olan metotlar tanımlanamaz.
- Fonksiyonlar yordamlarla, yordamlar da fonksiyonlarla aşırı yüklenebilir.
- Fonksiyonlar dönüş tiplerine göre aşırı yüklenemez.

Konu 3: String Fonksiyonları

`String` fonksiyonları, kullanıldığı **`String`** değeri üstünde verilen parametrelere göre değişen işlemler yaparlar. Sonuç olarak geriye döndürdükleri değerler, fonksiyonun işleyiş amacına göre değişir.

- **`CompareTo`.** Bu fonksiyon, işlemin yapılacağı değeri parametre olarak verilen değerle karşılaştırır. İki değer bir birine eşitse 0, parametredeki değer alfabetik olarak önde ise 1, değilse -1 değeri döndürülür.

```
Dim yazi1 As String = "BilgeAdam"
Dim yazi2 As String = InputBox("Yazı giriniz.")

Select Case yazi1.CompareTo(yazi2)
    Case 0
        MsgBox("Yazılar birbirine eşit")
    Case 1
        MsgBox(yazi1 & ", " & yazi2 & " kelimesinden sonra
geliyor")
    Case -1
        MsgBox(yazi1 & ", " & yazi2 & " kelimesinden önce
geliyor")
End Select
```

- **`Concat`. `String`** değerlerini birleştirmek için kullanılır. Parametre tipi **`ParamArray`** olduğu için, sınırsız **`String`** değişkeni birleştirilebilir.

```
Dim kurum As String = "BilgeAdam"

Label1.Text = String.Concat("Kurum: ", kurum, "Şubeler: ",
vbCrLf, "Fatih", "Bakırköy", "Kadıköy", "Beşiktaş", "Town
Center")
```

- **CopyTo.** Bu fonksiyon ile bir **String** değişkenin belli bir kısmı, bir karakter dizisine kopyalanır. Ayrıca kopyalanacak dizinin hangi indisten itibaren başlanacağı da belirtilir.

```
Dim yazi As String = "BilgeAdam"
Dim Karakterler(10) As Char

' Yazının 5. karakterinden itibaren alınan 4 karakter,
' karakterler dizisinin 3. indisinden başlanarak
' diziye kopyalanır.
yazi.CopyTo(5, Karakterler, 3, 4)

' Karakterler dizisinin son hali:
' _ _ _ A d a m _ _ _ _
```

Burada dikkat edilmesi gereken nokta, karakterlerin kopyalanacağı dizinin büyüklüğünün yeterli olup olmadığıdır. Dizinin kopyalanmaya başlanacak indisi ile kopyalanacak karakterlerin uzunluğunun toplamı, dizi büyüklüğünden küçük olmalıdır

- **EndsWith & StartsWith.** Bu fonksiyonlar, **String** değişkeninin, parametrede verilen **String** değeriyle başladığını ya da bittiğini gösterir. Geriye dönüş değeri **Boolean** tipindedir.

```
degisken.EndsWith(deger As String) As Boolean

degisken.StartsWith(deger As String) As Boolean

Dim HtmlTag As String = "<table>"
If HtmlTag.StartsWith("<") And HtmlTag.EndsWith(">") Then
    MsgBox("Yazım doğru")
End If
```

- **ToUpper & ToLower. ToUpper**, **String** değişkenin içindeki küçük karakterleri büyüğe; **ToLower**, büyük karakterleri küçüğe çevirir.

```
Dim yazi As String = "bilgeADAM"

MsgBox(yazi.ToUpper())
' Sonuç: BİLGEADAM
MsgBox(yazi.ToLower)
' Sonuç: bilgeadam
```

- **Join.** Bir **String** dizisindeki elemanları, parametre olarak verilen ayraç karakteri ile birleştirerek tek bir **String** değişkeni döndürür.

```
Dim yazi() As String = {"İsim", "Soyad", "Adres", "Email",
"Telefon"}
MsgBox(String.Join(";", yazi))

' Sonuç: İsim;Soyad;Adres;Email;Telefon
```

- **SubString.** Verilen bir **String** değerinin, bir bölümünü **String** olarak döndüren fonksiyondur. İstenen karakterlerin hangi indisten başlayacağı parametre olarak geçilir. Bu durumda, başlangıç karakterinden sona kadar okunur. Ancak fonksiyonun, kaç karakter okunacağını belirten bir parametre kabul eden aşırı yüklemesi de vardır.

```
Dim yazi As String = "BilgeAdam"
MsgBox(yazi.Substring(5))
' Sonuç : Adam

MsgBox(yazi.Substring(5, 2))
' Sonuç : Ad
```

- **Trim, TrimEnd, TrimStart. Trim** fonksiyonu, parametre olarak verilen bir karakteri, **String** değişkeninin başından ve sonundan kaldırır.

  TrimEnd fonksiyonu parametrede verilen karakteri **String** değişkeninin sadece sonundan, **TrimStart** ise sadece başından kaldırır.

```
Dim yazi As String = "--------Merhaba--------

MsgBox(yazi.Trim("-"))
' Sonuç: Merhaba

MsgBox(yazi.TrimEnd("-"))
' Sonuç: --------Merhaba

MsgBox(yazi.TrimStart("-"))
' Sonuç: Merhaba--------
```

Konu 4: Matematiksel Fonksiyonlar

Uygulamalarda çoğu zaman matematiksel hesaplamalara ihtiyaç duyulur. Bu hesaplamaları kolaylaştıran hazır matematik fonksiyonları vardır. Bu fonksiyonlar .NET Framework'te **`System.Math`** ad uzayının (namespace) içinde tanımlanmıştır.

- **`Abs`.** Verilen bir sayının mutlak değerini döndürür. Dönen değer her durumda pozitif olacaktır.

```
Math.Abs(-123)
' Sonuç: 123
```

- **`Ceiling`** & **`Floor`. `Ceiling`** fonksiyonu, **`Double`** veri tipinde verilen bir sayıdan büyük, en küçük tamsayıyı verir. **`Floor`** fonksiyonu verilen sayıdan küçük, en büyük tamsayıyı verir.

```
Math.Ceiling(-12.231231)
' Sonuç: -12

Math.Ceiling(12.231231)
' Sonuç: 13

Math.Floor(-12.231231)
' Sonuç: -13

Math.Floor(12.231231)
' Sonuç: 12
```

- **Cos, Sin, Tan.** Bu fonksiyonlar temel trigonometrik işlemleri gerçekleştirir. **Cos** fonksiyonu verilen derecenin kosinüsünü, **Sin** sayının sinüsünü ve **Tan** sayının tanjantını hesaplar. Parametre olarak verilen derece radyan (360 derece) değeri olarak kabul edilir.

```
Dim Derece  As Double =90
Math.Cos(Math.PI * Derece / 180)
Math.Sin(Math.PI * Derece / 180)
Math.Tan(Math.PI * Derece / 180)
```

- **Exp**. Bu fonksiyon, *e* sabitinin değerinin (yaklaşık 2,718281 değerinin), parametrede verilen sayı ile üssünü alır.

```
Math.Exp(4)
' Sonuç yaklaşık: 54,59815
```

- **Log**. Logaritmik hesaplamalar için kullanılan bir fonksiyondur. Taban parametresi verilmezse sayının *e* tabanında logaritmasını alır.

```
Math.Log(1000, 10)
' Sonuç: 3

Math.Log(Math.E)
' Sonuç: 1
```

- **Max & Min. Max** fonksiyonu verilen iki sayıyı karşılaştırarak büyük olanı, **Min** fonksiyonu ise sayılardan küçük olanı döndürür.

```
Math.Max(100, 200)
' Sonuç: 200
Math.Min(100, 200)
' Sonuç: 100
```

- **Pow**. İlk parametrede verilen bir sayının, ikinci parametredeki değer kadar üssünü alır.

```
Math.Pow(10,3)
' Sonuç: 1000
```

- **Sqrt.** Verilen sayının karekökünü hesaplar.

```
Math.Sqrt(441)
' Sonuç: 21
```

Konu 5: Tarih ve Zaman Fonksiyonları

Tarih ve zaman fonksiyonları **`Date`** veri tipi üzerinde hesaplamalar yapan fonksiyonlardır. Bu fonksiyonlar **`System.DateTime`** ad uzayında tanımlıdır. Ayrıca **`Microsoft.VisualBasic`** altında tanımlı **`DateAdd`** ve **`DateDiff`** fonksiyonları da vardır.

- **`DateAdd`.** Bu fonksiyon verilen bir zamana belli bir tarih veya zaman aralığı eklemek için kullanılır. Dönüş değeri, hesaplanan **`Date`** tipinde bir değer olacaktır.

```
Function DateAdd(ByVal Aralik As DateInterval, _
   ByVal Sayi As Double, _
   ByVal Zaman As DateTime _
) As DateTime
```

Fonksiyonun aldığı ilk parametre **`DateInterval`** tipinde bir zaman aralığıdır. Bu değerler aşağıdaki tiplerde olabilir:

- **`Day`.** Gün (1 – 31)
- **`DayOfYear`.** Gün (1 – 366)
- **`Hour`.** Saat (1 – 24)
- **`Minute`.** Dakika (1 – 60)
- **`Month`.** Ay (1 – 12)
- **`Quarter`.** Çeyrek yıl (1 – 4)
- **`Second`.** Saniye (1 – 60)
- **`WeekDay`.** Gün (1 – 7)

- **WeekOfYear.** Hafta (1 – 53)
- **Year.** Yıl

Zaman parametresine işlem yapılacak tarih, **Sayi** parametresine de eklenecek değer girilir.

```
Do
    Dim baslangic As Date = Now
    Dim tahmin As Integer
    tahmin = InputBox("3 saniye içinde bir sayı girin")

    If Now > DateAdd(DateInterval.Second, 3, baslangic) Then
        MsgBox("Süreniz Doldu")
        Exit Do
    End If

    ' Oyun kodları...
Loop
```

- **DateDiff.** Bu fonksiyon verilen iki tarihin arasındaki kalan süreyi hesaplar. Bu süre **DateInterval** tipinde parametre olarak verilir.

```
Function DateDiff( _
   ByVal Aralik As DateInterval, _
   ByVal Zaman1 As DateTime, _
   ByVal Zaman2 As DateTime
) As Long
```

Zaman aralığının hesaplanması, ikinci zaman değerinden ilkinin çıkarılması ile gerçekleşir: **Zaman2 – Zaman1**

```
Sub DogumGununeKalan(ByVal dogumGunu As Date, Optional ByVal
aralik As DateInterval = DateInterval.Day)
    ' Verilen doğum gününün senesi,
    ' içinde bulunan sene ile değiştirilir.

    dogumGunu = DateAdd(DateInterval.Year, Now.Year -
dogumGunu.Year, dogumGunu)

    Dim kalan As Long
    kalan = DateDiff(aralik, dogumGunu, Now.Date)

    Dim mesaj As String = "Doğumgününüz " & Math.Abs(kalan)
& " " & aralik.ToString()

    If kalan > 0 Then
        mesaj &= " geçmiş"
```

```
    ElseIf kalan < 0 Then
        mesaj &= " sonra"
    Else
    mesaj = "Doğumgününüz bu " & aralik.ToString() & "
içinde"
    End If
    MsgBox(mesaj)
End Sub
```

Bu örnek, verilen bir doğum gününe kalan zaman aralığını hesaplar. Zaman aralığı varsayılan değeri **Day** olan opsiyonel bir parametre olarak verilir. **DateDiff** fonksiyonu iki tarih arasındaki zaman aralığını hesaplar. Bu örnekte istenen, doğum gününe kalan zamandır. Dolayısıyla doğum gününün yılı önemli değildir. Bu değer içinde bulunan yıl değeri ile değiştirildiğinde, bu sene içinde kalan gün hesaplanabilir.

- **CompareTo. String** ifadelerinde olduğu gibi, tarih ve zaman değerleri üzerinde de karşılaştırma yapılabilir. **CompareTo** fonksiyonu, işlem yapılan tarih ile parametre olarak verilen tarihi karşılaştırır. Parametredeki tarih küçükse 1, büyükse -1 veya eşitse 0 döndürür.

```
Dim d As Date = #03/23/2002#
MsgBox(d.CompareTo(Now))
```

- **DaysInMonth.** İlk parametrede verilen yılın, ikinci parametrede verilen ayında kaç gün olduğunu döndürür.

```
Date.DaysInMonth(2002, 2)
' Sonuç: 28
```

- **IsLeapYear.** Verilen bir yılın artık yıl olup olmadığını hesaplar. Dönüş değeri **True** ya da **False** olabilir.

```
Date.IsLeapYear(1200))
' Sonuç: True
```

- **Parse**. Parametrede verilen **String** bir ifadeden **Date** veri tipine çevrim işlemini yapar. **String** ifadesinde verilen ifadenin doğru bir tarih ve zaman tipinde olması gerekir.

```
Date.Parse("23.04.2005 20:20:00")
Date.Parse("22 July 2005 02:00 PM")
Date.Parse("18 Haziran 1980")
```

İngilizce'den farklı bir dilde girilen ay isimlerinin tarih tipine çevrilmesi için, uygulamanın kültürünün o dilde ayarlanması gerekir.

```
' Uygulama kültürü Fransızca yapılır.
Application.CurrentCulture = New Globalization.CultureInfo("fr-FR")
' temps değişkeninin değeri 23/05/2005 olacaktır.
Dim temps As Date = Date.Parse("23 Mai 2005")
' Bu kod hata verecektir.
Dim zaman As Date = Date.Parse("23 Mayıs 2005")
```

- **`ToLongDateString` & `ToLongTimeString`.** Verilen tarihi uzun tarih ve zaman formatında gösteren fonksiyonlardır.

```
Dim d As Date = #1/29/2005 12:59:22 PM#

d.ToLongDateString()
' Sonuç: 29 Ocak 2005 Cumartesi

d.ToLongTimeString()
' Sonuç: 12:59:22
```

- **`ToShortDateString` & `ToShortTimeString`.** Verilen tarihi kısa tarih ve zaman formatında gösteren fonksiyonlardır.

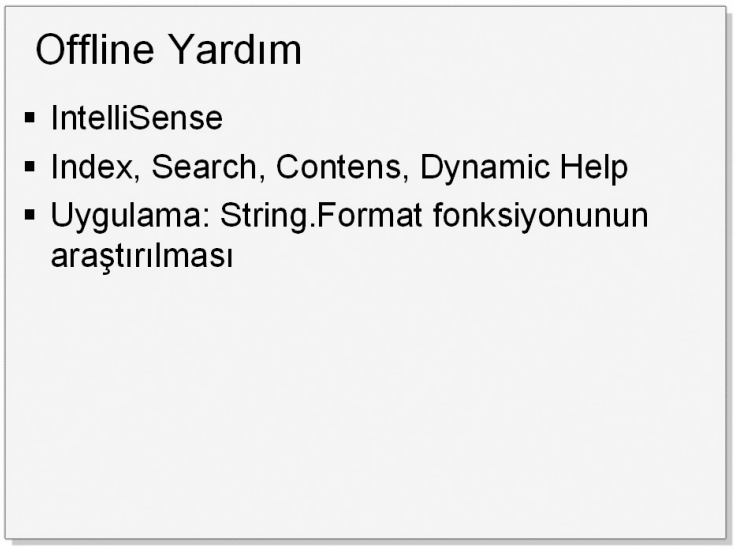
```
Dim d As Date = #1/29/2005 12:59:22 PM#

d.ToShortateString()
' Sonuç: 29 Ocak 2005

d.ToShortTimeString()
' Sonuç: 12:59
```

Konu 6: Offline ve Online Yardımın Etkin Kullanımı

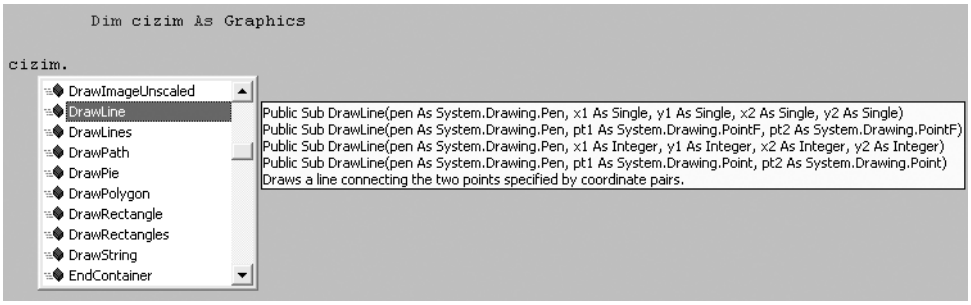
Visual Basic.NET dilinde uygulama geliştirirken .NET Framework içinde tanımlı bir çok nesnenin fonksiyon ve yordamları kullanılır. Ancak her yordam ve fonksiyonun aldığı parametrelerin ve ne işe yaradıklarının ezbere bilinmesi mümkün değildir.

Modül 3'teki Help Kullanımı bölümünde MSDN offline yardımının kullanılmasından ve öneminden bahsedilmişti. MSDN kütüphanelerinin Visual Studio içine kurulmaması durumunda online yardım araçları kullanılabilir. Visual Studio, başlangıç sayfasının Online Resources sekmesinde birçok arama kolaylığı sunar.

Offline Yardım



Uygulama geliştirirken, kodların yazılmasında IntelliSense aracından büyük ölçüde faydalanır. IntelliSense, bir kodun yazılması sırasında açıldığı zaman, yazılan kodlarla başlayan tüm metot, özellik ve nesneleri programcıya sunar. O anda üzerinde bulunulan öğenin açıklaması, aldığı parametreler gibi bilgileri de gösterir.



RESİM 7.4: IntelliSense.

Visual Studio içinde MSDN kütüphanelerinde istenen konuların aranması için Index, Search, Contents ve Dynamic Help panelleri kullanılır. Sonuç bulunduğu zaman yeni bir çalışma sayfasında gösterilir. Bu sayfada aranan kavram ile ilgili detaylı bilgiler ve örnekler mevcuttur.

Örnek: **`String`** veri tiplerinin değişik formatlarda yazdırılması için **`String.Format`** fonksiyonu kullanılır.

1. Visual Studio ortamında bir proje açın ve kod sayfasında **`String.Format`** yazın. Fonksiyonu yazdıktan sonra parantezi açın ve IntelliSense aracının çıkardığı menüyü inceleyin.

   Fonksiyon kaç parametre alabiliyor?

   Aşağı ve yukarı oklarla menü içinde ilerleyerek fonksiyonun aşırı yüklenmiş durumlarını inceleyin.

   Fonksiyonun kaç tane aşırı yüklemesi yazılmış?

2. Format yazısının üstüne geldikten sonra **F1** tuşuna basın ve dinamik yardımın açtığı sayfaya bakın. Bu sayfa fonksiyonun tüm aşırı yüklemelerini gösterir.

3. Parametre olarak **`String`** ve **`ParamArray Object`** alan fonksiyonu tıklayın. Çıkan sayfa fonksiyonun detaylarını listeler.

   - İlk olarak fonksiyonun söz dizimi verilmiştir. Burada parametre isimleri ve tipleri üzerinde bağlantılar görünür. Bu bağlantılar ile ilgili yardım dosyası açılır.
   - Parameters bölümünde bu fonksiyonun aldığı parametrelerin tipleri ve kullanım amaçlarını gösterilir.
   - Return Value fonksiyonun dönüş değerinin hangi tipte olduğu ve nasıl oluştuğu gösterilir.
   - Exceptions bölümünde bu fonksiyon kullanılırken meydana gelebilecek hatalar listelenir.
   - Remarks bölümü, fonksiyonun kullanım yerleri, parametrelerin nasıl kullanılacağı, parametreler kullanılırken dikkat edilmesi gereken yerler, bağlantılı konular gibi fonksiyon hakkında detaylı bilgiler verir.
   - Example bölümünde, fonksiyonun kullanımına örnekler verilir.
   - Requirements bölümünde, fonksiyonun çalışabilmesi için gereken araçlar ve platformlar listelenir.
   - See Also bölümü, fonksiyon ile ilişkili kavramlara bağlantılar sunar.

   `Remark` bölümündeki tanımlamalardan ve **`Examples`** bölümündeki örneklerden yararlanarak, fonksiyonun nasıl kullanıldığını inceleyin.

4. Kod sayfanıza geçin ve **`String.Format`** fonksiyona bir örnek yazın.

```
Dim ocak As Integer = 1000
Dim subat As Integer = 1100

MsgBox(String.Format("Ocak ayı maaşı {0:C} -- Şubat ayı
maaşı: {1:c}", ocak, subat))
```

5. Formatlama işlemleri hakkında daha fazla bilgi almak için, fonksiyonun yardım sayfasına gelin ve **`Remarks`** bölümünde Formatting Types

bağlantısını tıklayın ya da Index panelinde Formatting Types yazın ve yardım sayfasını açın.

Çıkan sayfada her veri tipi için kullanılan formatlama seçenekleri vardır.

Numeric Format Strings bağlantısını tıkladıktan sonra açılan sayfada NumberFormatInfo bağlantısını tıklayın.

Format Character tablosunda değişik formatlama seçeneklerini inceleyin ve kodunuzda deneyin.

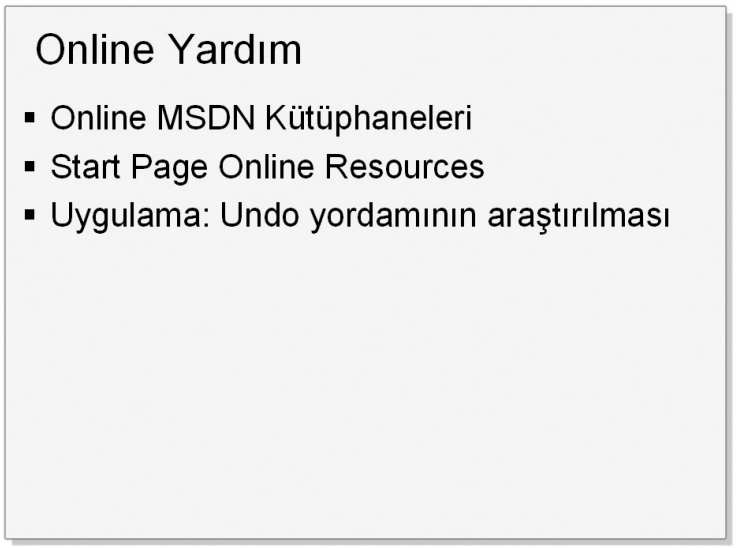
6. Web araç çubuğundan Geri düğmesini tıklayarak veya **ALT+Sola Ok** kısayolu ile **`String.Format Method`** başlıklı ilk açtığınız sayfaya dönün.
7. Parametre olarak **`IFormatProvider`**, **`String`** ve **`ParamArray Object`** alan fonksiyon tanımını tıklayın. Fonksiyonun kullanımını inceledikten sonra bu kullanıma bir örnek yazın.

```
MsgBox(String.Format(New Globalization.CultureInfo("it-IT"),
"Bugün: {0:dddd MM yyyy}", Now))
```

8. Bu örnekte uygulamanın kültür ayarları değiştirilmeden, tarihin istenen kültür ayarı ile gösterilmiştir. Kültür ayarlarının tanımlanmasını incelemek için Index yardım panelinde CultureInfo yazın ve About CultureInfo Class indeksini seçin. Çıkan Index Result penceresinde CultureInfo Class indeksini seçin.
9. Bu sayfada çıkan kültür isimlerini örneğinizde kullanarak değişik sonuçları inceleyin.

NOT **Türkçe dil ailesi için** `Globalization.CultureInfo("TR-tr")` **kullanılır**

Online Yardım



MSDN kütüphanelerinden offline olarak yardım almak hızlı ve etkili bir yöntemdir. Ancak bu yardım dosyalarının güncellenmesi için MSDN sürümünün yenilenmesi gerekir. Online yardım, MSDN kütüphanelerinin Internet ortamında yayınlanmasıdır. Yeni örnekler, makaleler ve düzeltmelerle güncellenen bu yardım dosyalarına `http://msdn.microsoft.com` adresinden ulaşılabileceği gibi, Visual Studio ortamından da bu dosyalar içinde arama yapılabilir.

Örnek: Windows uygulamasında kullanılan bir metin kutusunda "Geri Al" (`Undo`) işlemi yapılmak isteniyor, fakat fazladan kod yazılmak istenmiyor. Bunun için .NET Framework'te hazır bir metodun olup olmadığı kontrol edilmesi gerekir. Online yardım ile gerekli arama yapıldıktan sonra çıkan sonuçlar yorumlanır.

1. Başlangıç sayfasını (Start Page) açın ve Online Resources sekmesine gelin.
2. Sol paneldeki menüden Search Online menüsüne gelin ve Search For altındaki metin kutusuna "TextBox Undo" yazın. Sonuçların MSDN Online içinde hangi duruma göre filtrelenebildiğini gösteren bağlantılar çıkar. Sonuçlar;
   - Tüm MSDN içinde,
   - MSDN kod ve karşıdan yüklemelerde,
   - MSDN teknik makalelerinde,
   - Microsoft bilgi veri kaynağında,
   - Microsoft.com genelinde filtrelenebilir.

3. Search results for All of MSDN bağlantısını tıklayın ve çıkan sonuçları inceleyin. Aranan kaynağın .NET Framework içinde kullanılabilmesi istendiği için TextBoxBase.Undo Method (.NET Framework) yardım konusunu tıklayın.
4. MSDN Online kütüphanelerinin sayfa düzeni ve içeriği offline yardım ile aynıdır. **`TextBoxBase`** taban sınıfının **`Undo`** metodunu inceleyen bu yardım sayfasında, metot tanımlaması, Remarks, Examples, Requirements ve See Also bölümleri görülür. Examples bölümünde Visual Basic kodlarının altında **`Undo`** metodunun kullanımını inceleyin.
5. Undo yapıldıktan sonra silinen kelimelerin bir listede tutulması ve listeye ekleme işleminin kolay bir şekilde yapılması isteniyor. Bunun için Sol panelde bulunan menülerin üstündeki Search For metin kutusuna "ArrayList" yazın ve çıkan sonuçlardaki ilk bağlantıyı tıklayın.
6. **`ArrayList`** sınıfının **`Count`** ve **`Item`** özelliklerini ve **`Add`** metodunu inceleyin Uygulamanızı tamamlamak için bu özellikleri kodunuzda kullanın.

```
Dim silinenler As New ArrayList

Sub GeriAl()
    ' Metin kutusunda geri alınacak bir veri varsa
    If TextBox1.CanUndo Then

        ' Eski değerler listeye eklenir.
        silinenler.Add(TextBox1.Text)
        TextBox1.Undo()
        GeriAlinanKelimeler()
    End If
End Sub

' Listeleme işlemini yapan yordam
Sub GeriAlinanKelimeler()
    ListBox1.Items.Clear()
    For i As Integer = 0 To silinenler.Count - 1
        ' i indisli Item, liste kutusuna eklenir.
        ListBox1.Items.Add(silinenler.Item(i))
    Next
End Sub
```

Lab 1: Kelime Oyunu

Bu uygulamadaki oyun, girilen bir kelimenin son harfleriyle başlayan başka bir kelimenin girilmesidir. Oyunun seviyesi, girilecek kelimenin kontrol edilecek harf sayısıdır. Örneğin, ikinci seviyede ilk girilen kelime "Masa" ise, bir sonraki kelime "sa" ile başlamalıdır. Üçüncü seviyede bu kelime "asa" ile başlamalıdır.

Kullanıcı, oyuna ilk seviyeden başlar ve beş kelime bildiği zaman bir sonraki seviyeye geçer. Toplam alınan puan, bilinen kelime sayısının seviye kadar kuvveti alınarak hesaplanır.

Projenin Açılması

1. **KelimeOyunu** isminde bir Window projesi açın ve forma listedeki kontrolleri ekleyin.
   - **btnBasla** ve **btnGiris** isminde iki **Button**
   - **txtKelime** isminde bir **TextBox**
   - **lblMesaj** isminde bir **Label**
   - **tmrSure** isminde bir **Timer**
2. Projenizin kod sayfasına geçin ve uygulama boyunca kullanılacak global değişkenleri tanımlayın.

```
' Kontrol edilecek kelime
Dim kelime As String

' Oyunun seviyesi
Dim OyunSeviyesi As Byte = 1

' Timer kontrolünde kullanılacak süre
Dim kalanSure As Integer = 5

' Bilinen kelime sayısı
Dim tekrar As Integer = 0
```

3. Uygulamaya giriş **Sub Main** yordamından yapılır. Bu yordamda kullanıcıdan, formun başlığında görüntülenecek bir kullanıcı adı istenir. Eğer kullanıcı adı boş girilirse form yüklenmeden uygulamadan çıkılır.

```
' Uygulamanın giriş noktası
Shared Sub Main()
    Dim KullaniciAdi As String
    KullaniciAdi = InputBox("Kullanıcı Adı girin:")

    If KullaniciAdi = "" Then
        Exit Sub
```

```
    End If

    ' Form yükleniyor
    Dim oyun As New Form1
    oyun.Text = KullaniciAdi & " yarışıyor"
    oyun.ShowDialog()
End Sub
```

Yardımcı Yordam ve Fonksiyonlar

Uygulamanın tamamında kullanılacak kodlar yordam ve fonksiyonlar halinde yazılarak hem yönetilmesi hem de kullanılabilirliği artırılır. Uygulamada kullanılacak yordam ve fonksiyonlar Tablo 7.1'de listelenmiştir.

TABLO 7.1: Kelime Oyunu Uygulaması Yordam ve Fonksiyonları

İsim	Parametreler	İşlev
Temizle		Zamanı sıfırlar ve **TextBox** kontrolüne **Focus** verir
OyunuBaslat		Başlangıç kelimesi alınarak **Timer** başlatılır.
OyunuBitir	**Optional String neden**	Süreyi durdurur, puanı ve bitiş nedeni kullanıcıya gösterir.
Bilgi	**String mesaj**	**Label** kontrolünde mesaj görüntülenir.
SonrakiKelimeBilgi		Girilecek kelimenin hangi harflerle başlayacağını gösterir.
SeviyeAtla	**Byte seviye**	Oyunun seviyesini artırır.
Kontrol	**String kelime1, String kelime2**	İkinci kelimenin, ilk kelimenin harfleriyle başladığının kontrolü yapılır.
PuanHesapla	**Byte seviye, Short tekrar**	Tekrar değerinin, seviye kadar üssü alınır.

1. Yordamları ve fonksiyonları yazın.
   - **Temizle** yordamı

```
Sub Temizle()
    kalanSure = 5
    txtKelime.Text = ""
    txtKelime.Focus()
End Sub
```

- **OyunuBaslat** yordamı

```
Sub OyunuBaslat()
    Temizle()
    kelime = InputBox("Oyunun başlangıç kelimesini girin")

    tmrSure.Start()

    SonrakiKelimeBilgi()
End Sub
```

- **OyunuBitir** yordamı

```
Sub OyunuBitir(Optional ByVal neden As String = "")
    tmrSure.Stop()
    Bilgi(neden)

    Temizle()

    Dim puan As Integer
    puan = PuanHesapla(OyunSeviyesi, tekrar)
    MsgBox("Puanınız: " & puan)
End Sub
```

- **Bilgi** yordamı

```
Sub Bilgi(ByVal kelime As String)
    lblMesaj.Text = kelime
End Sub
```

- **SonrakiKelimeBilgi** yordamı

```
Sub SonrakiKelimeBilgi()
    Dim mesaj As String
    mesaj &= Microsoft.VisualBasic.Right(kelime,
OyunSeviyesi)
    mesaj &= " ile başlayan bir kelime girin"

    Bilgi(mesaj)
End Sub
```

- **SeviyeAtla** yordamı

```
Sub SeviyeAtla(ByVal seviye As Byte)
    OyunSeviyesi = seviye

    OyunuBitir(seviye & ". seviyeye geçildi")
```

```
    SonrakiKelimeBilgi()
End Sub
```

- **`Kontrol`** fonksiyonu

```
Function Kontrol(ByVal kelime1 As String, ByVal kelime2 As
String) As Boolean
    ' İkinci kelimenin başında oyun seviyesi kadar
    ' karakter alınır.
    Dim bas As String = kelime2.Substring(0, OyunSeviyesi)

    ' İkinci kelime, ilk kelimenin sonu ile başlıyorsa
    ' doğru girilmiştir. True değeri döner.
    Return kelime1.EndsWith(bas)
End Function
```

- **`PuanHesapla`** yordamı

```
Function PuanHesapla(ByVal seviye As Byte, ByVal tekrar As
Short) As Integer
    Return Math.Pow(tekrar, seviye)
End Function
```

Olayların Yazılması

1. **`tmrSure`** kontrolünün **`Tick`** olayına kalan süreyi kontrol eden kodları yazın.

```
Private Sub tmrSure_Tick(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles tmrSure.Tick
    If kalanSure <= 0 Then
        OyunuBitir("Süreniz doldu")
    Else
        kalanSure -= 1
    End If
End Sub
```

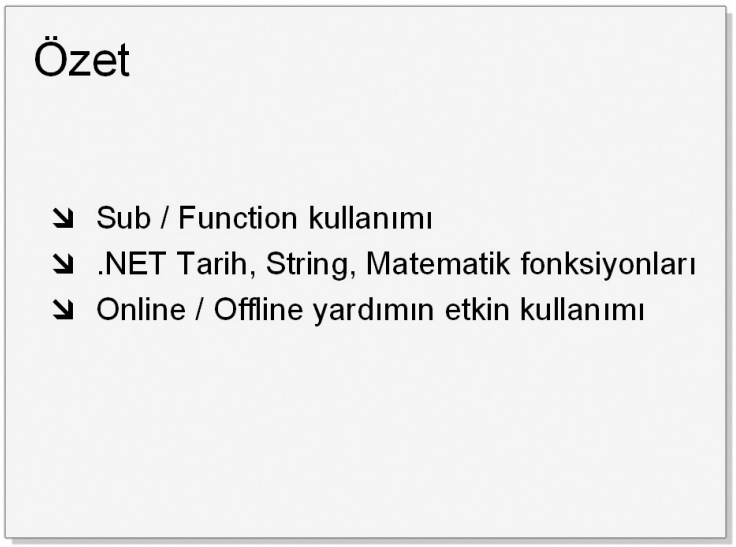
2. **`btnBasla`** düğmesinin **`Click`** olayına, oyunu başlatan yordamı yazın

```
Private Sub btnBasla_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnBasla.Click
    OyunuBaslat()
End Sub
```

3. **`btnGiris`** düğmesinin **`Click`** olayına, girilen kelimeyi alıp kontrolleri yapan kodu yazın. Burada dikkat edilmesi gereken nokta, tekrar sayısının seviye ile doğru orantılı olmasıdır.

```
Private Sub btnGiris_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnGiris.Click
    Dim girilen As String = txtKelime.Text

    If Not Kontrol(kelime, girilen) Then
        Dim neden As String
        neden = "Girilen kelime, ilk kelimenin son "
        neden &= OyunSeviyesi & " harfi ile başlamıyor"

        OyunuBitir(neden)
    ElseIf tekrar > 5 * OyunSeviyesi Then
        SeviyeAtla(OyunSeviyesi + 1)
    Else
        tekrar += 1
        kelime = girilen
        SonrakiKelimeBilgi()
        Temizle()
    End If
End Sub
```

Modül Sonu Soruları & Alıştırmalar



1. **`Sub`** ile **`Function`** arasındaki fark nedir?
2. **`Main`** yordamı formların ve modüllerin içinde nasıl tanımlanır? Kendi **`Main`** yordamınızı yazın.
3. Yordam ve fonksiyonlar uygulamalarda kod tekrarını nasıl önler?
4. Yordam ve fonksiyonların sınırsız parametre almasını sağlayan **`ParamArrays`** neden sonda tanımlanır?
5. Farklı kültürlerde tarih, zaman, metin değerlerini göstermek için gerekli olan sınıflar ve fonksiyonlar nelerdir?
6. Yordam ya da fonksiyon içinden yordam ya da fonksiyonlar çağırılabilir mi? Uygulamasını yazın.
7. Bir yordam ya da fonksiyon kendisini (Recursive) çağırabilir mi? Uygulamasını yazın.

Modül 8: Veri Tipleri Üzerine İleri Bakış

.NET içinde tanımlanabilen veri tipleri temel (primitive) veri tipleri ya da kullanıcının tanımladığı veri tipleridir. Temel veri tipleri .NET içinde tanımlanmış ve bazı önemli özellikleri olan tiplerdir. Örneğin 32 bitlik bir sayıyı temsil eden **`Int32`** değer tipi temel bir tiptir. Bu temel tipin üzerinde aritmetik işlemler yapılabilir. **`Structure`** olarak tanımlanan kullanıcı veri tipleri üzerinde aritmetik işlemler yapılamaz.

Temel ve kullanıcı tanımlı veri tipleri, değer tipi ve referans tipi olarak ikiye ayrılır. **`Structure`** bir **değer** tipi, **`Class`** ise bir **referans** tipidir. Değer tipleri belleğin stack bölgesinde, referans tipleri heap bölgesinde depolanır. Değer tiplerinin oluşturulması ve silinmesi, sadece değerleri üzerinde işlem yapıldığı için kolaydır. Değer tipinin ömrü bittiği zaman stack yapısından hemen kaldırılır. Referans tiplerinin oluşturulması ve yok edilmesi ekstra bir performans gerektirir. Ancak iki veri tipinin de birbirlerine göre avantajları vardır.

Bu modül tamamlandıktan sonra;

- Temel ve kullanıcı tanımlı değer tiplerini tanıyacak,
- Temel ve kullanıcı tanımlı referans tiplerini tanıyacak,
- Veri tiplerinin belleği kullanımı öğrenecek,
- **`ByVal`** ile **`ByRef`** arasındaki farkı öğrenecek,
- Referans ve değer tiplerinin nerede kullanılacağını öğreneceksiniz.

Konu 1: Değer Tipleri

Değer tipindeki değişkenlerin tuttukları değerler bellekte stack yapısında bulunur. Bir değer tipindeki değişkenin, başka bir değişkene atanması, değerin olduğu gibi kopyalanması ile gerçekleşir. Dolayısıyla ne zaman bir atama işlemi yapılsa, değer tipinin bir kopyası bellekte oluşturulur. Bu durum çok karmaşık değerler ve büyük veri blokları için performansı düşürür. Ancak değer tipleri, tanımlı olduğu yerden çıkıldığında bellekten hemen silinir.

Built-In Değer Tipleri

Built-In değer tipleri olarak bahsedilecek temel tipler, .NET içinde tanımlı olan veri tipleridir. Bu değer tipleri sayıları, ondalık sayıları, **`Boolean`** değerlerini, tarih ve zaman değerlerini ve karakterleri temsil eden yapılardır. Bu tipler, tüm .NET dilleri tarafından kullanılabilir şekilde tanımlanır. Ancak Visual Basic dilinde bu değer tiplerine belirli isimlerle ulaşılır (Tablo 8.1).

Tablo 8.1: Built-In Değer Tipleri

Visual Basic	.NET Framework	Değer
`Boolean`	`System.Boolean`	`True` / `False`
`Byte`	`System.Byte`	8 bit uzunluğunda sayı
`Char`	`System.Char`	16 bit uzunluğunda Unicode karakter
`Date`	`System.DateTime`	64 bit uzunluğunda tarih zaman değeri
`Decimal`	`System.Decimal`	128 bit uzunluğunda sayı
`Double`	`System.Double`	64 bit uzunluğunda kayan tipte sayı
`Integer`	`System.Int32`	32 bit uzunluğunda sayı

Tablo 8.1: Built-In Değer Tipleri

Visual Basic	.NET Framework	Değer
`Long`	`System.Int64`	64 bit uzunluğunda sayı
`Short`	`System.Int16`	16 bit uzunluğunda sayı
`Single`	`System.Single`	32 bit uzunluğunda kayan tipte sayı

```
' Visual Basic tanımı değer tipi
Dim sayi As Short = 10

' .NET Framework tanımı değer tipi
Dim sayi2 As Int32 = 10
```

Kullanıcı Tanımlı Değer Tipleri

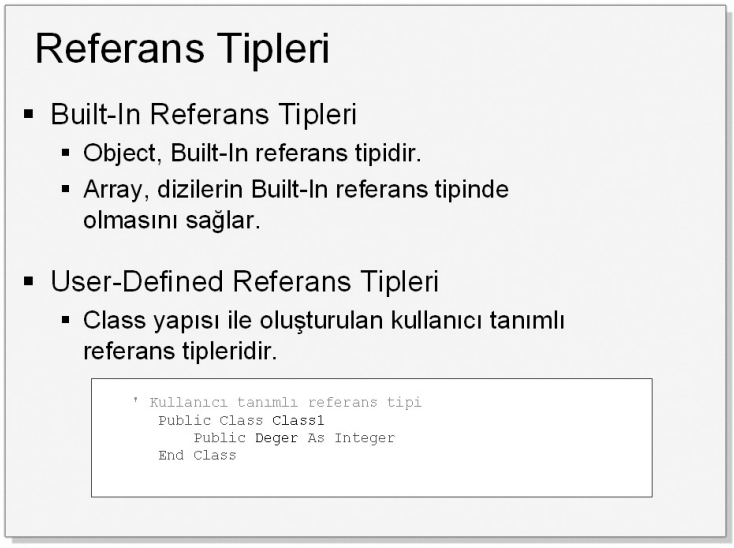
Uygulamalarda çoğu zaman Built-in değer tiplerinin sağlamadığı özel veri tiplerine ihtiyaç duyulur. Örneğin bir üçgen tipi, kenarları temsil eden üç tane sayı tutan bir değer tipi olarak oluşturulabilir.

Kullanıcı tanımlı değer tipleri Visual Basic .NET dilinde **`Structure`** ile oluşturulur.

```
Structure Ucgen
    Dim kenar1 As Integer
    Dim kenar2 As Integer
    Dim kenar3 As Integer

    Public Sub New(ByVal kenar_1 As Integer, ByVal kenar_2
As Integer, ByVal kenar_3 As Integer)
        Me.kenar1 = kenar_1
        Me.kenar2 = kenar_2
        Me.kenar3 = kenar_3
    End Sub

End Structure
```

- **`Structure`** tiplerinde en az bir veri tipi tanımlı olması gerekir.
- **`Structure`** tiplerinde boş parametreli constructor tanımlanamaz. Değer tipleri tanımlandıklarında bu constructor ile oluşturulur. Ancak bir veya daha fazla parametre alan constructor metotları kullanılabilir.
- **`Structure`** veri tipleri **`Class`** yapısına benzer, ancak değer tipi oldukları için oluşturulması ve yok edilmesi daha kolaydır.

Konu 2: Referans Tipleri



Referans tipindeki değerlere erişimler, bu değerlerin bellekte oluşturulduğu yerin adresi ile sağlanır. Bu değerler bellekteki heap bölgesinde oluşturulur. Referans tipindeki değişkenlerin, başka değişkenlere atanması işlemleri bellekteki adreslerin kopyalanması ile gerçekleşir. Dolayısıyla, aynı adresteki veriyi gösterir. Bu iki değişkenden herhangi biri değiştiğinde, diğeri de değişmiş olur.

Sınıf ve dizi yapıları referans tipleridir. Dizilerin tuttukları değerlerin sayısı çoğu zaman önceden bellidir, ama boyutları ve uzunlukları değişebilir. Dolayısıyla dizi değişkenlerinin ismi, elemanlarının bellekte tutuldukları ilk yerin adresini temsil eder. Ancak dizilerin tuttukları değerler referans tipinde olmayabilir.

Built-In Referans Tipleri

.NET içinde tanımlı olan **`Class`** ve **`Array`** yapıları **`Object`** sınıfından türetilmiştir. **`Object`** sınıfı .NET içinde tanımlı bir Built-In referans tipidir. Değişkenler tanımlandıkları sırada tipleri belirtilmezse **`Object`** tipinde oluşturulur.

Kullanıcı tarafından oluşturulan diziler, bir **`Array`** sınıfından türetilir. Bu sınıf, diziler üzerinde işlemleri kolaylaştıracak birçok özellik ve metot tanımlar. Örneğin **`Length`** özelliği, dizinin toplam eleman sayısını verir, **`Sort`** metodu ise dizideki elemanların sıralanması işlemini yapar. Dolayısıyla **`Array`** sınıfı, diziler için Built-In referans tipi sağlar.

```
' Parametre olarak verilen dizinin başlangıç adresidir.
' Dolayısıyla, bu adres bölgesinde tanımlı
' değerlere ulaşılabilir.
```

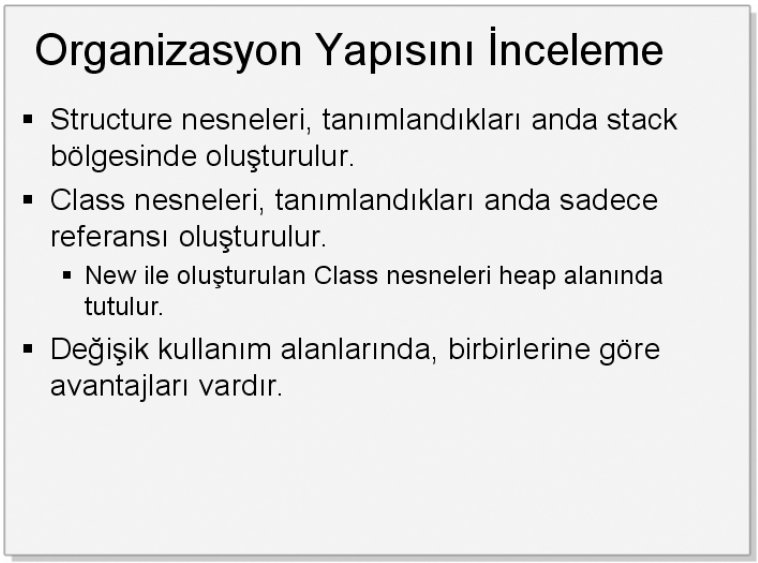
```
Sub Goruntule(ByVal dizi() As Integer)
    For i As Integer = 0 To dizi.Length - 1
        Label1.Text &= dizi(i)
    Next
End Sub
```

Kullanıcı Tanımlı Referans Tipleri

.NET içinde tanımlı sınıflar kullanılabildiği gibi, birçok nesne yönelimli programlama dilinde kullanıcılar kendi sınıflarını da oluşturabilirler. .NET Framework'te, kullanıcıların oluşturduğu bu sınıflar **`Object`** sınıfından türetilir. Dolayısıyla bu sınıflar kullanıcı tanımlı referans tipleridir.

