

**T.C.
MİLLÎ EĞİTİM BAKANLIĞI**

BİLİŞİM TEKNOLOJİLERİ

**NESNE TABANLI PROGRAMLAMAYA
GİRİŞ
482BK0168**

Ankara, 2011

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- **PARA İLE SATILMAZ.**

İÇİNDEKİLER

AÇIKLAMALAR	ii
GİRİŞ	1
ÖĞRENME FAALİYETİ-1	3
1. PROGRAMLAMA ORTAMI.....	3
1.1. Arayüz	3
1.2. Konsol Uygulaması Oluşturma	4
1.3. Konsol Ekranında Kod Yazma.....	6
UYGULAMA FAALİYETİ.....	11
ÖLÇME VE DEĞERLENDİRME.....	13
ÖĞRENME FAALİYETİ-2	14
2. İSİM UZAYLARI	14
2.1. Namespace	14
2.2. Using İfadesi	15
2.3. Grafikselsel Arayüz	18
2.4. Nesne Ekleme.....	21
2.5. Nesneye Kod Yazma.....	22
UYGULAMA FAALİYETİ.....	24
ÖLÇME VE DEĞERLENDİRME.....	26
ÖĞRENME FAALİYETİ-3	28
3. DEĞİŞKENLER VE VERİ TÜRLERİ	28
3.1. Değişken Tanımlama Kuralları	28
3.2. Değişken Tanımlama.....	29
3.3. Temel Veri Türleri	29
UYGULAMA FAALİYETİ.....	34
ÖLÇME VE DEĞERLENDİRME.....	35
ÖĞRENME FAALİYETİ-4	36
4. OPERATÖRLER	36
4.1. Aritmetiksel Operatörler	36
4.2. İşleçler ve Türler	36
4.3. İşlem Önceliği	37
4.4. Birleşim Özelliği	38
4.5. Birleşim ve Atama Operatörü	39
4.6. Artırma ve Azaltma Operatörleri	40
4.7. “var” Değişken Türü	41
UYGULAMA FAALİYETİ.....	42
ÖLÇME VE DEĞERLENDİRME.....	44
MODÜL DEĞERLENDİRME	45
CEVAP ANAHTARLARI.....	48
KAYNAKÇA	50

AÇIKLAMALAR

KOD	482BK0168
ALAN	Bilişim Teknolojileri
DAL/MESLEK	Veritabanı Programcılığı
MODÜLÜN ADI	Nesne Tabanlı Programlamaya Giriş
MODÜLÜN TANIMI	Nesne tabanlı programlama ortamı kullanılarak nesneye yönelik programlamanın temellerini basit kodlarla anlatan bir öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Programlama temelleri dersinin modüllerini tamamlamış olmak
YETERLİK	Nesne tabanlı programlama ortamını kullanarak programlamaya başlamak
MODÜLÜN AMACI	Genel Amaç Nesne tabanlı programlama ortamını kullanarak kod yazma işlemlerini yapabileceksiniz. Amaçlar <ol style="list-style-type: none">1. Nesne tabanlı programlama ortamını kullanabileceksiniz.2. Kod içerisinde isim uzaylarını(namespace) kullanabileceksiniz.3. Nesne tabanlı programlama ortamında değişkenleri kullanabileceksiniz.4. Aritmetiksel operatörleri kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Ortam: Nesne tabanlı programlama yazılımı Donanım: Bilgisayar
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma, eşleştirme vb.) kullanarak modül uygulamaları ile kazandığımız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Yazılan programların kapsadığı alanın gün geçtikçe genişlemesi, bir başka deyişle yazılan kodun satır sayısının artması programcıları yeni arayışlara itti. Ayrıca programlar büyüdükçe proje üzerinde çalışan insan sayısını da artmıştır. Her şey bir yana hâlâ yazılan programlarda belli bir standardın, belli bir modüleritenin kurulamamış olmasıdır.