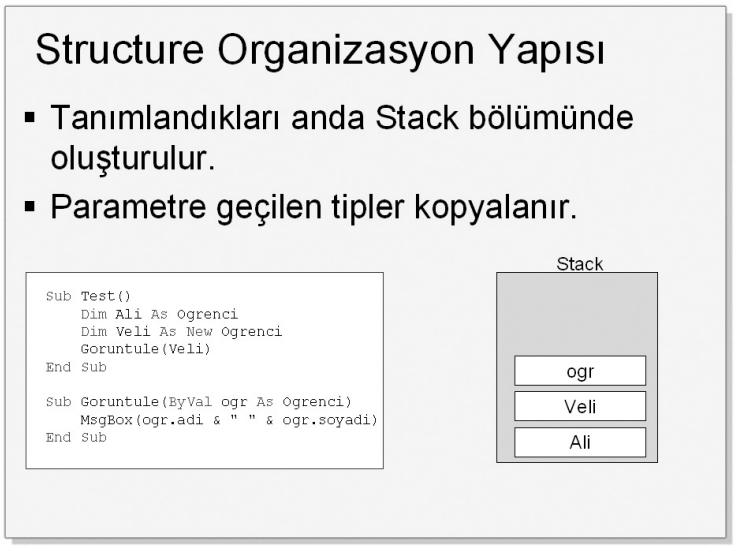
```
' Kullanıcı tanımlı referans tipi
Public Class Class1
    Public Deger As Integer
End Class

Sub Test()
    Dim sinif As New Class1
    sinif.Deger = 10

    Dim sinif2 As Class1

    ' Sinif değişkeninin tuttuğu adres bilgisi
    ' diğer değişkene aktarılır. Dolayısıyla Sinif2
    ' değişkeni de bellekte aynı yeri temsil eder.
    sinif2 = sinif

    ' Sinif değişkenin tuttuğu adres bölgesindeki
    ' değer değiştirilir.
    sinif.Deger = 15

    ' Sinif2 değişkeni de aynı adresi gösterdiği için
    ' sonuç 15 olur.
    MsgBox(sinif2.Deger)

End Sub
```

Konu 3: Organizasyon Yapısını İnceleme



`Structure` ve **`Class`** nesnelerini oluştururken, bu nesnelerin bilgilerini tutmak için bellek alanında yer açılır. Açılan yer için, **`Structure`** veya **`Class`** içindeki veri tiplerinin boyutu kadar kaynak gerekir. **`Structure`** bir veri tipi olduğu için, tanımlandığı anda veya parametre geçilirken belleğin stack alanında oluşturulur. Buna karşın, **`Class`** nesnesi referans tipi olduğu için, ancak **`New`** anahtar kelimesi ile tanımlandığı zaman belleğin heap alanında oluşturulur. Bu iki yapının, değişik kullanım alanlarında birbirlerine göre avantajları vardır.

Structure Organizasyon Yapısı ve Belleğin İncelenmesi



Structure veri tipi, değer tipi olduğu için, tanımlandığı anda bellekte **stack** bölümünde oluşturulur. Bellekte ayrılan yer, **Structure** içinde tanımlı olan Built-In veri tiplerinin toplam boyutu kadardır.

Visual Basic .NET dilinde **Structure** veri tipleri **New** anahtar kelimesiyle de oluşturulabilir. Ancak bu durumda constructor metotları parametre alacak şekilde tanımlanmalıdır. Varsayılan parametresiz constructor metotları CLR tarafından işlenir. Dikkat edilmesi gereken bir durum da, **New** ile oluşturulan değişkenlerin yine bir değer tipi olması ve **stack** alanında tutulmasıdır.

NOT **Sınıflardan nesne oluştururken** New **anahtar kelimesi kullanılır, ancak bu nesneler heap alanında tutulur.**

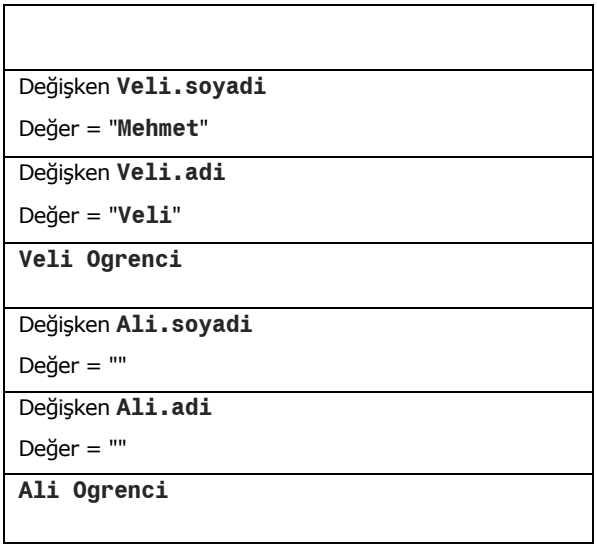
```
Public Structure Ogrenci
        Public adi As String
        Public soyadi As String

        Public Sub New(ByVal isim As String, ByVal soyisim
As String)
            Me.adi = isim
            Me.soyadi = soyisim
        End Sub

    End Structure

    Sub Test()
```

```
    ' 1 - Öğrenci değeri tanımlandığı sırada
    ' stack alanında yer ayrılır
    Dim Ali As Ogrenci

    ' 2 - New ile tanımlanan değişkenler de stack
    ' alanında oluşturulur.
    ' Farkı, bu değişkenin başlangıç değeri almasıdır.
    Dim Veli As New Ogrenci("Veli", "Mehmet")

    ' 3 - Parametre olarak sipariş nesnesinin
    ' adresi verilir
    Goruntule(Veli)
End Sub

Sub Goruntule(ByVal ogr As Ogrenci)
    MsgBox(ogr.adi & " " & ogr.soyadi)
End Sub
```

1. **`Ali`** değişkeni tanımlanırken Stack yapısı

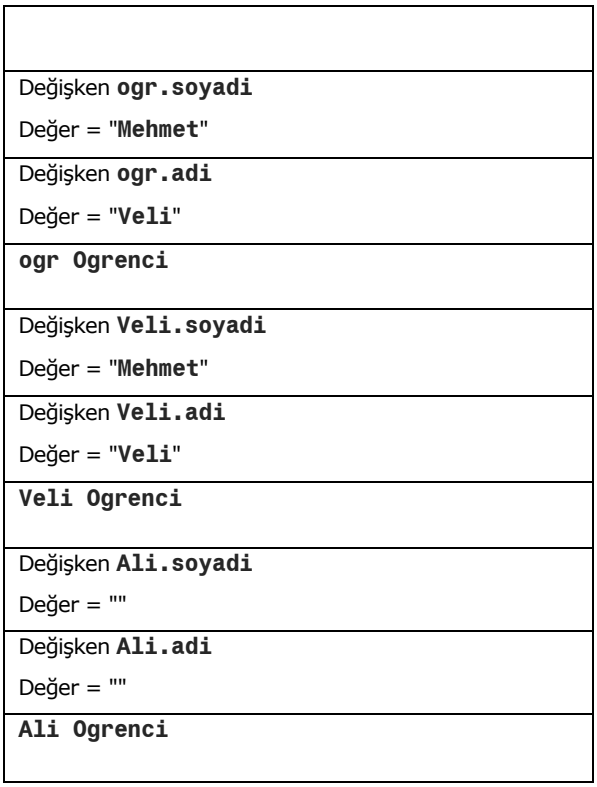
Değişken **`Ali.soyadi`** Değer = ""
Değişken **`Ali.adi`** Değer = ""
`Ali Ogrenci`

2. **Veli** değişkeni tanımlanırken Stack yapısı



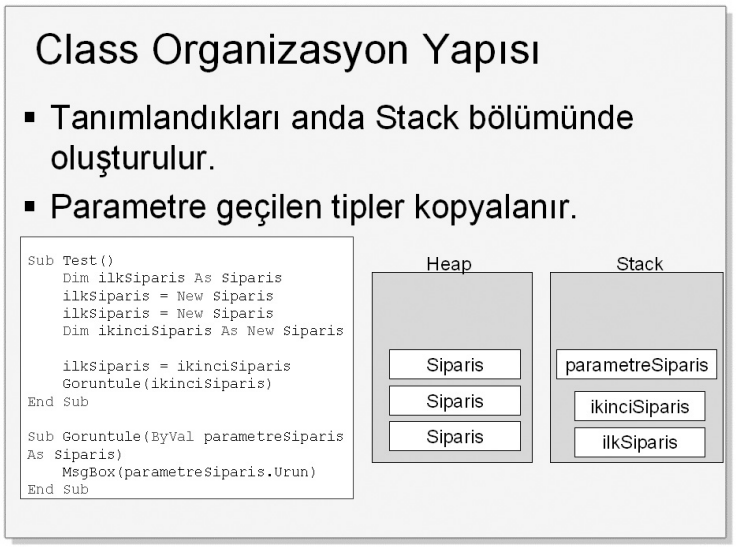
Bu değişkenler oluşturulduktan sonra, bir yordama parametre olarak verildiklerinde, tüm değerleri kopyalanır. Değer tipindeki değişkenler, atama işlemlerinde oldukları gibi kopyalanır.

3. **Goruntule** yordamı çağrıldığı zaman Stack yapısı



Test isimli yordamdan çıkılınca, bu değişkenler oluşturuldukları sırayla stack yapısından kaldırılır.

Class Organizasyon Yapısı ve Belleğin İncelenmesi



Class'lardan (Sınıf) nesneler oluşturuldukları zaman bu nesnelerin değerleri **heap** bölgesinde tutulur. Ancak bu nesneleri gösteren bir adres tutucusu oluşturulur ve bu adresin değeri de **stack** alanında depolanır.

```
    Public Class Siparis
        Public Tarih As Date
        Public AliciIsmi As String
        Public Urun As String

        Public Sub New()

        End Sub

        Public Sub New(ByVal Tarih As Date, ByVal Isim As
String, ByVal Urun As String)
            Me.Tarih = Tarih
            Me.AliciIsmi = Isim
            Me.Urun = Urun
        End Sub
    End Class


    Sub Test()
        ' 1 - Sipariş referansı oluşturulur
```

```
        Dim ilkSiparis As Siparis

        ' 2 - Yeni bir nesne oluşturulup,adresi
        ' bu referansa aktarılır
        ilkSiparis = New Siparis(Now, "Enis Günesen",
"Visual Studio.NET 2003")

        ' 3 - Yeni bir nesne daha oluşturulur ve adresi
        ' değişkene aktarılır
        ilkSiparis = New Siparis(Now.AddYears(-1), "Enis
Günesen", "Visual Studio.NET 2002")

        ' 4 - Yeni bir referans ve nesne oluşturulur
        Dim ikinciSiparis As New Siparis

        ' 5 - İki değişkenin aynı bellek alanını göstermesi
        ' sağlanır
        ilkSiparis = ikinciSiparis

        ' 6 - Bu alandaki Ürün ismi değiştirilir
        ilkSiparis.Urun = "BilgeAdam Yazılım Uzmanlığı"

        ' 7 - Parametre olarak sipariş nesnesinin
        ' adresi verilir
        Goruntule(ikinciSiparis)
    End Sub

    Sub Goruntule(ByVal parametreSiparis As Siparis)
        MsgBox(parametreSiparis.Urun)
    End Sub
```

1. **`ilkSiparis`** tanımlanırken Stack alanı

Değer = **0x00000000** (Bellekte boş bir alanı gösterir)
İlkSiparis

Heap alanı

Nesne	**Adres Bilgisi**

2. **ilkSiparis** oluşturulurken Stack alanı

Değer = **0x00000012**
ilkSiparis

Heap alanı

Nesne	**Adres Bilgisi**
Siparis Tarih = **10.05.2005** AliciIsmi = **Enis Günesen** Urun = **Visual Studio.NET 2003**	**0x00000012**

3. **ilkSiparis** referansına başka bir nesne verilirken Stack alanı

Değer = **0x00000056** (Gösterdiği adres değeri değişir)
ilkSiparis

Heap alanı

Nesne	**Adres Bilgisi**
Siparis Tarih = **10.05.2004** AliciIsmi = **Enis Günesen** Urun = **Visual Studio.NET 2002**	**0x00000056**
Siparis Tarih = **10.05.2005** AliciIsmi = **Enis Günesen** Urun = **Visual Studio.NET 2003**	**0x00000012** (Bu adres alanına artık hiçbir referans ulaşmıyor)

4. **`ikinciSiparis`** oluşturulurken Stack alanı

Değer = **0x00000088**
ikinciSiparis
Değer = **0x00000056**
ilkSiparis

Heap alanı

Nesne	**Adres Bilgisi**
Siparis Tarih = AliciIsmi = Urun =	**0x00000088**
Siparis Tarih = **10.05.2004** AliciIsmi = **Enis Günesen** Urun = **Visual Studio.NET 2002**	**0x00000056**
Siparis Tarih = **10.05.2005** AliciIsmi = **Enis Günesen** Urun = **Visual Studio.NET 2003**	**0x00000012**

5. **`ikinciSiparis`**'in adres bilgisi **`ilkSiparis`**'e atanırken Stack alanı

Değer = **0x00000088**
ikinciSiparis
Değer = **0x00000088** (Gösterdiği adres ikinci sipariş referansı ile aynı olur)
ilkSiparis

Heap alanı

Nesne	**Adres Bilgisi**
Siparis Tarih = AliciIsmi = Urun =	**0x00000088**
Siparis Tarih = **10.05.2004** AliciIsmi = **Enis Günesen** Urun = **Visual Studio.NET 2002**	**0x00000056** (Bu nesneyi gösteren referans kalmamış-tır)
Siparis Tarih = **10.05.2005** AliciIsmi = **Enis Günesen** Urun = **Visual Studio.NET 2003**	**0x00000012**

6. **ilkSiparis**'in gösterdiği nesnenin ürün ismi değiştirilirken Stack alanı

Değer = **0x00000088**
ikinciSiparis
Değer = **0x00000088**
ilkSiparis

Heap alanı

Nesne	Adres Bilgisi
Siparis Tarih = AliciIsmi = Urun = **BilgeAdam Yazılım Uzmanlığı**	**0x00000088**
Siparis Tarih = **10.05.2004** AliciIsmi = **Enis Günesen** Urun = **Visual Studio.NET 2002**	**0x00000056** (Garbage Collector, referanslarını kaybetmiş nesneleri bellekten siler)
Siparis Tarih = **10.05.2005** AliciIsmi = **Enis Günesen** Urun = **Visual Studio.NET 2003**	**0x00000012** (Garbage Collector, referanslarını kaybetmiş nesneleri bellekten siler)

7. **ikinciSiparis**'in gösterdiği değer **Goruntule** yordamı ile gösterilirken Stack alanı

Değer = **0x00000088**
parametreSiparis
Değer = **0x00000088**
ikinciSiparis
Değer = **0x00000088**
ilkSiparis

Heap alanı

Nesne	Adres Bilgisi
Siparis Tarih = AliciIsmi = Urun = **BilgeAdam Yazılım Uzmanlığı**	**0x00000088**

Sonuç olarak gösterilen değer, heap alanında **0x00000088** adres numaralı nesnenin ürün ismi olur. Parametre olarak verilen nesneler heap alanında tekrar oluşturulmazlar. Referans olarak geçen değişkenler aslında değer tipleridir. Ancak nesnenin tümü değil, sadece adres değerinin kopyası oluşturulur.

ByVal ve ByRef İncelemesi

Fonksiyon ve yordamlara parametre verilirken, varsayılan durumda değişkenlerin değerleri verilir. Parametre olarak verilen değişkenler üzerinde değişiklik yapılması için, bu parametrelerin bulunduğu adres bilgilerine ihtiyaç vardır. Referans tipindeki değerler parametre olarak geçildiklerinde, referansları verilir. Ancak değer tipleri parametre olarak verildiklerinde bu değerler kopyalanır ve asıl değişkenin tuttuğu değere ulaşılamaz. Bu karışıklıkları çözmek için, yordamlarda parametreler **`ByVal`** ve **`ByRef`** olarak belirtilir.

`ByVal`, parametre olarak verilecek değişkenin değeri ile işlem yapılacağını belirtir. Dolayısıyla bu parametrenin değeri değiştirilemez.

```
    ' Değişecek olan kelime ByVal olarak verilmiştir
    Sub Ekle(ByVal Kelime As String, ByVal eklenecek As
String)
        Kelime = Kelime.Insert(0, eklenecek)
    End Sub


    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim mesaj As String = "Hello"

        Ekle(mesaj, " World")
        MsgBox(mesaj)
    End Sub
```

mesaj değişkenin değeri, yordama değer olarak verilmiştir. Dolayısıyla yordamın üzerinde çalıştığı değer, **mesaj**'ın bir kopyasıdır. Bellek alanında fiziksel olarak farklı yerlerde dururular. Yani değişiklik yapılan değer, sadece bir kopyadır. Yordam sonlandığında kopya olarak oluşturulan değer silinecek ve asıl değer değişmemiş olarak kalacaktır. Bu durumda parametre olarak **mesaj** değişkeninin adresi verilmelidir. Dolayısıyla yordamdaki parametrenin **ByRef** olarak tanımlanması gerekir.

```
    Sub Ekle(ByRef Kelime As String, ByVal eklenecek As
String)
        Kelime = Kelime.Insert(0, eklenecek)
    End Sub
```

Parametre olarak referans tipinde bir değişken verilirse **ByVal** veya **ByRef** arasında bir fark olmaz. Referans tipleri, nesnelerin bulunduğu heap alanlarının adresini tutar. Dolayısıyla **ByVal** olarak tanımlanan parametreye referans tipinin değeri (adres bilgisi) kopyalanır. Bu kopya üzerinden aynı adres alanında değişiklik yapılır.

```
    ' Parametre ByVal ile tanımlı olsa dahi, değiştirir
    ' dizinin belirtilen index'teki değeri,
    ' adres olarak erişilir.

    Sub ElemanDegistir(ByVal dizi() As Integer, ByVal index
As Integer, ByVal yeniDeger As Integer)
        dizi(index) = yeniDeger
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim sayilar(2) As Integer

        sayilar(0) = 111
        sayilar(1) = 222
        sayilar(2) = 333

        ElemanDegistir(sayilar, 1, 1000000)
        MsgBox(sayilar(1))
        ' Sonuç = 1000000
    End Sub
```

Modül Sonu Soruları & Alıştırmalar

1. Değer değişkenleri ve referans değişkenleri arasındaki farkı açıklayın. Her iki değişken tipinin de yer aldığı parametreleri içeren bir yordam yazın. Değişkenlerin verilerinin değişip değişmediğinin gözlemleyin.
2. **`Structure`** yapısını açıklayın. Kompleks bir veri tipini **`Stucture`** yapısında kodlayın. Yazılan veri tipinin uzunluğunu hesaplayın.

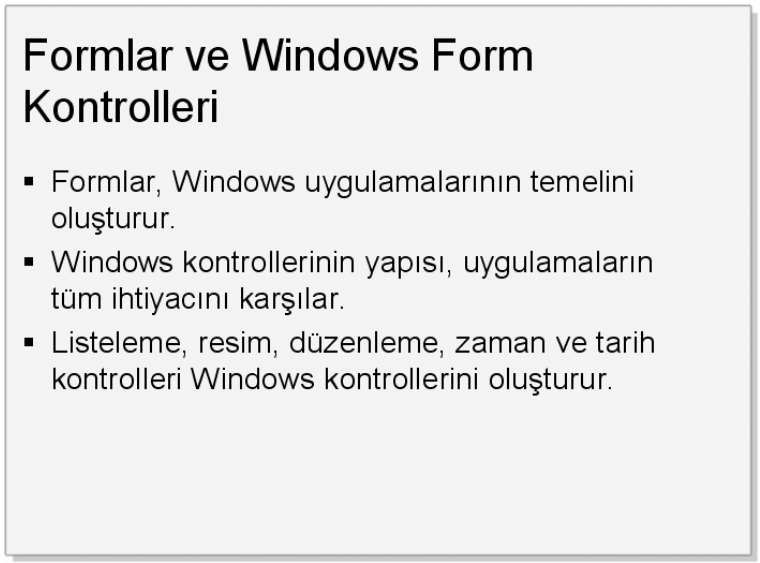
Modül 9: Windows Programlama

Visual Basic.NET ile Windows Tabanlı Programlama adlı Modül 4'te Windows formlarına ve kontrollerine giriş yapılmıştı. .NET Framework'te, Windows uygulamalarının görünüm ve kullanım zenginliğini artırmak için birçok kontrol vardır. Visual Studio ile varsayılan durumda gelen kontrollerin dışında birçok kontrol de Windows uygulamalarına eklenebilir.

Bu modül tamamlandıktan sonra;

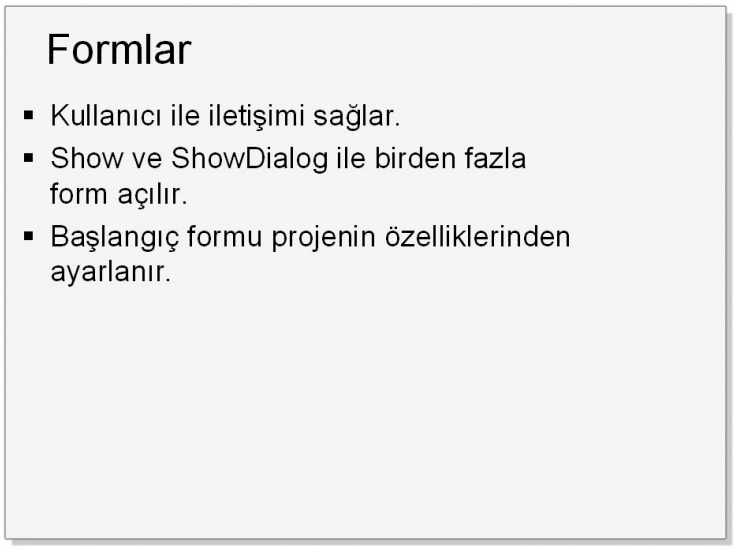
- **`ListBox`**, **`TreeView`** ve **`ComboBox`** gibi listeleme kontrollerini tanıyacak,
- **`PictureBox`** ve **`ImageList`** gibi resim kontrollerini tanıyacak,
- **`TabControl`**, **`Panel`**, **`HScrollBar`** ve **`VScrollBar`** gibi düzenleme kontrollerini tanıyacak,
- **`DateTimePicker`** ve **`MonthCalendar`** gibi zaman ve tarih kontrollerini tanıyacak,
- Çalışma anında forma yeni kontroller oluşturup ekleyebileceksiniz.

Konu 1: Formlar ve Windows Forms Kontrolleri



Windows uygulamalarının temelini Windows Form nesneleri oluşturur. Windows kontrolleri, kullanıcıya zengin uygulamalar geliştirmek için kolaylık sağlar. Bu kontroller, bir uygulamanın tüm ihtiyacını karşılayacak şekilde tasarlanmıştır. Listeleme kontrolleri, kullanıcıya bir dizi öğeyi değişik biçimlerde listelemeyi sağlar. Resim ve düzenleme kontrollerinin, forma görsel zenginlik sağlayan birçok özelliği vardır. Zaman ve tarih kontrollerinin yapısı, zaman ve tarih seçme işlemlerini kolaylaştırır. Bu Windows kontrolleri, tasarım anında eklenebileceği gibi, çalışma anında da eklenebilir.

Form Nesnesi



Windows uygulamaları, kullanıcı ile iletişimi **Form** nesneleri ile sağlar. Formlar, görünüm özellikleri, pencere stili değiştirilerek ve üzerine kontroller eklenerek özelleştirilir. Ayrıca birden çok **Form** nesnesi kullanılarak, uygulamalar zenginleştirilir.

Birden Fazla Form Oluşturmak

Windows uygulamaları birden fazla **Form** nesnesinden oluştuğu için, projelere form eklemek her zaman gereklidir. Bir Windows projesine yeni bir form eklemek için aşağıdaki adımları izleyin:

1. Solution Explorer panelinden projeyi sağ tıklayarak ya da Project menüsünden Add Windows Form komutunu seçin.
2. Açılan menüden Windows Form öğesinin seçili olduğunu kontrol edin ve bir isim vererek formu ekleyin.

Başlangıç formlarının ayarlanmasının yanı sıra, uygulamada bir formdan başka bir formun açılması ve ayarlanması sık karşılaşılan bir durumdur. **Form** nesneleri, **System.Windows.Forms** ad uzayı içinde bulunan **Form** sınıfından türetilmiş sınıflardır. Dolayısıyla yeni bir form oluşturmak için, istenen **Form** sınıfından bir nesne oluşturulması yeterlidir.

```
Dim yeniForm As New frmYeni
```

Yeni oluşturulan formların gösterilmesi, formun **Show** ve **ShowDialog** metotları ile yapılır. **ShowDialog** metodu, form gösterildikten sonra, kapanana kadar di-

ğer formlara erişimi engeller. **ShowDialog** metodundan sonra yazılan kodlar, form kapandıktan sonra çalıştırılır.

```
Dim yeniForm As New frmYeni
yeniForm.ShowDialog()

' Bu kodlar yeniForm kapandıktan sonra çalıştırılır
MsgBox("Form kapandı...")
```

ShowDialog ile gösterilen formlar, hangi durum ile kapandıklarını belirten bir **DialogResult** sonucu döndürürler. Bu kullanım **MsgBox** hazır fonksiyonu ile aynıdır.

```
Dim frm As New frmSatis
If frm.ShowDialog = DialogResult.Yes Then
    ' Verileri kaydet
End If
```

Formun hangi diyalog sonucu ile döneceğini, üzerindeki **Button** kontrollerinin **DialogResult** özelliği ile belirlenir. Eğer düğmenin bu özelliği **Yes** olarak ayarlanmışsa, bu düğme tıklanıp form kapatıldığı zaman, **DialogResult.Yes** değerini döndürür.

Windows uygulamalarının başlangıç nesnesi **Sub Main** olarak seçilebilir. Bu durumda, uyguluma yüklenirken herhangi bir form yerine **Main** yordamı aranır. Bu yordam ayrı bir modül içine yazılabilir, ancak bu modüldeki **Main** yordamı sonlandığı zaman uygulama da sonlanır. Dolayısıyla, bu yordamdan açılan formların **ShowDialog** metodu ile açılması gerekir.

Örneğin, bir Windows uygulamasının kullanıcının girdiği verilere göre değişik formları açması için, başlangıç nesnesinin **Sub Main** olarak ayarlanması gerekir. Bu yordamda, kullanıcının istediği form dinamik olarak yüklenir.

```
Module Giris
    Sub Main()
        Dim grup, parola As String

        grup = InputBox("Kullanıcı grubu:")
        parola = InputBox(grup & " grubuna giriş için parola
girin:")
        ' Grupların parolası kontrol edilir
        ' ve ilgili grubun formu açılır.
        ' Eğer parola veya grup ismi yanlış girilirse
        ' hata formu yüklenir.

        Select Case grup.ToUpper
            Case "SATIŞ"
                If parola.ToUpper <> "SATİS_PAROLA" Then
```

```
                    HataFormuYukle("Satış departmanı
parolası yanlış!")
                Else
                    Dim satisDepartmani As New frmSatis
                    satisDepartmani.ShowDialog()
                End If

            Case "YÖNETİM"
                If parola.ToUpper <> "YONETİM_PAROLA" Then
                    HataFormuYukle("Yönetim departmanı
parolası yanlış!")
                Else
                    Dim yonetimDepartmani As New frmYonetim
                    yonetimDepartmani.ShowDialog()
                End If

            Case Else
                HataFormuYukle(grup & " isminde bir grup
bulunamadı")

        End Select
    End Sub

    ' Hata formu, verilen parametredeki mesajı
    ' gösterecek şekilde ayarlanır ve yüklenir.
    Sub HataFormuYukle(ByVal mesaj As String)
        Dim hataFormu As New frmHata
        hataFormu.lblHataMesaji.Text = mesaj
        hataFormu.ShowDialog()
    End Sub
End Module
```

Formların üzerlerindeki kontroller, form sınıflarının birer üyesi oldukları ve **Friend** erişim seviyesinde tanımlandıkları için, aynı projeden ulaşılabilirler. Böylece, aynı Windows projesi içinde açılan formlar açılmadan önce kontrollerinin özellikleri değiştirilebilir. Örneğin, hata formu gösterilmeden önce, üzerindeki **Label** kontrolünün **Text** özelliği ilgili hata mesajını gösterecek şekilde ayarlanabilir.

Form Özellikleri

TABLO 9.1: `Form` Özellikleri

Özellik	Değer Tipi	Açıklama
`AcceptButton`	`Button`	Form üzerinde Enter tuşuna basıldığı zaman "tıklanacak" `Button` kontrolü
`CancelButton`	`Button`	Form üzerinde Esc tuşuna basıldığı zaman "tıklanacak" `Button` kontrolü
`Opacity`	`Double`	Formun şeffaflık oranı (0 -1 arası)
`MaximizeBox`	`Boolean`	Maximize düğmesinin görünürlüğü
`MinimizeBox`	`Boolean`	Minimize düğmesinin görünürlüğü
`ControlBox`	`Boolean`	Close, Maximize ve Minimize düğmelerinin tümünün görünürlüğü
`StartPosition`	`FormStartPosition`	Form açıldığı zaman, ekran üzerindeki konumu
`TopMost`	`Boolean`	Formun tüm pencerelerin üzerinde gözükmesi
`FormBorderStyle`	`FormBorderStyle`	Formun kenar stili
`MaximumSize`	`Size`	Formun alabileceği maksimum büyüklük
`MinimumSize`	`Size`	Formun alabileceği minimum büyüklük

Form Olayları

TABLO 9.2: `Form` Olayları

Olay	Açıklama
`Click`	Form üzerine tıklandığı zaman gerçekleşir
`Closing`	Form kapanmadan hemen önce gerçekleşir
`Closed`	Form kapandıktan sonra gerçekleşir
`Load`	Form yüklenirken gerçekleşir
`KeyDown`	Form üzerindeyken bir tuşun basılması ile gerçekleşir
`KeyUp`	Basılan tuşun kaldırılması ile gerçekleşir

Form Metotları

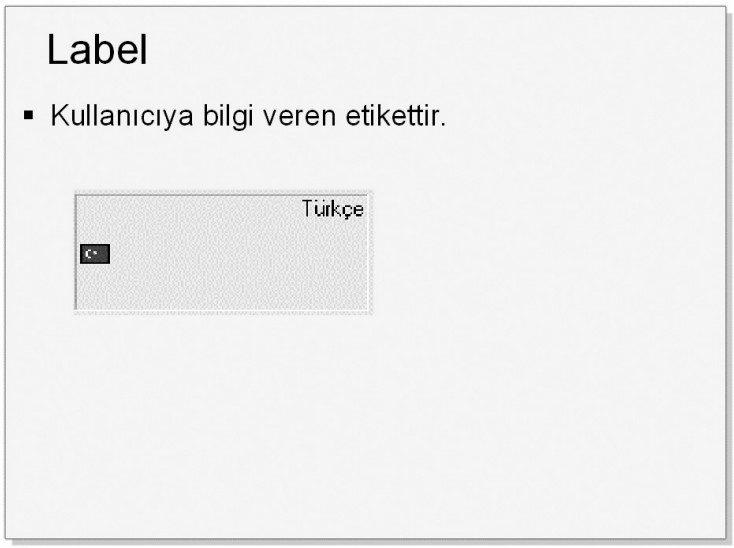
TABLO 9.3: `Form` Metotları

Metot	Açıklama
Hide	**Visible** özelliğini **False** yaparak formu gizler
Close	Formu kapatır. Eğer form başlangıç formuysa uygulama sonlanır
Show	Formu gösterir. **Hide** ile gizlenmişse, **Visible** özelliği **True** yapılır.
ShowDialog	Formu iletişim kutusu olarak gösterir.

Örnek: Bir Windows formunun kapanmasını yönetmek için, o formun **Closing** olayına ve **Close** metoduna ihtiyaç vardır. Kapanmasını yavaşlatmak için bir **Timer** kontrolü kullanılır ve formun şeffaflığı yavaşça azaltılır.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    Me.Text = "Hoşgeldiniz... " & TimeOfDay
End Sub

Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles
MyBase.Closing
    ' Kapanma olayı gerçekleşmeden önce iptal edilir
    e.Cancel = True
    Timer1.Start()
End Sub

Private Sub Form1_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles MyBase.KeyDown
    ' Shift+Ctrl+F3 tuşlarına basıldığında uygulama kapanır
    If e.Shift And e.Control And e.KeyCode = Keys.F3 Then
         Me.Close()
    End If
End Sub

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Timer1.Tick
    ' Form görünmez hale gelince uygulama kapanır
    If Me.Opacity = 0 Then
        Application.Exit()
    Else
        Me.Opacity -= 0.1
    End If
End Sub
```

Label



Label kontrolü form üzerinde kullanıcıya bilgi vermek amacıyla kullanılan etikettir.

Label Özellikleri

TABLO 9.4: **Label** Özellikleri

Özellik	Değer Tipi	Açıklama
TextAlign	**ContentAlignment**	Yazının etiket üzerindeki pozisyonu belirler.
BorderStyle	**BorderStyle**	Kontrolün kenar stilidir. **FixedSingle** değeri, kontrolün kenar çizgilerini gösterir. **Fixed3D** değeri, kenarların üç boyutlu olmasını sağlar.
Image	**Drawing.Image**	Etiket üzerinde görüntülenmek istenen resmi tutar.
ImageAlign	**ContentAlignment**	Etiket üzerindeki resmin nerede duracağını belirler.
RightToLeft	**RightToLeft**	Etiket üzerindeki yazının yönünü belirler. Eğer **Yes** değerini alırsa, yazılar sağdan sola gösterilir

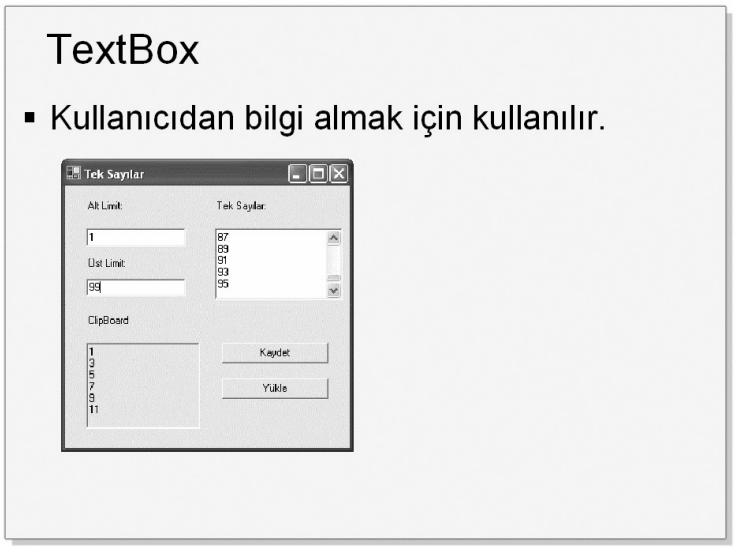
```
Label1.BorderStyle = BorderStyle.Fixed3D

' Visual Studio klasörü altındaki simgeler kullanılabilir
Label1.Image = Image.FromFile("C:\Program Files\ _
```

```
Microsoft Visual Studio .NET
2003\Common7\Graphics\icons\Flags\FLGTURK.ICO")

Label1.ImageAlign = ContentAlignment.MiddleRight
Label1.RightToLeft = RightToLeft.Yes
Label1.Text = "Türkçe"
```

RESİM 9.1: `Label` kontrolü örneği.

NOT **Resmin bulunduğu yer kontrolün sağ tarafında bulunacak şekilde ayarlanmasına rağmen, sol tarafta gözükür. Bu durum,** `RightToLeft` **özelliğinin** `Yes` **olarak atanmasından kaynaklanır.**

TextBox



Metin kutuları, kullanıcıdan bilgi almak için kullanılır.

TextBox Özellikleri

TABLO 9.4: `TextBox` Özellikleri

Özellik	Değer Tipi	Açıklama
`MultiLine`	`Boolean`	Metin kutusuna birden fazla satırda değer girilebilmesini sağlar. **`False`** durumunda ise, metin kutusunun yüksekliği değiştirilemez
`ScrollBars`	`ScrollBars`	Metin kutusunda kaydırma çubuklarının görünmesini kontrol eder. Varsayılan durumda kaydırma çubuğu görüntülenmez, ancak **`Horizontal`**, **`Vertical`** kaydırma çubukları ya da ikisi birden gösterilebilir.
`PasswordChar`	`Char`	Metin kutusuna parola girilecekse, girilen karakterlerin hangi karakter olarak görüneceğini belirler.
`WordWrap`	`Boolean`	Metin kutusuna girilen değerlerin, satır sonlandığında bir alt satıra geçmesini belirtir. Eğer **`MultiLine`** özelliği **`False`** ise, alt satırlar tanımlı olmayacağı için bu özelliğin bir etkisi görülmez.
`MaxLength`	`Integer`	Metin kutusunun alabileceği maksimum karakter sayısını belirtir.

TABLO 9.4: `TextBox` Özellikleri

Özellik	Değer Tipi	Açıklama
`ReadOnly`	`Boolean`	Metin kutusunun yazmaya karşı korumalı olduğunu belirtir.
`CharacterCasing`	`CharacterCasing`	Metin kutusuna karakterler girilirken büyük veya küçük harfe çevrilmesini sağlar. `Upper` değeri büyük, `Lower` değeri küçük harfe çevrimi sağlar.

TextBox Olayları

TABLO 9.5: `TextBox` Olayları

Olay	Açıklama
`TextChanged`	Metin kutusundaki yazı değiştiği zaman gerçekleşir.

TextBox Metotları

TABLO 9.6: `TextBox` Metotları

Metot	Açıklama
`Cut`	Seçilen karakterleri siler, ancak hafızada tutar.
`Copy`	Seçilen karakterleri kopyalar.
`Paste`	Hafızaya alınan karakterleri metin kutusuna yapıştırır.
`Clear`	Metin kutundaki yazıları temizler.
`SelectAll`	Metin kutusundaki tüm yazıyı seçer.

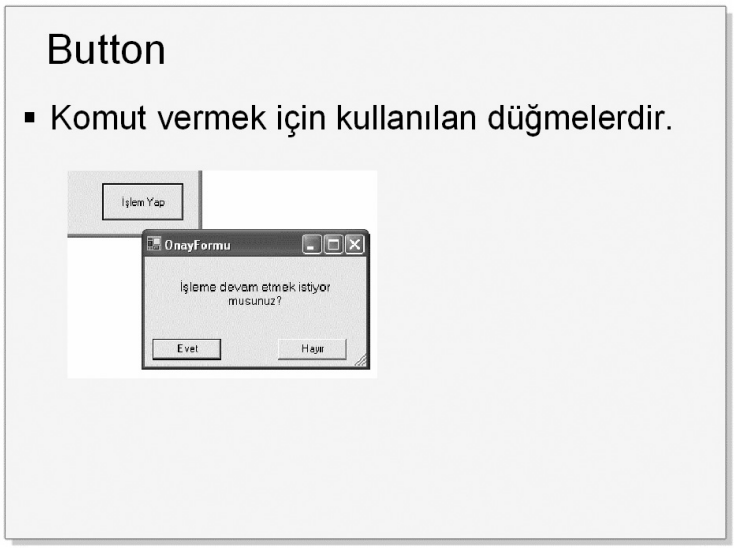
Örnek: Form üzerinde girilen değerlere göre tek sayıların hesaplanması ve görüntülenmesi işlemi için `TextBox` kontrolünün birçok olayından ve özelliğinden yararlanılır.

RESİM 9.2: `TextBox` kontrolü örneği.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    ' Form yüklenirken kontrollerin ayarlanması:
    txtAltSayi.MaxLength = 2
    txtUstSayi.MaxLength = 4

    txtSayilar.Multiline = True
    txtSayilar.ScrollBars = ScrollBars.Vertical

    txtClipBoard.ReadOnly = True
    txtClipBoard.Multiline = True
End Sub

' Bu olay hem txtUstSayi hem de txtAltSayi kontrolünün
' TextChanged olayında gerçekleşir.
' Handles ifadesinden sonra kontroller virgülle ayrılmıştır
Private Sub txtUstSayi_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
txtUstSayi.TextChanged, txtAltSayi.TextChanged
    TekSayiYazdir()
End Sub

Function Kontrol() As Boolean
    ' Metin kutularına sayı girildiyse
    If IsNumeric(txtUstSayi.Text) And
IsNumeric(txtAltSayi.Text) Then
        ' ve alt limit 0'dan büyük, ve üst limitten küçükse
        Dim ust As Integer = txtUstSayi.Text
        Dim alt As Integer = txtAltSayi.Text
        If ust > alt And alt > 0 Then
            ' giriş doğru yapılmıştır
            Return True
        End If
    End If

    ' Kod buraya gelirse, giriş yanlış yapılmıştır
    Return False
End Function

Sub TekSayiYazdir()
    If Not Kontrol() Then Exit Sub

    txtSayilar.Clear()
    Dim alt As Integer = txtAltSayi.Text
    Dim ust As Integer = txtUstSayi.Text
```

```
        ' Sayılar metin kutusuna, tek sayıların yazdırılması
        For i As Integer = alt To ust
            If i Mod 2 = 1 Then
                txtSayilar.Text &= i & vbCrLf
            End If
        Next
    End Sub

    ' Sayıların txtClipboard isimli metin kutusuna kaydedilmesi:
    Private Sub btnKaydet_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles btnKaydet.Click
        txtClipBoard.Text = txtSayilar.Text

        ' Sayıların kopyalanması için, önce seçilmesi gerekir
        txtSayilar.SelectAll()
        txtSayilar.Cut()
    End Sub

    ' Cut yordamı çağrıldıktan sonra veriler kopyalanır.
    ' Paste ile bu kopyalanan veriler geri yazdırılır.
    Private Sub btnYukle_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles btnYukle.Click
        txtSayilar.Clear()
        txtSayilar.Paste()
    End Sub
```

Button



Windows uygulamalarında, form üzerinde komut düğmeleri olarak kullanılır.

Button Özellikleri

TABLO 9.7: `Button` Özellikleri

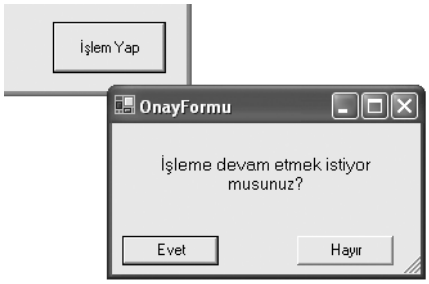
Özellik	Değer Tipi	Açıklama
`DialogResult`	`DialogResult`	Ait olduğu form **`ShowDialog`** metodu ile çağrıldığı zaman, dönüş değerini belirler
`FlatStyle`	`FlatStyle`	Düğme tıklandığında ve düğmenin üzerine gelindiğinde görünen formatı belirler

Button Olayları

TABLO 9.8: `Button` Olayları

Olay	Açıklama
`Click`	Düğme üzerine tıklandığı zaman gerçekleşir

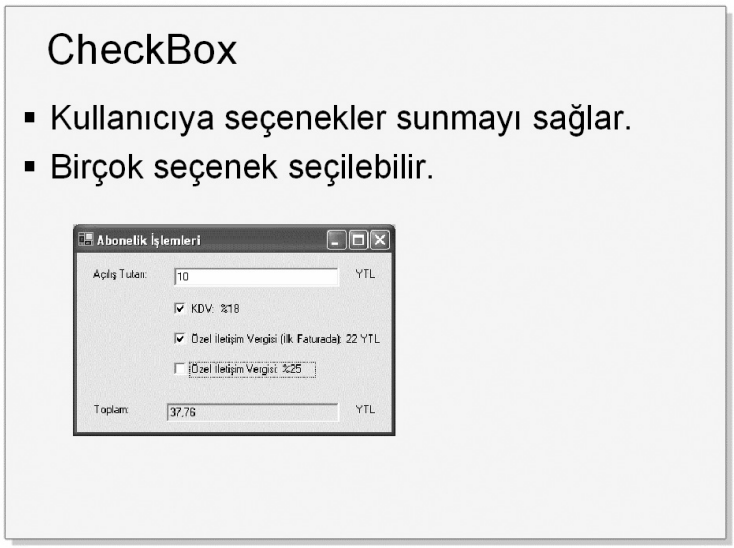
Örnek: Bir formun üzerindeki düğmelerin **`DialogResult`** özellikleri değiştirilerek, özel bir mesaj kutusu tasarlanabilir.



RESİM 9.3: **`Button`** kontrolü örneği.

```
Private Sub btnIslemYap_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnIslemYap.Click
    Dim onay As New OnayFormu

    With onay.btnHayir
        .DialogResult = DialogResult.No
        .FlatStyle = FlatStyle.Flat
    End With

    With onay.btnEvet
        .DialogResult = DialogResult.Yes
        .FlatStyle = FlatStyle.Flat
    End With

    If onay.ShowDialog = DialogResult.Yes Then
        ' Kayıt işlemleri...
    End If
End Sub
```

CheckBox



Kullanıcının birçok seçeneği birden seçmesi için kullanılır.

CheckBox Özellikleri

TABLO 9.9: `CheckBox` Özellikleri

Özellik	Değer Tipi	Açıklama
`Checked`	`Boolean`	Kontrolün seçili olup olmadığını belirler.
`CheckAlign`	`ContentAlignment`	Seçme kutusunun ve üzerinde yazan metnin birbirlerine göre konumlarını belirler.
`Appearance`	`Appearance`	Kontrolün seçme kutusu ya da düğme şeklinde olmasını belirler.
`ThreeState`	`Boolean`	Seçili olup olmaması dışında, **`Intermediate`** durumu da eklenir. Eğer kontrol **`Intermediate`** durumundaysa **`Checked`** özelliği **`True`** olur.
`AutoChecked`	`Boolean`	Kontrolün tıklandığı zaman seçili duruma geçileceğini belirtir. Eğer bu özellik **`False`** ise, kontrolün durumunu değiştirmek için, **`Click`** olayında, **`Checked`** özelliğini güncellemek gerekir.

CheckBox Olayları

TABLO 9.10: `CheckBox` Olayları

Olay	Açıklama
`CheckChanged`	Seçme kutusunun durumu değiştiği zaman gerçekleşir.

Örnek: Bir GSM şebekesinden faturalı hat açılışında toplam tutar hesaplanırken, bazı seçenekler `CheckBox` kontrolleri ile sunulabilir.

RESİM 9.4: `CheckBox` kontrolü örneği.

```
' Form üzerindeki tüm seçme kutularının durumu
' değiştiği zaman, toplam fiyat tekrar hesaplanır
Private Sub txtAcilisTutari_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
txtAcilisTutari.TextChanged, cbKDV.CheckedChanged,
cbOzelIletisim.CheckedChanged,
cbOzelIletisimIlkFatura.CheckedChanged
    Dim toplam As Double = txtAcilisTutari.Text
    ' İlk faturada 22 YTL açılış bedeli eklenir
    If cbOzelIletisimIlkFatura.Checked Then
        toplam += 22
    End If

    ' KDV eklenir
    If cbKDV.Checked Then
        toplam *= 1.18
    End If

    ' Özel İletişim vergisi eklenir
    If cbOzelIletisim.Checked Then
        toplam *= 1.25
    End If

    txtToplam.Text = toplam
End Sub
```

RadioButton

- RadioButton
  - Sunulan seçeneklerin bir tanesini seçmeyi sağlar.
  - GroupBox kontrolü ile gruplanır.
- GroupBox
  - Kontrollerin düzenlenmesini sağlar.
  - Başlık yazısı bulunur.
- Panel
  - Yatay – Dikey kaydırma çubukları bulunur.

RadioButton kontrolleri, kullanıcıya sunulan seçeneklerden sadece bir tanesinin seçilmesine izin verir. Form üzerinde birden fazla **RadioButton** konulduğunda, bu kontrollerin sadece bir tanesi seçili olabilir. Fakat bazı durumlarda, farklı seçenek grupları kullanılarak kullanıcının birden fazla seçim yapması istenebilir. Bu durumda, bazı seçenekler **GroupBox** kontrolü ile gruplanmalıdır.

Bu kontrolün özellikleri ve olayları **CheckBox** kontrolü ile aynıdır. Sadece bir seçenek işaretlenebildiği için, kontrollerin yapılması **CheckBox** kontrolüne göre daha kolaydır.

GroupBox

Bu kontrol, kontrollerin mantıksal bir düzende gruplanması için kullanılır. İçinde bulunan kontrollerin işleyişlerinde bir farklılık görünmez. Bir grup **RadioButton** kontrolünün, diğer **RadioButton** kontrollerinden etkilenmemesi için kullanılır.

Panel

GroupBox kontrolü gibi, kontrollerin belli bir düzende gözükmesini sağlamak için kullanılır. **GroupBox** kontrolünden farkı olarak yatay ve düşey kaydırma çubukları bulunur, ancak panel üzerinde başlık yazısı bulunmaz.

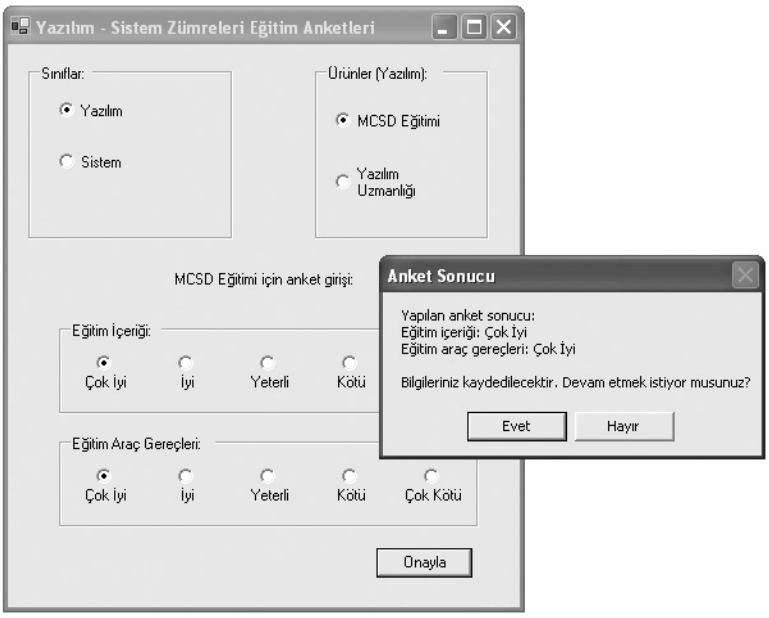
Panel Özellikleri

TABLO 9. 11: `Panel` Özellikleri

Özellik	Değer Tipi	Açıklama
`AutoScroll`	`Boolean`	Panelde kaydırma çubuklarının görünürlüğünü belirler

Paneller, seçeneklere göre bir grup kontrolün gizlenmesi veya görüntülenmesi aşamasında etkili bir rol oynar.

Örnek: `RadioButton`, `GroupBox` ve `Panel` kontrolleri, BilgeAdam eğitim anketi formunun tasarımında kullanılabilir. Anket, bir eğitimin ürünleri hakkında yapılır. Anket bilgileri eğitim araç gereçleri ve eğitim içeriği üzerinde "çok iyi"den "çok kötü"ye kadar bir değer verilmesiyle oluşturulur. Sonuç olarak elde edilen anket bilgileri kullanıcıya gösterilerek onaylaması beklenir.



RESİM 9.5: `GroupBox`, `Panel` ve `RadioButton` kontrolleri örneği.

- Global değişkenlerin oluşturulması:

```
' Ozet bilgilerinin tutulduğu değişken
Private AnketOzet As String

' Onaylama düğmesinin aktif hale gelmesi için
' tüm oylamaların yapılmış olması gerekir
Private IcerikOyuSecildi, AracOyuSecildi As Boolean
```

- Formun yüklenmesi sırasında kontroller üzerinde yapılan ayarlar:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    ' Sistem ve yazılım seçeneklerini tutan
    ' GroupBox kontrolleri gizlenir:
    grpSistem.Visible = False
    grpYazilim.Visible = False

    ' Anketleri tutan Panel kontrolü gizlenir
    pnlAnket.Visible = False

    ' Onayla düğmesi oylamadan önce pasif haldedir
    btnOnayla.Enabled = False
End Sub
```

- Eğitimler seçildiğinde, ilgili alt seçeneklerin görüntülenmesi sağlanır. Alt seçenekler, ayrı **GroupBox** kontrollerinde tutulur.

```
Private Sub rbYazilim_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
rbYazilim.CheckedChanged, rbSistem.CheckedChanged
    ' GroupBox kontrollerini görünümleri, eğitimlerin
    ' seçili olmasıyla doğru orantılıdır.
    grpYazilim.Visible = rbYazilim.Checked
    grpSistem.Visible = rbSistem.Checked

    UrunTemizle()
    pnlAnket.Visible = False
End Sub

' Ürünler başlangıç değerlerine çevrilir
Sub UrunTemizle()
    rbMCSD.Checked = False
    rbMCSE.Checked = False
    rbSistemUzmanligi.Checked = False
    rbYazilimUzmanligi.Checked = False
End Sub
```

- Alt ürünler seçildiğinde, anket paneli görüntülenir ve panelin karşılama mesajında ilgili ürünün ismi gösterilir.

```
Private Sub rbSistemUzmanligi_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
rbSistemUzmanligi.CheckedChanged,
rbYazilimUzmanligi.CheckedChanged, rbMCSD.CheckedChanged,
rbMCSE.CheckedChanged
```

```
    Dim panelMesaji As String

    ' Bu olayı tetikleyen RadioButton kontrolü alınır
    Dim basilan As RadioButton = sender

    lblKarsilamaMesaji.Text = basilan.Text & " için anket
girişi:"
    pnlAnket.Visible = True
End Sub
```

- Anketlerde, ilgili konularda oylama yapıldığı zaman, oylama düğmesi aktif hale getirilir ve anket mesajı oluşturulur.

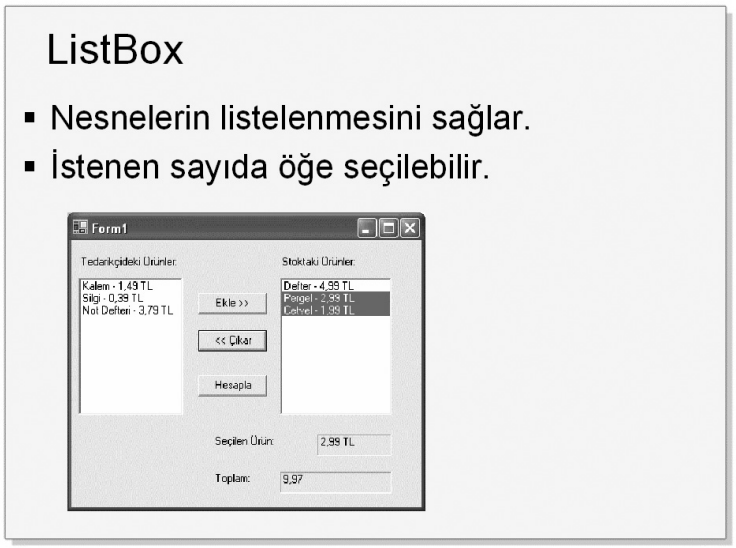
```
' Eğitim içeriği için verilen oy

Private Sub rbCokIyi_Icerik_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
rbCokIyi_Icerik.CheckedChanged,
rbCokKotu_Icerik.CheckedChanged,
rbIyi_Icerik.CheckedChanged,
rbYeterli_Icerik.CheckedChanged,
rbKotu_Icerik.CheckedChanged
    IcerikOyuSecildi = True

    Dim basilan As RadioButton = sender
    AnketOzetiCikar("Eğitim içeriği: " & basilan.Text)

End Sub

' Eğitim araç gereçleri için verilen oy

Private Sub rbCokIyi_Arac_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
rbCokIyi_Arac.CheckedChanged, rbCokKotu_Arac.CheckedChanged,
rbIyi_Arac.CheckedChanged, rbYeterli_Arac.CheckedChanged,
rbKotu_Arac.CheckedChanged
    AracOyuSecildi = True

    Dim basilan As RadioButton = sender
    AnketOzetiCikar("Eğitim araç gereçleri: " &
basilan.Text)
End Sub

Sub AnketOzetiCikar(ByVal ozet As String)
    AnketOzet &= ozet & vbCrLf

    If IcerikOyuSecildi And AracOyuSecildi Then
        btnOnayla.Enabled = True
```

```
    End If
End Sub
```

Anket bilgileri oluşturulduktan sonra, onay düğmesi aktif hale gelir. Bu düğme tıklandığı zaman kullanıcıya girdiği bilgiler mesaj kutusu ile gösterilir. Kullanıcı onayladıktan sonra kayıt işlemleri gerçekleşir.

```
Private Sub btnOnayla_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnOnayla.Click
    Dim mesaj As String
    mesaj = "Yapılan anket sonucu: " & vbCrLf & AnketOzet &
vbCrLf
    mesaj &= "Bilgileriniz kaydedilecektir. Devam etmek
istiyor musunuz?"

    If MsgBox(mesaj, MsgBoxStyle.YesNo, "Anket Sonucu") =
MsgBoxResult.No Then
        Exit Sub
    Else
        ' Anket kayıt işlemleri...
    End If
End Sub
```

ListBox



Kullanıcıya sunulan seçeneklerin bir liste halinde görünmesini sağlar. Liste kutusundan istenen sayıda öğe seçilebilir.

ListBox Özellikleri

TABLO 9.12: `ListBox` **Özellikleri**

Özellik	Değer Tipi	Açıklama
`Items`	`ListBox.ObjectCollection`	Liste kutusuna eklenen öğelerin tutulduğu koleksiyon nesnesidir.
`SelectedItem`	`Object`	Liste kutusundan seçilen öğeyi alır.
`SelectedItems`	`SelectedObjectCollection`	Liste kutusundan seçilen öğeleri alır. Seçilen öğeler dinamik bir dizide tutulur.
`SelectedIndex`	`Integer`	Liste kutusundan seçilen öğenin indisini alır.
`SelectedIndices`	`SelectedIndexCollection`	Liste kutusundan seçilen öğelerin indislerini bir koleksiyon nesnesinde tutar.
`DataSource`	`Object`	Listenin öğelerinin tutulduğu veri kaynağıdır. Veri kaynağı boş geçilirse `Items` koleksiyonuna eklenen öğeler görüntülenir.

TABLO 9.12: `ListBox` Özellikleri

Özellik	Değer Tipi	Açıklama
`DisplayMember`	`String`	Veri kaynağından gelen öğelerin, kullanıcıya gösterilecek özelliğidir.
`ValueMember`	`String`	Veri kaynağından gelen öğelerin, dönüş değerini belirleyen özelliğidir.
`SelectedValue`	`Object`	Seçilen öğenin, liste kutusunun **`ValueMember`** ile belirtilen özelliğidir.
`SelectionMode`	`SelectionMode`	Liste kutusundan kaç tane öğe seçilebileceğini belirtir. **`None`** değeri 0, **`One`** değeri 1, **`MultiSimple`** ve **`MultiExtended`** değerleri birden fazla öğenin seçilebileceğini belirtir.
`MultiColumn`	`Boolean`	Liste kutusundaki öğelerin biden fazla kolonda görüntülenmesini belirler.

ListBox Olayları

TABLO 9.13: `ListBox` Olayları

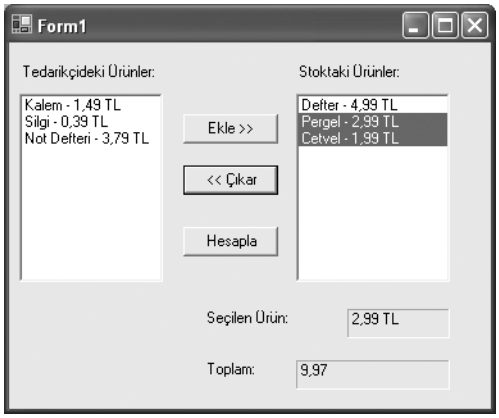
Olay	Açıklama
`SelectedIndexChanged`	Liste kutusunda bir öğe seçildiği zaman gerçekleşir.

ListBox Metotları

TABLO 9.14: `ListBox` Metotları

Metot	Açıklama
`GetItemText`	Parametre olarak verilen nesnenin liste kutusunda gösterilen yazısını döndürür.
`GetSelected`	Parametre olarak verilen indisteki öğenin seçili olup olmadığını döndürür.
`FindString`	Parametredeki **`String`** ifadesini liste kutusunda arayarak, bulduğu ilk öğenin indisini döndürür

Örnek: Tedarikçiden alınacak ve stokta bulunan ürünleri listelemek ve alım satım işlemi yapmak için **`ListBox`** kontrolleri kullanılabilir.



RESİM 9.6: **`ListBox`** kontrolü örneği.

- Ürünlerin tutulması için bir **`Structure`** oluşturulur. Bu ürün yapısının **`ToString`** metodu tekrar yazılmıştır. Bunun nedeni, **`ListBox`** kontrolünde listelenen nesnelerin görüntülendiği değerin **`ToString`** metodu çağırılarak belirlenmesidir. Dolayısıyla liste kutularında istenen formatta değerin gözükmesini sağlamak için **`ToString`** metodunun tekrar yazılması gerekir.

```
Structure Urun
    Public Ismi As String
    Public Fiyat As Double

    Public Sub New(ByVal UrunIsim As String, ByVal UrunFiyat
As Double)
        Ismi = UrunIsim
        Fiyat = UrunFiyat
    End Sub

    Public Overrides Function ToString() As String
        Return String.Format("{0} - {1:C}", Ismi, Fiyat)
    End Function
End Structure
```

- Liste kutularının özellikleri ayarlanır ve içine eleman doldurulur.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    lbTedarikci.SelectionMode = SelectionMode.MultiExtended
    lbStok.SelectionMode = SelectionMode.MultiExtended

    UrunEkle()
End Sub
```

```
Sub UrunEkle()
    Dim u As Urun

    u = New Urun("Kalem", 1.49)
    lbTedarikci.Items.Add(u)
    u = New Urun("Silgi", 0.39)
    lbTedarikci.Items.Add(u)
    u = New Urun("Defter", 4.99)
    lbTedarikci.Items.Add(u)
    u = New Urun("Cetvel", 1.99)
    lbTedarikci.Items.Add(u)
    u = New Urun("Pergel", 2.99)
    lbTedarikci.Items.Add(u)
    u = New Urun("Not Defteri", 3.79)
    lbTedarikci.Items.Add(u)

End Sub
```

- Tedarikçi liste kutusundan, stok liste kutusuna öğe aktarılması için, seçilen değerler önce liste kutusuna eklenir. Daha sonra bu seçilen değerler, diğer listede olmayacağı için tek tek çıkartılır.

```
Private Sub btnEkle_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnEkle.Click
    ' Tedarikçiden alınan ürünler stok listesine eklenir
    For Each item As Object In lbTedarikci.SelectedItems
        lbStok.Items.Add(item)
    Next

    ' Stok listesine eklenen tüm ürünler
    ' tedarikçi listesinden çıkartılır
    For Each item As Object In lbStok.Items
        lbTedarikci.Items.Remove(item)
    Next

    btnCikar.Enabled = True
    btnHesapla.Enabled = True
End Sub
```

- Stok listesinden öğe çıkarmak için, ekleme işlemine benzer kodlar çalıştırılır.

```
Private Sub btnCikar_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnCikar.Click
    ' Tedarikçiden alınan ürünler stok listesine eklenir
    For Each item As Object In lbStok.SelectedItems
```

```
        lbTedarikci.Items.Add(item)
    Next

    ' Stok listesine eklenen tüm ürünler
    ' tedarikçi listesinden çıkartılır
    For Each item As Object In lbTedarikci.Items
        lbStok.Items.Remove(item)
    Next

    If lbStok.Items.Count = 0 Then
        btnCikar.Enabled = False
        btnHesapla.Enabled = False
    End If
End Sub
```

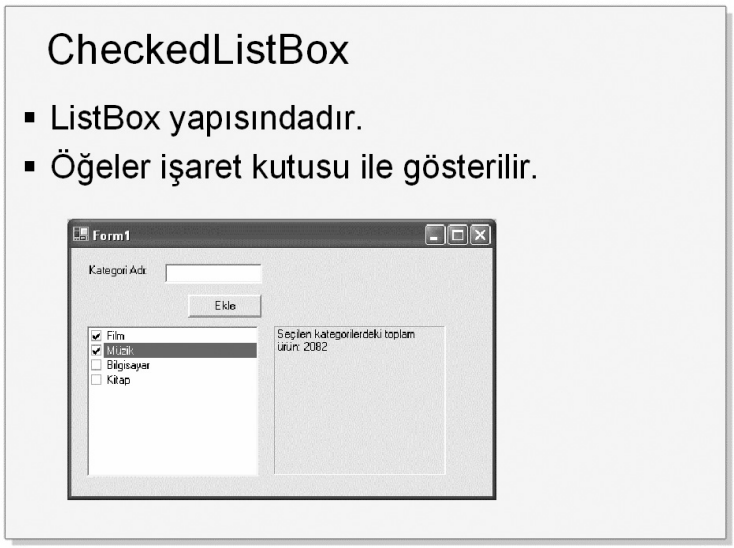
- Stoktaki toplam fiyatın hesaplanması işlemi, ürünlerin fiyatlarının alınıp toplanması ile gerçekleşir.

```
Private Sub btnHesapla_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnHesapla.Click
    Dim toplam As Double

    For i As Integer = 0 To lbStok.Items.Count - 1
        Dim urun As Urun = lbStok.Items(i)
        toplam += urun.Fiyat
    Next
    lblToplam.Text = toplam
End Sub
```

- Stok listesindeki bir öğenin seçildiği durumda, bu öğenin fiyatı görüntülenir.

```
Private Sub lbStok_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
lbStok.SelectedIndexChanged
    Dim secilen As Urun
    secilen = lbStok.SelectedItem

    lblUrunFiyat.Text = String.Format("{0:C}",
secilen.Fiyat)
End Sub
```

CheckedListBox



Liste kutusunun tüm özellik, metot ve olaylarını alır ve listedeki öğelerin işaret kutusu ile gösterilmesini sağlar.

CheckedListBox Özellikleri

TABLO 9.15: `CheckedListBox` Özellikleri

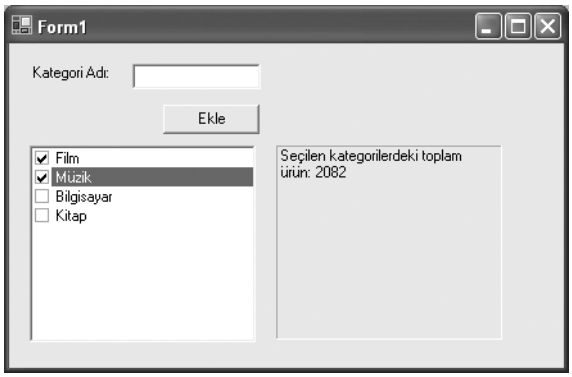
Özellik	Değer Tipi	Açıklama
`CheckedItems`	`CheckedItemCollection`	Liste kutusunda işaretlenmiş öğeleri tutar.
`CheckedIndices`	`CheckedIndexCollection`	Liste kutusunda işaretlenmiş öğelerin indislerini tutar.
`CheckOnClick`	`Boolean`	Liste kutusunda öğe tıklandığı zaman işaretlenmesini belirler. `False` ise ilk tıklamada öğe seçilir, ikinci tıklamada seçme kutusu işaretlenir.

CheckedListBox Metotları

TABLO 9.16: `CheckedListBox` Metotları

Metot	Açıklama
`GetItemSelected`	Parametre olarak verilen indisteki öğenin seçili olup olmadığını döndürür.
`SetItemSelected`	İlk parametrede verilen indisteki elemanın seçili olup olmadığını, ikinci parametrede verilen Boolean değeri ile belirler.

Örnek: Kategori başına stoktaki toplam ürünlerin gösterildiği bir uygulamada, listelenen kategorileri seçmek için bu kontrol uygun olur.



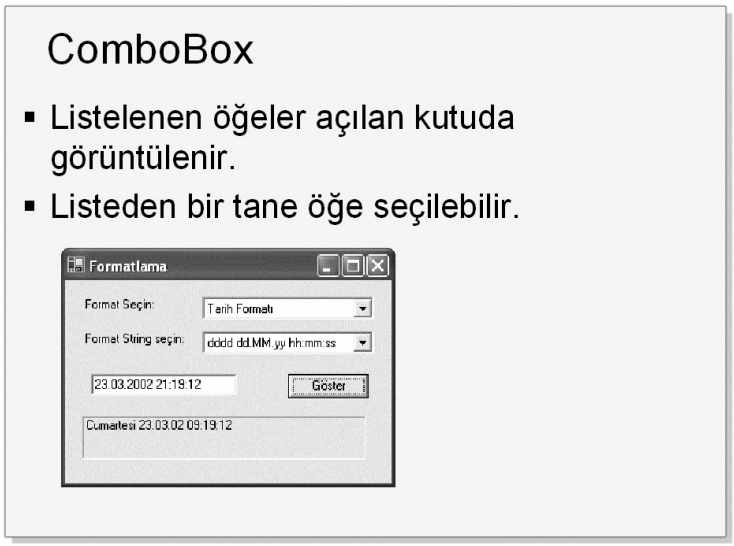
RESİM 9.7: **`CheckedListBox`** kontrolü örneği.

- Listede bir öğe seçildiği zaman, seçilen tüm kategorilerin ürün stok durumu alınır ve toplam ürün sayısı kullanıcıya gösterilir.

```
Private Sub chlistKategoriler_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs)
Handles chlistKategoriler.SelectedIndexChanged
    Dim toplam As Integer

    ' Listedeki seçilen öğelerin ürün adeti toplanır.
    For i As Integer = 0 To chlistKategoriler.Items.Count -
1
        If chlistKategoriler.GetItemChecked(i) Then
            Dim secilen As Object
            secilen = chlistKategoriler.Items(i)

            ' Stok durumunu gösteren fonksiyon çağırılır
            toplam += StokDurumu(secilen.ToString)
        End If
    Next

    lblToplamUrun.Text = "Seçilen kategorilerdeki toplam
ürün: " & toplam
End Sub


' Kategoriye göre, stoktaki ürünlerin belirlenmesi
Function StokDurumu(ByVal kategori As String) As Integer
    Select Case kategori
        Case "Film"
            Return 1100
```

```
        Case "Müzik"
            Return 982
        Case "Bilgisayar"
            Return 302
        Case "Kitap"
            Return 1222
        Case Else
            Return 10
    End Select
End Function

' Ekleme işlemi
Private Sub btnKategoriEkle_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnKategoriEkle.Click
    chlistKategoriler.Items.Add(txtKategoriAdi.Text)
End Sub
```

ComboBox



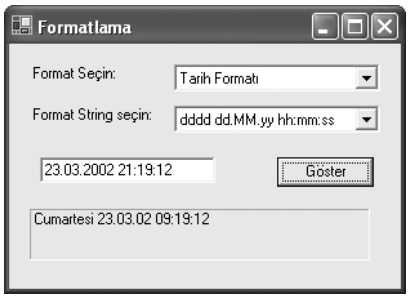
Liste kutusu ile aynı özelliklere sahiptir, ancak listelenen öğeler açılan bir kutuda görüntülenir ve listeden en fazla bir tane öğe seçilebilir. Liste kutusuna göre bir başka farklılığı ise, isteğe bağlı olarak, kullanıcının açılan kutu üzerinde değer girebilmesidir. Dolayısıyla bir **`TextBox`** kontrolü gibi de davranabilir.

ComboBox Özellikleri

TABLO 9.17: **`ComboBox`** Özellikleri

Özellik	Değer Tipi	Açıklama
`DropDownStyle`	**`ComboBoxStyle`**	Kontrolün listeleme stilini belirler. **`Simple`** stili, listedeki sadece bir öğeyi görüntüler. **`DropDown`** stili, listenin tüm elemanlarını görüntüleyerek seçilmelerini ve kullanıcının değer girmesini sağar. **`DropDownList`** kullanıcının değer girmesini engeller.
`DropDownWidth`	**`Integer`**	**`ComboBox`** kontrolünün açılan listesinin genişliğini belirler.
`MaxDropDownItems`	**`Integer`**	Kontrole eklenebilecek maksimum öğe sayısını belirler.
`MaxLength`	**`Integer`**	Kullanıcının girebileceği maksimum karakter sayısını belirler.
`SelectedText`	**`String`**	Seçilen öğenin görüntülenen yazısını belirler.

Örnek: Tarih ve sayı formatlarını, kullanıcının seçimine bırakarak bir sayı veya tarih yazdırma işlemi **ComboBox** kontrolleri ile yapılabilir.



RESİM 9.8: **ComboBox** kontrolü örneği.

- **ComboBox** kontrollerinin özelliklerinin ayarlanması ve format tiplerine öğe eklenmesi.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    cmbFormat.DropDownStyle = ComboBoxStyle.DropDownList
    cmbFormatString.DropDownStyle =
ComboBoxStyle.DropDownList

    cmbFormat.Items.Add("Tarih Formatı")
    cmbFormat.Items.Add("Sayı Formatı")
End Sub
```

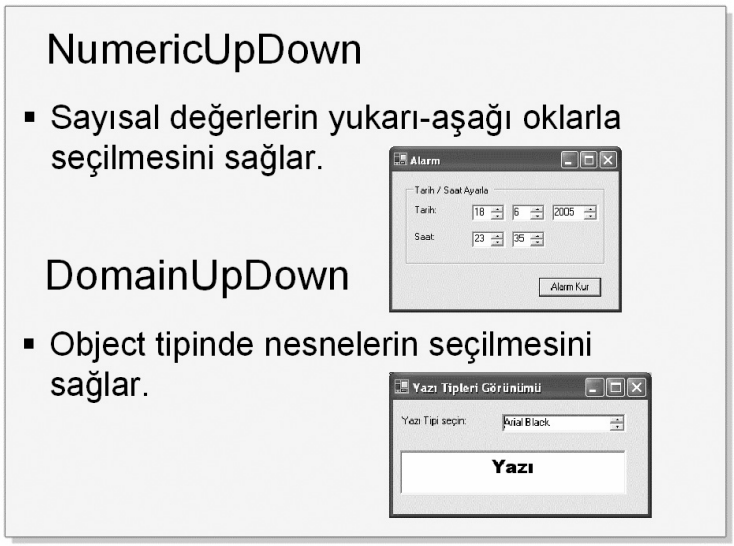
- Tarih ya da sayı formatlarından biri seçildiği zaman, ikinci **ComboBox** kontrolüne değişik format seçenekleri eklenir.

```
Private Sub cmbFormat_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmbFormat.SelectedIndexChanged
    cmbFormatString.Items.Clear()

    Select Case cmbFormat.SelectedIndex
        Case 0
            cmbFormatString.Items.Add("dd - MM - yyyy")
            cmbFormatString.Items.Add("yyyy*MM*dd hh:mm")
            cmbFormatString.Items.Add("dddd dd.MM.yy
hh:mm:ss")
        Case 1
            cmbFormatString.Items.Add("C")
            cmbFormatString.Items.Add("P")
            cmbFormatString.Items.Add("N")
    End Select
End Sub
```

- Format seçildikten sonra metin kutusuna girilen değer alınır ve ilgili formatta gösterilir.

```
Private Sub btnGoster_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnGoster.Click
    Select Case cmbFormat.SelectedIndex
        Case 0
            Dim d As Date = txtYazi.Text
            lblSonuc.Text = d.ToString(cmbFormatString.Text)
        Case 1
            Dim i As Integer = txtYazi.Text
            lblSonuc.Text = i.ToString(cmbFormatString.Text)
    End Select
End Sub
```

NumericUpDown



Bu kontrol kullanıcının, sayısal bir değeri girmesini veya yukarı aşağı okları ile seçmesini sağlar.

NumericUpDown Özellikleri

TABLO 9.18: `NumericUpDown` Özellikleri

Özellik	Değer Tipi	Açıklama
`HexaDecimal`	`Boolean`	Sayıların onaltılık tabanda görüntülenmesini belirler.
`Increment`	`Decimal`	Aşağı yukarı oklar kullanıldığında, sayıların artma ve azalma adımlarını belirler.
`Maximum`	`Decimal`	Kontrolde gösterilen sayıların alabileceği maksimum değeri belirler.
`Minimum`	`Decimal`	Kontrolde gösterilen sayıların alabileceği minimum değeri belirler.
`ThousandSeparators`	`Boolean`	Sayıların basamak ayracının gösterilmesini belirler.
`Value`	`Decimal`	Kontrolün gösterdiği sayı değerini belirler.
`ReadOnly`	`Boolean`	`True` değerini alırsa kullanıcının giriş yapmasını engeller.

NumericUpDown Olayları

TABLO 9.19: `NumericUpDown` Olayları

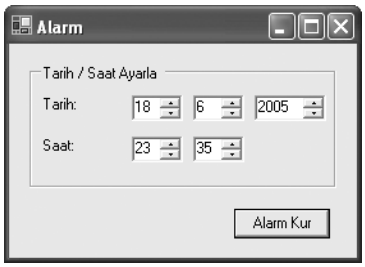
Olay	Açıklama
`ValueChanged`	Kontrolün sayı değeri değiştiği zaman gerçekleşir

NumericUpDown Metotları

TABLO 9.20: `NumericUpDown` Olayları

Metot	Açıklama
`DownButton`	Aşağı düğmesine basar ve sayı değerini düşürür.
`UpButton`	Yukarı düğmesine basar ve sayı değerini artırır.

Örnek: Alarm kurarken, tarih ve zaman değerlerinin ayarlanması `NumericUpDown` kontrolü ile yapılabilir.



RESİM 9.9: `NumericUpDown` kontrolü örneği.

- Tarih ve zaman değerlerinin alabileceği maksimum ve minimum değerler ayarlanır.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    nYil.Minimum = 1
    nAy.Minimum = 1
    nGun.Minimum = 1

    nYil.Maximum = 2099
    nAy.Maximum = 12
    nGun.Maximum = 31

    nSaat.Minimum = 0
    nDakika.Minimum = 0

    nSaat.Maximum = 23
    nDakika.Maximum = 59
```

```
    nYil.Value = Now.Year
    nAy.Value = Now.Month
    nGun.Value = Now.Day
    nSaat.Value = Now.Hour
    nDakika.Value = Now.Minute
End Sub
```

- Bu değerlerden herhangi biri değiştiği zaman, doğru tarih ve zaman değerinin girilmesi kontrol edilir.

```
Private Sub nGun_ValueChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles nGun.ValueChanged,
nAy.ValueChanged, nYil.ValueChanged, nSaat.ValueChanged,
nDakika.ValueChanged
    Dim tarih As String
    tarih = nGun.Value & "." & nAy.Value & "." & nYil.Value

    If Not IsDate(tarih) Then
        MsgBox(tarih)
    End If

    Dim zaman As String
    zaman = nSaat.Value & ":" & nDakika.Value

    If Not IsDate(zaman) Then
        MsgBox(zaman)
    End If

End Sub
```

DomainUpDown

NumericUpDown kontrolü ile aynı yapıdadır, ancak sayısal değerler yerine **Object** tipinde değerler tutar. Bu değerler kontrolün **Items** koleksiyonunda tutulur. Kontrol, bu özelliği ile liste kutusuna benzer.

DomainUpDown Özellikleri

TABLO 9.21: **DomainUpDown** Özellikleri

Özellik	Değer Tipi	Açıklama
Items	**DomainUpDownItemCollection**	Kontrolün öğelerinin tutulduğu dinamik bir listedir.
SelectedItem	**Object**	Kontrolde seçilen öğeyi tutar.

TABLO 9.21: `DomainUpDown` Özellikleri

Özellik	Değer Tipi	Açıklama
`Wrap`	`Boolean`	Liste sonuna gelindiğinde baştaki veya sondaki öğeye geri dönülmesini belirler.

DomainUpDown Olayları

TABLO 9.22: `DomainUpDown` Olayları

Olay	Açıklama
`SelectedItemChanged`	Kontrolde seçilen öğe değiştiği zaman gerçekleşir.

Örnek: Metin kutularının değiştirilmek istenen yazı tipleri `DomainUpDown` kontrolünde tutulabilir.

RESİM 9.10: `DomainUpDown` kontrolü örneği.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    For i As Integer = 0 To 10
dFont.Items.Add(System.Drawing.FontFamily.Families(i).Name)
    Next

    dFont.Wrap = True
End Sub

Private Sub dFont_SelectedItemChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
dFont.SelectedItemChanged
    If dFont.SelectedIndex >= 0 Then
        TextBox1.Font = New
Font(dFont.SelectedItem.ToString, 15)
    End If
End Sub
```

HScrollBar / VScrollBar

Horizontal – Vertical ScrollBar kontrolleri, sayısal bir değer taşıyan kaydırma çubuklarıdır. Tuttukları değerlerin sayısal olması bakımından **NumericUpDown** kontrolüne benzerler. Bu kontroller, üzerlerinde kaydırma çubukları olmayan kontroller üzerinde kullanılabilir. Örneğin **ListBox** ve **Panel** gibi kontrollerin kendi **ScrollBar** kontrolleri vardır. **TextBox** kontrolünün de ilgili özellikleri ayarlanarak yatay ve düşey **ScrollBar** kontrolleri gösterilebilir.

ScrollBar Özellikleri

TABLO 9.23: **ScrollBar** Özellikleri

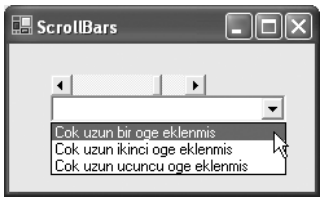
Özellik	Değer Tipi	Açıklama
Value	**Integer**	Kaydırma çubuğunun pozisyonuna göre alınan değeri tutar.
SmallChange	**Integer**	Kontrolü, üstündeki oklar ile kaydırıldığı zaman eklenecek ya da çıkartılacak değeri tutar.
LargeChange	**Integer**	Kontrolü, kaydırma çubuğundaki boşluğa tıklanarak kaydırıldığında zaman eklenecek ya da çıkartılacak değeri tutar.
Minimum	**Integer**	**Value** özelliğinin alabileceği maksimum değeri tutar.
Maximum	**Integer**	**Value** özelliğinin alabileceği minimum değeri tutar.

ScrollBar Olayları

TABLO 9.24: `ScrollBar` Olayları

Olay	Açıklama
`Scroll`	Çubuklar kaydırıldıkları zaman gerçekleşir.
`ValueChanged`	Kod ile ya da çubuklar kaydırılınca `Value` özelliği değiştiği zaman gerçekleşir.

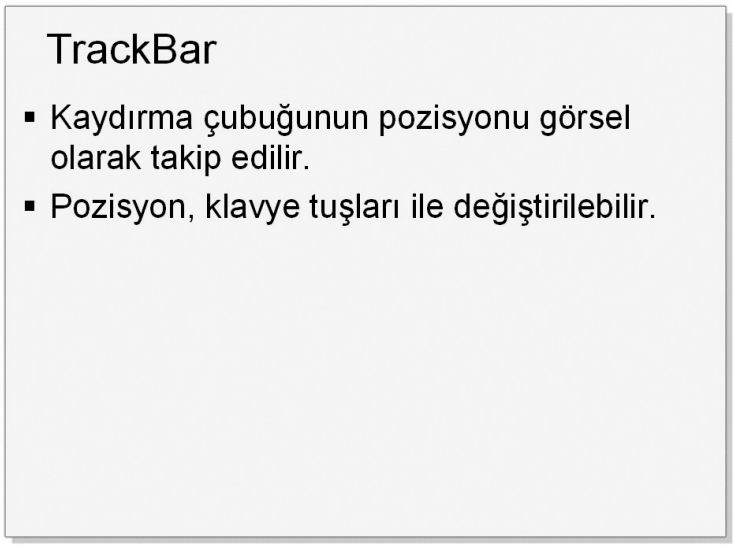
Örnek: Bir `ComboBox` kontrolünün öğelerini listelemek için, aşağıya doğru bir kaydırma çubuğu görüntülenir. Ancak listedeki bazı elemanlar kontrole sığmıyorsa, çalışma anında bu kontrolün genişliği artırılabilir.



RESİM 9.11: `ScrollBar` kontrolü örneği.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    hsGenislik.Maximum = ComboBox1.Width * 2
    hsGenislik.Value = ComboBox1.Width

End Sub

Private Sub hsGenislik_Scroll(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.ScrollEventArgs) Handles
hsGenislik.Scroll
    ComboBox1.Width = hsGenislik.Value
End Sub
```

TrackBar



Bu kontrol, **`ScrollBar`** kontrollerine benzer yapıdadır, ancak kontrol, bir cetvel biçiminde olduğu için, üzerinde durulan pozisyon görsel olarak takip edilebilir. Kontrolün kaydırma çubuklarından bir farkı da üzerine odaklanılabilir olmasıdır. Dolayısıyla, kontrolün **`Value`** değeri klavyede bulunan yukarı, aşağı, sağa, sola oklar ve PageUp, PageDown tuşları ile değiştirilebilir.

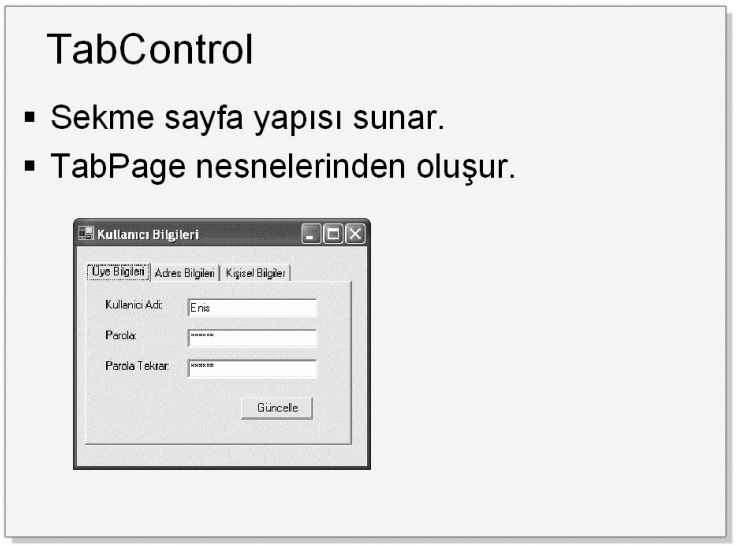
TrackBar Özellikleri

`TrackBar` kontrolünün birçok özelliği **`ScrollBar`** kontrollerinin özellikleriyle aynıdır. Fakat kontrolü daha esnek hale getiren birkaç özelliği vardır.

TABLO 9.25: **`TrackBar`** Özellikleri

Özellik	Değer Tipi	Açıklama
`TickStyle`	**`TickStyle`**	Kontrolün değerini gösteren çizgilerin pozisyonunu belirler.
`TickFrequency`	**`Integer`**	Çizgiler arasında kalan değerlerin sayısını belirler.
`Orientation`	**`Orientation`**	Kontrolün yönünün yatay veya düşey olmasını sağlar.

TabControl



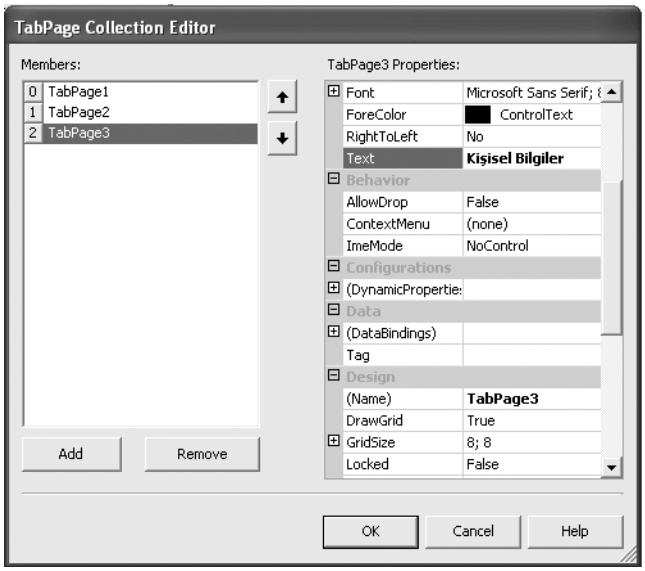
TabControl nesnesi, içinde sekme sayfaları tutan yapıdır. Bu sayfalar, **TabPage** nesneleri olarak oluşturulup yapılandırıldıktan sonra **TabControl** nesnesinin **TabPages** koleksiyonuna eklenir. Ekleme işlemi, Properties paneli ile tasarım anında da yapılabilir.

TabControl Özellikleri

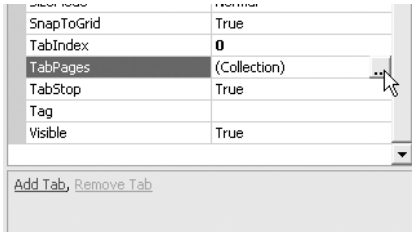
TABLO 9.26: TabControl Özellikleri

Özellik	Değer Tipi	Açıklama
HotTrack	**Boolean**	Fare ile sekme sayfalarının üzerine gelindiğinde, isimlerinin görsel olarak değişmesini belirler.
ItemSize	**Size**	Sekme sayfalarının büyüklüğünü belirler.
Multiline	**True**	Eklenen sekmelerin birden fazla satırda üst üste gözükmesini belirler.
ShowToolTips	**Boolean**	Fare sekme sayfalarının üzerindeyken bilgi mesajının gösterilmesini belirler.
SelectedTab	**TabPage**	Seçilen sekme sayfasını belirler.
SelectedIndex	**Integer**	Seçilen sekme sayfasının indisini belirler.
TabCount	**Integer**	Sekme sayısını belirler.
TabPages	**TabPageCollection**	Kontrolün içinde bulunduğu sekme sayfalarının koleksiyonudur.

TabControl nesnesine **TabPage** sayfaları eklemek için tasarım anında TabPage Collection Editor penceresinden yararlanılabilir (Resim 9.12).



RESİM 9.12: TabPage Collection Editor penceresi.



RESİM 9.13: **TabPage** eklemek.

TabPage Özellikleri

Sekme sayfaları, normal form tasarımları gibi kontroller eklenerek yapılır. **TabPage** kontrolü **Panel** kontrolünden türetilir ve **Panel** kontrolünün tüm özelliklerini alır.

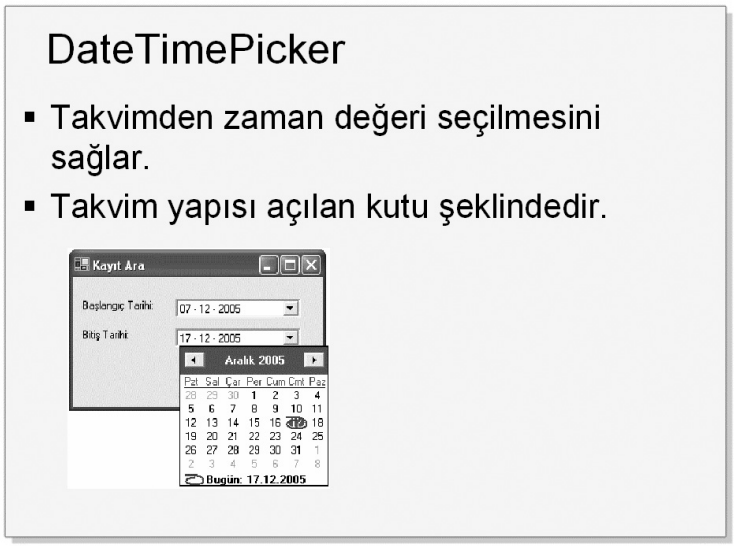
TABLO 9.27: **TabPage** Özellikleri

Özellik	Değer Tipi	Açıklama
ToolTipText	**String**	Bu özelliğin değeri, fare sayfanın üzerindeyken, bilgi mesajı olarak gösterilir. Ait olduğu **TabControl** nesnesinin **ShowToolTip** özelliği **True** olmalıdır.

Örnek: Bir kullanıcı kaydının tek bir formda görüntülenmesi isteniyorsa, bu form **TabControl** ile küçük sayfalara bölünebilir.

RESİM 9.14: `TabControl` örneği.

DateTimePicker



Bir açılan kutudan zaman değeri seçmeyi sağlar. Seçilen değer **`Date`** tipinde olur.

DateTimePicker Özellikleri

TABLO 9.28: **`DateTimePicker`** Özellikleri

Özellik	Değer Tipi	Açıklama
`CalendarTrailingForeColor`	**`Color`**	Bir önceki ve bir sonraki ayın günlerinin görüntülenme rengidir.
`CalendarTitleForeColor`	**`Color`**	Takvim başlığının ön plan rengidir.
`CalendarTitleBackColor`	**`Color`**	Takvim başlığının arka plan rengidir.
`CalendarMonthBackground`	**`Color`**	Takvim arka plan rengidir.
`CalendarForeColor`	**`Color`**	Takvimdeki yazıların ön plan rengidir.
`CalendarFont`	**`Font`**	Takvimin gösterileceği yazı tipi ayarlarıdır.
`ShowCheckBox`	**`Boolean`**	Tarih değerinin yanında seçme kutusunun gösterilmesi.
`Checked`	**`Boolean`**	Seçme kutusu görüntülendiği zaman, tarihin seçili olup olmadığını gösterir

TABLO 9.28: `DateTimePicker` Özellikleri

Özellik	Değer Tipi	Açıklama
`Format`	`DateTime PickerFormat`	Kontrolün görüntüleneceği formatı belirler. **`Long`**, **`Short`** değerleri uzun ve kısa tarih formatını, **`Time`** sadece zamanı gösterir. **`Custom`** değeri, **`CustomFormat`** özelliğine girilen formatta gösterileceğini belirler.
`CustomFormat`	`String`	Tarihin hangi formatta gösterileceğini belirler.
`Value`	`Date`	Seçilen tarih değerini belirler.
`MaxDate`	`Date`	Kontrolün alabileceği maksimum tarih değeridir.
`MinDate`	`Date`	Kontrolün alabileceği minimum tarih değeridir.
`ShowUpDown`	`Boolean`	Kontrolün formunu açılan kutu ya da yukarı aşağı okları formatında gösterir. Bu özellik **`True`** olduğunda, kontrolün formatı, **`NumericUpDown`** kontrolünün formatında olur.

Örnek: Veritabanından bir kaydın belli tarih aralıkları ile sorgulanması sırasında, kullanıcının başlangıç ve bitiş tarihlerini seçmesi için bu kontrol kullanılabilir.