Nesne tabanlı programlamada (object oriented programming) yenilikçi olan şey, yordamsal programlamada yer alan birçok şeyi yapabildiğimiz gibi sınıflar ile bilgisayar için soyut bir kavramı, hem onun hem de programcının anlayacağı bir şekilde modelleyebilmektir. Bu özellik güncel yazılımlarda, nesne tabanlı programlama dillerini popüler hâle getirmektedir.

Bu modülle OOP'ye giriş yaparak programlama ortamını kullanmayı, veri türleri ve değişken tanımlamayı, nesnelere içerisine kod yazmayı ve aritmetiksel operatörleri kullanmayı öğreneceksiniz.

ÖĞRENME FAALİYETİ-1

AMAÇ

Nesne tabanlı programlama ortamını kullanabileceksiniz.

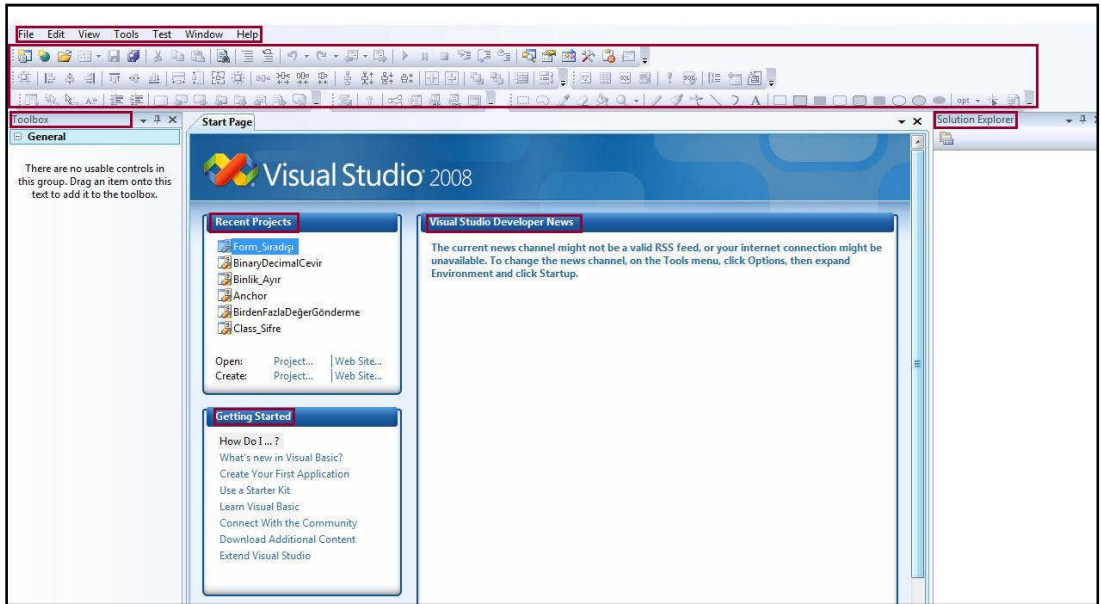
ARAŞTIRMA

- Günümüzdeki yaygın olarak kullanılan programlama dillerinin hangileri olduğunu araştırınız.
- Nesne tabanlı programlama dillerinin avantajlarının neler olduğunu araştırınız.
- Konsol ve grafiksel arayüz kavramlarını araştırınız.
- Günlük hayatta kullanılan, nesne tabanlı programlama dillerinden herhangi biriyle oluşturulmuş yazılım örneklerini inceleyerek arkadaşlarınızla paylaşınız.

1. PROGRAMLAMA ORTAMI

1.1. Arayüz

Nesne tabanlı programlama yazılımı çalıştırıldığında Resim 1.1'deki ekranla karşılaşılır.