RESİM 9. 15: `DateTimePicker` kontrolü örneği.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    TarihAyarla(dtBaslangic)
    TarihAyarla(dtBitis)

End Sub
```

```
' Tarih kontrollerinin ayarlanması
Sub TarihAyarla(ByVal dtTarih As DateTimePicker)
    dtTarih.Format = DateTimePickerFormat.Custom
    dtTarih.CustomFormat = "dd - MM - yyyy"
    dtTarih.MaxDate = Now.AddYears(2)
    dtTarih.MinDate = Now.AddYears(-2)
End Sub

' Gerekli kontroller yapıldıktan sonra
' Sql cümlesi seçilen tarihlere göre oluşturulur
Private Sub btnAra_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnAra.Click
    Dim basTarih, sonTarih As Date
    basTarih = dtBaslangic.Value
    sonTarih = dtBitis.Value

    If Date.Compare(basTarih, sonTarih) = 1 Then Exit Sub

    Dim Sql As String
    Sql = "Select * From Siparisler Where SiparisTarih
Between "
    Sql &= basTarih & " And " & sonTarih

    ' Sql komutunu çalıştır
End Sub
```

MonthCalendar

DateTimePicker kontrolünün açılan takvimi biçimindedir. Bu kontrol kullanıcıya, tarih alanları üzerinde daha esnek çalışma olanağı sağlar.

MonthCalendar Özellikleri

DateTimePicker kontrolünün birçok özelliğini almasına rağmen, bazı özelliklerinde değişiklikler görülür. Örneğin **Value** özelliği bu kontrolde yoktur. Bu kontrolden seçilen değerler, bir tarih aralığıdır. Dolayısıyla tek bir **Date** tipini tutan bir özellik yoktur.

TABLO 9.29: **MonthCalendar** Özellikleri

Özellik	Değer Tipi	Açıklama
MaxSelectionCount	**Integer**	Bir seferde maksimum kaç gün seçileceğini belirler.
SelectionRange	**SelectionRange**	Başlangıç ve bitiş tarihlerinden oluşan bir seçim aralığı nesnesidir.
SelectionBegin	**Date**	Seçilen tarih aralığının hangi tarihten itibaren başladığını belirler.
SelectionEnd	**Date**	Seçilen tarih aralığının hangi tarihte bittiğini belirler.
ScrollChange	**Integer**	İleri geri düğmeleri tıklandığı zaman kaç ay atlanacağını belirler.
MonthlyBoldedDates	**Date()**	Takvimde hangi günlerin kalın yazı tipinde gösterileceğini belirler. İşaretlenen günler, her ay için kalın gösterilir.

TABLO 9.29: `MonthCalendar` Özellikleri

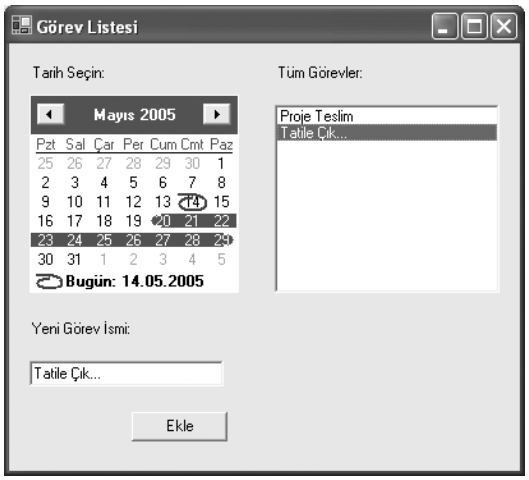
Özellik	Değer Tipi	Açıklama
`ShowToday`	`Boolean`	Takvimin alt kısmında, sistem takvimine göre hangi günde olduğunu gösterir.
`ShowTodayCircle`	`Boolean`	Takvimde, o günün seçili olmasını belirler.
`ShowWeekNumbers`	`Boolean`	Takvimin sol tarafında, yılın hafta numaralarını gösterir.

MonthCalendar Olayları

TABLO 9.30: `MonthCalendar` Olayları

Olay	Açıklama
`DateChanged`	Seçilen tarihten farklı bir tarih seçildiğinde gerçekleşir.
`DateSelected`	Yeni bir tarih seçildiği zaman gerçekleşir. **`DateChanged`** olayı gerçekleştikten hemen sonra bu olay gerçekleşir.

Örnek: Yapılacak görevlerin tutulduğu bir Windows uygulamasında, görevin başlangıç ve bitiş tarihleri tek bir `MonthCalendar` kontrolünden kolaylıkla seçilebilir.



RESİM 9.16: `MonthCalendar` kontrolü örneği.

- Görevlerin tanımlanması için bir **`Gorev`** sınıfı oluşturulur.

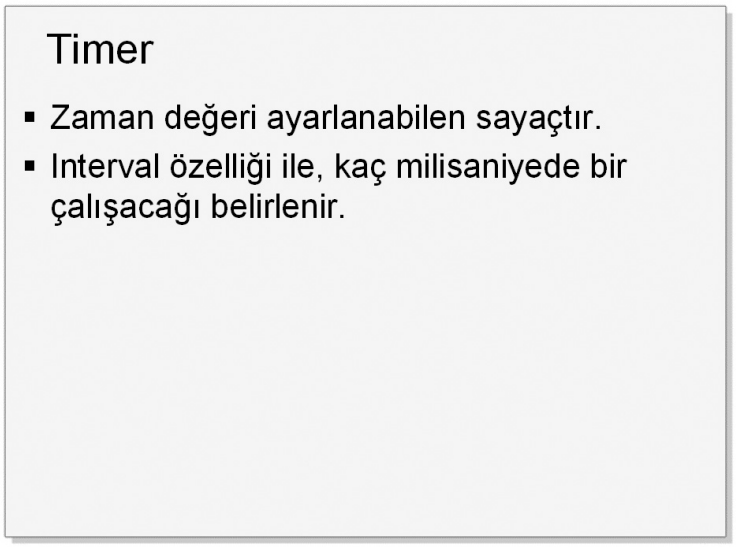
```
Public Class Gorev
    Public GorevIsmi As String
    Public BaslangicTarihi As Date
    Public BitisTarihi As Date
```

```
    ' Liste kontrollerinde görevin isminin görüntülenmesi
    ' için, ToString metodunu tekrar yazmak gerekir.
    Public Overrides Function ToString() As String
        Return GorevIsmi
    End Function

    Public Sub New(ByVal Isim As String, ByVal basTarihi As
Date, ByVal bitTarihi As Date)
        Me.GorevIsmi = Isim
        Me.BaslangicTarihi = basTarihi
        Me.BitisTarihi = bitTarihi
    End Sub
End Class
```

- Görevler ekleneceği zaman, yeni bir **Gorev** nesnesi oluşturulur ve görevin başlangıç-bitiş tarihleri ayarlanır.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    ' Maksimum iki hafta seçilsin
    MonthCalendar1.MaxSelectionCount = 14
End Sub

Private Sub btnEkle_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnEkle.Click
    Dim baslangicTarihi As Date =
MonthCalendar1.SelectionStart
    Dim bitisTarihi As Date = MonthCalendar1.SelectionEnd
    Dim gorevIsmi As String = txtYeniGorev.Text

    Dim yeniGorev As New Gorev(gorevIsmi, baslangicTarihi,
bitisTarihi)
    ListBox1.Items.Add(yeniGorev)

End Sub

Private Sub ListBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ListBox1.SelectedIndexChanged
    Dim secilen As Gorev
    secilen = ListBox1.SelectedItem

    MonthCalendar1.SelectionStart = secilen.BaslangicTarihi
    MonthCalendar1.SelectionEnd = secilen.BitisTarihi
    txtYeniGorev.Text = secilen.GorevIsmi
End Sub
```

Timer



Windows uygulamalarında sayaç görevini görür.

Timer Özellikleri

TABLO 9.31: `Timer` Özellikleri

Özellik	Değer Tipi	Açıklama
`Enabled`	`Boolean`	Kontrolün aktif olup olmadığını belirler.
`Interval`	`Integer`	Sayacın hangi zaman aralığında bir çalışması gerektiğini belirler. Milisaniye cinsindedir.

Timer Olayları

TABLO 9.32: `Timer` Olayları

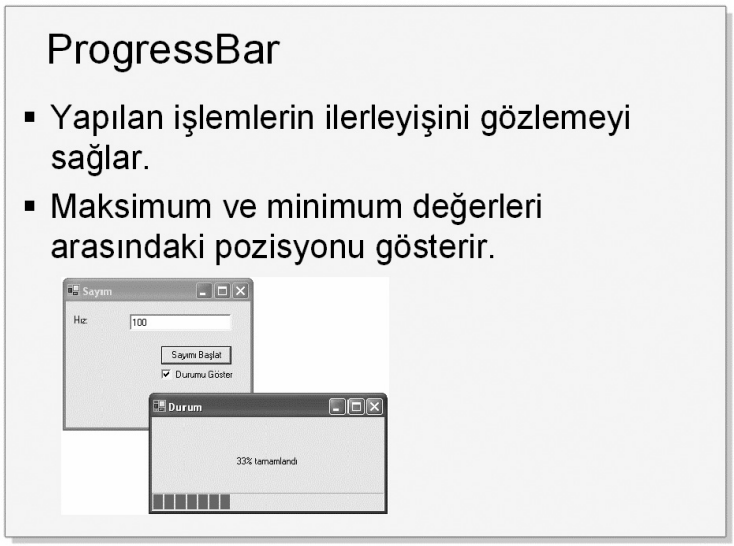
Olay	Açıklama
`Tick`	`Interval` özelliğinde belirtilen zaman değeri geçtiğinde gerçekleşir.

Timer Metotları

TABLO 9.33: `Timer` Metotları

Metot	Açıklama
`Start`	Sayacı başlatır
`Stop`	Sayacı durdurur

ProgressBar



ProgressBar, belli bir andaki değerin, alabileceği değer aralığına göre yüzdesini gösterir. Yapılan bir işlemin ilerleyişini göstermesi açısından oldukça kullanışlı bir kontroldür.

ProgressBar Özellikleri

TABLO 9.34: **ProgressBar** Özellikleri

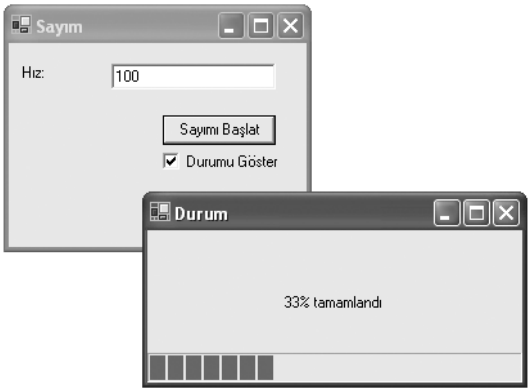
Özellik	Değer Tipi	Açıklama
Minimum	**Integer**	Kontrolün alabileceği minimum değeri belirler.
Maximum	**Integer**	Kontrolün alabileceği minimum değeri belirler.
Value	**Integer**	Kontrolün verilen değer aralığındaki pozisyonunu belirler.

Örnek: **ProgressBar** bir sayım işleminde kalan durumu göstermek için kullanılabilir.

- **ProgressBar** ile durumun gösterileceği ayrı bir form eklenir. Burada sayma işleminin hızı için bir **Timer** bulunur. Sayaç her işlediğinde yeni değer **ProgressBar** kontrolünde gösterilir.

```
Dim kalan As Integer

Private Sub Durum_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    kalan = ProgressBar1.Maximum
```

```
    Timer1.Start()
End Sub

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Timer1.Tick
    If kalan = 0 Then
        Timer1.Stop()
        Me.Close()
    End If

    Dim aralik As Integer
    aralik = ProgressBar1.Maximum - ProgressBar1.Minimum

    ' Kalan sayım işleminin yüzdesi hesaplanır
    Dim oran As Integer = (aralik - kalan) / aralik * 100
    Label1.Text = oran & "% tamamlandı"

    ProgressBar1.Value = ProgressBar1.Maximum - kalan
    kalan -= 1
End Sub
```



RESİM 9.17: **`ProgressBar`** kontrolü örneği.

- Oluşturulan bu form, başlangıç formundan çağırılarak durum gösterilir.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    CheckBox1.Checked = True
End Sub

Private Sub btnBaslat_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnBaslat.Click
    Dim frmDurum As New Durum
    frmDurum.Timer1.Interval = TextBox1.Text
```

```
        If CheckBox1.Checked Then
            frmDurum.ShowDialog()
        End If
    End Sub
```

ErrorProvider

Form üzerindeki kontrollerin yanında hata mesajları gösterilmesini sağlar.

ErrorProvider Özellikleri

TABLO 9.35: `ErrorProvider` Özellikleri

Özellik	Değer Tipi	Açıklama
`BlinkRate`	`Integer`	Hata simgesinin kaç milisaniyede bir yanıp söneceğini belirler.
`BlinkStyle`	`ErrorBlinkSytle`	Hata simgesinin yanıp sönme stilini belirler. **`AlwaysBlink`**, her zaman, **`BlinkIfDifferentError`** farklı bir hata meydana geldiğinde yanıp söneceğini belirler. **`NeverBlink`** ise simgenin yanıp sönmeden görüntüleneceğini belirler.
`Icon`	`Icon`	Hata mesajlarının gösterilmesi sırasında çıkan simgeyi belirler.

ErrorProvider Metotları

TABLO 9.36: `ErrorProvider` Metotları

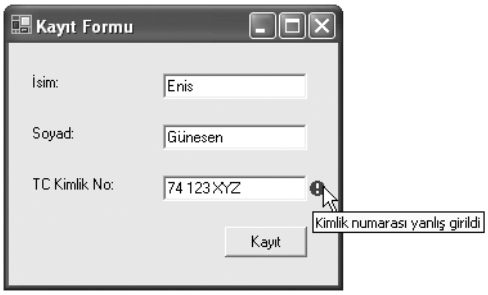
Metot	Açıklama
`SetError`	Kontrollerin hata mesajlarının belirlenmesi için kullanılır.

ErrorProvider kontrolü forma eklendiği zaman, **Properties** panelinde, kontrollerin ekstra özellikleri görünür. Bu özellikler, forma eklenen her **ErrorProvider** için oluşturulur.

TABLO9.37: **ErrorProvider** Ekstra Özellikleri

Özellik	Açıklama
IconAlignment On ErrorProviderIsmi	Hata simgesinin, kontrolün üzerinde nerde bulunacağını belirler.
IconPadding On ErrorProviderIsmi	Hata simgesinin, kontrolden kaç piksel uzakta duracağını belirler.
Error On ErrorProviderIsmi	Varsayılan hata mesajını belirler

Örnek: Kayıt işlemlerinin yapıldığı sırada, isim soyadı ve TC kimlik numaralarının girişleri **ErrorProvider** kontrolü ile denetlenebilir.



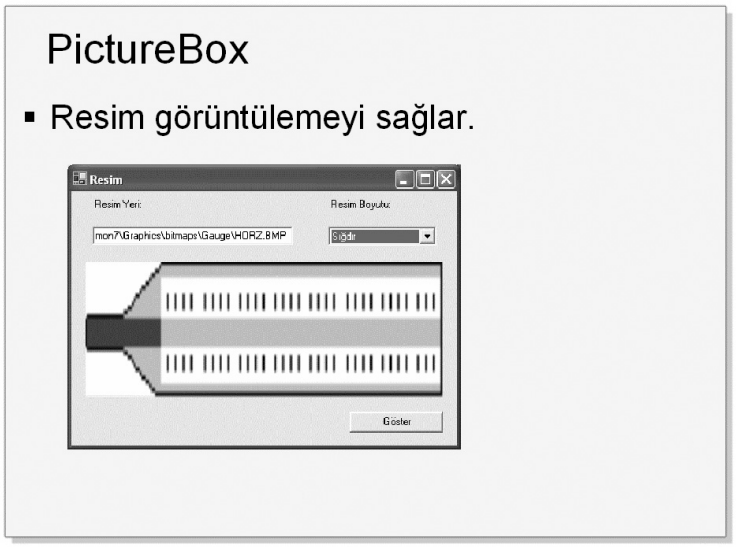
RESİM 9.18: **ErrorProvider** kontrolü örneği.

- Metin kutularının **Validating** olayında, girilen verilerin kontrolleri yapılır ve gerektiği durumlarda **ErrorProvider** ile hata mesajları gösterilir.

```
Private Sub txtIsim_Validating(ByVal sender As Object, ByVal
e As System.ComponentModel.CancelEventArgs) Handles
txtIsim.Validating
    If txtIsim.Text = "" Then
        ErrorProvider1.SetError(txtIsim, "İsim alanı boş
girilemez")
        ' Bu komut olayın gerçekleşmesini engeller
        ' Dolayısıyla veri girilmeden bu alandan çıkılamaz
        e.Cancel = True
    Else
        ' Eğer beri doğru girilmişse, Error simgesini
        ' gizlemek için, hata mesajı boş girilir
        ErrorProvider1.SetError(txtIsim, "")
    End If

End Sub
```

```
Private Sub txtSoyad_Validating(ByVal sender As Object,
ByVal e As System.ComponentModel.CancelEventArgs) Handles
txtSoyad.Validating
    If txtSoyad.Text = "" Then
        ErrorProvider1.SetError(txtSoyad, "Soyad alanı boş
girilemez")
        e.Cancel = True
    Else
        ErrorProvider1.SetError(txtSoyad, "")
    End If

End Sub

Private Sub txtTCKimlik_Validating(ByVal sender As Object,
ByVal e As System.ComponentModel.CancelEventArgs) Handles
txtTCKimlik.Validating
    If Not IsNumeric(txtTCKimlik.Text) Then
        ErrorProvider1.SetError(txtTCKimlik, "Kimlik
numarası yanlış girildi")
        e.Cancel = True
    Else
        ErrorProvider1.SetError(txtTCKimlik, "")
    End If

End Sub
```

PictureBox



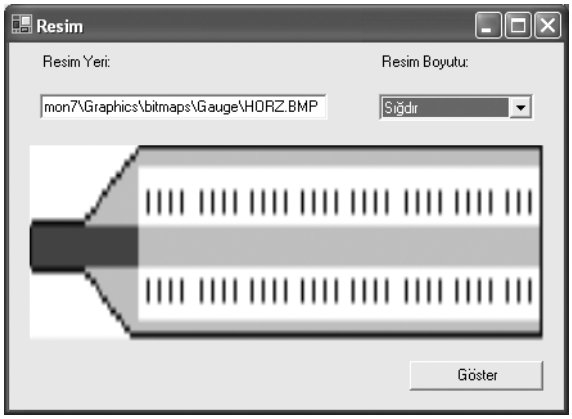
Form üzerinde bir resim görüntülemek için kullanılır.

PictureBox Özellikleri

TABLO 9.38: `PictureBox` Özellikleri

Özellik	Değer Tipi	Açıklama
`Image`	`Image`	Kontrolün resim kaynağını belirler.
`SizeMode`	`PictureBoxSizeMode`	Kontrolün, resmi nasıl görüntüleyeceğini belirler. **`AutoSize`** değeri, kontrolün büyüklüğünü resmin büyüklüğüne göre ayarlar. **`CenterImage`** değeri, resmi kontrolün ortasına gelecek şekilde ayarlar. **`Normal`** değeri, kontrolün sol üst köşesine göre konumlandırır. **`StretchImage`** değeri, resmi kontrolün büyüklüğüne göre boyutlandırır ve resmin tam görünmesini sağlar.

Örnek: Form üzerinde bir resmin değişik boyutlarda gösterilmesi için `PictureBox` kontrolü tercih edilir.



RESİM 9.19: **`PictureBox`** kontrolü örneği.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    ComboBox1.Items.Add("Normal")
    ComboBox1.Items.Add("Ortala")
    ComboBox1.Items.Add("Sığdır")
    ComboBox1.Items.Add("Otomatik Boyutlandır")
End Sub

' ComboBox kontrolünden resmin görüntülenme modu seçilir
Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
    Select Case ComboBox1.SelectedIndex
        Case 0
            PictureBox1.SizeMode = PictureBoxSizeMode.Normal
        Case 1
            PictureBox1.SizeMode =
PictureBoxSizeMode.CenterImage
        Case 2
            PictureBox1.SizeMode =
PictureBoxSizeMode.StretchImage
        Case 3
            PictureBox1.SizeMode =
PictureBoxSizeMode.AutoSize
    End Select
End Sub

Private Sub btnGoster_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnGoster.Click
    PictureBox1.Image = Image.FromFile(txtResimYeri.Text)
End Sub
```

ImageList

`ImageList` kontrolü, form kontrolleri ve içinde bulunan öğeleri için arka plan resmi sağlayan bir listesi görevini görür.

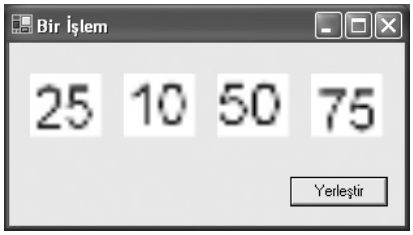
ImageList Özellikleri

TABLO 9.39: **`ImageList`** Özellikleri

Özellik	Değer Tipi	Açıklama
`Images`	**`ImageCollection`**	Kontrolün içinde bulunan resimlerin listelendiği dinamik bir koleksiyondur. Bu özellik bir koleksiyon olduğu için, diğer liste kontrollerinin öğelerinin resmini belirleme işlemi büyük ölçüde kolaylaşır.
`ImageSize`	**`Size`**	Kontrolün tuttuğu resimlerin büyüklüğünü belirler
`TransparentColor`	**`Color`**	Listedeki resimlerin bu özellikte belirtilen renkteki bölgeleri saydam olur.

Windows uygulamalarında **`ImageList`** kontrolünün kullanımı, diğer kontrollerin **`ImageList`** özelliği olarak belirlendikten sonra gerçekleşir. Bu kontrollerin listelediği öğelerin arka plan resimleri **`ImageList`** kontrolü ile belirlenir.

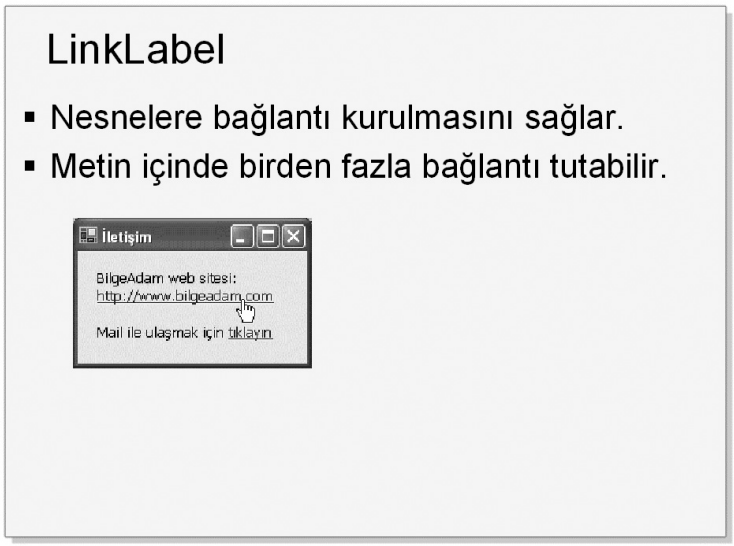
Örnek: **`ImageList`** kontrolünde tutulan resimler bir sayı oyununda rasgele resim göstermek için kullanılabilir.



RESİM 9.20: ImageList kontrolü örneği.

```
Private Sub btnYerlestir_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnYerlestir.Click
    Dim max As Integer = ImageList1.Images.Count - 1

    Randomize()
    PictureBox1.Image = ImageList1.Images(Rnd() * max)
    PictureBox2.Image = ImageList1.Images(Rnd() * max)
    PictureBox3.Image = ImageList1.Images(Rnd() * max)
    PictureBox4.Image = ImageList1.Images(Rnd() * max)
End Sub
```

LinkLabel



Bu kontrol, nesnelere bağlantı kurmak için kullanılır. **`Text`** özelliğinde birden fazla nesneye bağlantı kurulabilir. Bu durumda, kontrol tıklandığı zaman hangi bağlantının işleneceği **`Click`** olayında belirlenir.

LinkLabel Özellikleri

TABLO 9.40: **`LinkLabel`** Özellikleri

Özellik	Değer Tipi	Açıklama
`LinkArea`	**`LinkArea`**	Bağlantının hangi karakterler arasında aktif olacağını belirler.
`LinkBehavior`	**`LinkBehavior`**	Bağlantının yazısında bulunan çizginin ne zaman gösterileceğini belirler. **`HoverUnderline`** değeri fare üzerinde durduğu zaman, **`AlwaysUnderline`** değeri her zaman altı çizili olduğunu belirler. **`NeverUnderline`** değeri ise bağlantı yazısının altının çizilmeyeceğini belirler.
`LinkColor`	**`Color`**	Bağlantının **`LinkVisited`** özelliği **`False`** olduğu zaman gösterilecek rengini belirler.
`LinkVisited`	**`Boolean`**	Bağlantının en az bir kere tıklandığını belirler.
`VisitedLinkColor`	**`Color`**	Bağlantının **`LinkVisited`** özelliği **`True`** olduğu zaman gösterilecek rengini belirler.
`Links`	**`LinkLabel.LinkCollection`**	Kontrolün **`Text`** özelliğinde bulunan bağlantıları tutar.

LinkLabel Olayları

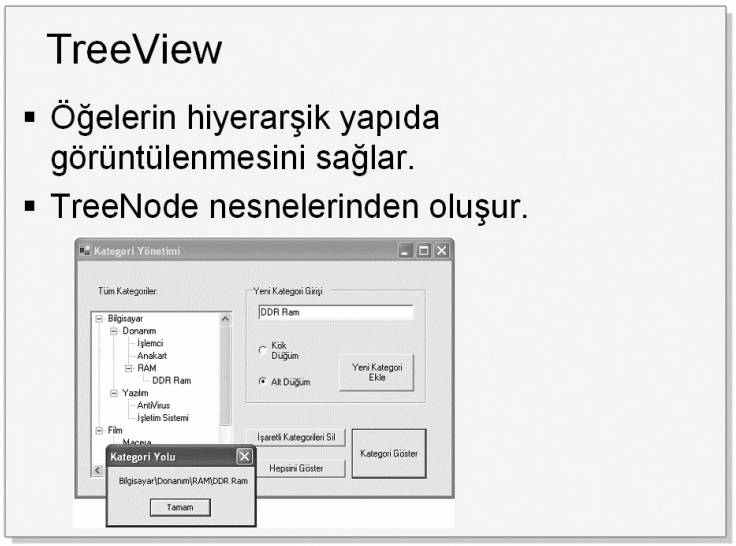
TABLO 9.41: `LinkLabel` Olayları

Olay	Açıklama
`Click`	Kontrolün üzerine tıklandığı zaman gerçekleşir. Diğer kontrollerin tıklama olayından farklı olarak, LinkLabel üzerinde hangi bağlantının tıklandığı anlaşılır.

Örnek: İletişim bilgi formunda e-posta ve Internet adresleri gibi bağlantıları göstermek için `LinkLabel` kullanılır.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    Dim bilgi As String

    bilgi = "BilgeAdam web sitesi: http://www.bilgeadam.com"
    bilgi &= vbCrLf & "Mail ile ulaşmak için tıklayın"

    LinkLabel1.Text = bilgi

    ' İnternet adresinin başladığı karakterden
    ' itibaren link eklenir
    LinkLabel1.Links.Add(22, 24, "http://www.bilgeadam.com")

    ' Mail adresinin başladığı karakterden
    ' itibaren link eklenir
    LinkLabel1.Links.Add(72, 8,
"mailto:postakutusu@bilgeadam.com")
End Sub

Private Sub LinkLabel1_LinkClicked(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles
LinkLabel1.LinkClicked
    Dim tiklanan As Integer
    tiklanan = LinkLabel1.Links.IndexOf(e.Link)

    'Tıklanan linkin ziyaret edildiği belirtilir
    LinkLabel1.Links(tiklanan).Visited = True

    ' Linki çalıştırmak için ilgili işlem gerçekleştirilir
    System.Diagnostics.Process.Start(e.Link.LinkData)
End Sub
```

TreeView



Bu kontrol, içinde bulunan öğeleri hiyerarşik bir yapıda görüntüler. Her eklenen öğe bir düğümü temsil eder. Düğümler birleşerek ağaç yapısını oluştururlar. Düğümler, kök ve alt düğüm olarak ikiye ayrılır. Kök düğümler, kontrolün ilk sırasında yer alır ve aynı seviyededir. Alt düğümler, kök düğümlerin ve diğer alt düğümlerin altına eklenebilir.

TreeNode Nesnesi

`TreeView` kontrolünde gösterilen öğeler, özelliklerini **`TreeNode`** sınıfından alır. Kök ve alt düğümlerin tümü **`TreeNode`** tipindedir. Her düğümün bir **`Nodes`** özelliği vardır. Bu özellik, düğümün alt düğümlerinin tutulduğu koleksiyondur. Alt düğümler oluşturulup bu özelliğe eklenebilir.

`TreeNode` düğümleri oluşturulup, özellikleri atandıktan sonra **`TreeView`** kontrolünde gösterilmesi için, **`TreeView`** nesnesinin **`Nodes`** koleksiyonuna eklenmesi gerekir.

TreeNode Özellikleri

TABLO 9.42: **`TreeNode`** Özellikleri

Özellik	Değer Tipi	Açıklama
`Text`	**`String`**	Düğümün üstünde gösterilen yazıyı belirler.
`Nodes`	**`TreeNodeCollection`**	Düğümün alt düğümlerini tutan koleksiyondur.

TABLO 9.42: `TreeNode` Özellikleri

Özellik	Değer Tipi	Açıklama
`Checked`	`Boolean`	`TreeView` kontrolünde seçim kutuları gösteriliyorsa, düğümün işaretli olup olmadığını belirler.
`NextNode`	`TreeNode`	Aynı seviyedeki bir sonraki düğümü gösterir.
`PrevNode`	`TreeNode`	Aynı seviyedeki bir önceki düğümü gösterir.
`LastNode`	`TreeNode`	Alt düğümlerinin en sonuncusunu gösterir.
`FirstNode`	`TreeNode`	Alt düğümlerinin ilkini gösterir.
`NodeFont`	`Font`	Düğümün yazı tipini belirler.
`FullPath`	`String`	Düğümün, kökten kendisine kadar olan tüm düğümlerin `Text` özelliklerini sıralar.
`Parent`	`TreeNode`	Düğümün ait olduğu `TreeNode` nesnesini belirtir.

TreeNode Metotları

TABLO 9.43: `TreeNode` Metotları

Metot	Açıklama
`Collapse`	Düğümün ilk seviyedeki alt düğümlerini gizler. Eksi işaretinin tıklanması ile aynı görevi görür.
`Expand`	Düğümün ilk seviyedeki alt düğümlerini gösterir. Artı işaretinin tıklanması ile aynı görevi görür.
`ExpandAll`	Düğümün alt düğümlerini son seviyeye kadar gösterir.
`Toggle`	Düğümün durumunu açıksa kapalı, kapalıysa açık duruma getirir.
`GetNodeCount`	Verilen parametre `True` ise tüm alt düğümlerin, `False` ise sadece ilk seviyedeki düğümlerin sayısını verir.

TreeView Özellikleri

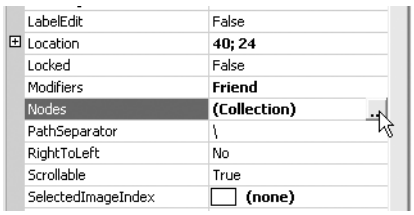
TABLO 9.44: `TreeView` Özellikleri

Özellik	Değer Tipi	Açıklama
`CheckBoxes`	`Boolean`	Düğümlerin yanında işaret kutularının gösterilmesini belirler.
`ImageIndex`	`Integer`	Kontrolün tüm öğeleri için varsayılan resmin `ImageList` içindeki indisini belirler. Bu özelliğin kullanılması için, kontrolün `ImageList` özelliğinin belirlenmesi gerekir.

TABLO 9.44: `TreeView` Özellikleri

Özellik	Değer Tipi	Açıklama
`SelectedImageIndex`	`Integer`	Öğenin üzerine gelip seçildiğinde gösterilecek resmin `ImageList` içindeki indisini belirler.
`SelectedNode`	`TreeNode`	Seçilen düğümü belirler.
`TopNode`	`TreeNode`	Kontrolün ilk kök düğümünü gösterir.
`ShowLines`	`Boolean`	Düğümler arasında çizgilerin gözükmesini belirler.
`ShowPlusMinus`	`Boolean`	Alt düğümleri gösterip gizlemek için kullanılan artı-eksi işaretlerinin gözükmesini belirler.
`ShowRootLines`	`Boolean`	Kök düğümlerinin çizgilerinin ve artı-eksi işaretlerinin gözükmesini belirler.
`PathSeparator`	`String`	Bir düğümün `FullPath` özelliğinde gösterilen düğümleri ayıran karakterleri belirler.

`TreeView` kontrolüne kod ile düğüm eklenebildiği gibi, tasarım anında Visual Studio TreeNode Editor penceresi kullanılarak da düğüm eklenebilir.



RESİM 9.21: Properties panelinde düğüm eklemek.

TreeView Metotları

TABLO 9.45: `TreeView` Metotları

Metot	Açıklama
`CollapseAll`	Kontrolün tüm düğümlerini gizler.
`ExpandAll`	Kontrolün tüm düğümlerini gösterir.

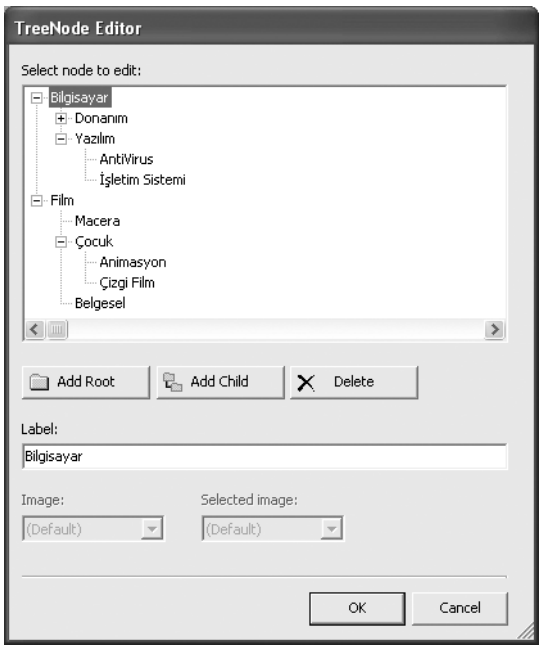
TreeView Olayları

TABLO 9.46: `TreeView` Olayları

Olay	Açıklama
`BeforeSelect`	Düğüm seçilmeden önce gerçekleşir.
`AfterSelect`	Düğüm seçildikten sonra gerçekleşir.

TABLO 9.46: `TreeView` Olayları

Olay	Açıklama
`BeforeCollapse`	Düğüm kapanmadan önce gerçekleşir.
`AfterCollapse`	Düğüm kapandıktan sonra gerçekleşir.
`BeforeExpand`	Düğüm açılmadan önce gerçekleşir.
`AfterExpand`	Düğüm açıldıktan sonra gerçekleşir.



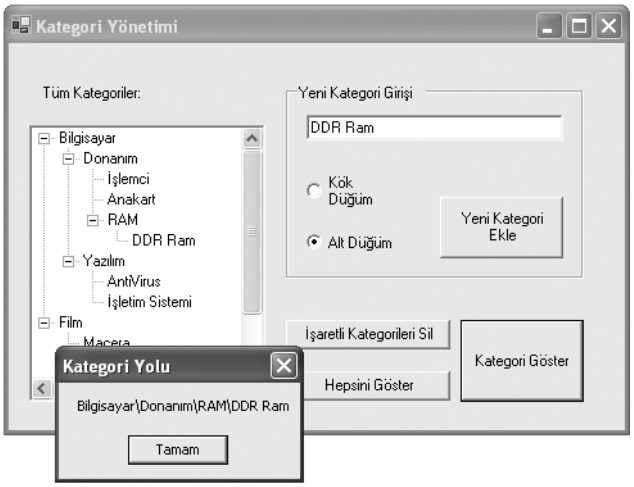
RESİM 9.22: TreeNode Editor penceresi.

Örnek: Ürün kategorileri, genelde tek kategori olarak ele alınsa da, aslında hiyerarşik bir yapıda incelenmeleri gerekir. Her kategorinin sonsuz sayıda alt kategorisi olabilir. Bu tip kategoriler, en iyi şekilde `TreeView` kontrolü ile görüntülenebilir.

- Yeni kategori ekleme işlemi kök düğüm ve alt düğüm olarak yapılabilir. Eğer `RadioButton` kontrollerinde kök düğüm seçilmişse ana kategori; alt düğüm seçilmişse, seçilen kategorinin altına bir alt kategori eklenir.

```
Private Sub btnYeniKategoriEkle_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnYeniKategoriEkle.Click
    Dim secilen As TreeNode
    secilen = TreeView1.SelectedNode

    If RadioButton1.Checked Then
        ' Kök düğüm eklenir
```

```
        TreeView1.Nodes.Add(txtYeniKategori.Text)

    ElseIf RadioButton2.Checked Then
        ' Seçilen kategoriye alt kategori eklenir
        secilen.Nodes.Add(txtYeniKategori.Text)
    End If
End Sub
```



RESİM 9.23: `TreeView` kontrolü örneği.

- Seçilen bir kategorinin silinmesi işlemi için, o düğümün hangi ana düğüme ait olduğu bulunmalıdır.

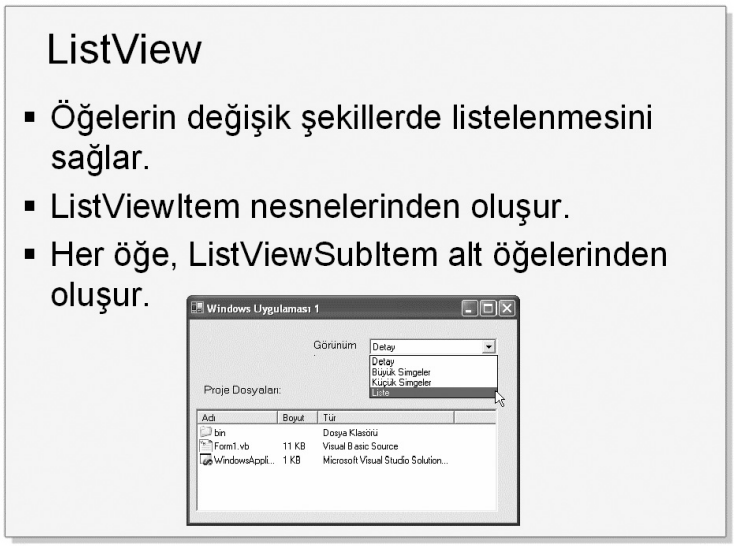
```
Private Sub btnSil_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSil.Click
    Dim secilen As TreeNode = TreeView1.SelectedNode

    If Not secilen.Parent Is Nothing Then
        ' Seçilen düğüm, Parent düğümünün Nodes
        ' koleksiyonundan çıkartılır.
    secilen.Parent.Nodes.Remove(secilen)
   Else
        ' Eğer Parent yok ise Kök düğümdür.
        TreeView1.Nodes.Remove(secilen)
   End If
End Sub
```

- Tüm düğümlerin gösterilmesi ve seçilen düğümün hiyerarşik yapısının gösterilmesi:

```
Private Sub btnGoster_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnGoster.Click
```

```
    TreeView1.ExpandAll()
End Sub

Private Sub btnKategoriGoster_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnKategoriGoster.Click
    Dim secilen As TreeNode = TreeView1.SelectedNode

    MsgBox(secilen.FullPath)
End Sub
```

ListView



Kullanıcıya değişik listeleme seçenekleri sunan bir kontroldür. İçinde bulunan öğeler, tek bir nesne olarak veya detayları ile gösterilebilir. Dolayısıyla öğeler **`ListViewItem`** nesnesi, detayları ise **`ListViewSubItem`** nesnesi olarak tanımlanır.

ListView Özellikleri

TABLO 9.47: **`ListView`** Özellikleri

Özellik	Değer Tipi	Açıklama
`View`	**`View`**	Listenin görünümünü belirler. **`LargeIcons`** değeri listedeki öğelerin büyük resimle, **`SmallIcons`** küçük resimle görünmesini sağlar. **`List`** değeri, öğelerin küçük resimle, fakat alt alta görünmesini sağlar. **`Details`** değeri, alt öğelerin kolonlar altında görüntülendiği detay görünümü sağlar.
`AllowColumnReorder`	**`Boolean`**	Detay görünümünde, kolonların kullanıcı tarafından düzenlenebilmesini belirler.

TABLO 9.47: `ListView` Özellikleri

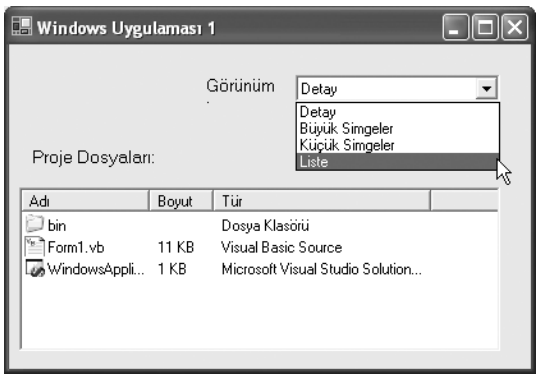
Özellik	Değer Tipi	Açıklama
`Activation`	`ItemActivation`	Öğelerin ne zaman etkinleştirileceğini belirler. **`OneClick`** değeri, öğenin tek tıklamayla, **`Standard`** değeri, öğenin çift tıklamayla aktif hale geleceğini belirler. **`TwoClick`** değeri seçiliyken, ilk tıklandığında öğe seçilir, daha sonra ikinci defa tıklandığında ise öğe aktif hale gelir.
`CheckBoxes`	`Boolean`	Öğelerin yanında seçme kutularının bulunmasını belirler.
`Columns`	`ColumnHeader Collection`	Detay görünümündeyken, öğelerin alt öğelerinin gösterileceği kolonları tutan koleksiyondur.
`FullRowSelect`	`Boolean`	Detay görünümde, öğenin tüm detay satırının seçilmesini belirler.
`GridLines`	`Boolean`	Kolonlar ve satırlar arasında ayırıcı çizgilerin gözükmesini belirler.
`LabelEdit`	`Boolean`	Çalışma anında, kullanıcın, liste öğelerinin yazısını değiştirmesini belirler. Bu özelliğin kullanılması için, **`Activation`** özelliğinin **`Standard`** olması gerekir.

ListView Olayları

TABLO 9.48: `ListView` Olayları

Olay	Açıklama
`AfterLabelEdit`	Öğenin yazısı değiştikten sonra gerçekleşir
`BeforeLabelEdit`	Öğenin yazısı değişmeden önce gerçekleşir

Örnek: Windows Explorer ile dosya görünümleri, `ListView` ile gerçekleştirilir.



RESİM 9.24: `ListView` kontrolü örneği.

- Form yüklenirken **ListView** kontrolüne kolon ve öğeler eklenir. Ayrıca **ComboBox** kontrolüne görünüm seçenekleri eklenir.

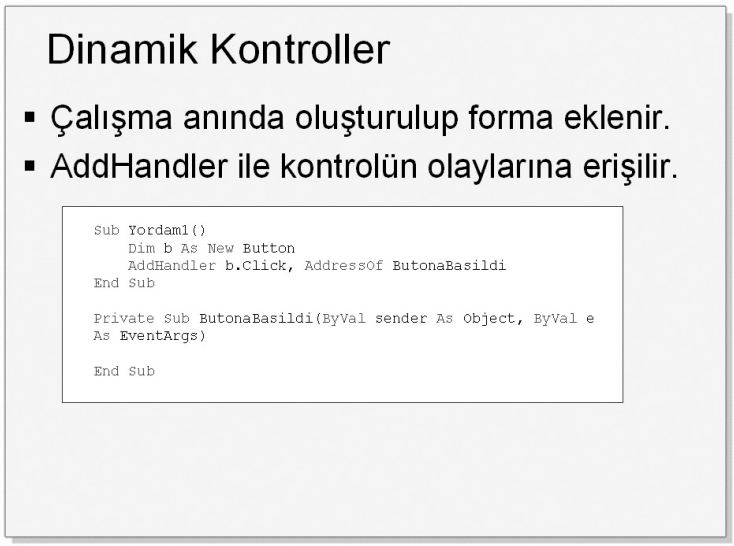
```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    ComboBox1.Items.Add("Detay")
    ComboBox1.Items.Add("Büyük Simgeler")
    ComboBox1.Items.Add("Küçük Simgeler")
    ComboBox1.Items.Add("Liste")
    ComboBox1.DropDownStyle = ComboBoxStyle.DropDownList

    ListView1.Columns.Add("Adı", 100,
HorizontalAlignment.Left)
    ListView1.Columns.Add("Boyut", 50,
HorizontalAlignment.Left)
    ListView1.Columns.Add("Tür", 170,
HorizontalAlignment.Left)

    ListView1.View = View.Details

    Dim oge As New ListViewItem("bin")
    oge.SubItems.Add("")
    oge.SubItems.Add("Dosya Klasörü")
    oge.ImageIndex = 0
    ListView1.Items.Add(oge)

    oge = New ListViewItem("Form1.vb")
    oge.SubItems.Add("11 KB")
    oge.SubItems.Add("Visual Basic Source")
    oge.ImageIndex = 2
    ListView1.Items.Add(oge)

    oge = New ListViewItem("WindowsApplication1.sln")
    oge.SubItems.Add("1 KB")
    oge.SubItems.Add("Microsoft Visual Studio Solution
Object")
    oge.ImageIndex = 1
    ListView1.Items.Add(oge)
End Sub
```

- **ComboBox** kontrolünde seçilen değer değiştiği zaman, **ListView** görünümü değişir.

```
Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
    Select Case ComboBox1.SelectedIndex
```

```
            Case 0
                ListView1.View = View.Details
            Case 1
                ListView1.View = View.LargeIcon
            Case 2
                ListView1.View = View.SmallIcon
            Case 3
                ListView1.View = View.List
        End Select
    End Sub
```

Dinamik Kontroller



Kontroller tasarım anında eklenip ayarlanabildiği gibi, çalışma anında da oluşturulup forma eklenebilir. Kontrollerin, Properties panelinde gözüken tüm özelliklerine kod tarafından ulaşılabildiği için, çalışma anında önceden oluşturulmuş bir kontrolün özelliği değiştirilebilir. Bununla birlikte, yeni bir form oluşturup gösterme işlemi gibi, çalışma anında yeni bir kontrol oluşturulup özellikleri atanarak form üzerinde gösterilebilir.

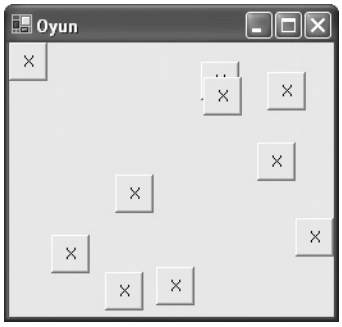
Yeni eklenen kontrollerin olaylarına erişmek için **`AddHandler`** anahtar kelimesi kullanılır. Kontrolün olayı gerçekleştiği zaman çalıştırılacak kodlar ise **`AddressOf`** anahtar kelimesi ile belirtilmelidir.

```
AddHandler b.Click, AddressOf ButonaBasildi
```

Bu şekilde tanımlanan yordamların, kontrolün olay tanımlayıcısı ile aynı parametrelere sahip olmalıdır.

```
Private Sub ButonaBasildi(ByVal sender As System.Object,
ByVal e As System.EventArgs)


End Sub
```

Örnek: Form üzerinde sürekli düğme eklenen ve düğmelerin tıklandıklarında yok edildiği bir oyunun yazılması için, bu düğmelerin dinamik bir şekilde oluşturulması gerekir.



RESİM 9.25: Dinamik kontrol örneği.

- Form üzerindeki bir **`Timer`** kontrolü, iki saniyede bir düğme oluşturup forma ekler.

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Timer1.Tick
    ' Yeni bir düğme oluşturulur.
    Dim b As New Button
    b.Height = 30
    b.Width = 30
    b.Text = "X"

    Dim maxLocation_Y, maxLocation_X As Integer

    ' Yeni düğmenin yeri form dışında bir yerde olamaz
    maxLocation_X = Me.Width - b.Width
    maxLocation_Y = Me.Height - b.Height

    Randomize()

    ' Düğmenin bulunacağı yer rasgele ayarlanır.
    b.Location = New Point(Rnd() * maxLocation_X, Rnd() *
maxLocation_Y)

    AddHandler b.Click, AddressOf ButonaBasildi

    ' Oluşturulan kontrol, Formun kontroller
    ' listesine eklenmelidir.
    Me.Controls.Add(b)
End Sub
```

- Oluşturulan kontroller tıklandığı zaman çalıştırılacak yordam yazılır.

```
Private Sub ButonaBasildi(ByVal sender As System.Object,
ByVal e As System.EventArgs)
    ' Kontrolün, üzerine basıldığı zaman yok edilmesi
```

```
    sender.Dispose()
End Sub
```

- Form yüklendiği zaman **Timer** nesnesi çalışmaya başlar.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    Timer1.Interval = 500
    Timer1.Start()
End Sub
```

Lab 1: Internet Tarayıcısı

Bu labda, Windows altında bulunan Microsoft Web Tarayıcısı kontrolü projeye eklenerek Internet tarayıcısı gerçekleştirilir.

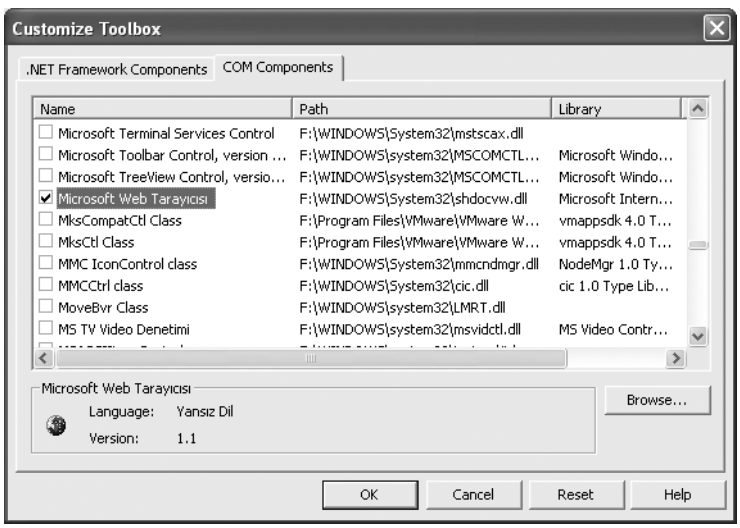
Bu labda kullanılan kontroller ve teknikler:

- **`LinkLabel`.** Ana sayfaya bağlantı sağlar.
- **`RadioButton`.** Bağlantıların yeni ya da aynı pencerede açılması seçeneğini sunar.
- **`GroupBox`. `RadioButton`** kontrollerini gruplamak için kullanılır.
- **`TabControl`.** Tarayıcıların farklı pencerelerde gözükmesini sağlar.
- **Microsoft Web Tarayıcısı.** Internet sitelerinin görüntülenmesini sağlar.
- **`Dispose` metodu**. **`TabPage`** sayfalarının silinmesi için kullanılır.
- **`For Each`.** Sayfaların tümünün kapanması için kullanılır.
- **Dinamik kontroller.** Bağlantılar yeni bir sayfada açıldığı zaman, yeni bir **`TabPage`** oluşturulur. Bu sayfanın içine yeni bir tarayıcı kontrolü oluşturulup eklenir. Daha sonra bu sayfa **`TabControl`** nesnesine eklenir.

Kontrollerin Eklenmesi

Yeni bir Windows projesi açın ve Toolbox paneline Microsoft Web Tarayıcısını ekleyin.

NOT **Toolbox paneline kontrol ekleme işlemleri için Modül 3'e bakın**

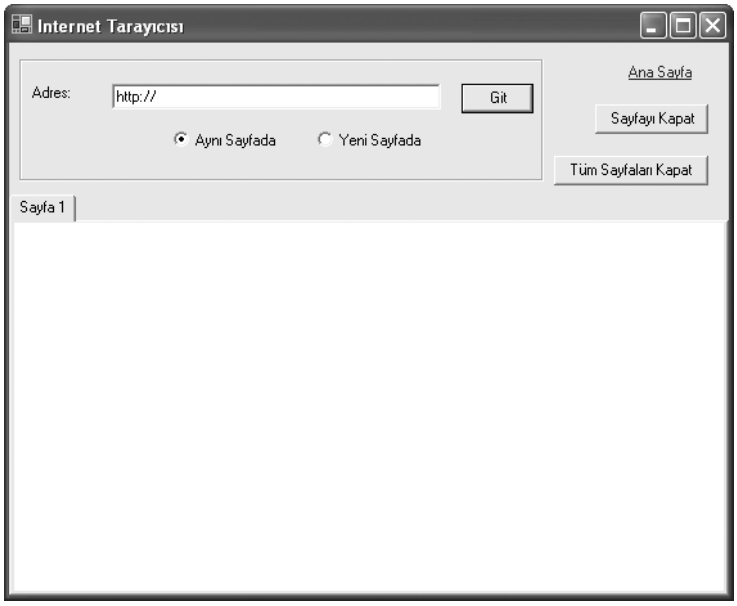


RESİM 9.26: Microsoft Web Tarayıcısı kontrolünün eklenmesi.

Form üzerine Tablo 9.49'daki kontrolleri ekleyin ve belirtilen özellikleri ayarlayın.

TABLO 9.49: Eklenecek Kontroller

Kontrol – Kontrol İsmi	Özellik	Değer
TextBox – txtAdres	Text	http://
RadioButton - rbAyniSayfa	Checked	True
RadioButton - rbYeniSayfa		
GroupBox – GroupBox1	Text	
LinkLabel – LinkLabel1	Text	Ana Sayfa
Button – btnSayfaKapat	Text	Sayfayı Kapat
Button – btnTumunuKapat	Text	Tüm Sayfaları Kapat
TabControl – TabControl1	TabPages	Yeni bir sayfa ekleyin
	Dock	Bottom
TabPage – TabPage1	Text	Sayfa 1
Tarayıcı – AxWebBrowser1	Dock	Fill



RESİM 9.27: Internet tarayıcısının tasarımı.

Kodların Yazılması

1. Form yüklenirken **LinkLabel** kontrolünün göstereceği bağlantıyı ve formun **AcceptButton** özelliğini ayarlayın.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
    LinkLabel1.Links.Add(0, 9, "http://www.bilgeadam.com")
    Me.AcceptButton = btnGit
End Sub
```

2. Yazılan Internet adresine gitmek için kullanıcı, aynı sayfayı veya yeni açılacak bir sayfayı kullanabilir. Seçilen duruma göre aynı sayfada ya da farklı sayfada Internet sitesini görüntüleyen kodları yazın.

```
Private Sub btnGit_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnGit.Click
    ' Girilen bağlantının başında http ifadesi
    ' bulunmuyorsa bu ifade eklenir
    If Not txtAdres.Text.StartsWith("http://") Then
        txtAdres.Text = txtAdres.Text.Insert(0, "http://")
    End If

    ' TabControl nesnesinde sayfa yoksa ya da Yeni Sayfa
    ' seçeneği seçilmişse, adres yeni sayfada gösterilir.
    If rbYeniSayfa.Checked OrElse TabControl1.TabPages.Count
= 0 Then
        YeniSayfa(txtAdres.Text)
    Else
        AyniSayfa(txtAdres.Text)
    End If
End Sub

Sub YeniSayfa(ByVal link As String)
    ' Dinamik kontroller oluşturulur.
    Dim sayfa As New TabPage(link)
    Dim tarayici As New AxSHDocVw.AxWebBrowser
    tarayici.Dock = DockStyle.Fill

    ' Tarayıcı TabPage kontrolüne eklenir
    sayfa.Controls.Add(tarayici)

    ' Oluşturulan sayfa TabControl nesnesine eklenir.
    TabControl1.TabPages.Add(sayfa)

    ' Yeni açılan sayfa seçili olarak gösterilir
    TabControl1.SelectedTab = sayfa

    ' Tarayıcı, verilen bağlantıyı görüntüler
    tarayici.Navigate(link)
End Sub


Sub AyniSayfa(ByVal link As String)
    ' Internet sitesi, seçilen sayfada gösterilir.
    Dim sayfa As TabPage
```

```
    sayfa = TabControl1.SelectedTab

    Dim tarayici As AxSHDocVw.AxWebBrowser
    ' Tarayıcı, sayfanın kontrolleri içinde bulunur.
    ' Sayfada başka kontrol bulunmadığı için, ilk
    ' kontrol tarayıcıdır.
    tarayici = sayfa.Controls(0)

    sayfa.Text = link
    tarayici.Navigate(link)
End Sub
```

3. Ana sayfa bağlantısı tıklandığı zaman, BilgeAdam Internet sitesinin yeni bir sayfada açılmasını sağlayan kodları yazın.

```
Private Sub LinkLabel1_LinkClicked(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles
LinkLabel1.LinkClicked
    YeniSayfa(e.Link.LinkData)
End Sub
```

4. Seçilen sayfayı ve tüm sayfaları kapatan kodları yazın.

```
Private Sub btnSayfaKapat_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnSayfaKapat.Click
   Dim sayfa As TabPage
   sayfa = TabControl1.SelectedTab

   If Not sayfa Is Nothing Then
        sayfa.Dispose()
   End If
End Sub

Private Sub btnTumunuKapat_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnTumunuKapat.Click
    For Each sayfa As TabPage In TabControl1.TabPages
        sayfa.Dispose()
    Next
End Sub
```

Lab 2: Dört Haneli Sayı Bulma Oyunu

Bu labda, MasterMind oyunundan uyarlanmış 4 haneli sayı bulma oyunu programlanacaktır. Oyunun işleyişi rakamları farklı ya da aynı olarak tutulan 4 haneli sayının tahmin edilmesidir. Tahmin edilen sayıyla ilgili ipuçları verilir. Yerini tutan rakamlar + ile, yerini tutmayan, ancak sayı içinde geçen rakamlar – ile belirtilir.

Örnek:

Tutulan sayı: 1980

Tahmin 1: 4952

İpucu: +1 (Sadece 9 rakamı yerini tuttu)

Tahmin 2: 9820

İpucu: +1 -2 (0 yerini tuttu, 9 ve 8 bulundu, ancak yeri tutturulamadı)

Bu labda kullanılan kontroller ve teknikler:

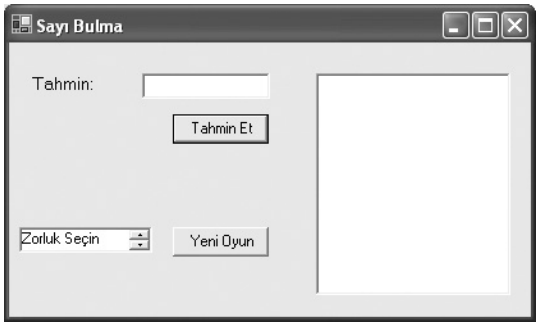
- **`ListBox`.** Yapılan tahminleri tutmayı sağlar.
- **`DomainUpDown`.** Oyunun zorluk derecesinin seçilmesini sağlar.
- **`ErrorProvider`.** Kullanıcının, tahminleri düzgün formatta girip girmediğini kontrol eder.
- **İç İçe Döngüler.** Farklı rakamları olan sayılar üretmek ve tahminleri kontrol etmek için kullanılır.

Kontrollerin Eklenmesi

Form üzerine Tablo 9.50'deki kontrolleri ekleyin ve belirtilen özellikleri ayarlayın.

TABLO 9.50: Eklenecek Kontroller

Kontrol – Kontrol İsmi	Özellik	Değer
`TextBox – txtTahmin`		
`ListBox – ListBox1`		
`DomainUpDown – DomainUpDown1`	`Items`	`Farklı Sayılar`
		`Tekrarlı Sayılar`
	`Text`	`Zorluk Seçin`
`Button – btnTahminEt`	`Text`	`Tahmin Et`
`Button – btnYeniOyun`	`Text`	`Yeni Oyun`
`Label – lblMesaj`		



RESİM 9.28: Dört haneli sayı bulma oyunu tasarımı.

Kodların Yazılması

Sistem tarafından tutulacak sayılar, **DomainUpDown** kontrolünde yapılan seçime göre farklı ya da aynı rakamlara sahip olacaktır.

```
Function SayiUret() As Integer
    Dim sayi As Integer = DortHaneliSayi()

    ' Sayıdaki rakamlar tekrar edilebilirse
    If DomainUpDown1.SelectedIndex = 1 Then
        Return sayi
    End If

    ' Sayının rakamları birbirinden farklı
    ' olana kadar sayı üretilir
    While Not SayiKontrol(sayi)
        sayi = DortHaneliSayi()
    End While

    Return sayi
End Function
```

- Rakamları birbirinden farklı dört haneli sayı üretir.

```
Function DortHaneliSayi() As Integer
    Randomize()
    Dim sayi As Integer = Rnd() * 10000

    ' Sayı 4 haneli olana kadar tekrar üretilir
    While sayi < 1000
        sayi = Rnd() * 10000
    End While

    Return sayi
End Function
```

- Sayının rakamlarının birbirinden farklı olmasını kontrol eder.

```
Function SayiKontrol(ByVal sayi As Integer) As Boolean
    Dim rakamlar() As Char = CStr(sayi).ToCharArray

    ' Rakamlar tek tek bir birleriyle kontrol edilir
    ' Tekrarlanan rakam varsa False döner
    For i As Integer = 0 To rakamlar.Length - 2
        For j As Integer = i + 1 To rakamlar.Length - 1
            If rakamlar(i) = rakamlar(j) Then
                Return False
            End If
        Next
    Next
    Return True
End Function
```

- Yeni oyun düğmesine tıklandığı zaman sayı üretilir ve oyun başlar.

```
Private Sub btnYeniOyun_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnYeniOyun.Click
    BulunacakSayi = SayiUret()
    lblMesaj.Text = "Yeni Oyun! Sayı üretildi..."
End Sub
```

- Metin kutusunun **`Validating`** olayında, girilen değerler kontrol edilir.

```
Private Sub txtTahmin_Validating(ByVal sender As Object,
ByVal e As System.ComponentModel.CancelEventArgs) Handles
txtTahmin.Validating
    If txtTahmin.Text.Length = 4 And
IsNumeric(txtTahmin.Text) Then
        ErrorProvider1.SetError(txtTahmin, "")
    Else
        ErrorProvider1.SetError(txtTahmin, "Sayı yanlış
girildi")
        e.Cancel = True
    End If
End Sub
```

- Tahmin edilen sayının hangi rakamlarının tutuğu kontrol edilir.

```
    Public Function TahminKontrol(ByVal sayi As Integer) As
String
        Dim sonuc As String

        ' Sonuç kümesindeki artı ve eksi sayısı
        Dim arti As Byte = 0
```

```
        Dim eksi As Byte = 0
        Dim i, j As Byte

        Dim sdizi() As Char
        sdizi = CStr(sayi).ToCharArray

        Dim sBulunacak() As Char
        sBulunacak = CStr(BulunacakSayi).ToCharArray

        ' Yerleri tutan sayılar bulunur
        For i = 0 To 3
            If sdizi(i) = sBulunacak(i) Then
                arti += 1
            End If
        Next

        ' Yerleri tutmayan sayıların kontrolü
        For i = 0 To 3
            For j = 0 To 3
                If i <> j Then
                    If sdizi(i) = sBulunacak(j) Then
                        eksi += 1
                        Exit For
                    End If
                End If
            Next
        Next

        If arti = 0 And eksi = 0 Then
            sonuc = "0"
        ElseIf arti = 4 Then
            sonuc = "Tebrikler!"
        ElseIf arti <> 0 AndAlso eksi <> 0 Then
            sonuc = "+" & arti & " -" & eksi
        ElseIf arti = 0 Then
            sonuc = "-" & eksi
        End If

        Return sonuc
    End Function

    Private Sub btnTahmin_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnTahmin.Click
        ListBox1.Items.Add(TahminKontrol(txtTahmin.Text))
    End Sub
```

Lab 3: Hafıza Oyunu

Hafıza oyunu, belli sayıda kart içinden aynı resme sahip olanların bulunmasına dayalıdır. Bu labda, form üzerine seçilen seviye kadar kart ekleme işlemi yapılır. Kontroller çalışma anında ekleneceği için dinamik olarak oluşturulmalıdır.

Bu labda kullanılan kontroller ve teknikler:

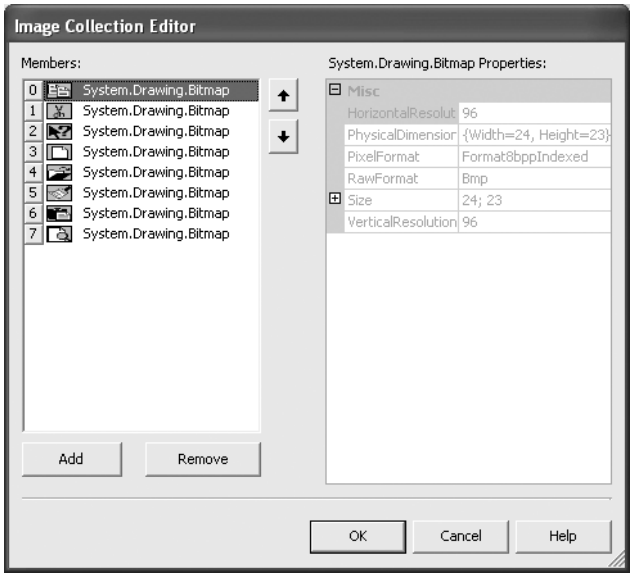
- **`ComboBox`.** Seviyenin seçilmesi için kullanılır.
- **`ImageList`.** Eklenen kartların resimlerini tutar.
- **Dinamik kontroller.** Kullanıcının seçtiği seviye kadar kart ekleme işlemi için kullanılır.
- **`Tag`.** Kontrollerin **`Tag`** özelliği, o kontrole ait bilgi tutmak için kullanılır. Bu labda, yeni eklenen kartların hangi resmi taşıyacağı kontrolü **`Tag`** özelliğinde tutulur.

Kontrollerin Eklenmesi

Form üzerine Tablo 9.51'deki kontrolleri ekleyin ve belirtilen özellikleri ayarlayın.

TABLO 9.51: Eklenecek Kontroller

Kontrol – Kontrol İsmi	Özellik	Değer
`ComboBox`– `ComboBox1`	`Items`	`4 Kart`
		`8 Kart`
		`16 Kart`
`ImageList` – `ImageList1`	`Images`	`8 tane resim ekleyin`



RESİM 9.29: Image Collection Editor.

Kodların Yazılması

1. **ComboBox** kontrolünden seviye seçildiği zaman, form üzerinde varolan tüm düğmelerin silinip, seçilen seviye kadar düğme eklenmesi gerekir. Bu işlem oyunu baştan başlatır.

```
Sub KartYerlestir(ByVal kartSayisi As Integer)
    DugmeleriSil()
    Dim x As Integer = 10
    Dim y As Integer = 50

    For i As Integer = 1 To kartSayisi
        ' Dinamik bir düğme oluşturulur ve özellikleri
        ' ayarlanır
        Dim kart As New Button
        kart.Height = 30
        kart.Width = 30
        kart.Location = New Point(x, y)

        ' Düğme tıklandığı zaman gerçekeşecek olay
        AddHandler kart.Click, AddressOf ButonaTiklandi

        Me.Controls.Add(kart)

        ' Bir sonraki eklenecek olan düğme
        ' ilk kontrolün 70 piksel sağında olacaktır
        x += 70

        ' Düğmenin Form sınırları içinde olması gerekir.
        If x > Me.Width Then
            x = 10
            y += 50
        End If
    Next
    KartResimYukle()
End Sub
```

2. Düğmeleri silme işlemi, form üzerindeki tüm düğmelerin bir listeye atılıp daha sonra formun kontrollerinden kaldırılarak yapılır.

```
Sub DugmeleriSil()
    Dim silinecek As New ArrayList

    ' Form içindeki Button kontrolleri bir listede
tutulur
    For Each c As Control In Me.Controls
```

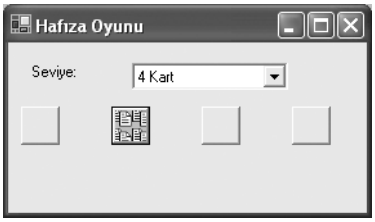
```
        If TypeOf c Is Button Then
            silinecek.Add(c)
        End If
    Next

    For i As Integer = 0 To silinecek.Count - 1
        Me.Controls.Remove(silinecek(i))
    Next
End Sub
```

3. Kartlara resim yüklerken, her resmin iki karta yüklenmesi gerekir.

```
Sub KartResimYukle()
    ' Düğmeler bir listeye alınır.
    Dim dugmeler As New ArrayList
    For Each c As Control In Me.Controls
        If TypeOf c Is Button Then
            dugmeler.Add(c)
        End If
    Next

    Randomize()
    Dim i As Integer = 0

    ' Kartlar ikişer ikişer ele alınır. İki karta da
    ' aynı resim atanır. Ve bu iki kart düğmeler
    ' listesinden çıkartılır.
    While dugmeler.Count > 0
        Dim kart1, kart2 As Button

        kart1 = dugmeler(Rnd() * (dugmeler.Count - 1))
        kart1.Tag = i
        dugmeler.Remove(kart1)

        kart2 = dugmeler(Rnd() * (dugmeler.Count - 1))
        kart2.Tag = i
        dugmeler.Remove(kart2)

        i += 1
    End While
End Sub
```

4. Eklenen kartlar tıklandığı zaman, ilk seferde bir kart açılır ve resmi gösterilir. İkinci kart açıldığı zaman bu iki kartın resmi aynıysa kart formdan kaldırılır.

```
    Private AcikKart As Button
    Private acik As Boolean = False

    Private Sub ButonaTiklandi(ByVal sender As Object, ByVal
e As EventArgs)
        Dim kart As Button = sender

        ' Eğer ilk kart açılıyorsa
        If Not acik Then
            ' Kartı görüntüle
            kart.BackgroundImage =
ImageList1.Images(kart.Tag)
            AcikKart = kart
            acik = True

            'Eğer ikinci kart açılıyorsa
        Else
            ' Açılmış kartın resmi, yeni açılan kartın
            ' resmi ile aynıysa, bu kartlar silinir
            If kart.Tag = AcikKart.Tag Then
                Me.Controls.Remove(kart)
                Me.Controls.Remove(AcikKart)
            Else
                AcikKart.BackgroundImage = Nothing
            End If
            acik = False
        End If
    End Sub
```



RESİM 9.30: Hafıza oyunu.

Lab 4: Hesap Makinesi

Bu labda, bir hesap makinesinde kullanılan genel fonksiyonlar gerçekleştirilecektir.

Bu labda kullanılan kontroller ve teknikler:

- **`Button`.** Hesap makinesindeki her işlem ve sayı için bir düğme kullanılır.
- **`Try Catch Finally`.** Hesaplamalar yapılırken, kullanıcın yanlış bir değer girmesi durumunda çıkacak hataları yakalamak için kullanılır.

Kontrollerin Eklenmesi

Form üzerine Tablo 9.52'deki kontrolleri ekleyin ve belirtilen özellikleri ayarlayın.

TABLO 9.52: Eklenecek Kontroller

Kontrol – Kontrol İsmi	Özellik	Değer
`Button` – 0 – 9 arası, her sayı için **`Button(Sayi)`** isminde bir düğme ekleyin. Örnek: 5 sayısı için **`Button5`**	**`Text`**	Temsil ettikleri sayılar
`Button` – Her işlem için bir düğme ekleyin: Çarpma, bölme, toplama çıkarma, eşitlik, temizleme	**`Text`**	Temsil ettikleri işlemler. **`* + / - = C`**

RESİM 9. 31: Hesap makinesi.

Kodların Yazılması

1. İşlemin türünü ve seçildiğini belirleyen, girilen bir önceki sayıyı tutan global değişkenleri yazın.

```
Private IslemSecildi As Boolean = False
Private Sayi As Double
Private Islem As String
```

2. Sayı düğmelerinden herhangi biri tıklandığı zaman, metin kutusunun görünümünü değiştiren işlemi yazın.

```
 Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click,
Button0.Click, Button2.Click, Button3.Click, Button4.Click,
Button5.Click, Button6.Click, Button7.Click, Button8.Click,
Button9.Click
        If Not IslemSecildi Then
            txtSayi.Text &= sender.Text
        Else
            txtSayi.Text = sender.Text
            IslemSecildi = False
        End If
    End Sub
```

3. İşlem seçildiği zaman, bir önceki girilen sayıyı tutan kodları yazın.

```
    Private Sub btnCarp_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnCarp.Click,
btnBol.Click, btnCikar.Click, btnTopla.Click
        Islem = sender.Text
        Try
            Sayi = txtSayi.Text
            IslemSecildi = True
        Catch ex As Exception
            MsgBox("Sayı düzgün formatta girilmedi")
        Finally
            txtSayi.Text = ""
            txtSayi.Focus()
        End Try
    End Sub
```

4. Eşittir düğmesi tıklandığı zaman aritmetik operasyonu yapan kodları yazın.

```
    Private Sub btnEsit_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnEsit.Click
        Select Case Islem
            Case "*"
                Sayi *= txtSayi.Text
            Case "/"
                Sayi /= txtSayi.Text
            Case "-"
                Sayi -= txtSayi.Text
            Case "+"
                Sayi += txtSayi.Text
```

```
        End Select
        txtSayi.Text = Sayi
    End Sub
```

5. **C** (temizle) düğmesi tıklandığı zaman, metin kutusunu temizleyen ve global değişkenleri başlangıç değerlerine getiren kodları yazın.

```
    Private Sub btnTemizle_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnTemizle.Click
        Sayi = 0
        IslemSecildi = False
        txtSayi.Text = ""
        txtSayi.Focus()
    End Sub
```

Modül Sonu Soruları & Alıştırmalar

1. Formun kapanmasını, şeffaflığını yavaşça azaltarak sağlamak için, formun hangi olay, özellik ve metotlarından faydalanır? Uygulamasını yazın.
2. Fiziksel olarak bulundukları yerler bir dizide tutulan resimlerin, slayt gösterisi şeklinde gösterilmesi hangi kontroller ile sağlanır? Uygulamasını yazın.
3. Kurumsal bir şirketin elemanlarının bağlı oldukları departmanlar ve müdürler hiyerarşik olarak hangi kontrol ile gösterilebilir? Her müdür ve departman başka bir müdür ve departmana bağlıdır. Uygulamasını `Structure` yapısını kullanarak ve ilgili kontrollerle birlikte yazın.
4. Microsoft Excel ile oluşturulan sayfalar, aynı pencerede tutulur. Bir Windows uygulamasında sınırsız sayıda sayfanın aynı form üzerinde tutulmasını hangi kontrol sağlar? Bu sayfalar çalışma anında oluşturulmak istenirse, kontrolün hangi özelliklerinden faydalanılır?

Modül 10: Menü Tasarımı ve MDI Formlar

Konu 1: Menü Tasarımı

Windows uygulamalarında en çok kullanılan tasarım araçları menülerdir. Dosya, Düzen, Görünüm gibi menüler neredeyse tüm Windows uygulamalarında, belli başlı işlerin yapılmasında kullanıcıya kolay erişim sağlar.

Uygulamalarda, menülerde tanımlanan işlemlere görsel kısayollar sunulur. Bu işlem araç kutuları ile sağlanır.

Bu bölüm tamamlandıktan sonra:

- **`MainMenu`** ve **`ContextMenu`** kontrolleri ile menü tanımı yapabilecek,
- **`ToolBar`** kontrolü ile tasarımda araç çubuklarını kullanabilecek,
- **`ToolTip`** kontrolü ile menü araçlarının kullanımı hakkında bilgi sağlayacak,
- **`StatusBar`** ve **`NotifyIcon`** kontrolleri ile uygulamaların tasarımını zenginleştireceksiniz.

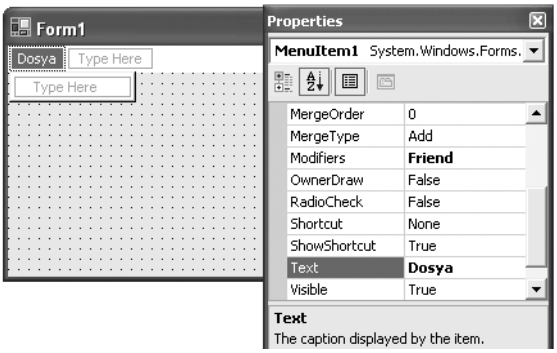
Menüler

Windows uygulamalarında kullanılan iki tip menü vardır. **`MainMenu`**, formların başında duran sabit menüdür. **`ContextMenu`**, fare ile sağ tıklandığında çıkan menüdür.

MainMenu

Windows uygulamasına bir menü eklemek için, Toolbox panelinden bir **`MainMenu`** kontrolünü forma sürükleyin. Eklenen menü bir bileşen olarak formun alt bölümünde gözükecektir. Ancak üstüne gelindiğinde formun başlığının hemen altında belirir. Menü öğesi eklemek veya ismini değiştirmek için üstüne

gelinir ve başlık yazısı yazılır. Properties panelinde bu menünün **MenuItem** olarak eklendiği görülür (Resim 10.1).



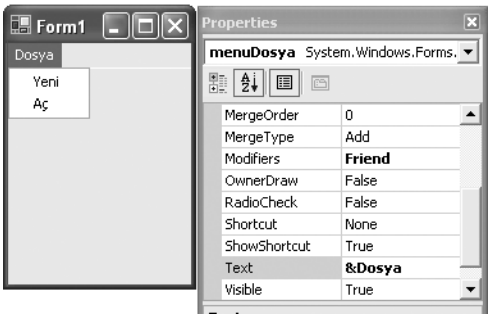
RESİM 10.1: **MenuItem**.

Menüye **MenuItem** eklendiğinde hemen altında ve yanında, menü eklemek için bir yer açılır. Bu açılan yere de menü ismi girilip, alt menü öğeleri oluşturulabilir.

Menü öğeleri tıklandığı zaman bir işlemin gerçekleşmesi için, kontrol çift tıklanarak bu öğenin **Click** olayına geçilir. Çalıştırılmak istenen kodlar buraya yazılır.

```
Private Sub menuYeni_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles menuYeni.Click

End Sub
```

Menü öğelerine isim verirken **&** işareti kullanılarak, kullanıcın klavyenin **ALT** tuşuyla bu öğeyi çalıştırması sağlanabilir. **&** işareti hangi karakter ile kullanılırsa, kısayol olarak o karakter kullanılır (Resim 10.2).



RESİM 10.2: Kısayol tuşunun belirlenmesi.

MenuItem Özellikleri

TABLO 10.1: `MenuItem` Özellikleri

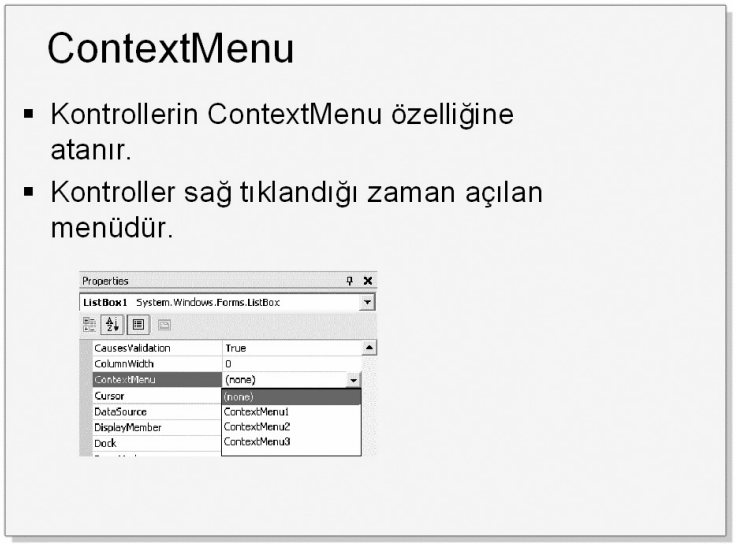
Özellik	Değer Tipi	Açıklama
Checked	**Boolean**	Menü öğesinin yanında seçili olduğuna dair bir işaretin gözükmesini sağlar.
Enabled	**Boolean**	Menü öğesinin aktif durumda olup olmadığını belirler.
RadioCheck	**Boolean**	Öğenin seçilme stilinin **RadioButton** düğmesi olarak gözükmesini sağlar.
ShortCut	**ShortCut**	Menüye ulaşım için bir kısayol tanımlar.
ShowShortCut	**Boolean**	Menünün kısayolunun, isminin yanında gözükmesini belirler.
MenuItems	**MenuItemCollection**	Alt menülerin tutulduğu koleksiyondur.

Örnek:

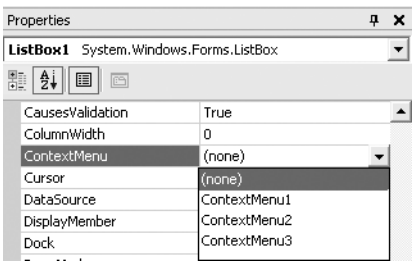
```
    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Dim dosya As New MenuItem("D&osya")

        ' Yeni işleminin yapılması için bir menü eklenir.
        Dim yeni As New MenuItem("&Yeni")
        yeni.Shortcut = Shortcut.CtrlN
        yeni.ShowShortcut = True
        AddHandler yeni.Select, AddressOf YeniClick
        dosya.MenuItems.Add(yeni)

        ' Açma işleminin yapılması için bir menü eklenir.
        Dim ac As New MenuItem("&Aç")
        ac.Shortcut = Shortcut.CtrlO
        ac.ShowShortcut = False
        AddHandler ac.Select, AddressOf AcClick
        dosya.MenuItems.Add(ac)

        MainMenu1.MenuItems.Add(dosya)
    End Sub

    Private Sub AcClick(ByVal sender As System.Object, ByVal
e As System.EventArgs)

    End Sub


    Private Sub YeniClick(ByVal sender As System.Object,
ByVal e As System.EventArgs)


    End Sub
```

RESİM 10.3: Menü örneği.

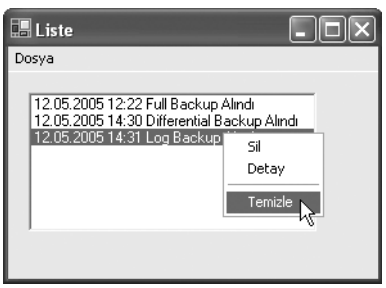
ContextMenu



ContextMenu, bir kontrol sağ tıklandığı zaman açılan menüdür. Bu menü uygulamaya eklendiği zaman Properties panelinde, kontrollerin **ContextMenu** özelliği olarak bu menü atanabilir (Resim 10.4).

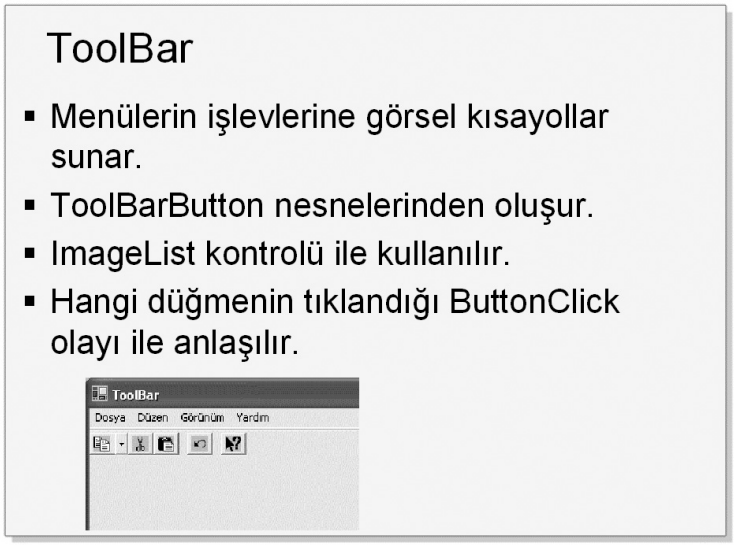


RESİM 10. 4: ContextMenu özelliği.



RESİM 10.5: ContextMenu örneği.

ToolBar



ToolBar kontrolü menülerin altında kullanıcıya kısayollar ve kullanım kolaylığı sunan bir kontroldür. Kontroldeki öğeler çoğu zaman **ImageList** kontrolünün sağladığı resimlerle gösterilir. Resim yerine yazı da gösterilebilir, ancak yazı ile işlemlerin listelenmesi menülerle sağlanır.

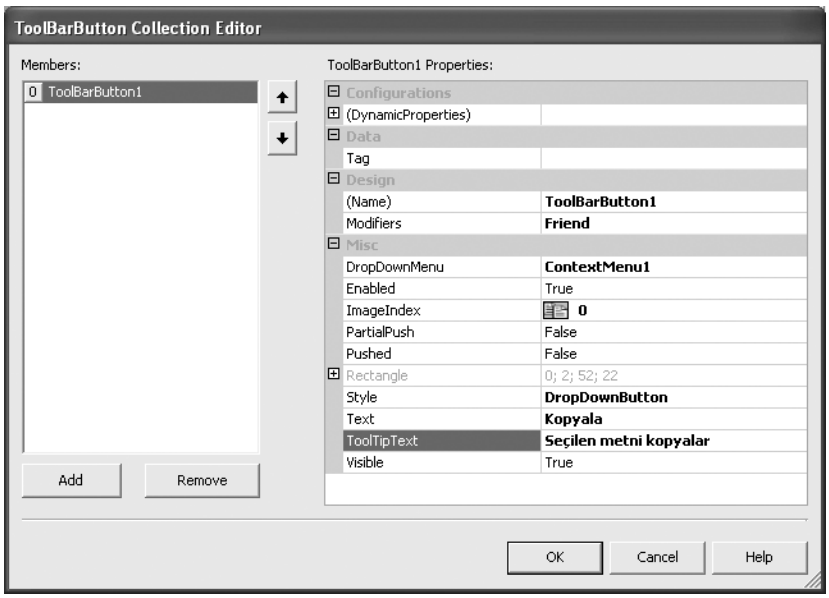
ToolBar kontrolünde yapılacak işlemler bir **ToolBarButton** olarak gösterilir.

ToolBar Özellikleri

TABLO 10.2: **ToolBar** Özellikleri

Özellik	Değer Tipi	Açıklama
Buttons	**ToolBarButtonCollection**	Kontrolün düğmelerinin tutulduğu koleksiyon
ButtonSize	**Size**	Kontroldeki düğmelerin büyüklüğünü belirler. Düğmelerin büyüklükleri ayrı ayrı belirlenemez.
DropDownArrows	**Boolean**	Stili **DropDownButton** olarak seçilmiş **ToolBarButton** düğmelerinin alt menüsünün görünmesini belirler.

ToolBar kontrolüne **ToolBarButton** düğmeleri eklemek için kontrolün **Buttons** özelliğinden faydalanılır. Tasarım anında Properties panelinden **Buttons** özelliği tıklandığı zaman açılan pencerede, kontrole düğme eklenir (Resim 10.6).



RESİM 10.6: ToolBarButton Collection Editor penceresi.

ToolBarButton Özellikleri

TABLO 10.3: `ToolBarButton` Özellikleri

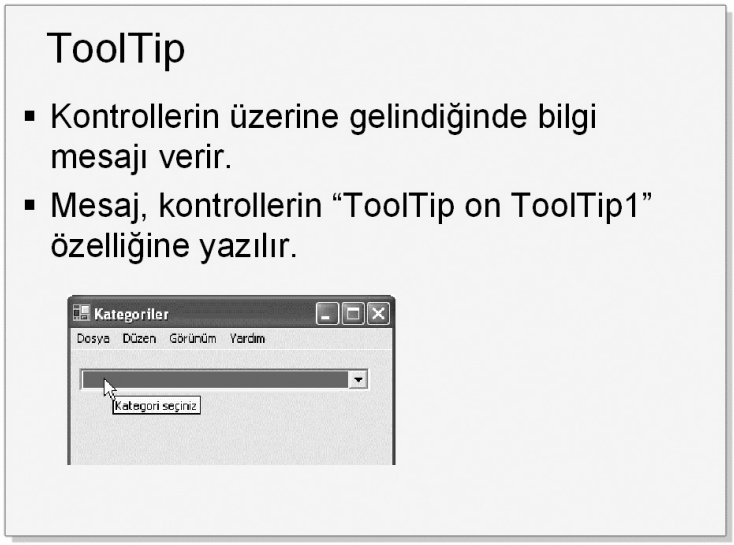
Özellik	Değer Tipi	Açıklama
`Style`	`ToolBarButtonStyle`	Düğmenin görünüm stilini belirler. **`PushButton`** değeri standart bir düğmeyi, **`ToggleButton`** tıklandığı zaman basılı kalan bir düğmeyi, **`Separator`** değeri düğmeler arasında bir ayracı temsil eder. **`DropDownButton`** değeri düğmenin yanında bir menünün açılacağını belirler.
`DropDownMenu`	`Menu`	Kontrolün stili **`DropDownButton`** olarak seçilmişse, yanında çıkacak menüyü belirler. Bu menü sadece **`ContextMenu`** cinsinden olabilir.
`Pushed`	`Boolean`	Düğmenin basılı olup olmadığını belirler.
`Text`	`String`	Düğmenin üzerinde yazan yazıyı belirler.
`ImageIndex`	`Integer`	**`ToolBar`** kontrolüne bir **`ImageList`** bağlanmışsa, bu özellik düğmenin hangi resmi göstereceğini belirler.
`ToolTipText`	`String`	Düğmenin üzerinde durulduğu zaman gösterilecek ipucunu belirler.

Düğmeler tıklandığı zaman çalışması istenen kodlar, **ToolBar** kontrolünün **ButtonClick** olayına yazılır. Ancak burada hangi düğmenin tıklandığının kod yazarak bulunması gerekir.

RESİM 10.7: **ToolBar** örneği.

```
    Private Sub ToolBar1_ButtonClick(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.ToolBarButtonClickEventArgs) Handles
ToolBar1.ButtonClick
        Select Case ToolBar1.Buttons.IndexOf(e.Button)
            ' Ayraçlar da bir ToolBarButton olduğu için
            ' indisler kontrol edilirken buna dikkat
edilmelidir
            Case 0
                ' Kopyala
            Case 1
                ' Kes
            Case 2
                ' Yapıştır
            Case 4
                ' Geri Al
            Case 6
                ' Yardım
        End Select
    End Sub
```

ToolTip

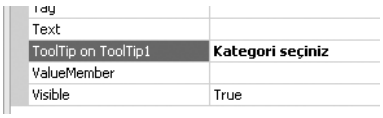


Bu kontrol, form üzerindeki kontrollerin üzerine gelindiği zaman ipucu göstermek için kullanılır (Resim 10.9). **`ToolTip`** forma eklendiği zaman, kontrollerin özelliklerinde **`ToolTip on [ToolTip kontrolünün ismi]`** şeklinde bir özellik belirir. Bu özelliğe verilen yazılar, çalışma anında kontrollerin ipucunu belirler (Resim 10.8).

ToolTip Özellikleri

TABLO 10.4: **`ToolTip`** Özellikleri

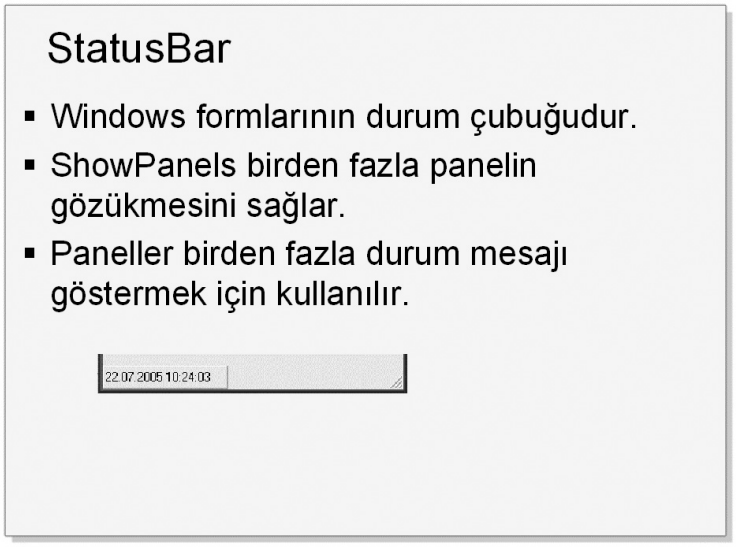
Özellik	Değer Tipi	Açıklama
`Active`	**`Boolean`**	Kontrolün aktif olup olmadığını belirler. **`False`** değerini alırsa, form üzerinde ipuçları görüntülenmez.
`AutomaticDelay`	**`Integer`**	**`AutoPopDelay`**, **`InitialDelay`**, **`ReshowDelay`** değerleri için otomatik süreleri ayarlar.
`AutoPopDelay`	**`Integer`**	İpucunun görüntülenme süresini belirler.
`InitialDelay`	**`Integer`**	İpucunun gözükmesi için, fare imlecinin kontrol üzerinde durması gereken süreyi belirler
`ReshowDelay`	**`Integer`**	Yeni bir kontrolün üzerine gelindiği zaman, bu kontrole ait ipucunun gösterilmesi için gereken süreyi belirler.
`ShowAlways`	**`Boolean`**	Seçilen kontrol aktif olmadığı zamanlarda dahi ipucunun gösterilmesini sağlar.



RESİM 10.8: `ToolTip` özellikleri.

RESİM 10.9: `ToolTip` kontrolü örneği.

StatusBar



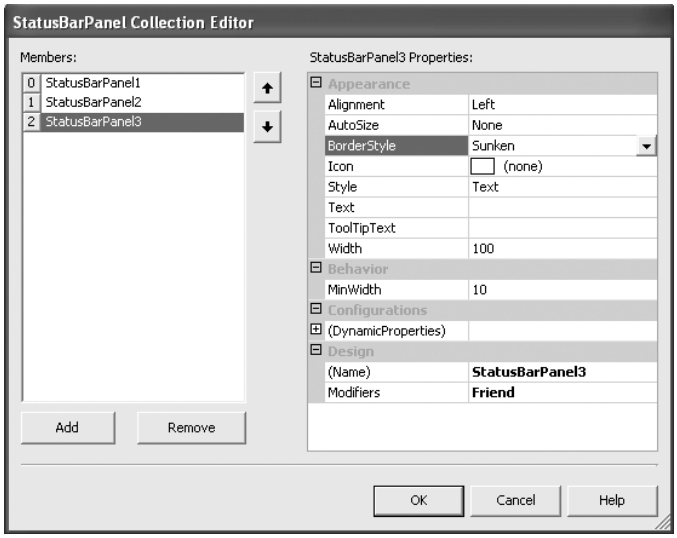
Bu kontrol, Windows uygulamalarında formların altında bulunan durum çubuğunu temsil eder. Durum çubuklarında sadece bir yazı görüntülenebildiği gibi, içindeki paneller ile birden fazla durum yazısı görüntülenebilir.

StatusBar Özellikleri

TABLO 10.5: `StatusBar` Özellikleri

Özellik	Değer Tipi	Açıklama
`Panels`	`StatusBarPanel Collection`	Kontrolün içinde birden fazla yazı görüntülemek için kullanılan panelleri tutar.
`ShowPanels`	`Boolean`	Birden fazla panelin gözükmesini belirler.
`SizingGrip`	`Boolean`	`StatusBar` kontrolünün yanında, formun büyüklüğünü değiştirmek için kullanılan simgenin gözükmesini belirler.
`Text`	`String`	`StatusBar` üzerinde yazan yazıyı belirler. Eğer `ShowPanels` özelliği `True` ise, bu özellikte yazılan yazı gözükmez.

`StatusBar` kontrolüne panel eklemek için kontrolün `Panels` özelliğinden yararlanılır.