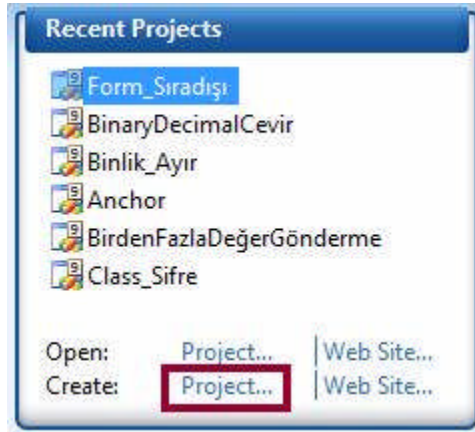
Resim 1.1:Nesne tabanlı programlama yazılımı ekranı

Resim 1.1’de işaretlenmiş bölümler ve işlevleri şöyle tanımlanabilir:

- **Menü çubuğu:** Yazılıma ait komutlar ve alt seçenekleri bulunmaktadır.
- **Araç çubukları:** Menülerden de yapılan işlemlerin daha hızlı bir şekilde yapılabilmesi için oluşturulmuş bölümdür.
- **Toolbox:** Kullanılabilecek hazır nesnelere yararlanmak için oluşturulmuştur(Label, Button, TextBox vb.).
- **Recent Projects:** Daha önce oluşturulan projelerin listelendiği bölüm, projelere hızlı erişim imkânı sağlar.
- **Solution Explorer:** Üzerinde çalışılan projeye ait dosyalara hızlı erişim için oluşturulmuştur.
- **Visual Studio Developer News:** Kullanılan yazılıma ait yenilikler ve güncellemelerin görüntülediği bölümdür. Yenilikler ve güncellemelerin görüntülenebilmesi için internet bağlantısının olması gerekmektedir.
- **Getting Started:** Yazılımı kullanmaya yeni başlayanlar için yardımcı araçların bulunduğu bölümdür. İçerik İngilizcedir.

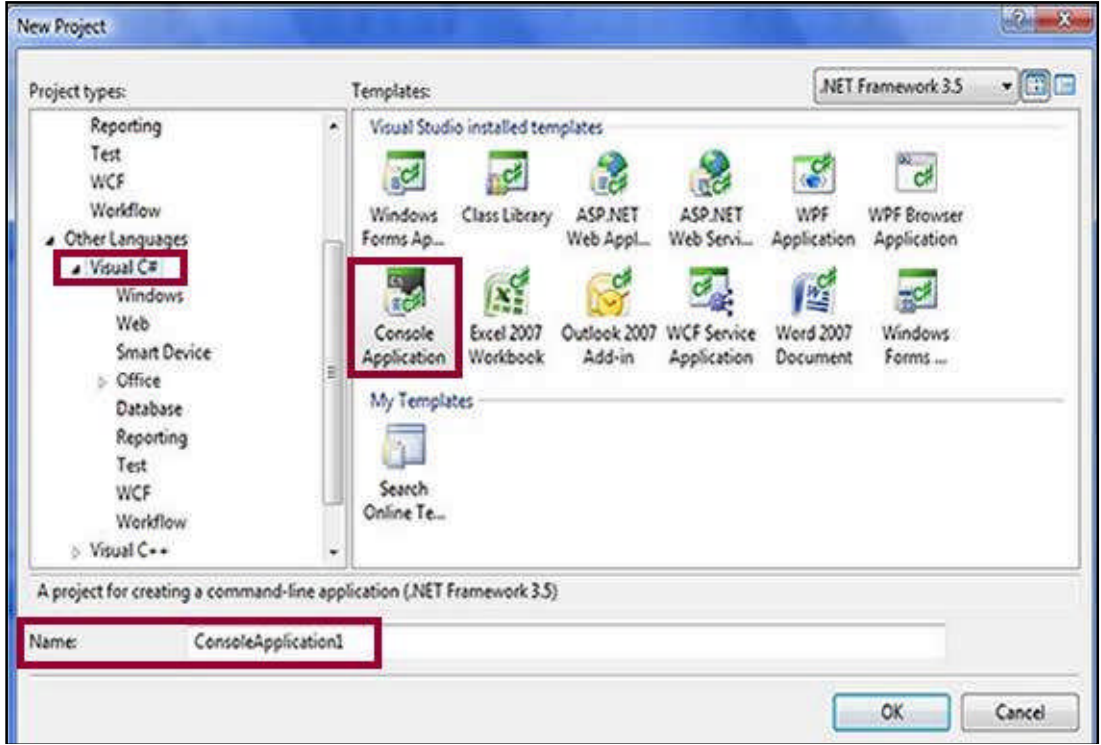
1.2. Konsol Uygulaması Oluşturma

Konsol uygulaması oluşturmak için program açıldıktan sonra Resim 1.2’de görülen “Recent Projects” bölümündeki “Create: Project” seçeneği tıklanabilir.



Resim 1.2: Recent Projects

Tıkladıktan sonra Resim 1.3’ te görülen New Project penceresi açılacaktır.



Resim 1.3: New Project penceresi

New Project penceresini açmak için farklı yollar da vardır. Bunlardan birisi, klavyeden Ctrl+N kısayol tuşunu kullanmak, diğeri ise menü çubuğundan “File-New Project” adımlarını takip etmektir.

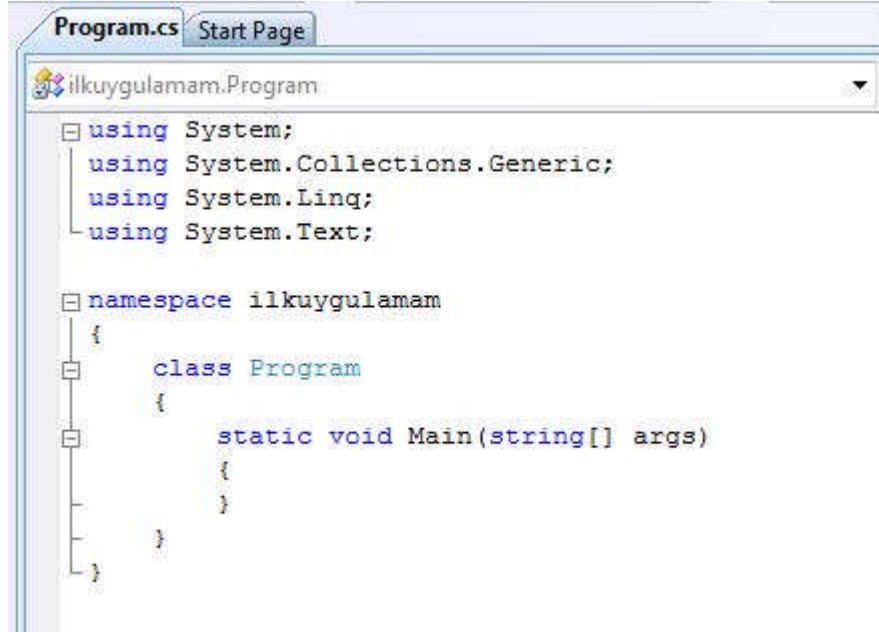
Artık ilk uygulama oluşturulabilir. Resim 1.3'te işaretlenmiş bölümlere dikkat edilmelidir.

Project Types: Projeye uygun programlama dilinin seçildiği bölümdür.

Templates: Proje oluşturmak için uygulama yazılımı ile birlikte gelen hazır şablonların bulunduğu bölümdür. Bu bölümden “Console Application” seçeneği seçilir.

Name: Uygulamalara isim vermek için kullanılan alandır. Bu bölüme ilk uygulama adını verdikten sonra “OK” butonuna tıklayıp uygulama oluşturuluyor.