RESİM 10.10: StatusBarPanel Collection Editor penceresi.

Panel Özellikleri

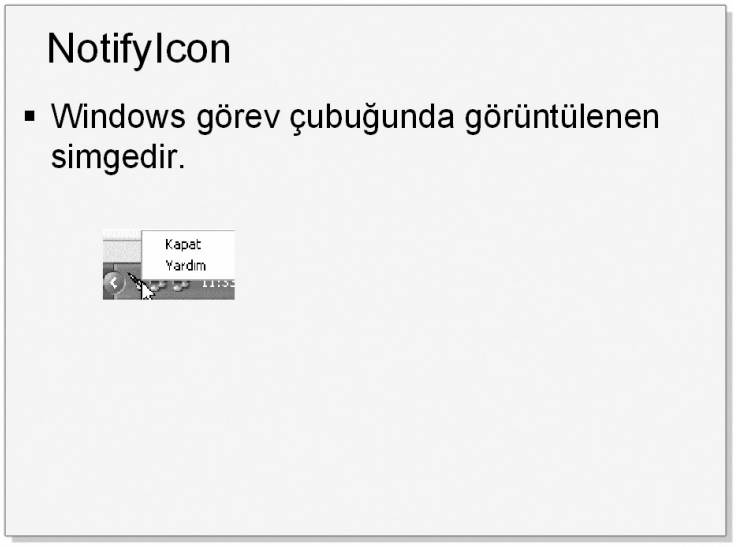
TABLO 10.6: Panel Özellikleri

Özellik	Değer Tipi	Açıklama
`AutoSize`	`StatusBarPanel AutoSize`	Panelin bazı durumlara göre otomatik boyutlandırmasını sağlar. **`None`** değeri, panelin büyüklüğünün değişmeyeceğini, **`Contents`** değeri, panelin içerdiği yazıya göre değişeceğini belirler. **`Spring`** değeri, durum çubuğundaki boş alanların paylaşılmasını sağlar.
`BorderStyle`	`StatusBarPanel BorderStyle`	Panelin kenarlık stilidir. **`Raised`** değeri, panelin bir düğme gibi gözükmesini, **`Sunken`** değeri, panelin basık gözükmesini sağlar. **`None`** değeri, kenarların gözükmesini engeller.
`Alignment`	`Horizontal Alignment`	Panelin yazısının hizalanmasının belirler.
`Text`	`String`	Panel üzerinde yazan yazıyı belirler.
`Width`	`Integer`	Panelin genişliğini belirler.
`MinWidth`	`Integer`	Panel büyüklüğünün minimum değerini belirler.
`Style`	`StatusBar PanelStyle`	Panelin üzerindeki yazıların stilini belirler. **`Text`** değeri, normal yazı gözükmesini sağlar. **`OwnerDraw`**, değişik font ve renklerde yazıların görüntülenmesini sağlar.
`Icon`	`Icon`	Panel üzerinde görüntülenen simgeyi belirler.

```
  Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Dim p As New StatusBarPanel
        p.MinWidth = 100
        p.AutoSize = StatusBarPanelAutoSize.Contents
        p.Alignment = HorizontalAlignment.Left
        p.BorderStyle = StatusBarPanelBorderStyle.Raised
        p.Style = StatusBarPanelStyle.Text

        StatusBar1.Panels.Add(p)

        Timer1.Interval = 1000
        Timer1.Start()
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Timer1.Tick
        Dim panel As New StatusBarPanel
        panel = StatusBar1.Panels(0)
        panel.Text = Now
    End Sub
```

RESİM 10.11: **`StatusBar`** kontrolü örneği.

NotifyIcon



Windows uygulamalarının Windows görev çubuğunda görüntülenen simgesini belirler.

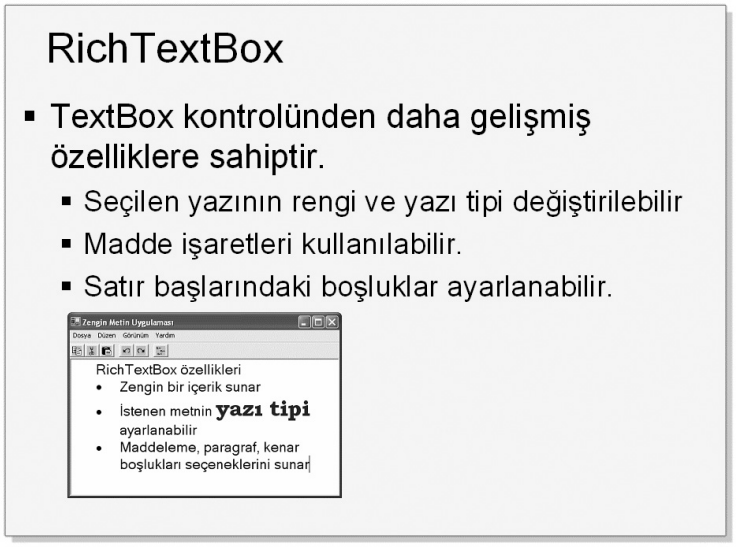
RESİM 10.12: `NotifyIcon` kontrolü örneği.

NotifyIcon Özellikleri

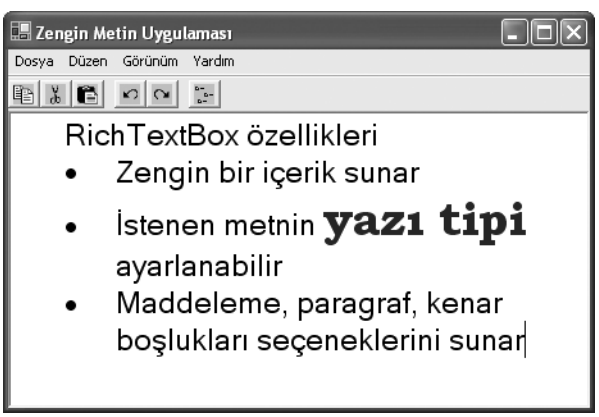
TABLO 10.7: `NotifyIcon` Özellikleri

Özellik	Değer Tipi	Açıklama
`Icon`	`Icon`	Görev çubuğunda gözükecek simgeyi belirler.
`ContextMenu`	`Menu`	Simge sağ tıklandığı zaman açılacak menüyü belirler.
`Text`	`String`	Simge üzerine gelindiğinde görüntülenecek yazıyı belirler.

RichTextBox



Normal bir metin kutusundan daha gelişmiş özelliklere sahip bir kontroldür. **`TextBox`** kontrolünde yazının yazı tipi, büyüklüğü gibi ayarlar yapılabilir. Ancak sadece seçilen yazının rengi, yazı tipi, satır başı genişliği ve madde işaretleri kullanımı gibi ayarlar yapmak mümkün değildir. **`RichTextBox`** kontrolü, bu tip zengin özelliklerin kullanılmasını sağlar (Resim 10.12).



RESİM 10.12: **`RichTextBox`** kontrolü örneği.

RichTextBox Özellikleri

`RichTextBox` kontrolü kullanıcıya birçok seçenek sunar; dolayısıyla tasarım ve çalışma sırasında erişilebilen birçok özelliği bulunur.

Tasarım sırasında ulaşılabilen özellikler:

TABLO 10.8: `RichTextBox` Tasarım Sırasında Ulaşılabilen Özellikler

Özellik	Değer Tipi	Açıklama
`ZoomFactor`	`Single`	Metnin büyüklüğünü belirler. 1 – 64 arası bir değer alır.
`WordWrap`	`Boolean`	Uzun yazıların bir sonraki satıra geçerek görüntülenmesini sağlar.
`DetectUrls`	`Boolean`	Bağlantı olarak girilen yazıların `LinkLabel` şeklinde algılanmasını belirler.
`Lines`	`String()`	Satırları `String` dizisi olarak tutar.
`BulletIntend`	`Integer`	Satırların madde işaretinden kaç piksel açıkta duracağını belirler.
`AcceptsTab`	`Boolean`	**Tab** tuşunun bir karakter olarak algılanmasını, dolayısıyla bu tuşa basıldığında kontrolden çıkılmasının engellenmesini belirler.
`ShowSelectionMargin`	`Boolean`	Satır başındaki boşluğun gösterilmesini belirler.
`RightMargin`	`Integer`	Satırların maksimum uzunluğunu piksel cinsinden belirler.

Çalışma sırasında ulaşılabilen özellikler:

TABLO 10.9: `RichTextBox` Çalışma Sırasında Ulaşılabilen Özellikler

Özellik	Değer Tipi	Açıklama
`Capture`	`Boolean`	Kontrol içine yazı yazarken farenin gizlenmesini belirler.
`UndoActionName`	`String`	En son yapılabilecek Undo işleminin tipini tutar.
`RedoActionName`	`String`	Undo işlemi yapıldıktan sonra, en son yapılabilecek Redo işleminin ismini tutar.
`SelectedText`	`String`	Seçilen metni belirler.
`SelectionBullet`	`Boolean`	Seçilen satırın madde işaretli olarak görüntülenmesini belirler.
`SelectionAlignment`	`Boolean`	Seçilen satırın hizalanmasını belirler.
`SelectionColor`	`Color`	Seçilen metnin rengini belirler.
`SelectionFont`	`Font`	Seçilen metnin yazı tipini belirler.
`SelectionIntend`	`Integer`	Seçilen satırın, sol kenara olan uzaklığını belirler.
`SelectionLength`	`Integer`	Seçilen metnin uzunluğunu belirler.

RichTextBox Metotları

TABLO 10.10: `RichTextBox` Metotları

Metot	Açıklama
`Find`	Metin kutusu içinde, parametre olarak verilen bir yazıyı arar. Yazıyı ilk gördüğü yerin indisini döndürür.
`LoadFile`	Bir dosyadan alınan metni yükler.
`SaveFile`	Parametre olarak verilen konumdaki dosyaya, metni yazar. Dosyanın `*.rtf` veya `*.doc` uzantılarında kaydedilmesi, zengin içeriğin görüntülenmesi açısından önemlidir.
`Undo`	Yapılan işlem geriye alınır.
`Redo`	Geri alınan işlem tekrar yapılır.

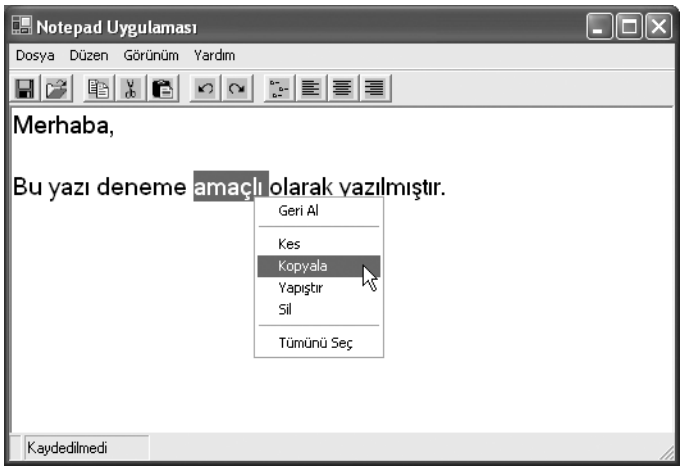
RichTextBox Olayları

TABLO 10.11: `RichTextBox` Olayları

Olay	Açıklama
`TextChanged`	Metin kutusundaki yazı değiştiği zaman gerçekleşir.
`LinkClicked`	Metin içindeki bir bağlantıya tıklandığı zaman gerçekleşir.

Lab 1: Notepad Uygulaması

Bu labda, **RichTextBox** kontrolünün sağladığı kolaylıklarla bir metin editörü uygulaması geliştirilecektir. Bu uygulamanın kullanımını kolaylaştırmak için menüler, araç çubuğu ve durum çubuğundan faydalanılır.



RESİM 10.13: Notepad uygulaması.

Bu labda kullanılan kontroller ve teknikler:

- **MainMenu.** Dosya, düzen, görünüm ve yardım işlemleri için kullanılır.
- **ContextMenu.** Araç çubuğunu gizlemek ve kopyala, yapıştır, kes gibi metin işlemleri için kullanılır.
- **RichTextBox.** Yazılan metnin tutulması için kullanılır.
- **NotifyIcon.** Uygulamanın simgesinin görev çubuğunda gözükmesini sağlar.
- **ToolBar.** Kaydetme, dosya açma, hizalama gibi işlemlere kısayollar sağlamak için kullanılır.
- **ImageList.** Araç çubuğundaki düğmeleri resimlerini belirlemek için kullanılır.
- **SaveFileDialog.** Dosyaların kaydedilmesi sırasında kullanılır.
- **OpenFileDialog.** Dosyaları açmak için kullanılır.
- **FontDialog.** Yazı tipini değiştirmek için kullanılır.
- **StatusBar.** Dosyalar açıldığı zaman isimlerini ve kayıt durumlarını görüntülemek için kullanılır.

Kontrollerin Eklenmesi

Form üzerine Tablo 10.12'deki kontrolleri ekleyin ve belirtilen özellikleri ayarlayın.

TABLO 10.12: Eklenecek Kontroller

Kontrol – Kontrol İsmi	Özellik	Değer
ContextMenu – **ContextMenu1**		Geri Al, Kes, Kopyala, Yapıştır, Sil, Tümünü Seç değerlerini içeren menü öğeleri ekleyin.
ContextMenu – **ContextMenu2**		Gizle değerini içeren bir menü öğesi ekleyin.
ToolBar – **ToolBar1**	**Buttons**	Kaydet, Aç, Kopyala, Kes, Yapıştır, Undo, Redo, Madde İşaretle, Sola Hizala, Sağa Hizala, Ortala komutları için düğmeler ekleyin. Her düğmenin **ImageIndex** özelliğine, **ImageList** içinde bulunan resimlerden uygun olanın indisini atayın.
ImageList – **ImageList1**	**Images**	Araç çubuğundaki öğeleri temsil eden resimler ekleyin.
OpenFileDialog - **OpenFileDialog1**		
SaveFileDialog - **SaveFileDialog1**		
FontDialog - **FontDialog1**		
StatusBar - **StatusBar1**	**ShowPanels**	**True**
	Panels	İki tane panel ekleyin. İlk panelin **AutoSize** özelliğini **Contents** olarak belirleyin.
NotifyIcon - **NotifyIcon1**	**Icon**	Uygulamanız için bir simge seçin
	Text	"**Notepad Uygulaması**"
RichTextBox – **RichTextBox1**	**Dock**	**True**

Uygulamaya son olarak bir **MainMenu** ve ilgili alanlara **MenuItem** öğelerini ekleyin. Parantez içinde belirtilen tuşlar, menü öğelerine erişmek için kullanılacak kısayollardır. Bu değerleri, menü öğelerinin **ShortCut** özelliğine ekleyin.

- Dosya
  - Yeni (**Ctrl+ N**)
  - Aç (**Ctrl+O**)
  - Kaydet (**Ctrl+ S**)
  - Farklı Kaydet
  - Çıkış

- Düzen
  - Geri Al (**Ctrl+ Z**)
  - Kes (**Ctrl+X**)
  - Kopyala (**Ctrl+C**)
  - Yapıştır (**Ctrl+V**)
  - Sil
  - Bul
  - Yazı Tipi
  - Tümünü Seç
- Görünüm
  - Sola Hizala
  - Sağa Hizala
  - Ortala
  - Madde İşaretle
  - Araç çubuğunu gizle
- Yardım
  - Hakkında

Uygulamaya **frmBul** isminde yeni bir form ekleyin. Bu form, metin kutusunda aranan değeri bulmak için kullanılacaktır.

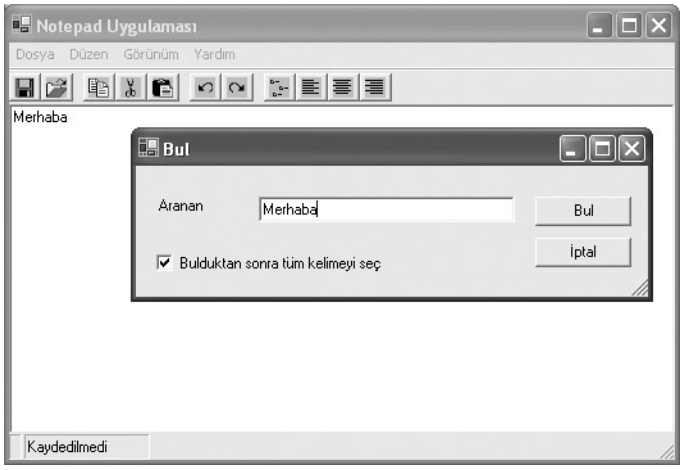
Forma, arama işlemleri için gereken kontrolleri ekleyin.

TABLO 10.13: Eklenecek Kontroller

Kontrol – Kontrol İsmi	Özellik	Değer
Button – Button1	DialogResult	DialogResult.OK
	Text	"Bul"
Button – Button2	Text	"İptal"
CheckBox – cbTumKelimeyiSec	Text	"Bulduktan sonra tüm kelimeyi seç"
	Checked	True
TextBox – txtAranan		

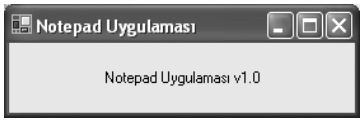
Kontrolleri ekledikten sonra **Button1** ve **Button2** düğmelerinin **Click** olayına, formu kapatan kodları yazın:

```
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click,
Button2.Click
        Me.Close()
    End Sub
```



RESİM 10.14: Notepad Bul işlemi.

Uygulamaya **`frmHakkinda`** isminde yeni bir form ekleyin. Bu form, uygulama bilgilerini içerir.



RESİM 10.15: Notepad Hakkında penceresi.

Kodların Yazılması

- Dosya ismini ve dosyanın kaydedilip edilmediğini tutan değişkenleri tanımlayın.

```
Private Kaydedildi As Boolean = True
Private DosyaIsmi As String
```

Menü öğelerine kod eklemeden önce, yapılacak işlemler yordamlar içine yazılır. Böylece kodun karmaşıklığı azalır ve değişiklik yapmak kolaylaşır.

- Durum çubuğunda değişiklik yapma işlemlerini yazın. Durum çubuğu, dosya açma kaydetme gibi işlemler sonunda değişecektir.

```
Sub DurumDegistir()
    ' İlk panele dosya ismi yazılır
    StatusBar1.Panels(0).Text = DosyaIsmi

    ' İkinci panele kayıt durumu yazılır
    If Kaydedildi Then
        StatusBar1.Panels(1).Text = "Kaydedildi"
    Else
```

```
            StatusBar1.Panels(1).Text = "Kaydedilmedi"
        End If
    End Sub
```

- Dosyaya kaydetme ve farklı kaydetme işlemlerini yazın.

```
    ' Kaydetme işlemi
    Sub Kaydet()
        If DosyaIsmi = "" Then
            FarkliKaydet()
        Else
            RichTextBox1.SaveFile(DosyaIsmi)
            Kaydedildi = True
        End If
        DurumDegistir()
    End Sub

    ' Farklı kaydetme işlemi
    Sub FarkliKaydet()
        Dim dosya As String
        ' Kaydedilecek yeri seçmek için
        ' SaveFileDialog kutusu gösterilir

        ' Dosya yoksa otomatik olarak oluşturulması sağlanır
        SaveFileDialog1.CreatePrompt = True

        If SaveFileDialog1.ShowDialog = DialogResult.OK Then
            dosya = SaveFileDialog1.FileName()
            RichTextBox1.SaveFile(dosya)
            DosyaIsmi = dosya
            Kaydedildi = True
        End If
        DurumDegistir()
    End Sub
```

- Yeni bir dosya veya var olan bir dosyayı açma işlemlerini tanımlayın.

```
  Sub DosyaAc(ByVal yeniDosya As Boolean)
        If Not Kaydedildi Then
            Select Case MsgBox("Dosya kaydedilsin mi?",
MsgBoxStyle.YesNoCancel)
                Case MsgBoxResult.OK
                    ' Kaydetme işlemi yapılır
                    Kaydet()

                Case MsgBoxResult.Cancel
```

```
                    ' İşlem iptal edildi
                    Exit Sub
                End Select
            End If

            If Not yeniDosya Then
                ' Varolan bir dosya açılır.
                Dim dosya As String
                If OpenFileDialog1.ShowDialog = DialogResult.OK
Then
                    dosya = OpenFileDialog1.FileName
                    RichTextBox1.LoadFile(dosya)
                    DosyaIsmi = dosya
                End If

            Else
                ' Yeni bir dosya açılır
                RichTextBox1.Clear()
                DosyaIsmi = ""
            End If

            Kaydedildi = True
            DurumDegistir()
        End Sub
```

- Bulma işlemlerini gerçekleştiren kodları yazın. Burada yeni bir form açılıp, orda girilen değerlere göre arama işlemi yapılır.

```
    Sub Bul()
        ' Bulma formu görüntülenir, iptal tuşuna basıldıysa
çıkılır
        Dim bul As New frmBul
        If Not bul.ShowDialog = DialogResult.OK Then Exit
Sub

        Dim aranan As String = bul.txtAranan.Text
        If aranan = "" Then Exit Sub

        ' Bulduktan sonra kelimenin tümünü işaretlenmesi
bilgisi alınır
        Dim TumKelimeyiSec As Boolean =
bul.cbTumKelimeyiSec.Checked

        ' Bulunan ilk indis alınır.
        Dim start As Integer = RichTextBox1.Find(aranan)
```

```
        If Not TumKelimeyiSec Then
            ' Sadece aranan kelime seçilir.
            RichTextBox1.Select(start, aranan.Length)
        Else
            Dim son As Integer = start
            Dim bas As Integer = start

            While son < RichTextBox1.Text.Length - 1 AndAlso
RichTextBox1.Text.Substring(son, 1) <> " "
                son += 1
            End While

            While bas > -1 AndAlso
RichTextBox1.Text.Substring(bas, 1) <> " "
                bas -= 1
            End While

            RichTextBox1.Select(bas + 1, son - bas - 1)
        End If
    End Sub
```

- **ToolBar** düğmeleri tıklandığı zaman gerçekleşecek kodları yazın.

DİKKAT **Bu kodda belirtilen indis numaraları, uygulamanızda** ToolBar **kontrolüne eklediğiniz düğmelerin indis numaraları ile farklılık gösterebilir. Yapılan işlemler yorum satırı olarak geçilmiştir. Bu işlemleri, düğmelerin indislerine göre tekrar düzenleyin. Düğmelerin indislerini öğrenmek için** ToolBar **kontrolünün** Buttons **özelliğine bakın.**

```
Private Sub ToolBar1_ButtonClick(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.ToolBarButtonClickEventArgs) Handles
ToolBar1.ButtonClick
        ' Basılan düğmenin indisine göre işlem yapılır.
        Select Case ToolBar1.Buttons.IndexOf(e.Button)
            Case 0
                'Kaydet
                Kaydet()
            Case 1
                ' Ac
                DosyaAc(False)
            Case 3
                ' Kopyala
                RichTextBox1.Copy()
            Case 4
                ' Kes
                RichTextBox1.Cut()
            Case 5
```

```
                ' Yapıştır
                RichTextBox1.Paste()
            Case 7
                ' Geri Al
                RichTextBox1.Undo()
            Case 8
                ' Tekrarla
                RichTextBox1.Redo()
            Case 10
                ' Madde işaretle
                RichTextBox1.SelectionBullet = Not
RichTextBox1.SelectionBullet
            Case 11
                ' Sola Hizala
                RichTextBox1.SelectionAlignment =
HorizontalAlignment.Left
            Case 12
                ' Ortala
                RichTextBox1.SelectionAlignment =
HorizontalAlignment.Center
            Case 13
                ' Sağa Hizala
                RichTextBox1.SelectionAlignment =
HorizontalAlignment.Right
        End Select
    End Sub
```

- Dosya içinde bulunan bir bağlantı tıklandığı zaman, bu bağlantıyı ilgili tarayıcıda açan kodları yazın.

```
    ' Linke git
    Private Sub RichTextBox1_LinkClicked(ByVal sender As
Object, ByVal e As
System.Windows.Forms.LinkClickedEventArgs) Handles
RichTextBox1.LinkClicked
        System.Diagnostics.Process.Start(e.LinkText)
    End Sub
```

- Dosya içine yazılan yazı değiştiği zaman gereken kodları yazın.

```
    Private Sub RichTextBox1_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RichTextBox1.TextChanged
        Kaydedildi = False
        DurumDegistir()
    End Sub
```

- Uygulama kapanırken dosyanın kaydedilip kaydedilmeyeceğini soran kodları yazın.

```
    ' Kapanırken dosyanın kaydedilmesi kontrol edilir.
    Private Sub Form3_Closing(ByVal sender As Object, ByVal
e As System.ComponentModel.CancelEventArgs) Handles
MyBase.Closing
        If Not Kaydedildi Then
            Select Case MsgBox("Dosya kaydedilsin mi?",
MsgBoxStyle.YesNoCancel)
                Case MsgBoxResult.OK
                    ' Kaydetme işlemi yapılır
                    Kaydet()

                Case MsgBoxResult.Cancel
                    ' İşlem iptal edildi
                    e.Cancel = True
            End Select
        End If
    End Sub
```

- Her menü öğesinin altına, ilgili işlemleri yazın. Burada dikkat edilmesi gereken nokta, bazı **`ContextMenu`** öğelerinin ve **`MainMenu`** öğelerinin aynı işlemi yaptığıdır. Örneğin "Geri Al" komutu, her iki menüde de vardır. Bu kodları farklı yordamlar yerine, aynı yordamın içine yazarak **`Handles`** ifadesine iki menü öğesinin **`Click`** olayı yazılır.

Örnek:

```
    Private Sub MenuItem19_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem19.Click, MenuItem5.Click
        RichTextBox1.Undo()
    End Sub
```

DİKKAT **Bu kodda belirtilen menü isimleri, uygulamanızda** `MainMenu` **veya** `ContextMenu` **kontrolüne eklediğiniz menülerin isimleri ile farklılık gösterebilir. Yapılan işlemler yorum satırı olarak geçilmiştir. İlgili menü öğesini çift tıklayarak** `Click` **olayında, burada belirtilen işlemleri yazın.**

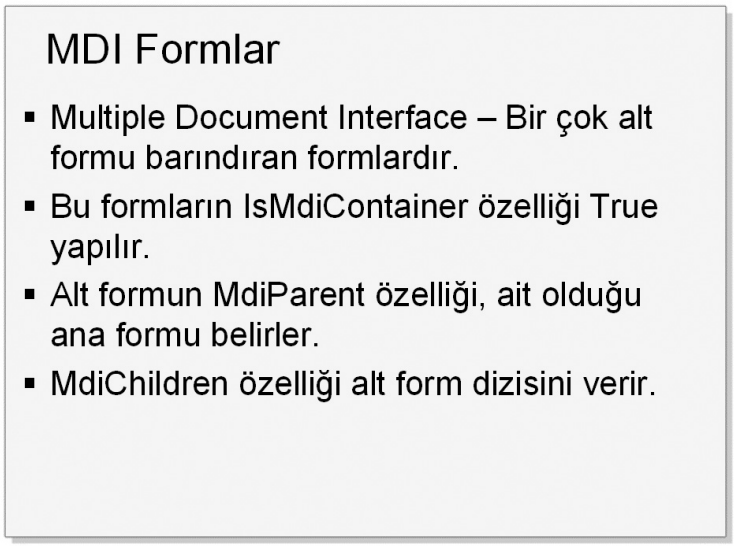
```
    ' Yeni Dosya aç
    Private Sub MenuItem13_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem13.Click
        DosyaAc(True)
    End Sub

    ' Dosya Aç
```

```
    Private Sub MenuItem14_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem14.Click
        DosyaAc(False)
    End Sub

    ' Kaydet
    Private Sub MenuItem15_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem15.Click
        Kaydet()
    End Sub

    ' Farklı Kaydet
    Private Sub MenuItem16_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem16.Click
        FarkliKaydet()
    End Sub

    ' Çık
    Private Sub MenuItem18_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem18.Click
        Application.Exit()
    End Sub

    ' Geri al
    Private Sub MenuItem19_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem19.Click, MenuItem5.Click
        RichTextBox1.Undo()
    End Sub

    ' Kes
    Private Sub MenuItem21_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem21.Click, MenuItem7.Click
        RichTextBox1.Cut()
    End Sub

    ' Kopyala
    Private Sub MenuItem22_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem22.Click, MenuItem8.Click
        RichTextBox1.Copy()
    End Sub
```

```
    ' Yapıştır
    Private Sub MenuItem23_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem23.Click, MenuItem9.Click
        RichTextBox1.Paste()
    End Sub

    ' Yazı sil
    Private Sub MenuItem24_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem24.Click, MenuItem10.Click
        ' silinecek kelime RichTextBox kontrolünde seçilen
kelimedir
        Dim silinecek As String = RichTextBox1.SelectedText

        ' secilen kelimenin indisi bulunur
        Dim i As Integer = RichTextBox1.SelectionStart

        RichTextBox1.Text = RichTextBox1.Text.Remove(i,
silinecek.Length)
    End Sub

    ' Tüm yazıyı seç
    Private Sub MenuItem28_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem28.Click, MenuItem12.Click
        RichTextBox1.SelectAll()
    End Sub

    ' Yazı tipini seç
    Private Sub MenuItem36_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem36.Click
        ' Font seçerken, renklerin de görünmesi sağlanır.
        FontDialog1.ShowColor = True

        If FontDialog1.ShowDialog = DialogResult.OK Then
            RichTextBox1.SelectionFont = FontDialog1.Font
        End If
    End Sub

    ' Sola Hizala
    Private Sub MenuItem29_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem29.Click
        RichTextBox1.SelectionAlignment =
HorizontalAlignment.Left
    End Sub
```

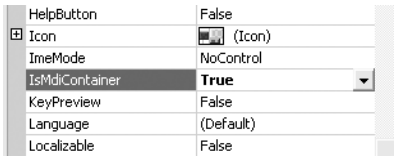
```
    ' Sağa Hizala
    Private Sub MenuItem30_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem30.Click
        RichTextBox1.SelectionAlignment =
HorizontalAlignment.Right
    End Sub

    ' Ortala
    Private Sub MenuItem31_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem31.Click
        RichTextBox1.SelectionAlignment =
HorizontalAlignment.Center
    End Sub


    ' Madde işaretle
    Private Sub MenuItem33_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem33.Click
        RichTextBox1.SelectionBullet = Not
RichTextBox1.SelectionBullet
    End Sub


    ' Hakkında formunun gösterilmesi
    Private Sub MenuItem34_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem34.Click
        Dim hakkinda As New frmHakkinda
        hakkinda.ShowDialog()
    End Sub


    ' Araç çubuğunun gizlenmesi, MainMenu ve Toolbar
kontrolüne
    ' atanan ContextMenu yapılır.
    Private Sub MenuItem37_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem37.Click, MenuItem38.Click
        ToolBar1.Visible = MenuItem37.Checked
        MenuItem37.Checked = Not MenuItem37.Checked
    End Sub


    ' Dosya bulunması
    Private Sub MenuItem26_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem26.Click
        Bul()
    End Sub
End Class
```

MDI Formlar



Multiple Document Interface formları, içinde birden fazla form barındıran formlardır. **`MDIChild`** olarak eklenen bu formlar birbirinden tamamen bağımsızdır. Örneğin bir Excel dosyası içinde birden fazla sayfa olabilir. Bu sayfalar ana forma bağlıdır. Ana form kapandığı zaman bu sayfalar da kapanır. **`MDIParent`** olarak nitelendirilen bu ana formların, **`MDIChild`** formlarını açmak ve yönetmek için menülere ihtiyaçları vardır.

Formları MDI olarak tanımlamak için **`IsMdiContainer`** özelliğinin **`True`** olarak ayarlanması gerekir (Resim 10.16).



RESİM 10.16: **`isMDIcontainer`** özelliği.

MDI formlara alt formlar eklemek için, form oluşturma işlemleri bilinen şekilde yapılır. Ancak formun **`MDIParent`** özelliği belirlenmelidir.

```
Dim f As New AltForm

' Oluşturulan form, ana forma bağlanır.
f.MdiParent = Me
f.Show()
```

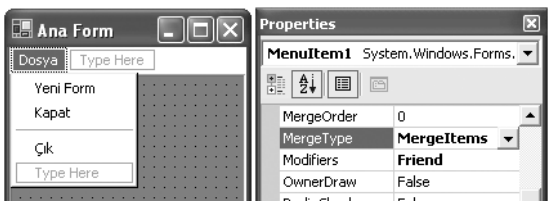
Bir formun sahip olduğu alt formlara ulaşmak için, **MDIChildren** özelliğinden yararlanılır. Bu özellik tek boyutlu bir form dizisidir.

```
' Tüm formları kapatır.
' Alt formlar kapandığı zaman, dizi otomatik olarak
' yeniden boyutlandırılır.
While Me.MdiChildren.Length > 0
    Me.MdiChildren(0).Close()
End While

' Tüm formları Minimize eder
For i As Integer = 0 To Me.MdiChildren.Length - 1
    Me.MdiChildren(i).WindowState =
FormWindowState.Minimized
Next
```

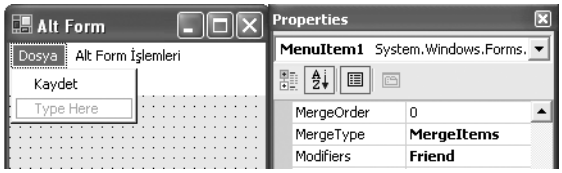
RESİM 10.17: MDI formlar.

Alt formlar genişletildiklerinde, form üzerinde yazan başlık ana forma taşınır. Alt formda tanımlı bir menü, ana formun menüsü ile birleşir. Bu menü birleşim işlemine **Merge** denir. Menü öğeleri, varsayılan durumda ana formdaki menülerin yanına eklenir. Ancak menü öğelerinin **MergeType** özelliği ile varsayılan değer değiştirilebilir.

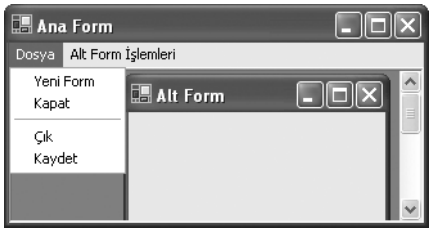
- **MergeType.Add.** Varsayılan değerdir. Bu değeri alan menü öğeleri, birleşme sonucunda menüye eklenir.
- **MergeType.MergeItem.** Bu değeri alan menüler, sonuç menüsünde aynı **MergeOrder** değerindeki menülerle birleşir.
- **MergeType.Replace**. Birleşme sonucunda bu menü, aynı **MergeOrder** değerinde olan öğe ile değiştirilir.
- **MergeType.Remove.** Birleşme sonucunda bu menü çıkartılır.



RESİM 10.18: Ana formdaki menü.



RESİM 10.19: Alt formdaki menü.



RESİM 10.20: Birleştirilmiş menü.

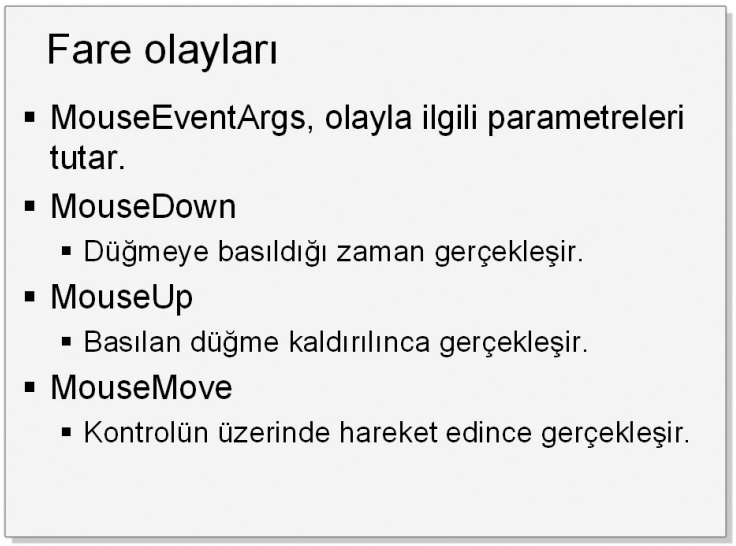
Alt formlar, ana forma basamak şeklinde eklenir. Birçok alt form ile çalışılıyorsa, bu formların düzenlenmesine ihtiyaç duyulur. Alt formları düzenlemek için formun **`LayoutMdi`** metodu kullanılır.

```
Me.LayoutMdi(MdiLayout.TileHorizontal)
Me.LayoutMdi(MdiLayout.TileVertical)
Me.LayoutMdi(MdiLayout.Cascade)
Me.LayoutMdi(MdiLayout.ArrangeIcons)
```

MDI Form içindeki alt formlardan seçili olana ulaşmak için, formun **`ActiveMdiChild`** özelliği kullanılır.

```
If Not Me.ActiveMdiChild Is Nothing Then
    Me.Text = Me.ActiveMdiChild.Text
End If
```

Fare Olayları



Fare olayları, formlar üzerinde farenin bir tuşunun tıklanması ya da bir kontrolün üzerine gelinmesi gibi olaylardır. Bu olayla ilgili parametreler, olay gerçekleştiği zaman **`MouseEventArgs`** nesnesi ile kullanıcıya bildirilir.

`MouseEventArgs` özellikleri:

- **`Button`.** Hangi fare düğmesinin tıklandığını gösterir.
- **`Click`.** Olay gerçekleşene kadar, düğmeye kaç defa tıklandığını belirler. Örneğin fareye çift tıklanmışsa 2 değerini alacaktır.
- **`Delta`.** Farenin ortadaki düğmesinin dönme oranını gösterir.
- **`X`.** Kontrole göre, farenin tıklandığı pozisyonun x koordinatını gösterir.
- **`Y`.** Kontrole göre, farenin tıklandığı pozisyonun y koordinatını gösterir.

NOT **Fare olayları MDI formlar üzerinde gerçekleşmez.**

MouseDown olayı

Farenin herhangi bir düğmesine basıldığı zaman gerçekleşir. Kontrolün **`Click`** olayından önce çalışır.

MouseUp olayı

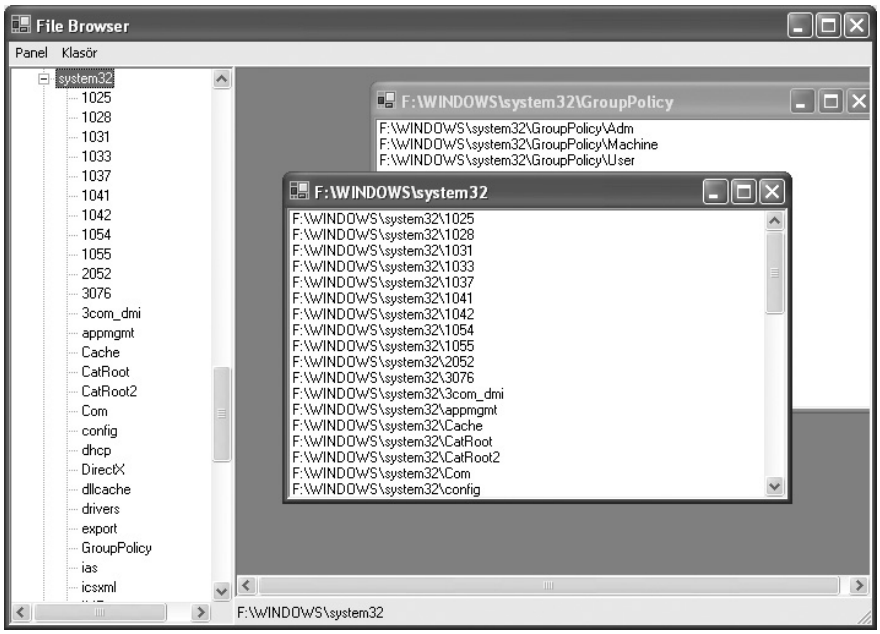
Farenin basılan düğmesi kaldırıldığı zaman gerçekleşir.

MouseMove olayı

Farenin, kontrol üzerinde hareket etmesi ile gerçekleşir.

Lab 2: File Browser

Bu labda, verilen bir konumdaki klasörlerin listelenmesi ve seçilen klasörün bilgilerinin alt formlarda görünmesi uygulaması gerçekleştirilecektir.



RESİM 10.21: File Browser uygulaması.

Bu labda kullanılan kontroller ve teknikler:

- **`MainMenu`**. Klasörlerin görüntüleneceği konumu belirlemek, yeni klasör eklemek ve klasör silmek gibi işlemler için kullanılır.
- **`ContextMenu`.** Seçilen klasörün alt klasörlerini listelemek ve klasörü listeden kaldırmak için kullanılır.
- **`TreeView`.** Belirtilen konumdaki klasörleri ve alt klasörleri listelemeyi sağlar.
- **`StatusBar`.** Seçilen dosyaların konumlarını görüntülemeyi sağlar.
- **`ListBox`.** Alt klasörlerin listelenmesi için kullanılır.

Kontrollerin Eklenmesi

Form üzerine Tablo 10.14'teki kontrolleri ekleyin ve belirtilen özellikleri ayarlayın.

TABLO 10.14: Eklenecek Kontroller

Kontrol – Kontrol İsmi	Özellik	Değer
`Form`	`isMDIContainer`	`True`
`ContextMenu` – `ContextMenu1`		Alt Klasörler ve Kaldır değerlerini içeren menü öğelerini ekleyin.

TABLO 10.14: Eklenecek Kontroller

Kontrol – Kontrol İsmi	Özellik	Değer
MainMenu – **MainMenu1**		Yeni Konum ve Dosya Bilgileri değerlerini içeren menü öğelerini ekleyin.
StatusBar – **StatusBar1**		
TreeView – **TreeView1**		

Uygulamaya **DosyaBilgileri** isminde yeni bir form ekleyin. Form içine Tablo 10.15'teki kontrolleri ekleyin ve özelliklerini ayarlayın.

TABLO 10.15: Eklenecek Kontroller

Kontrol – Kontrol İsmi	Özellik	Değer
MainMenu – **MainMenu1**		Yeni, Sil ve Kapat değerlerini içeren menü öğelerini ekleyin.
ListBox – **ListBox1**	**Dock**	**Fill**

Kodların Yazılması

Ana Form

- Belirtilen konumdaki klasörleri listeleyen kodları yazın.

```
    Function KlasorleriAl(ByVal konum As String) As String()
        Dim klasorler() As String =
System.IO.Directory.GetDirectories(konum & "\")

        For i As Integer = 0 To klasorler.Length - 1
            klasorler(i) = klasorler(i).Remove(0,
konum.Length + 1)
        Next

        Return klasorler
    End Function
```

- Form üzerinde görüntülenecek klasörlerin bulunduğu yeri tutan değişkeni ve yeni formun açılmasını yazın.

```
    Private YeniKonum As String

    Sub FormBilgileri()
        Dim f As New DosyaBilgileri
        f.MdiParent = Me
        f.Text = YeniKonum & TreeView1.SelectedNode.FullPath
```

```
        f.KlasorleriListele()
        f.Show()
    End Sub
```

- Yeni konumu seçen menü altına, **TreeView** kontrolünde alt klasörleri listeleyen kodları yazın.

```
    ' Yeni konumun seçilmesi
    Private Sub MenuItem2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem2.Click
        YeniKonum = InputBox("Konum girin:", "Yeni Konum",
"C:\")

        Dim klasorler() As String = KlasorleriAl(YeniKonum)

        For i As Integer = 0 To klasorler.Length - 1
            TreeView1.Nodes.Add(klasorler(i))
        Next
        TreeView1.SelectedNode = TreeView1.Nodes(0)
    End Sub
```

- **TreeView** kontrolünde bir klasör seçildiği zaman durum çubuğunda klasörün ismini görüntüleyen kodları yazın.

```
Private Sub TreeView1_AfterSelect(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles
TreeView1.AfterSelect

    StatusBar1.Text = YeniKonum &
TreeView1.SelectedNode.FullPath

End Sub
```

- **ContextMenu** içinde tanımlanan işlemleri yazın.
  - Alt klasörlerin listelenmesi.

```
    ' Alt klasörler
    Private Sub MenuItem3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem3.Click
        Dim secilen As TreeNode = TreeView1.SelectedNode
        secilen.Nodes.Clear()

        Dim konum As String = YeniKonum & secilen.FullPath
        Dim altKlasorler() As String = KlasorleriAl(konum)
```

```
        For i As Integer = 0 To altKlasorler.Length - 1
            secilen.Nodes.Add(altKlasorler(i))
        Next
    End Sub
```

- Klasörün kaldırılma işlemi.

```
    ' Seçilen klasörün listeden kaldırılma işlemi
    Private Sub MenuItem4_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem4.Click
        Dim secilen As TreeNode = TreeView1.SelectedNode
        If secilen Is Nothing Then Exit Sub

        If secilen.Parent Is Nothing Then
            TreeView1.Nodes.Remove(secilen)
        Else
            secilen.Parent.Nodes.Remove(secilen)
        End If
    End Sub
```

- Dosya bilgilerini görüntüleyen kodları yazın.

```
    ' Dosya bilgileri – MainMenu öğesine tıklandığında
    Private Sub MenuItem5_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem5.Click
        FormBilgileri()
    End Sub

    ' Dosya bilgileri – TreeView öğesine çift tıklandığında
    Private Sub TreeView1_MouseDown(ByVal sender As Object,
ByVal e As System.Windows.Forms.MouseEventArgs) Handles
TreeView1.MouseDown
        If e.Clicks = 2 Then
            FormBilgileri()
        End If
    End Sub
```

- Farenin ortadaki tekerleğinin döndürülmesi işleminde, **`TreeView`** içinde seçilen öğeden bir önceki veya bir sonraki öğeye gidilmesi için gereken kodları yazın.

```
    Private Sub TreeView1_MouseWheel(ByVal sender As Object,
ByVal e As System.Windows.Forms.MouseEventArgs) Handles
TreeView1.MouseWheel
        If TreeView1.SelectedNode Is Nothing Then Exit Sub
```

```
        If e.Delta < 0 Then
            Dim sonraki As TreeNode =
TreeView1.SelectedNode.NextNode
            If Not sonraki Is Nothing Then
                TreeView1.SelectedNode = sonraki
            End If
        Else
            Dim onceki As TreeNode =
TreeView1.SelectedNode.PrevNode
            If Not onceki Is Nothing Then
                TreeView1.SelectedNode = onceki
            End If
        End If

    End Sub
```

DosyaBilgileri formunda yazılacak kodlar:

- Alt klasörlerin listelendiği kodları yazın.

```
    Sub KlasorleriListele()
        ListBox1.Items.Clear()
        Dim klasorler() As String =
System.IO.Directory.GetDirectories(Me.Text & "\")

        For i As Integer = 0 To klasorler.Length - 1
            ListBox1.Items.Add(klasorler(i))
        Next
    End Sub
```

- Yeni klasörün eklenmesi için gereken kodları yazın.

```
    Private Sub MenuItem3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem3.Click
        Dim yeniKlasor As String = InputBox("Yeni klasör
ismi girin:")
        yeniKlasor = yeniKlasor.Insert(0, Me.Text & "\")
        System.IO.Directory.CreateDirectory(yeniKlasor)

        KlasorleriListele()
    End Sub
```

- Seçilen klasörün silinmesini sağlayan kodları yazın.