Oluşan kod satırları Resim 1.4'te görülebilir.



```
Program.cs Start Page
ilkuygulamam.Program
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ilkuygulamam
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

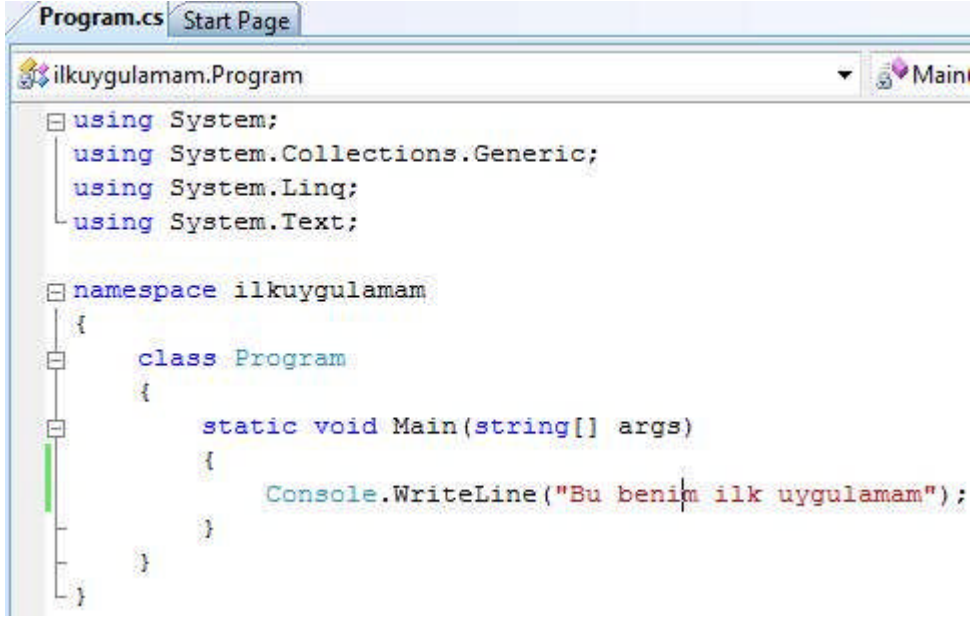
Resim 1.4: Uygulama ekranı

1.3. Konsol Ekranında Kod Yazma

Konsol uygulaması, grafiksel kullanıcı arayüzünden çok, komut satırı penceresinde çalışan uygulamadır.

UYGULAMA: Ekranı “Bu benim ilk uygulamam” mesajı yazan program kodları yazılsın.

Resim 1.5’te kodlar görülmektedir.