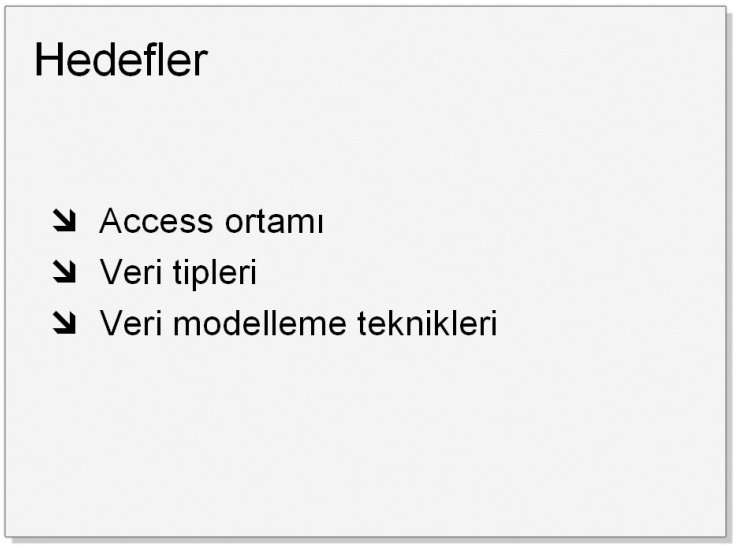
```
    Private Sub MenuItem2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem2.Click
        Dim silinecek As String
```

```
        silinecek = ListBox1.SelectedItem

        System.IO.Directory.Delete(silinecek, True)
        KlasorleriListele()
    End Sub
```

Modül Sonu Soruları & Alıştırmalar

1. **`MainMenu`** ve **`ContextMenu`** nesnelerini ve kullanım alanlarını açıklayın. Kontrolleri içeren bir uygulama geliştirin.
2. **`ImageList`** kontrolünün kullanım amacını ve nasıl kullanıldığını açıklayın. Kontrolü içeren bir uygulama geliştirin.
3. **`SDI`** ve **`MDI`** form yapılarını açıklayınız ve her iki tür için birer örnek uygulama geliştirin.

Modül 11: Veri Yapıları



Birçok şirket, kurum ve kayıtlarını tutan yapılar için verilerin önemi çok büyüktür. Verilerin kağıt üzerinde tutulması, hem aramaların yapılması, hem de kayıt düzeni açısından çok zor bir yöntemdir. Bilgisayarların iş yaşamında kullanılmaya başlanması ile verilen yönetimi daha da kolaylaştı. Ancak bu teknoloji ilerledikçe kullanılması zorlaşmaya başladı. Verilerin tutulması metin dosyalarından tablolara aktarıldı. Günümüzde veri ve tablo yapılarının yönetimi artık veritabanı yöneticilerin eline bırakılmış durumdadır.

Windows ve Web uygulamaların çoğu veri üzerine yoğunlaşır. Uygulamalarda veriye hızlı bir şekilde ulaşmak ve veriyi yönetmek için tablo yapılarının iyi bir şekilde modellenmesi gerekir. Bu modülde Microsoft Access veritabanı üzerinde veri yapılarının kullanılması işlenecektir.

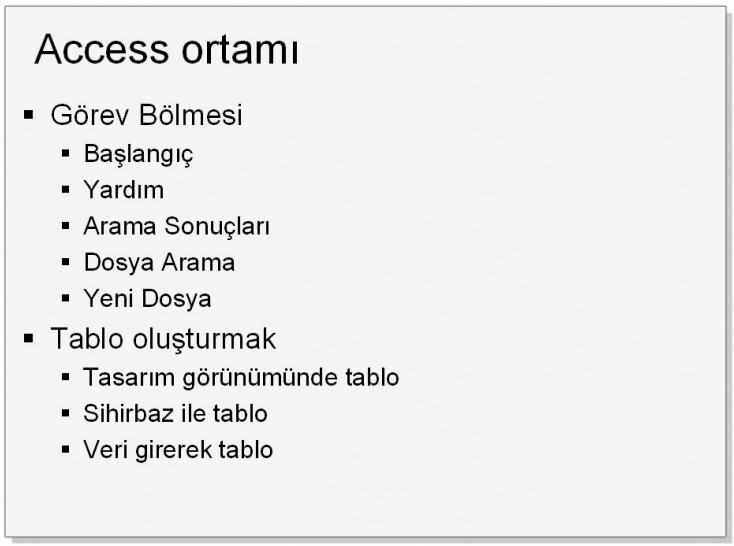
Bu modülü tamamladıktan sonra;

- Microsoft Access ortamını tanıyacak,
- Veritabanlarında kullanılan değişik veri tiplerini tanıyacak,
- Veri modelleme tekniklerini öğreneceksiniz.

Konu 1: Access' e Giriş

Access Microsoft'un ilişkisel veritabanı uygulamasıdır. İçindeki birçok sihirbaz yardımı ile kullanım kolaylığı ve tablo tasarımlarının hızlı bir şekilde yapılabilmesini sağlar. Access tasarım görünümlerinde, tabloların yapısını analiz etmek için kullanılan sorguların kolay bir şekilde oluşturması sağlanır. Karmaşık bir dosya yapısı olmaması, veritabanının taşınabilirliğini kolaylaştırır ve her platformda çalışabilmesini sağlar.

Access Ortamı



Access ortamı, veritabanı geliştirirken kullanıcıya birçok kolaylık sunar. Access açıldığı zaman sağ panelde "Görev Bölmesi" çıkar (Resim 11.1). Bu panel birçok işleme kısayol sağlar.

- **Başlangıç.** Access Office Online başlangıç sayfasıdır. Microsoft haber sitelerine bağlantıları ve en son açılan veritabanlarını listeler.
- **Yardım.** Online yardım seçeneklerini sunar.
- **Arama sonuçları.** Online yardımda bulunan sonuçları listeler.
- **Dosya arama.** Belirtilen yerde, belli tipte dosyaları aramayı sağlar.
- **Yeni dosya.** Yeni bir veritabanı dosyası veya veri erişim dosyası açmak için kullanılır.

RESİM 11.1: Görev Bölmesi.

Boş veritabanı açma komutu verildiği zaman, "Yeni Veritabanı Dosyası" iletişim kutusunda dosya ismi girilip yeni veritabanı oluşturulur. Oluşturulan veritabanı dosyalarının uzantısı **`*.mdb`** olur.

Daha önceden oluşturulmuş bir veritabanını açmak için Dosya menüsünden Aç komutu verilir. **Ctrl+O** kısayolu da dosyaları açmak için kullanılabilir.

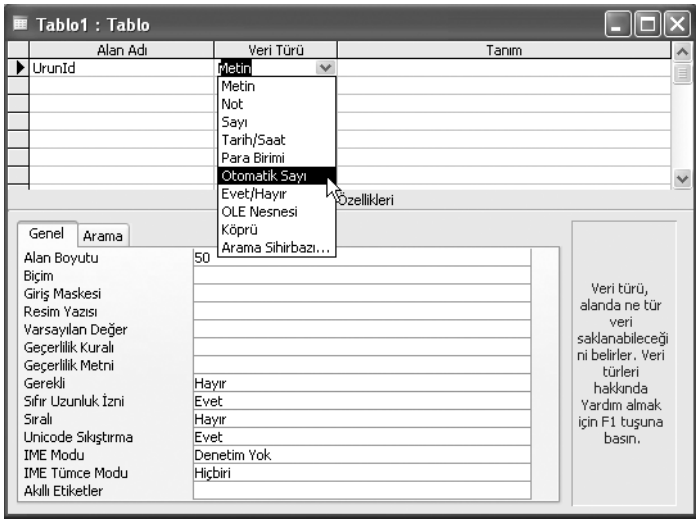
Veritabanı açıldığı zaman, veritabanı üzerinde yapılabilecek tüm işlemleri sunan bir pencere çıkar (Resim 11.2). Veritabanı nesnelerini yönetilmesi bu pencere ile yapılır. Sol panelde bulunan Nesneler sekmesinde, veritabanında bulunabilecek tüm nesneler listelenir. Bir nesne tipi seçildiğinde, veritabanında bulunan bu tipteki tüm öğeler görüntülenir. Örneğin, Tablolar sekmesine gelindiğinde veritabanı üzerindeki tablolar görüntülenir, yeni tablo oluşturmak için seçenekler sunulur.

RESİM 11.2: Veritabanı penceresi.

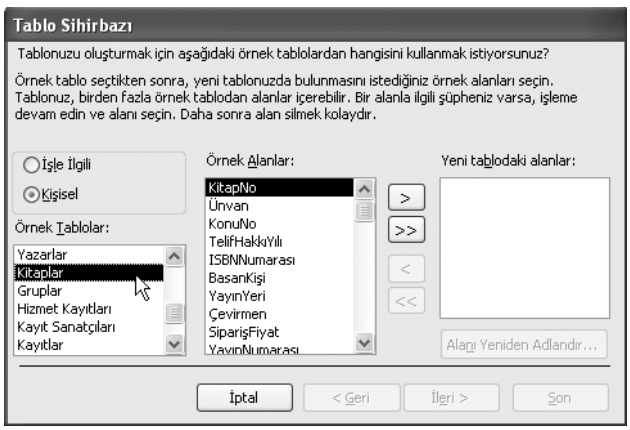
Veritabanı Nesnesi Oluşturmak

Veritabanı penceresinde nesneleri oluşturmak için farklı kısayollar bulunur. Tabloları oluşturmak için bu kısayollardan yararlanılabilir.

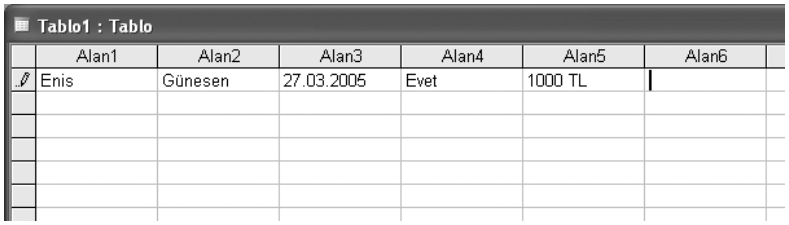
- **Tasarım görünümünde tablo oluştur.** Bu seçenek ile tablodaki verilerin tasarımı tamamen kullanıcıya bırakılmıştır. Kullanıcı alan adlarını kendisi girip, ilgili veri tipini ve gerekli ayarları seçebilir (Resim 11.3).
- **Sihirbaz kullanarak tablo oluştur.** Access içinde çok sık karşılaşılan, kullanıcıya büyük hız sağlayan sihirbaz yardımı ile tablo oluşturulur. Sihirbaz, hangi tipte tablo oluşturulacağını, önceden hazırlanmış zengin şablonları kullanıcıya sunarak belirler (Resim 11.4).
- **Veriler girerek tablo oluştur.** Bu seçenek ile tablolar veri girişi ile oluşturulur (Resim 11.5). Access kullanıcının girdiği verilere göre alan sayısını ve tiplerini belirler. Ancak alan adları daha sonradan değiştirilmelidir (Resim 11.6).



RESİM 11.3: Tasarım görünümünde tablo oluşturmak.



RESİM 11.4: Sihirbaz ile tablo oluşturmak.



RESİM 11.5: Veri girerek tablo oluşturmak.

Oluşturulan tablolar tasarım ve veri sayfası görünümlerinde incelenebilir. Veri sayfası görünümü, veri girişinde kullanıcıya büyük kolaylık sağlar.

Örneğin, Evet/Hayır veri tipindeki bir alana veri girilmesi için bir **CheckBox** görüntülenir. Ayrıca tablonun ilişkide olduğu tablolar bulunur ve alt tablo olarak kullanıcıya sunulur (Resim 11.7).

RESİM 11.6: Alan adlarının değiştirilmesi.

RESİM 11.7: Veri sayfasında ilişkili tabloların görüntülenmesi.

Tablolar oluşturulduktan sonra, aralarındaki ilişkilerin kurulması ve görüntülenmesi için araç çubuğundaki "İlişkiler" düğmesi kullanılır (Resim 11.8).

RESİM 11.8: İlişkiler düğmesi.

Konu 2: Veri Yapılarına Giriş

Veritabanlarında tüm veriler aynı tipte tutulmaz. Bu durum, küçük veriler için fazla yer alanları açmayı engellediği gibi, değişik formatlardaki verilerin yönetilebilirliğini de artırır. Örneğin kategori tablosunda tutulan verilerin sayısı genellikle azdır ve çok fazla artmaz. Dolayısıyla bu verilerin tekil alanında tutulan sayının çok büyük veri tipinde olması gerekmez. Ancak makalelerin tutulduğu bir alanın kapasitesinin çok büyük olması gerekir.

Metin Veri Tipleri

- **Metin (`Text`).** Metin bilgilerini tutmak için tanımlanan veri tipidir. Bu değere girilebilecek maksimum karakter sayısı 255'tir. Bir alana belirtilen uzunluktan küçük bir değer girildiğinde, kalan boş yerler için kaynak ayrılmaz. Metin veri tipi sayısal değerler de içerebilir.
- **Not (`Memo`).** Maksimum 65535 karakter tutar. Büyük metinsel veriler için tercih edilmelidir.

Sayısal Veri Tipleri

Sayı veri tipinin birden fazla alan büyüklüğü vardır.

- **Bayt (`Byte`).** 0 ile 255 arasında bir sayı.
- **Tamsayı (`Integer`).** –32,768 ile 32,767 arasında bir sayı.
- **Uzun Tamsayı (`Long Integer`).** –2,147,483,648 ile 2,147,483,647 arasında bir sayı.
- **Tek (`Single`).** Negatif sayı aralığı: –3.402823E+38 ile –1.401298E–45 Pozitif sayı aralığı: 1.401298E–45 ile 3.402823E38.
- **Çift (`Double`).** Negatif sayı aralığı: 1.79769313486231E+308 ile –4.94065645841247E–324. Pozitif sayı aralığı: 1.94065645841247E–324 ile 1.79769313486231E+308.
- **Ondalık (`Decimal`).** –10^38–1 ile 10^38–1 arasında sayı.
- **Otomatik Sayı (`AutoNumber`)** veri tipi, alana veri girildiği zaman otomatik olarak belirlenen sayıları ifade eder. Sayılar rasgele ya da birden başlayarak girilir.

Tarih Veri Tipi

Tarih alanları için değişik büyüklüklerde depolama seçenekleri sunar.

- **Genel Tarih.** Kısa Tarih ve Uzun Saat birleşimi bir görünümdür.
- **Uzun Tarih.** `12 Aralık 2004 Pazar` formatında görünür.
- **Orta Uzunlukta Tarih.** `12 Ara 2004` formatında görünür.
- **Kısa Tarih.** `12.12.2004` formatında görünür.
- **Uzun Saat.** `15:11:19` formatında görünür.
- **Kısa Saat.** `15:11` formatında görünür

Evet/Hayır Veri Tipi

Bir bit değerinde, Evet ve Hayır değerlerini alan veri tipidir. Veri sayfalarında veya sorgu sonucunda bir `CheckBox` ile ifade edilir. Eğer seçili ise bu alanın değeri -1, değilse 0 olur. Bu alan sorgulanırken -1 ve 0 değerleri kontrol edilmelidir.

OLE Veri Tipi

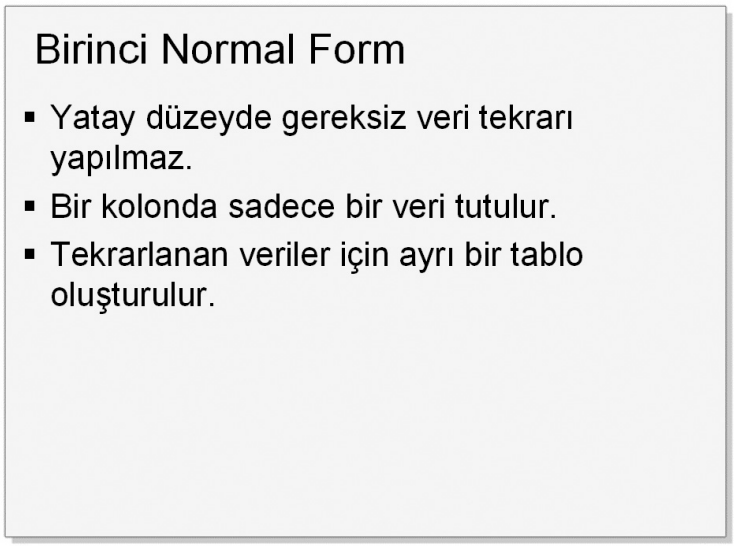
Alana bir nesne eklemek veya bağlamak için kullanılan veri tipidir. Resimler, Excel dosyaları veya bir dosyadan seçebileceğiniz herhangi bir nesne bağlanabilir.

Konu 3: Veri Modelleme Gereksinimleri

Verileri tablolarda tutarken bazı modellemelere gereksinim duyulur. Örneğin, yazılan bir verinin tekrarlamaması önemlidir. Ürünler tablosunda kategori isim olarak tutulursa, aynı kategorideki ürünler için bu ismin tekrar yazılması gerekecektir. Bu durum, hem tabloya veri girişini zorlaştırır, hem de değişiklik yapılmak istenirse, her ürünün kategorisini değiştirmek gerekir. Bu tip sorunlar, normalizasyon kurallarını ortaya çıkarmıştır.

Normalizasyon, yer alanından kazanma, veri tutarlılığı ve ölçeklenebilirlik amacıyla tablolardan gereksiz verilerin çıkartılması işlemleridir. Bu işlemler, tabloların üç etapta normal formlara getirilmesi ile gerçekleşir.

Birinci Normal Form



Bu işlem, yatay düzeyde gereksiz veya tekrarlanan verilerin çıkartılmasıdır. Satırlarda en az düzeyde veri tutulması ve bir bilginin sadece bir kolonda bulunması sağlanır.

Örnek: Bu örnekte bir eğitmen grubunun yaptığı projeler bir veritabanında tutulur. Verilerin tek bir tabloda tutulması bazı problemlere yol açar.

Eğitmen1	Eğitmen2	Proje	Konu	Saat	Kurum
Ali	Veli	Uzmanlık Kitabı	Windows	300	BilgeAdam
Ali	Veli	Mühendislik Kitabı	Windows, Web	350	BilgeAdam

Bu örnekte, projelerin eğitmenleri iki ayrı alanda tutulmuştur. Bu durum 1NF (birinci normal form) kuralını ihlal etmiştir. Yani bir satırda, bir verinin tekrar etmesi söz konusudur. Bu tabloda projeler iki eğitmenle sınırlanmış durumdadır. Ancak bir kitabı birçok eğitmenin yazdığı durumlar da olabilir.

Ayrıca, proje konularında birden fazla bilgi tutulur. Mühendislik Kitabı projesinin Windows ve Web olmak üzere iki tane konusu bulunur. Belli bir konuya göre arama yapmak zorlaşır.

Eğitmenler tek bir alanda toplanıp, konular kitaplara göre tekrar düzenlenebilir.

Eğitmen	Proje	Konu	Saat	Kurum
Ali	Uzmanlık Kitabı	Windows	300	BilgeAdam
Veli	Uzmanlık Kitabı	Windows	300	BilgeAdam
Ali	Mühendislik Kitabı	Web	350	BilgeAdam
Veli	Mühendislik Kitabı	Web	350	BilgeAdam

Yeniden düzenlenen bu tabloda ise bir kitap projesi için iki tane satır oluşuyor. Ayrıca Mühendislik kitabının sadece Web konusunda olduğu görülüyor. Diğer konu için de ayrıca iki satır eklenmesi gerekir.

Eğitmen	Proje	Konu	Saat	Kurum
Ali	Uzmanlık Kitabı	Windows	300	BilgeAdam
Veli	Uzmanlık Kitabı	Windows	300	BilgeAdam
Ali	Mühendislik Kitabı	Web	350	BilgeAdam
Veli	Mühendislik Kitabı	Web	350	BilgeAdam
Ali	Mühendislik Kitabı	Windows	350	BilgeAdam
Veli	Mühendislik Kitabı	Windows	350	BilgeAdam

Ancak bu şekilde, verilerin gereksiz yere tekrarlandığı görülür. Veriler bu şekilde tekrar yazıldıkları zaman hata olasılığı artar. Dolayısıyla, veri bütünlüğü bozulur. Örneğin Mühendislik Kitabı yerine Muhendis Kitapi gibi bir veri girildiği zaman, alınacak raporlarda çelişkiler ortaya çıkar.

Dolayısıyla tekrarlanan bu verilerin ayrı bir tabloda tutulması gerekir.

Eğitmen No	Eğitmen
1	Ali
2	Veli

Konu No	Konu
500	Windows
501	Web

Proje No	Proje	Saat	Kurum
100	Uzmanlık Kitabı	300	BilgeAdam
101	Mühendislik Kitabı	350	BilgeAdam

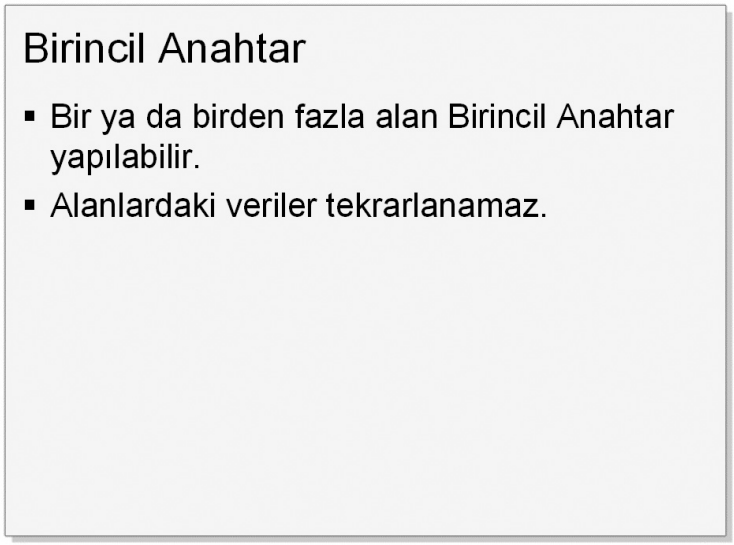
Eğitmenler ve konular tablosundaki verilerin birer numarası vardır. Bu verilere erişmek için konu veya eğitmenin ismiyle değil, numara kullanılır. Dolayısıyla, tablolarda onlarca karakterin tekrarlanması yerine, verileri temsil eden numaralar tekrarlanacaktır. Bu durum, hem veritabanının büyümesini engeller, hem de tablo üzerinde kayıt aramalarını hızlandırır.

Tablolar birbirinden ayrıldıktan sonra, projelerin hangi eğitmenler tarafından yapıldığı ve hangi konularda olduğu bilgileri kaybedilmiştir. Bu bilgilerin elde edilmesi için tablolar arasında ilişkiler kurmak gereklidir.

İlişkilerin kurulması için, tabloların birbirlerine referans vermesi gerekir. Yani bir tablodan diğerine ulaşmak için bir bilgi gerekir. Örneğin, bir projenin hangi konuda olduğunu belirlemek için konu numarasına ihtiyaç vardır. Bu numara, projenin hangi konuda olduğunu belirleyecektir.

Tablolar arasında ilişkileri kurmak için bu numaraların doğru biçimde kullanılması gerekir. Bu numaralar davranışlarına göre ikiye ayrılır.

Birincil Anahtar



Tablonun bir ya da birden fazla alanı, tek bir veriyi temsil etmesi için Birincil Anahtar (Primary Key) yapılır. Bu anahtar verinin bir daha tekrarlanmamasını sağlar ve ilişkiler kurulurken ana tabloyu belirler.

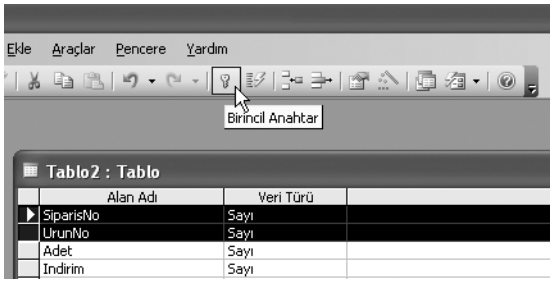
Örnekteki Birincil Anahtar olan alanlar Eğitmen No, Proje No ve Konu No alanlarıdır.

Birden fazla alanın Birincil Anahtar olarak tanımlanması, alanların tuttuğu verinin birleşik olarak tekliğini sağlar. Örneğin, sipariş detayları tablosunda, sipariş numarası ile ürün numarasının beraber tekrarlanmaması gerekir. Aksi halde, bir siparişteki ürünün iki farklı adet, indirim vs. bilgileri olacaktır.

Sipariş No	**Ürün No**	**Adet**	**İndirim**
100	**680**	**1**	**0**
100	679	2	10
102	680	1	15
100	**680**	**2**	**5**

Bu tabloda sipariş ve ürün numarası beraber Birincil Anahtar yapılmıştır. Dolayısıyla, bu alanların herhangi birisinde bir veri tekrarı olabilir. Önemli olan, bu iki alanın beraber aynı veriyi tutmamasıdır. Örnekte, 100 numaralı siparişte 680 numaralı ürün kaydı iki defa geçmiştir. Yapılacak sorgularda, bu ürünün siparişte 1 adet olduğu ve 0 YTL indirim yapıldığı, aynı zamanda 2 adet olduğu ve indirimin 5 YTL olduğu görülür. Bu da verinin tutarlılığını bozar.

Access ile tablolarda Birincil Anahtar oluşturmak için, istenen alanlar seçilerek araç çubuğundaki Birincil Anahtar düğmesi tıklanır (Resim 11.9).



RESİM 11.9: Birincil Anahtar düğmesi.

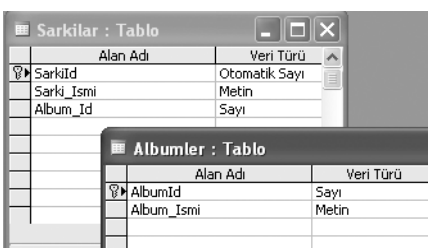
Yabancı Anahtar

Yabancı Anahtar

- Başka bir tablonun Yabancı Anahtar alanına referans gösterir.
- İlişkideki Birincil Anahtar üzerinde güncelleme ve silme işlemleri, bu alanda da yapılabilir.
  - Ardarda Güncelle
  - Ardarda Sil
- İlişkilerde, Yabancı Anahtar alanındaki değer kontrol edilebilir.
  - Bilgi Tutarlılığına Zorla

Bir tablo içinde başka bir tabloya referans vermek için, o tablonun numarası kullanılır. Yani o tablonun Birincil Anahtar alanına gönderme yapılır. Bu işlemin yapılması için, referans gönderen tabloda bu verinin tutulması gerekir. Farklı bir tablonun birincil anahtarını tutan alana Yabancı Anahtar (Foreign Key) denir.

Örneğin, şarkı listesinin tutulduğu bir tabloda albüm numarası, albümler tablosundaki Birincil Anahtar olan alana referans verir.

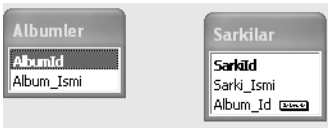


RESİM 11.10: Yabancı Anahtar.

Bu anahtarların kullanımı, ilişkilerin tanımlanmasında büyük öneme sahiptir. Tabloların normalizasyonunun sağlanması için birbirleriyle ilişkilendirilmeleri gerekir. Üç çeşit ilişki vardır.

1. Bire bir ilişki (One to One)
2. Bire sonsuz ilişki (One to Many)
3. Sonsuza sonsuz ilişki (Many to Many)

Access ile, tablolar arasındaki ilişkiler, bir alanın sürüklenip diğer tablodaki bir alanın üzerine bırakılması ile kurulur (Resim 11.11). Access bu alanların Birincil Anahtar olup olmadığına bakarak ilişkinin cinsini belirler.



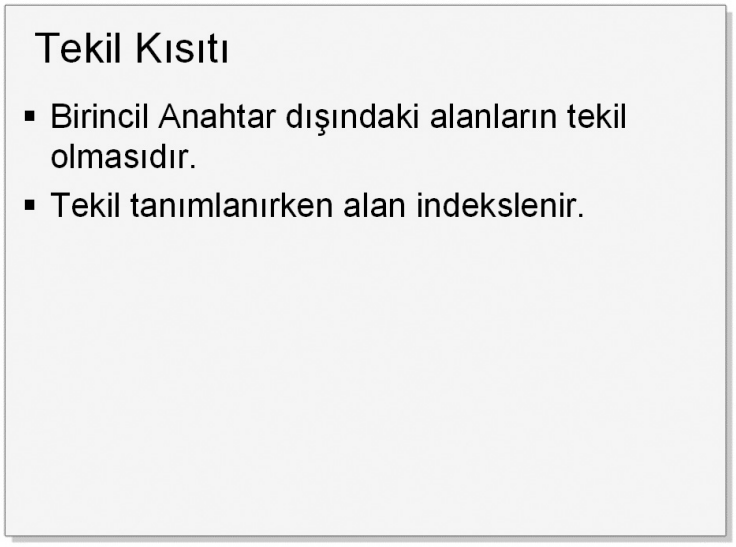
RESİM 11.11: İlişki oluşturmak.

İlişki tanımlanırken açılan İlişkileri Düzenle penceresinde, tablolardaki hangi alanlar üzerinde ilişki kurulacağı gösterilir. Buradan, ilişkinin türü ve davranışı hakkında özel ayarlamalar yapılır (Resim 11.12).

- **Bilgi tutarlılığına zorla.** Bir tablodaki verinin diğer tabloda varolup olmadığını kontrol eder.
- **İlişkili alanları ardarda güncelle (Cascade Update).** Birincil Anahtar üzerinde bir değişiklik yapılmışsa, ilişkide olduğu tablolardaki Yabancı Anahtar alanları da değiştirir.
- **İlişkili kayıtları ardarda sil (Cascade Delete).** Tabloda bir kayıt silindiği zaman, ilişkide olduğu tablolardaki veriler de silinir.

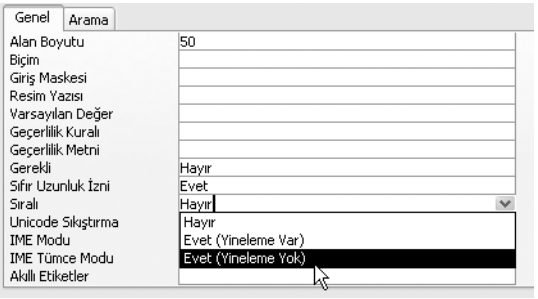
RESİM 11.12: İlişkileri Düzenle penceresi.

Tekil Kısıtı (Unique Constraint)



Bazı durumlarda, Birincil Anahtar olmayan alanların bazılarının da tabloda birden fazla kez geçmesi istenmez. Örneğin, öğrenci tablosundaki bir numara başka bir öğrenci için geçerli değildir. Ya da sicil tablosundaki bir TC kimlik numarası tekrarlanmaz. Bu alanların Tekil olarak tanımlanması gerekir.

Access ile tablo tasarlarken, alanların Tekil olarak tanımlanması, indekslemeyi gerektirir. Bir alanın indekslenmesi, tabloda aramaların o alan üzerinden daha hızlı yapılmasını sağlar. Ancak her alan üzerinde indeks kullanılmamalıdır. Bu durum, sorguların performansını artırmak yerine düşürür. Üzerinde sıkça sorgu çalıştırılan alanlar indekslenebilir (Resim 11.13).



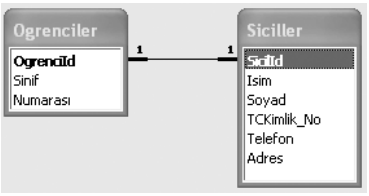
RESİM 11.13: Tekil tanımlaması.

Bire Bir İlişki

Bir tablodaki bir kayıt, diğer tablodaki bir veri için ancak bir kez kullanılabilir. Örneğin Sicil tablosu, bir kişinin ismini, soyadını ve kimlik numarasını tutuyor olsun. Öğrenci tablosu ise öğrencinin okul numarası, sınıfı gibi kayıt bilgilerini

tutuyor olsun. Öğrenci ile Sicil arasında bire bir ilişki vardır. Öğrenci tablosundaki bir veri, Sicil tablosunda sadece bir veriyi referans gösterebilir. Sicil tablosundaki bir veri de, Öğrenci tablosundaki bir veri için kullanılabilir. Dolayısıyla, bir öğrencinin bir sicili olabilir, bir sicil ise sadece bir öğrenciye ait olabilir (Resim 11.14).

Tablolar arasındaki bu ilişkiler iki Birincil Anahtar üzerinden yapılır.

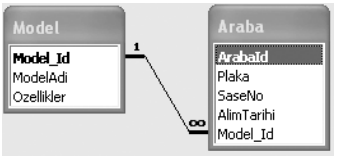


RESİM 11.14: Bire bir ilişki.

Bire Sonsuz İlişki

Tablodaki bir verinin, ilişkide olduğu tabloda birden fazla kez kullanılabilmesidir. Örneğin, bir araba ve model tabloları arasında bire sonsuz bir ilişki vardır. Araba tablosundaki bir veri, model tablosundaki bir veriyi bir kez kullanabilir. Ancak model tablosundaki bir veri, araba tablosunda birden fazla veri tarafından kullanılabilir. Yani, bir arabanın sadece bir modeli olur ve bir model birden fazla arabanın modeli olabilir (Resim 11.15).

Tablolar arasında bire sonsuz bir ilişki oluşturmak için, birden fazla veride geçecek olan tabloda Birincil Anahtar, bu değerin bir kere tutulacağı tabloda Yabancı Anahtar olmak zorundadır.



RESİM 11.15: Bire sonsuz ilişki.

Sonsuza Sonsuz İlişki

İki tablo arasında sonsuza sonsuz bir ilişkiyi temsil eder. Tablolardaki her veri diğeri için birden fazla kullanılıyorsa, iki taraflı sonsuz bir ilişki vardır. Örneğin, bir film ve oyuncu tabloları arasındaki ilişki sonsuza sonsuzdur. Film tablosundaki bir veri, oyuncular tablosunda birden fazla veri için kullanılabilir. Aynı şekilde, oyuncu tablosundaki bir veri, filmler tablosunda birden fazla veri için kullanılabilir. İlişki şu şekilde tanımlanabilir:

Bir oyuncu birden fazla filmde oynayabilir. Bir filmde birden fazla oyuncu bulunabilir.

Tablolar arasında sonsuza sonsuz bir ilişki kurmak için, ara tabloya ihtiyaç duyulur. Bunun nedeni, her iki tablodaki verilerin birden fazla eşinin bulunabilmesidir. Yapılan ara tabloya, iki tablodan alınan Birincil Anahtar alan-

ları konur. Bu alanlar ikili Birincil Anahtar yapılarak veri bütünlüğü sağlanmış olur (Resim 11.16).

RESİM 11.16: Sonsuza sonsuz ilişki.

Tabloların birinci normal forma getirilmesi için ilişkilerin kurulması gerekir. Bu durumda, ayrılan tabloların birbirleri ile ilişkilerinin saptanması ve bunun sonucunda Yabancı Anahtar alanlarının eklenmesi veya ara tabloların oluşturulması gerekir.

Örneğin, Proje ile Konular arasında bir sonsuza sonsuz ilişki vardır. Bir projenin birden fazla konusu olabilir ve bir konuda birden fazla proje yapılabilir. Bunun için ara tablonun kurulması gerekir.

Konu No	**Konu**
500	Windows
501	Web

Proje No	**Proje**	**Saat**	**Kurum**
100	Uzmanlık Kitabı	300	BilgeAdam
101	Mühendislik Kitabı	350	BilgeAdam

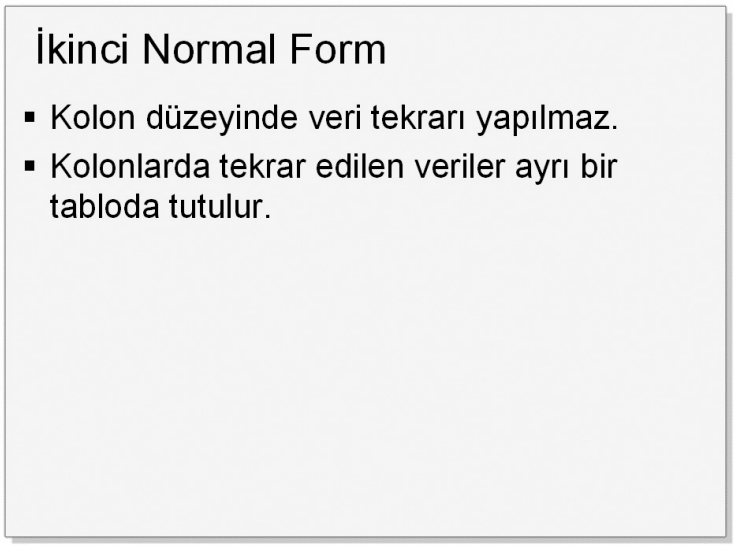
Proje No	**Konu No**
100	500
101	500
101	501

Bu tablo ile 100 numaralı Uzmanlık Kitabı projesinin 500 numaralı Windows konusunda olduğu görülür. Bu tablo biçimi, belli konulardaki projelerin sorgulanmasını da destekler.

Eğitmenler ile projeler arasında da sonsuza sonsuz bir ilişki vardır. Bir eğitmen birden fazla projede bulunabilir. Bir projeyi birden fazla eğitmen yürütebilir. Dolayısıyla, bu ilişki için de bir ara tablo yapılması gerekir.

Eğitmen No	**Proje No**
1	101
2	101
1	100
2	100

İkinci Normal Form



Birinci normal form satır bazında gereksiz verilerin çıkartılmasıydı. İkinci normal form ise kolon bazında veri tekrarını kontrol eder. Eğer bir kolonda bir veri birden fazla kez kullanılıyorsa, bu verilerin ayrı bir tabloda tutulması gerekir.

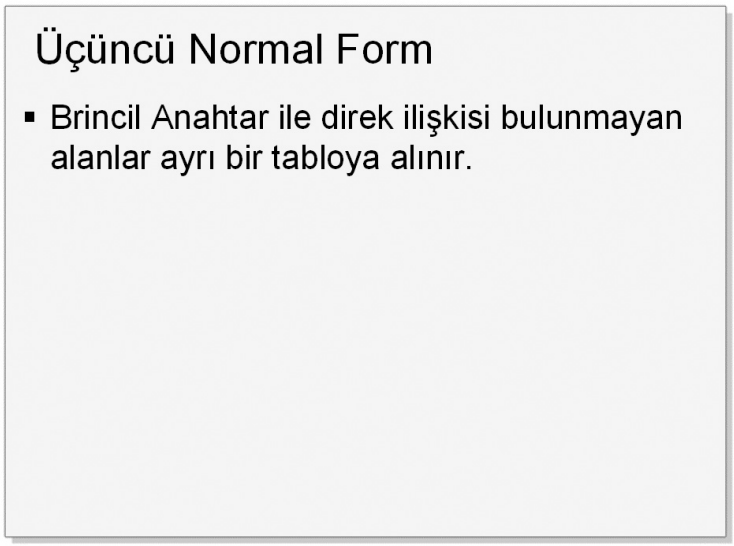
Örnekte kurum ismi olan BilgeAdam, tüm satırlar için yazılmıştır. Dolayısıyla bu kolonda veri tekrarı yapılmıştır. Bu kurum ismi ayrı bir tabloda tutulup, ana tabloda numarası ile referans gösterilmelidir.

Kurum No	Kurum İsmi	Şehir	Adres
221214	BilgeAdam	İstanbul	Barbaros Bulvarı Beşiktaş

Bu durumda, projeler ve kurum tablosu arasında bire sonsuz bir ilişki olduğu için, projeler tablosuna hangi kuruma ait olduğunu belirtmek üzere bir Yabancı Anahtar eklenir.

Proje No	Proje	Saat	Kurum No
100	Uzmanlık Kitabı	300	221214
101	Mühendislik Kitabı	350	221214

Üçüncü Normal Form



Üçüncü normal formda, tablonun Birincil Anahtar ile direk ilişkisi bulunmayan, ancak diğer alanlara bağlı alanları bulunur. Örneğin, kurumlar tablosunda şehir ismi alanının kurum ile doğrudan bağlantısı yoktur. Adres alanı ile daha çok bağlantılıdır. Bu alanların ayrı bir tabloya alınması, üçüncü derece normalizasyondur.

Tablolar ayrıldıktan sonra aralarındaki ilişkiler belirlenmelidir. Bu örnekte bir kurumun birden fazla adresi olabilir. Ancak bir adres, sadece bir kuruma aittir.

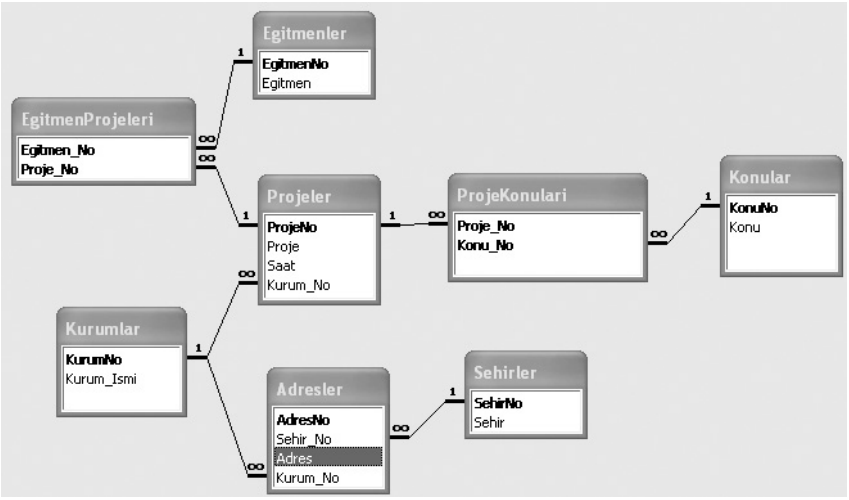
Kurum No	**Kurum İsmi**
221214	BilgeAdam

Adres No	**Şehir**	**Adres**	**Kurum No**
17982	İstanbul	Barbaros Bulvarı Beşiktaş	221214

Üçüncü normal forma getirilen tabloların, diğer formların da kısıtlarını sağlaması gerekir. Adres tablosundaki Şehirler alanı, her adres için tekrarlanacaktır. Bu da ikinci normal form kuralının ihlali demek olur. Dolayısıyla Şehir alanını ayrı bir tablo olarak ayırmak gerekir.

Şehir No	**Şehir İsmi**
34	İstanbul

Adres No	**Şehir No**	**Adres**	**Kurum No**
17982	34	Barbaros Bulvarı Beşiktaş	221214



RESİM 11.17: Örnekteki ilişkiler.

Uygulama: Alışveriş Modeli

Bir e-ticaret internet sitesinin hedefi, ürünlerin büyük kitlelere satışını gerçekleştirmektir. İnternet kullanıcıları bu hedef kitleyi oluşturur. Satılan ürünler, bu kullanıcılara çeşitli hizmetler sunularak pazarlanmalıdır. Veritabanında ürünlerin tutulması, stok durumlarının ve siparişlerin gözlenmesi kadar, kullanıcı kayıtlarının tutulması, yeni kampanyaların açılması, ürünler hakkındaki yorumların tutulması gibi kavramlar da önemlidir. Veritabanının tasarlanmasında bu kavramlar tek tek ele alınıp incelenmelidir.

Kaynak Yönetimi Modülü

E-ticaret firmasının ürünlerinin yönetimi, stok, sipariş ve firma bilgilerinden oluşur. Ürünlerin stoklardaki durumları takip edilmeli ve gerektiği zaman firmalardan tedarik edilmelidir. Dolayısıyla ürünler, firmalar, siparişler ve stoklar bu modülde işlenmelidir.

- **Ürünler.** Bu tablo, ürünlerin detaylı bilgilerini tutar. Ürünün ismi, birim fiyatı, eklenme tarihi, özellikleri, üretimde olup olmadığı, incelenme sayısı gibi bilgilerin tutulur (Resim 11.18).

  Ürünlerin hangi kategoride oldukları ve sağlayıcı firma bilgileri de tutulmalıdır. Ancak kategori ismi kolon bazında birçok defa tekrarlanacağı için, ikinci normalizasyon kuralına göre ayrı bir tabloya alınmalıdır. Sağlayıcı firma bilgileri de aynı şekilde ayrı bir tabloda tutulmalıdır. Bu durumda bire sonsuz bir ilişki oluşur. Yani bir firma birden fazla ürün sağlar, ancak bir ürün sadece bir firma tarafından sağlanır. Dolayısıyla bu iki alan Yabancı Anahtar olarak tanımlanmalıdır.

RESİM 11.18: Ürünler tablosu.

- **Firmalar.** Firma bilgileri ayrı bir tablo olarak tutulur. Bilgi olarak adres, müşteri temsilcisi ismi, e-posta ve Web sayfası adresleri tutulur (Resim 11.19).

RESİM 11.19: Firmalar tablosu.

- **Siparişler.** Ürünler satın alındıktan sonra, sipariş bilgisi olarak kayda geçer. Siparişlerin nakliye ücreti, sipariş verilme ve gönderilme tarihi, havale ile ödeme durumlarında son ödeme tarihi, gönderilecek adres, ödenip ödenmediği ve siparişin iptal edilip edilmediği gibi bilgileri tutulur. Ayrıca siparişi hangi kayıtlı kullanıcının verdiğini de tutmak gerekir. Bir siparişi sadece bir kullanıcı verebilir ve bir kullanıcı birden fazla sipariş verebilir. Dolayısıyla bire sonsuz bir ilişki oluşturmak için kullanıcı numarası Yabancı Anahtar olarak tanımlanmalıdır (Resim 11.20).

RESİM 11.20: Siparişler tablosu.

Siparişler ile Ürünler arasında sonsuza sonsuz bir ilişki vardır. Yani bir siparişte birden fazla ürün bulunabilir. Bir kullanıcı aynı anda birden fazla ürün almak isteyebilir. Aynı şekilde bir ürün birden fazla siparişte bulunabilir. Yani bir ürün birden fazla kullanıcıya satılabilir. Bu durumda Siparişler ile Ürünler arasında ayrı bir tablo yapılması gerekir.

Bu ara tablo, bir siparişteki bir ürün bilgisini tutacaktır. Dolayısıyla bu tablo daha etkin bir şekilde kullanılabilir. Örneğin belli bir siparişte bir üründen kaç tane alındığı ancak bu tabloda tutulabilir. Ayrıca, bu ürün,

yapılan bir kampanyadan alınıyor olabilir. Böylece, bu kayıtta kampanya bilgilerinin de tutulması gerekir (Resim 11.21).

RESİM 11.21: SiparişDetayları tablosu.

- **StokMerkezleri.** Ürünlerin belirli şehirlerde veya belirli merkezlerdeki stokları tutulması gerekir. Siparişin verildiği yere en yakın stoktan ürün göndermek için bu stok bölgelerinin tutulması gerekir. Bu stokun adres, e-posta, merkez olup olmadığı bilgileri tutulmalıdır (Resim 11.22).

RESİM 11.22: StokMerkezleri tablosu.

StokMerkezleri ile Ürünler arasında sonsuza sonsuz bir ilişki vardır. Yani bir stok merkezinde birden fazla ürün bulunabilir ve bir ürün birden fazla stok merkezinde bulunabilir. Bu ilişki için ara bir tablo yapılmalıdır (Resim 11.23).

Müşteri Yönetim Modülü

Kaynak planlamaları yapıldıktan sonra, bu kaynakların müşteriye ne şekilde sunulacağına karar verilmelidir. Kullanıcılar Internet sitesini kullanırken kendilerine bir hesap açabilirler ve siparişlerini bu hesap ile yaptıklarında, kendilerine ait istatistikleri kolayca elde edebilirler. Örneğin bir kullanıcı, en çok hangi kategoride ürünleri satın aldığını sorgulayabilir. Kullanıcılar, siparişlerini vermeden önce ürünlerle ilgili bilgi almak isteyebilir. Bu ürünleri daha önce alan

kullanıcıların yazdıkları yorumlardan faydalanmaları için, ürün yorumlarının da tutulması gerekir. Ayrıca, kullanıcıya değişik tarihlerde açılan, belli süreli kampanyaların sunulması, e-ticaret sitesinin kullanımını artıracaktır. Kullanıcılar ürünleri incelerken, satın almadan önce sepetlere ekleyebilirler. Böylece siteyi tekrar ziyaret edince, daha önceden inceledikleri ve sepete ekledikleri ürünleri görebilirler.

RESİM 11.23: StokDurumu tablosu.

- **Kullanıcılar.** Bu tabloda kullanıcı hakkında bilgiler tutulur. İsim, soyadı, e-posta, kayıt tarihi gibi bilgilerin yanı sıra siteye giriş yapmak için kullanıcı adı ve parolanın da tutulması gerekir. Bu parolanın değişikliği durumda güvenlik sorusu ve cevabı da ayrıca tutulmalıdır (Resim 11.24).

RESİM 11.24: Kullanicilar tablosu.

- **Yorumlar.** Kullanıcıların yaptıkları yorumların bir tabloda tutulması gerekir. Ancak burada dikkat edilmesi gereken nokta, bir kullanıcının yorum yazması için sisteme giriş yapmasının gerekmemesidir. Dolayısıyla, burada Kullanıcılar tablosuna bir referans göstermeye gerek yoktur. Yorumları yazan kişileri takma adları, yazdığı yorumlar, tarih ve verdikleri puan tutulmalıdır (Resim 11.25).

  Ayrıca, yorumun hangi ürün hakkında yapıldığını belirten ve ürünler tablosuna referans gösteren bir Yabancı Anahtar alanının tutulması gerekir.

RESİM 11.25: Yorumlar tablosu.

- **Sepetim.** Kullanıcıların ürünleri inceledikten sonra sepetlerinde saklaması için oluşturulan bir tablodur. Bu tabloda ürün numarası ve kullanıcı numarasına referans gösterilmelidir. Bu ürünlerin eklenme tarihi ve adeti de tabloda tutulmalıdır.

  Kullanıcılar, ürünleri sürekli sepete ekleyip çıkartabilir. Çıkartma işleminde, verinin tablodan silinmesi gerekir. Ancak bir kaydın sürekli eklenip silinmesi performansı düşürür. Dolayısıyla ürünün sepetten çıkartıldığını belirleyen bir Evet/Hayır veri tipinde alan belirlenebilir. Bu alanın değeri Evet ise ürün sepettedir ve kullanıcıya gösterilir. Ürünün tekrar ekleme işleminde ise sadece bu alan güncellenir (Resim 11.26).

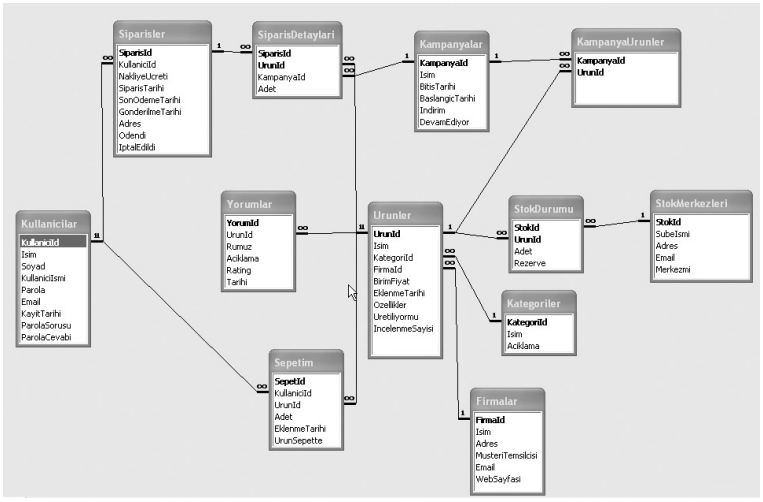
RESİM 11.26: Sepetim tablosu.

- **Kampanyalar.** Kullanıcıya sunulan kampanyalar e-ticaret kavramında önemli bir yer alır. Bu kampanyalar bir ya da birden fazla ürünün toplam fiyatında belli tarihler arasında belli bir indirim yapılmasıyla gerçekleşir. Kampanya tablosunda kampanyanın başlangıç bitiş tarihleri, devam edip etmediği ve yapılan indirim birer alan olarak tutulmalıdır (Resim 11.27).

  Bu durumda, bir kampanyada birden fazla ürün olabilir. Bir ürün ise birden fazla kampanya dahilinde olabilir. Dolayısıyla ara tablo eklenerek sonsuza sonsuz bir ilişki kurulmalıdır (Resim 11.28).

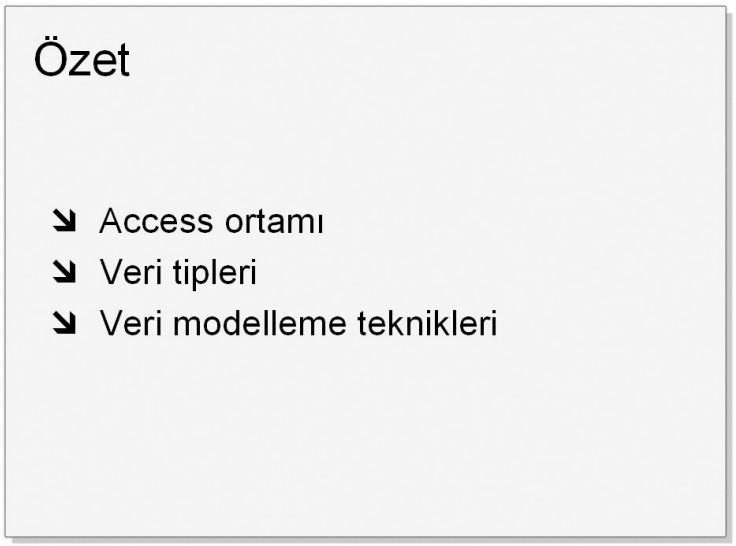
RESİM 11.27: Kampanyalar tablosu.

RESİM 11.28: KampanyaUrunler tablosu.



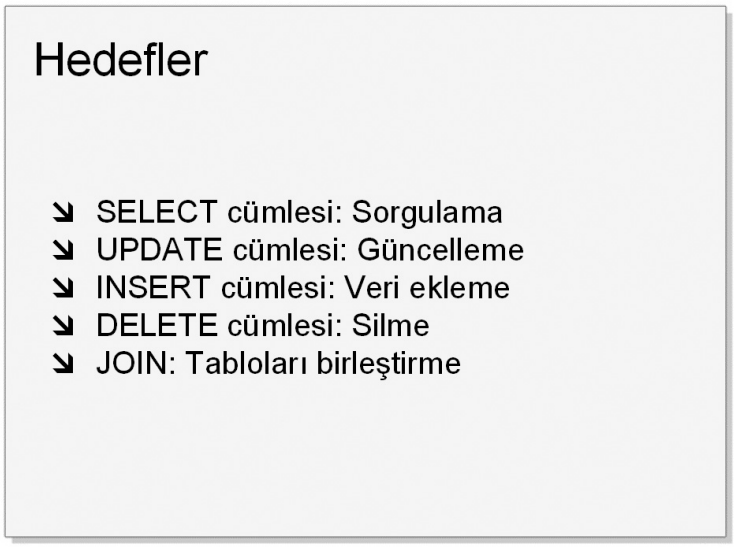
RESİM 11.29: Uygulamadaki ilişkiler.0

Modül Sonu Soruları & Alıştırmalar



1. Veritabanı yönetim sistemi kavramını ve bu sistemlere neden ihtiyaç duyulduğunu açıklayın.
2. Microsoft Access platformunun avantajlarını açıklayın.
3. Microsoft Access'te yer alan veri türlerini ve kullanım alanlarını açıklayın.
4. Birincil Anahtar ve Yabancı Anahtar yapılarını ve kullanım alanlarını açıklayın. Örnek bir veritabanı geliştirin.

Modül 12: SQL'e Giriş

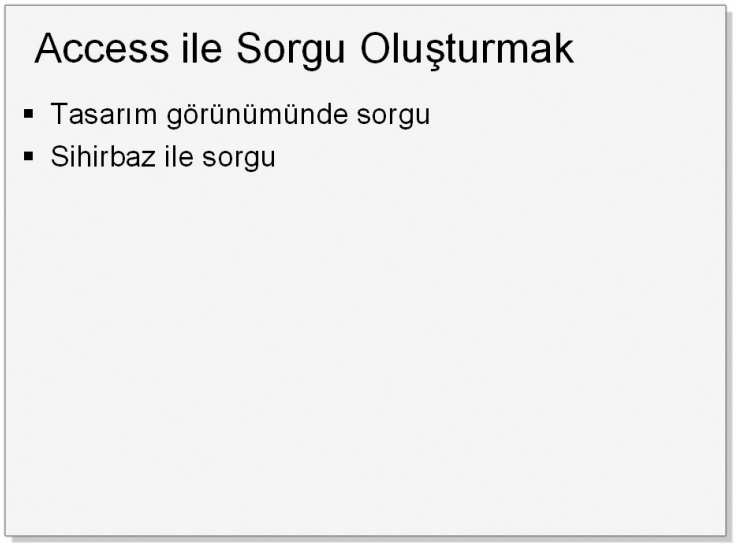


SQL (Structured Query Language) dili, veritabanları üzerinde sorgu yapmak için kullanılan bir dildir. Sorgular, analiz aşamalarında, veri eklerken, güncellerken ve silerken kullanılır. Sorgular tek bir tablo üzerinde yapılabileceği gibi, birçok tablodan veri okunmasını da sağlar. Sorgular üzerinde konan kriterler, detaylı veri analizi yapmak için kullanılır.

Bu modülü tamamladıktan sonra;

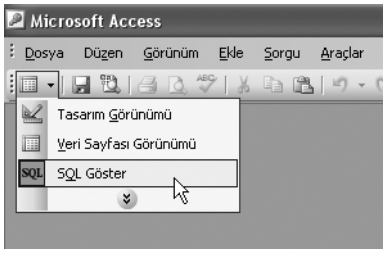
- **`SELECT`** cümleleri ile tablo sorgulayabilecek,
- Kriterler ve hesaplama fonksiyonları kullanarak sorguları şekillendirebilecek,
- **`UPDATE`** sorgusu ile tabloları güncelleyebilecek,
- **`INSERT`** sorgusu ile tablolara veri ekleyebilecek,
- **`DELETE`** sorgusu ile tablolardan veri silebilecek,
- **`JOIN`** ile birden fazla tabloyu birleştirip sorgu çalıştırabileceksiniz.

Konu 1: Access ile Sorgu Oluşturmak



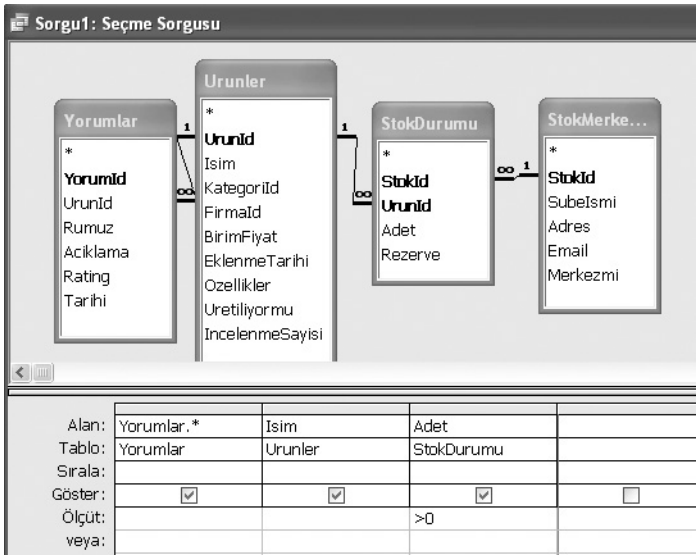
Access ile sorguları görüntülemek ve oluşturmak için, veritabanı penceresinden Sorgular sekmesi seçilir. Sorgular iki şekilde oluşturulabilir.

- **Tasarım görünümünde sorgu.** Sorgular, istenen tablolar ve gerekli alanlar eklenerek oluşturulur. Burada sorgunun üç farklı görünüm şekli vardır. Tasarım görünümü, SQL görünümü ve Veri sayfası görünümü (Resim 12.1).



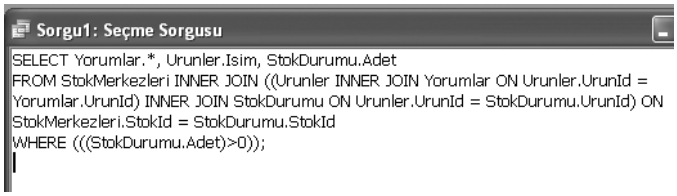
RESİM 12.1: Sorgu görünüm seçenekleri.

Tasarım görünümünde sorgular, tabloların görsel olarak eklenip, alanlarının seçilmesi ile oluşturulur. Tabloları bağlama işlemleri, kriterler ve alan isimlerinin SQL diline çevrilmesi Access tarafından yapılır (Resim 12.2).



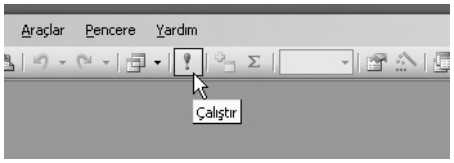
RESİM 12.2: Tasarım görünümü.

SQL görünümünde sorgular, SQL cümlesi kullanıcı tarafından yazılarak oluşturulur. Bu modüldeki sorgular bu görünümde oluşturulacaktır (Resim 12.3).



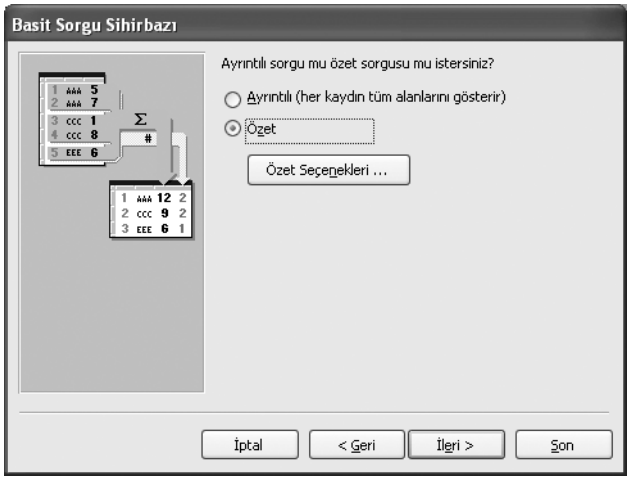
RESİM 12.3: SQL görünümü.

Veri sayfası görünümü, SQL sorgusu çalıştırıldıktan sonra verilerin görünümüdür. SQL sorguları çalıştırıldıktan sonra bu görünüme geçilir (Resim 12.4).



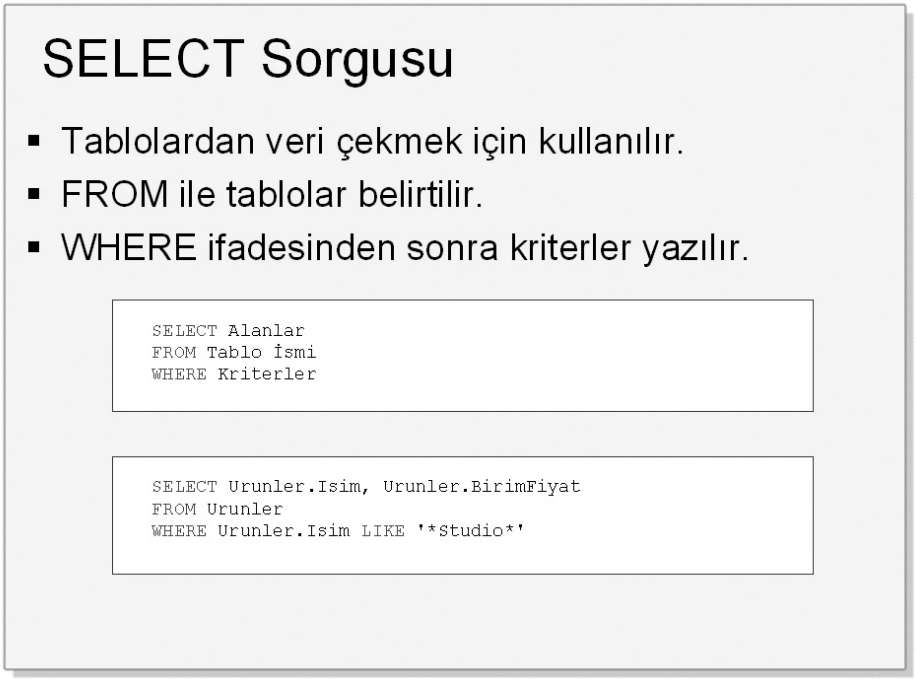
RESİM 12.4: Çalıştır düğmesi.

- **Sihirbaz ile sorgu.** Access sihirbazı, tablolar üzerinde yapılacak sorguların kolay ve hızlı bir şekilde oluşturulmasını sağlar.



RESİM 12.5: Sihirbaz ile sorgu oluşturmak.

SELECT FROM WHERE



SELECT sorgusu tablolardan veri kümesi çekmek için kullanılan sorgudur. Sorgunun yapısı **SELECT** ***Alanlar*** **FROM** ***Tablo İsmi*** **WHERE** ***Kriterler*** şeklindedir.

Bu cümlede **SELECT** kelimesinden sonra gelen alanlar, tabloları oluşturulan kolonlardır. Sonuç kümesinde, tablonun hangi alanlarının olacağını gösterir. Burada yapılan, kolon bazında filtrelemedir.

FROM ifadesi, sorgunun hangi tablo veya tablolar üzerinde yapılacağını gösterir.

WHERE ifadesinden sonra, sorgu kümesinde, verilen kritere uyan satırlar görüntülenir. Burada yapılan, satır bazında filtrelemedir.

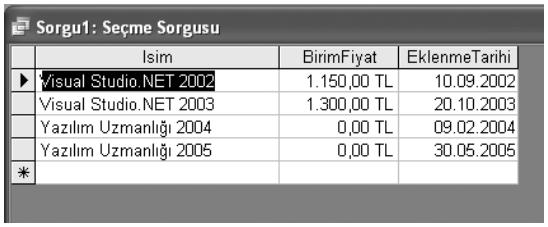
```
SELECT * FROM Urunler
```

Buradaki yıldız ifadesi, tüm alanların listeleneceği anlamına gelir.

```
SELECT
Urunler.Isim,
Urunler.BirimFiyat,
Urunler.EklenmeTarihi
FROM Urunler
```

SELECT ifadesinde alanların ismi verilirken, hangi tabloya ait olduğu da yazılır. Ancak bu durum tek tablo üzerinden yapılan işlemler için gerekli değildir. Birkaç tablo üzerinde sorgu yapıldığı zaman, alanları tablo ismiyle belirtmek gerekir.

```
SELECT
Isim,
BirimFiyat,
EklenmeTarihi
FROM Urunler
```



RESİM 12.6: Sorgu sonucu.

WHERE ifadesinden sonra yazılan kriterler mantıksal karşılaştırmalardır. Bu karşılaştırmalar alanlardaki değerler üzerinde yapılır. Karşılaştırmalar aritmetik olabildiği gibi metinsel de olabilir.

- **Büyük.** Alandaki değerin verilen bir değerden veya başka bir alandan büyük olup olmadığını kontrol eder.

```
SELECT Urunler.*
FROM Urunler
WHERE Urunler.IncelenmeSayisi > 100
```

- **Büyük Eşit.** Verilen bir alanın veya değerin, kontrol edilen alandan büyük veya alana eşit olup olmadığını kontrol eder.

```
SELECT Urunler.*
FROM Urunler
WHERE Urunler.IncelenmeSayisi >= 100
```

- **Küçük.** Alandaki değerin verilen bir değerden veya başka bir alandan büyük olup olmadığını kontrol eder.

```
SELECT Urunler.*
FROM Urunler
WHERE Urunler.IncelenmeSayisi < 100
```

- **Küçük Eşit.** Verilen bir alanın veya değerin, kontrol edilen alandan küçük veya alana eşit olup olmadığını kontrol eder.

```
SELECT Urunler.*
FROM Urunler
WHERE Urunler.IncelenmeSayisi <= 100
```

- **BETWEEN – AND**. Alandaki değerin iki değer arasında olup olmadığını kontrol eder. Değerlere eşit oldukları durumlar da sonuç kümesine dahil edilir.

```
SELECT Urunler.*
FROM Urunler
WHERE Urunler.IncelenmeSayisi BETWEEN 100 AND 200
```

- **NOT**. Verilen kritere **uymayan** kayıtları döndürür.

```
SELECT Urunler.*
FROM Urunler
WHERE NOT Urunler.IncelenmeSayisi = 0
```

- **LIKE**. Alandaki değerin belirli bir metin biçimde olup olmadığını kontrol eder.

```
SELECT Alanlar FROM Tablo WHERE AlanIsmi LIKE 'Pattern'
```

`'Pattern'` ifadesinde yazılan karakterler, alanların içinde kesin olarak geçecek karakterlerdir. Örneğin `Isim LIKE 'Enis'`. Ancak bazı özel karakterler farklı anlam ifade ederler. Örneğin, `*` karakteri sıfır veya daha fazla karakteri temsil eder. `Isim LIKE '*ni*'` ifadesi sıfır veya daha fazla karakter ile başlayan, `ni` ile devam eden ve yine sıfır veya daha fazla karakter ile biten kelimeleri kontrol eder. Örneğin Deniz, Nil, Seni, Ni değerleri bu biçime uyacaktır Tablo 12.1'de bazı kural ve örnekler verilmiştir.

TABLO 12.1: Özel Karakter Örnekleri

Pattern	Örnek	True değeri döndüren örnek	False değeri döndüren örnek
Sıfır veya birden fazla karakter *	Nu*	Nuray, Nuri	Banu
Özel karakterlerin kullanımı	Beş [*]	Beş *	Beşiktaş
Tek karakter ?	Ç?n	Çan, Çin	Çıban, Çanak
Tek Sayı #	Versiyon #	Versiyon 5, Versiyon 1	Versiyon 10, Versiyon Üç
Karakter Aralığı	[a-z]	a, b, c	43, 2
Aralık Dışı	[!0-9]	a, b, c	1, 2, 3

Örnek: Microsoft Studio ürünlerin listelenmesi (Resim 12.7).

```
SELECT Urunler.Isim
FROM Urunler
WHERE Urunler.Isim LIKE '*Studio*'
```

RESİM 12.7: Sorgu sonucu.

- **Is NULL**. Bazı alanların değerleri boş bırakılmış olabilir. Boş bırakılan alanların değerleri **NULL** olarak geçer. Sorgularda boş alanların kontrolü **Is NULL** ifadesi ile yapılır.

```
SELECT Urunler.*
FROM Urunler
WHERE Urunler.Ozellikler Is NULL
```

Bir sorguda birden fazla kriter kullanılabilir. Ancak bu kriterlerin **AND** veya **OR** ifadeleri ile ayrılmaları gerekir. **AND** ifadesi ile ayrılan kriterlerin hepsinin sağlandığı satırlar sonuca dahil edilir. **OR** ifadesi ile ayrılan kriterlerin herhangi biri sağlandığı satırlar sonuca dahil edilir.

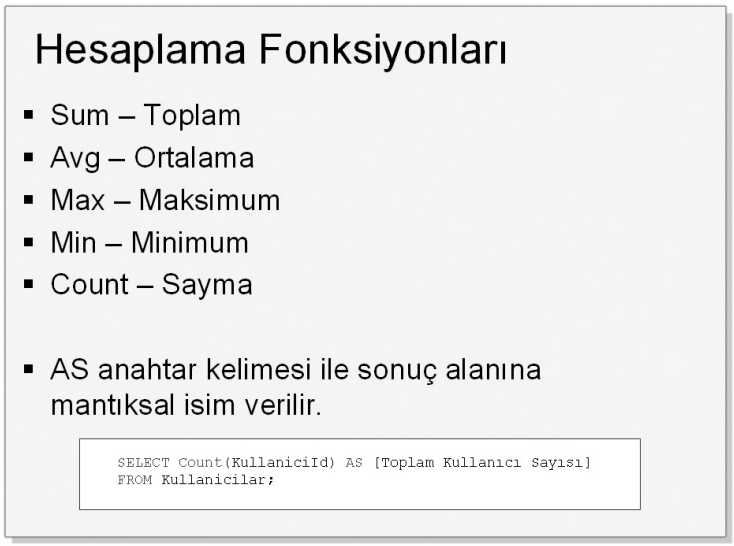
Örnek: 12.12.2002'den sonra kaydolmuş, ismi E ile başlayan kullanıcılar:

```
SELECT *
FROM Kullanicilar
WHERE Kullanicilar.KayitTarihi > #12/12/2002# AND
Kullanicilar.Isim Like 'E*';
```

E-posta adresi veya Web adresi olan firmalar:

```
SELECT Firmalar.Isim, Firmalar.Email, Firmalar.WebSayfasi
FROM Firmalar
WHERE ((Not (Firmalar.Email) Is Null)) OR ((Not
(Firmalar.WebSayfasi) Is Null));
```

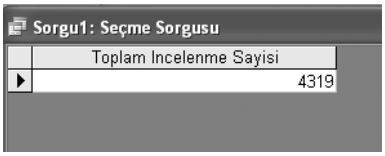
Hesaplama Fonksiyonları



Alanlar üzerinde sayma, toplama, ortalama alma gibi aritmetik işlemlerin yanı sıra minimum ve maksimum değerlerin alınması gibi işlemler de yapılabilir. Bu işlemlerin sonucunda sayısal bir sonuç ortaya çıkar. Bu sayı, sonuç tablosunda gösterilirken herhangi bir alan ismi ifade etmez. Dolayısıyla, sonuç tablosunda sayısal değerler gösterilirken mantıksal bir isim verilmesi gerekir. Bu ifade ise **AS** anahtar kelimesi ile belirtilir.

- **Sum.** Kriterlerin sağlandığı alanlar üzerinde toplama işlemi yapar.

```
SELECT Sum(IncelenmeSayisi) AS [Toplam Incelenme Sayisi]
FROM Urunler WHERE Uretiliyormu = -1;
```



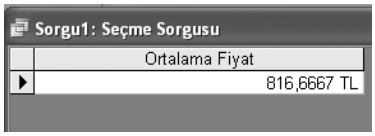
RESİM 12.8: Toplama işleminin sonucu.

- **Avg.** Kriterlerin sağlandığı alanların ortalama değerini alır.

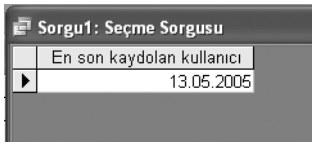
```
SELECT Avg(Urunler.BirimFiyat) AS [Ortalama Fiyat]
FROM Urunler WHERE Uretiliyormu = -1;
```

- **Max.** Kriterlerin sağlandığı alanların maksimum değerini alır. Metinsel değerlerde alfabetik olarak sıralama yapar.

```
SELECT Max(Isim) AS [En son geçen kullanıcı]
FROM Kullanicilar;

SELECT Max(KayitTarihi) AS [En son kaydolan kullanıcı]
FROM Kullanicilar;
```



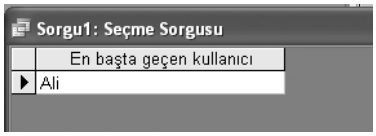
RESİM 12.9: Ortalama işleminin sonucu.



RESİM 12.10: Maksimum değerin alınması.

- **`Min`.** Kriterlerin sağlandığı alanların minimum değerini alır.

```
SELECT Min(Isim) AS [En başta geçen kullanıcı]
FROM Kullanicilar;

SELECT Min(KayitTarihi) AS [İlk kaydolan kullanıcı]
FROM Kullanicilar;
```



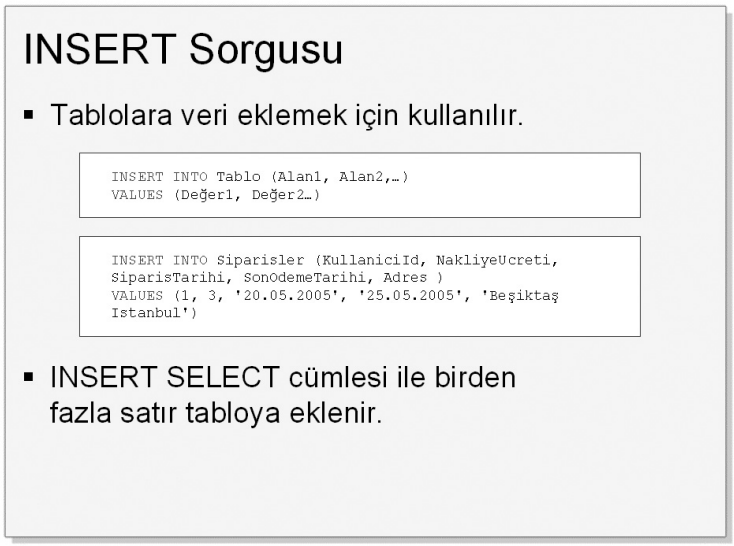
RESİM 12.11: Minimum değerin alınması.

- **`Count`.** Değeri **`NULL`** olmayan satırların kaç tane olduğunu verir. Genellikle tablolardaki satır sayısı istendiğinde bu fonksiyon kullanılır. Ancak bu tip bir sorguda, sayılan alanın boş bir değer almaması gerekir. Birincil Anahtar alanının üzerinden bir sayım yapılabilir.

```
SELECT Count(KullaniciId) AS [Toplam Kullanıcı Sayısı]
FROM Kullanicilar;
```

RESİM 12.12: Sayma işleminin sonucu.

INSERT



INSERT sorguları tablolara kayıt eklemek için kullanılır. Bu kayıtlar eklenirken tablo isimi, alan adı ve hangi değerlerin ekleneceği belirtilmelidir. **INSERT** sorgularında, gerekli olan (**NULL** kabul etmeyen) alanlara değer eklenmesi unutulmamalıdır.

Sözdizimi:

```
INSERT INTO Tablo (Alan1, Alan2,...) VALUES (Değer1,
Değer2...)
```

VALUES ifadesinde verilen değerlerin, tablonun yazılan alanlarıyla aynı sırada olması gerekir.

```
INSERT INTO Siparisler ( KullaniciId, NakliyeUcreti,
SiparisTarihi, SonOdemeTarihi, Adres )
VALUES (1, 3, '20.05.2005', '25.05.2005', 'Beşiktaş
Istanbul')
```

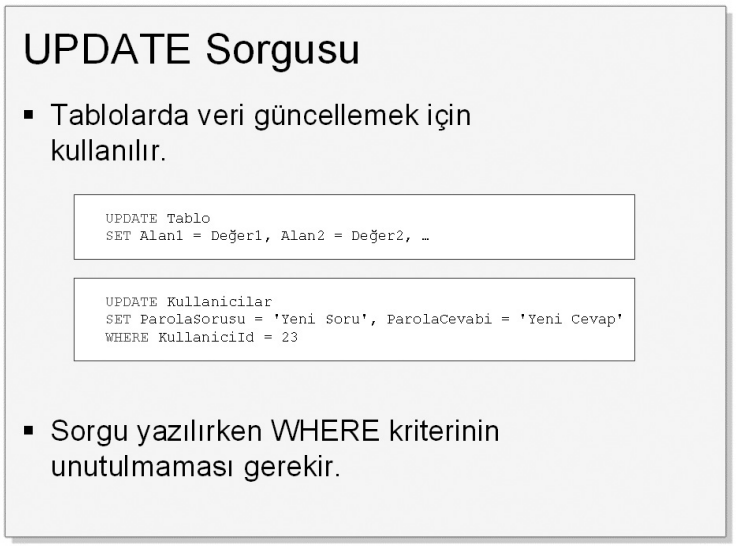
Bu tip **INSERT** sorgularında sadece tek bir değer girilebilir. Ancak bazı durumlarda birden fazla verinin girilmesi istenebilir. Bu durumda, girilecek değerler **SELECT** cümlesiyle başka bir tablodan alınır.

Örnek: Ödenen siparişlerin tutulduğu ayrı bir tablo oluşturulur. Sipariş tablosundan bu tabloya tüm ödenen kayıtların aktarılması işlemi **INSERT SELECT** cümlesi ile yapılır.

```
INSERT INTO OdenenSiparisler ( SiparisId, KullaniciId,
NakliyeUcreti, SiparisTarihi, GonderilmeTarihi, Adres)
```

```
SELECT SiparisId, KullaniciId, NakliyeUcreti, SiparisTarihi,
GonderilmeTarihi, Adres
FROM Siparisler
WHERE Odendi = -1;
```

UPDATE



UPDATE sorguları tablolarda varolan kayıtların belirli alanlarının güncellenmesi işlemini yapar. Bu sorguda da tablo, alan ve yeni değerlerin belirtilmesi gerekir.

Sözdizimi:

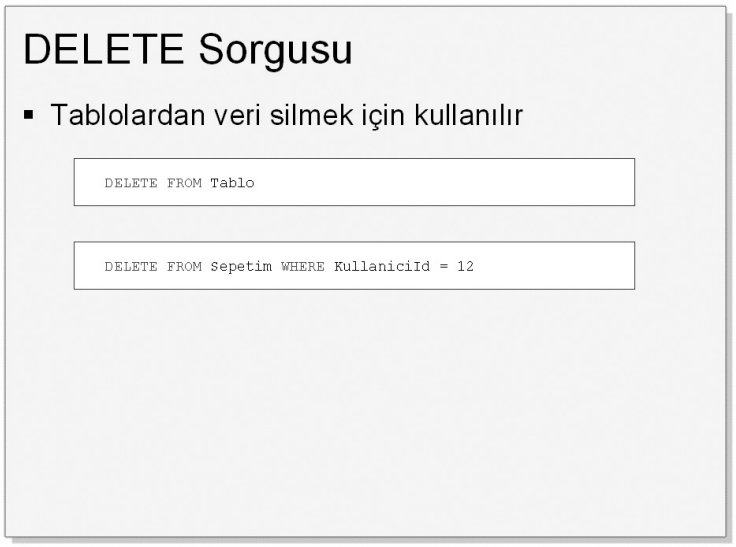
```
UPDATE Tablo SET Alan1 = Değer1, Alan2 = Değer2, ...
```

Bu sorguda dikkat edilmesi gereken en önemli nokta, belli kayıtlarda güncelleme işlemi yapılıyorsa **WHERE** kriterinin unutulmamasıdır. Aksi halde tablodaki tüm kayıtlar, sorguda belirlenen değerleri alacaktır.

Örnek:

```
UPDATE Kullanicilar
SET ParolaSorusu = 'Yeni Soru', ParolaCevabi = 'Yeni Cevap'
WHERE KullaniciId = 23
```

DELETE



Tablodan veri silmek için kullanılır. Bu sorguda alan isimleri belirtilmez, ancak **WHERE** kriterinin unutulmaması gerekir.

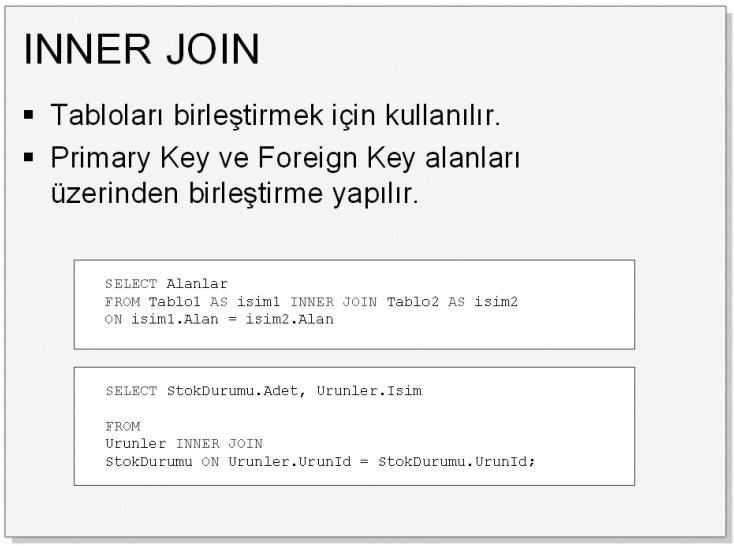
Sözdizimi:

```
DELETE FROM Tablo İsmi
```

Örnek:

```
DELETE FROM Sepetim Where KullaniciId = 12
```

Konu 2: INNER JOIN ile Tablo Birleştirmek



Birden fazla tablodan kayıt çekilmek istendiğinde, bu tabloların Birincil Anahtar ve Yabancı Anahtar alanları üzerinden birleştirilmeleri gerekir. Tabloları birleştirmek, birçok bilgiyi sonuç kümesinde tek bir tablo olarak göstermeyi sağlar. Örneğin, bir ürünün hangi kategoride olduğu bilgisi Ürünler tablosunda vardır. Ancak bu değer o kategori numarasını belirttiği için, son kullanıcıya bir şey ifade etmez. Kategori ismi ise, Kategoriler tablosunda durur. Sonuç kümesinden kategori ismini görüntülemek için bu tabloların birleştirilmesi gerekir.

Sözdizimi:

```
SELECT Alanlar
FROM Tablo1 AS isim1 INNER JOIN Tablo2 AS isim2
ON isim1.Alan = isim2.Alan
```

Burada tablo isimlerine birer takma isim verilmiştir. Bu isimler alanların seçiminde yazım kolaylığı sağlar. Bazı alanlar birbirleriyle aynı isimde oldukları için bu alanın hangi tabloya ait olduğu belirtilmelidir.

```
SELECT isim1Alan1, isim1.Alan2, ..., isim2.Alan1,
isim2.Alan2
FROM Tablo1 AS isim1 INNER JOIN Tablo2 AS isim2
ON isim1.Alan = isim2.Alan
```

İki tablonun birleştirme işlemi, **ON** ifadesinden sonra belirtilen alanlar üzerinden yapılır. Burada, iki tablo arasında ilişki kurulan alanlar belirtilmelidir.

Örnek: Ürünlerin stoklardaki miktarını öğrenmek için StokDurumu ve Ürünler tablolarını birleştirmek gerekir.

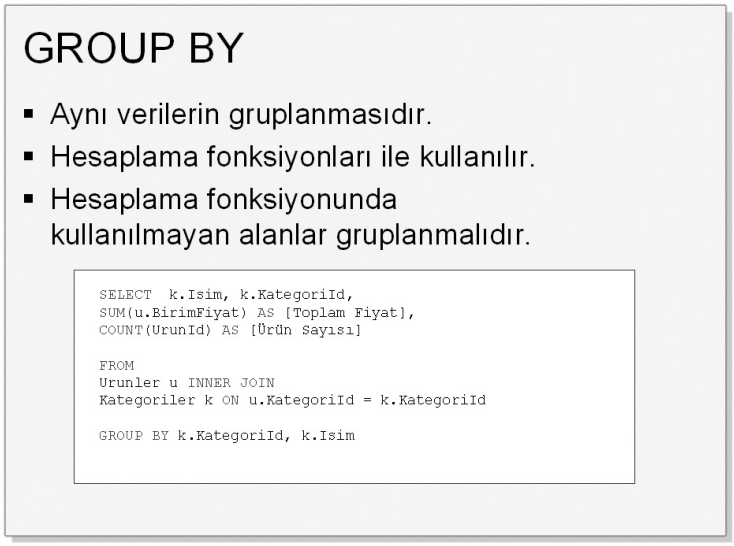
```
SELECT
StokDurumu.Adet,
Urunler.Isim

FROM
Urunler INNER JOIN
StokDurumu ON Urunler.UrunId = StokDurumu.UrunId;
```

İkiden fazla tablodan bilgi çekmek için, önce iki tablo birleştirilir. Sonuç olarak çıkan tablo ile de diğer tablolar tek tek birleştirilir. Birleştirme işlemi **((Tablo1 + Tablo2) + Tablo3) + Tablo4...** şeklindedir. **INNER JOIN** kullanılırken parantezlerin unutulmaması gerekir.

Örnek: Bir kullanıcının sepetindeki ürünlerin birim fiyatları sorgulanmak istendiği zaman, Kullanıcılar, Sepetim ve Ürünler tabloları ilişkide oldukları alanlar üzerinden birleştirilmelidir.

```
SELECT
k.Isim,
k.Soyad,
s.Adet,
u.BirimFiyat

FROM
(Urunler u INNER JOIN
Sepetim AS s ON u.UrunId = s.UrunId)  INNER JOIN
Kullanicilar AS k ON k.KullaniciId = s.KullaniciId

WHERE k.KullaniciId = 1
```

Konu 3: GROUP BY



Hesaplama fonksiyonlarının kullanıldığı sorgularda **SELECT** ifadesinden sonra sadece hesaplanan alan sonuç kümesine eklenmişti. Ancak çoğu zaman, hesaplanan alanlarla birlikte diğer alanların da sonuç kümesinde olması istenir.

Örneğin, belli bir kategoride kaç tane ürünün bulunduğu, kategori numarası sonuç kümesinde olacak şekilde isteniyor. Bu durumda, Ürünler tablosundaki kayıtların sayma işleminin gerçekleştirilmesi için önce kategori numarasına göre gruplanması gerekir.

```
SELECT
k.KategoriId,
COUNT(UrunId) AS [Ürün Sayısı]
FROM
Urunler u INNER JOIN
Kategoriler k ON u.KategoriId = k.KategoriId
GROUP BY k.KategoriId
```

Tablodan kategori numarası dışında başka herhangi bir alan daha isteniyorsa, bu alan **GROUP BY** ifadesine ya da bir hesaplama fonksiyonunun içine alınmalıdır.

```
SELECT
k.Isim,
k.KategoriId,
Sum(u.BirimFiyat) AS [Toplam Fiyat],
COUNT(UrunId) AS [Ürün Sayısı]
```

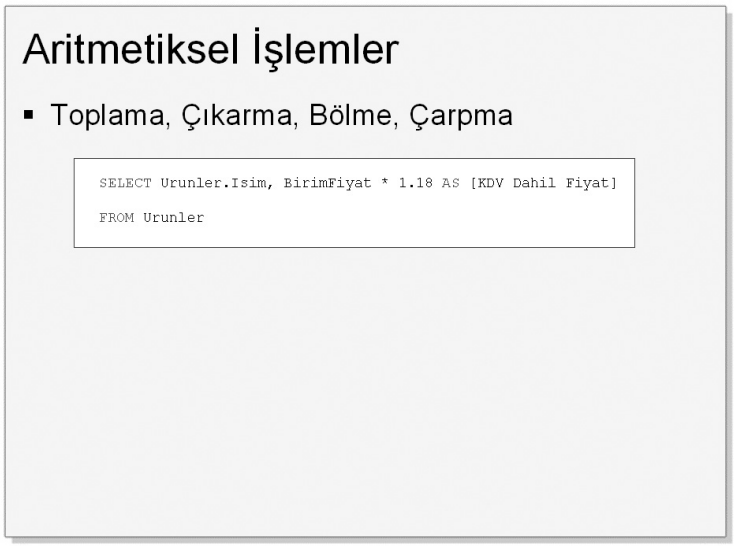
```
FROM
Urunler u INNER JOIN
Kategoriler k ON u.KategoriId = k.KategoriId
GROUP BY k.KategoriId, k.Isim
```


Sorgu1: Seçme Sorgusu

	Isim	KategoriId	Toplam Fiyat	Ürün Sayısı
	Kitap	1	0,00 TL	2
▶	Yazılım	2	2.450,00 TL	2

RESİM 12.12: `GROUP BY` kullanımı.

Konu 4: Aritmetiksel İşlemler



Sorgular sırasında, alanlar üzerinde toplama, çıkarma, çarpma, bölme gibi aritmetiksel işlemler yapılabilir. Bu işlemler sabit değerler ile yapılabildiği gibi başka alanlardaki değerler ile de yapılabilir.

Örnek: Birim fiyatlarının KDV eklenmiş halini gösteren sorgu.

```
SELECT
Urunler.Isim,
BirimFiyat * 1.18 AS [KDV Dahil Fiyat]
FROM Urunler

SELECT
Sum(BirimFiyat) * 1.18 AS [Toplam Ürünler Fiyatı KDV Dahil]
FROM Urunler
```

Örnek: Stoklarda rezerve edilmemiş toplam ürün sayısı.

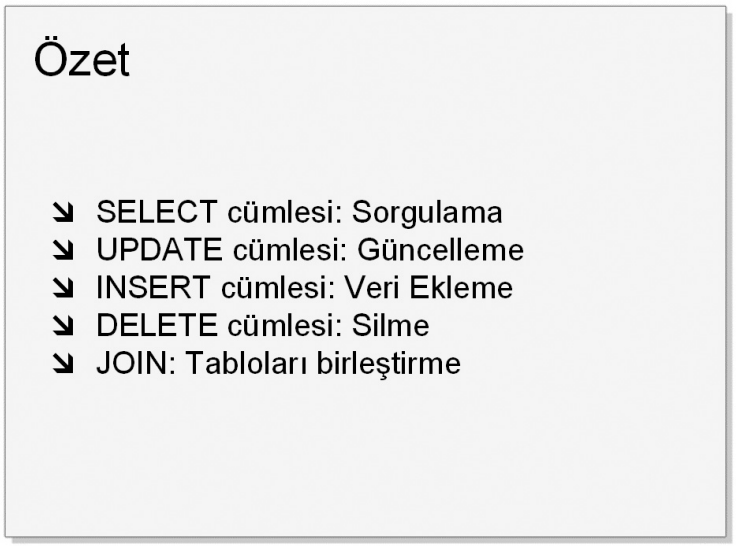
```
SELECT Urunler.Isim, Sum(StokDurumu.Adet -
StokDurumu.Rezerve) AS [Açık Ürün Sayısı - Tüm Stoklar]
FROM Urunler INNER JOIN StokDurumu ON Urunler.UrunId =
StokDurumu.UrunId
GROUP BY Urunler.Isim;
```

Toplama işlemi, sayılar üzerinde yapılabildiği gibi metinsel değerler üzerinde de birleştirme görevi görür.

Örnek: Kullanıcıların isim ve soyadlarının beraber görüntülenmesi

```
SELECT
Kullanicilar.Isim + ' ' + Kullanicilar.Soyad AS [İsim Soyad]

FROM Kullanicilar
```

Modül Sonu Soruları & Alıştırmalar



1. **SELECT** ifadesinin kullanım alanını açıklayın ve bir örnek SQL cümlesi geliştirin.
2. **INSERT** ifadesinin kullanım alanını açıklayın ve bir örnek SQL cümlesi geliştirin.
3. **UPDATE** ifadesinin kullanım alanını açıklayın ve bir örnek SQL cümlesi geliştirin.
4. **DELETE** ifadesinin kullanım alanını açıklayın ve bir örnek SQL cümlesi geliştirin.
5. **DELETE** ve **UPDATE** ifadelerini kullanırken dikkat etmemiz gereken noktaları açıklayın.
6. **CASCADE DELETE** ve **CASCADE UPDATE** ifadelerini içeren bir veritabanı uygulaması geliştirin ve silme durumunu gözlemleyin.