```
Program.cs Start Page
ilkuygulamam.Program
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ilkuygulamam
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Bu benim ilk uygulamam");
        }
    }
}
```

Resim 1.5: Uygulama kodları

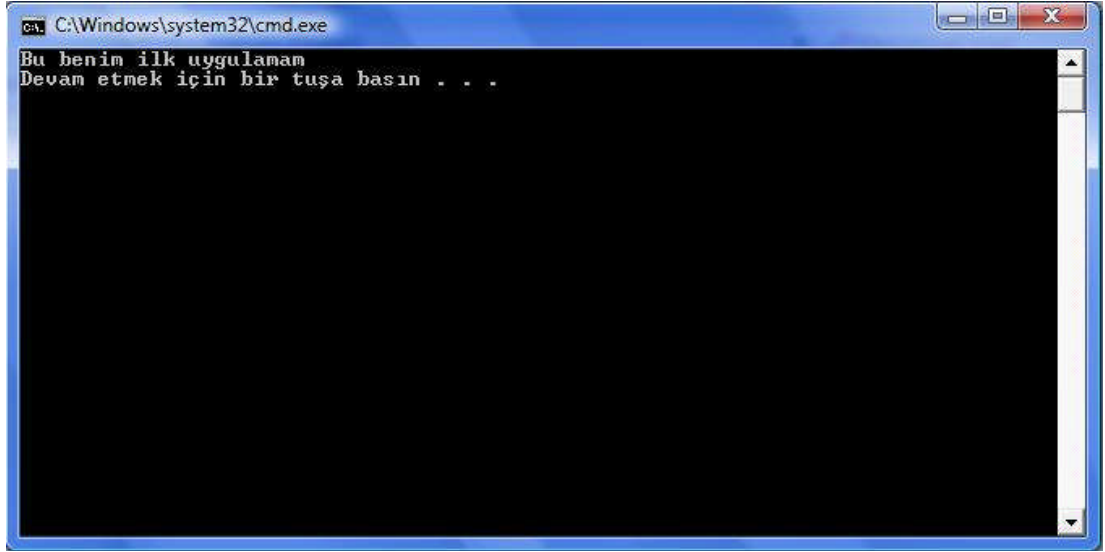
DİKKAT EDİLECEK NOKTALAR

- Program büyük-küçük harf duyarlıdır. Kodlar yazılırken bu durum göz önünde bulundurulmalıdır.
- Program kodu **static void Main** ifadesi altındaki küme parantezleri içerisine yazılmıştır.
- Kod satırının sonuna noktalı virgül(;) konulmuştur. Her bir ifade noktalı virgül ile bitirilmelidir. Aksi takdirde ifade derlenemeyecektir.
- **Console.WriteLine** ifadesi çift tırnak(“ ”) içine yazılan metni ekrana mesaj olarak yazar.

Yazılan kodlar test edilmelidir. Bunun için **F5** kısayol tuşu kullanılabilir ya da araç çubuklarındaki “**StartDebugging**” ()butonu veya menüden **Debug-Start Debugging** seçeneği de kullanılabilir.

Program çalışmaktadır. Fakat program çıktısını göremeden konsol ekranı kaybolmaktadır. Bunun için **Ctrl+F5** kısayol tuşu kullanılabilir veya yazılan kod satırının altına **Console.ReadLine();** ifadesi eklenebilir.

Program çıktısı Resim 1.6’ da görülmektedir.



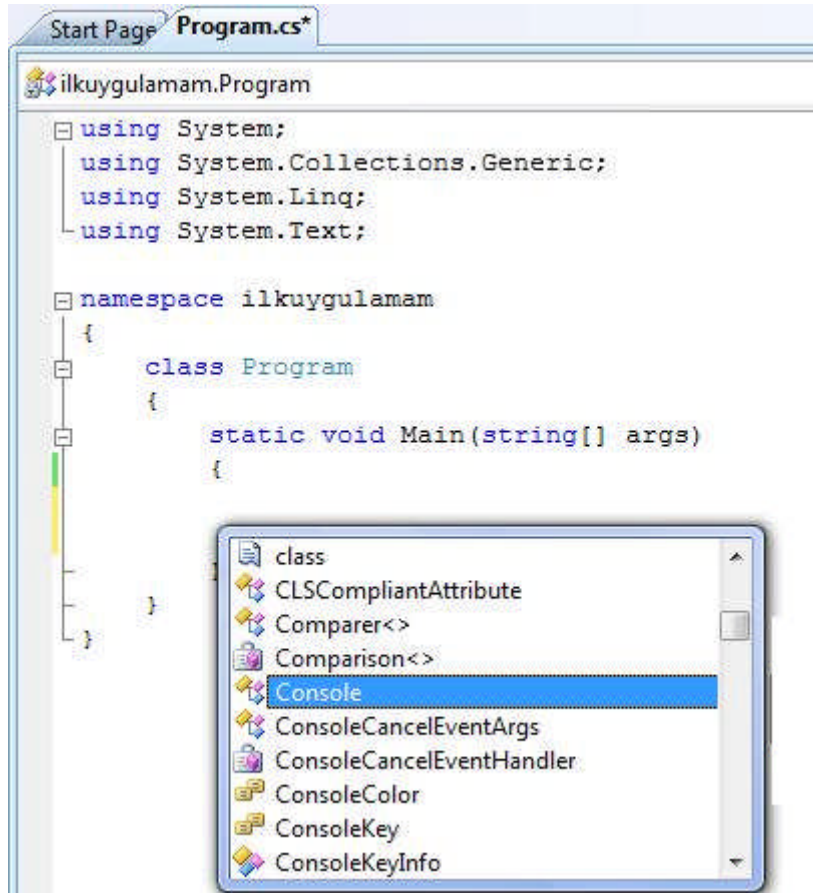
Resim 1.6: Program çıktı ekranı

➤ **IntelliSense kullanarak kod yazma**

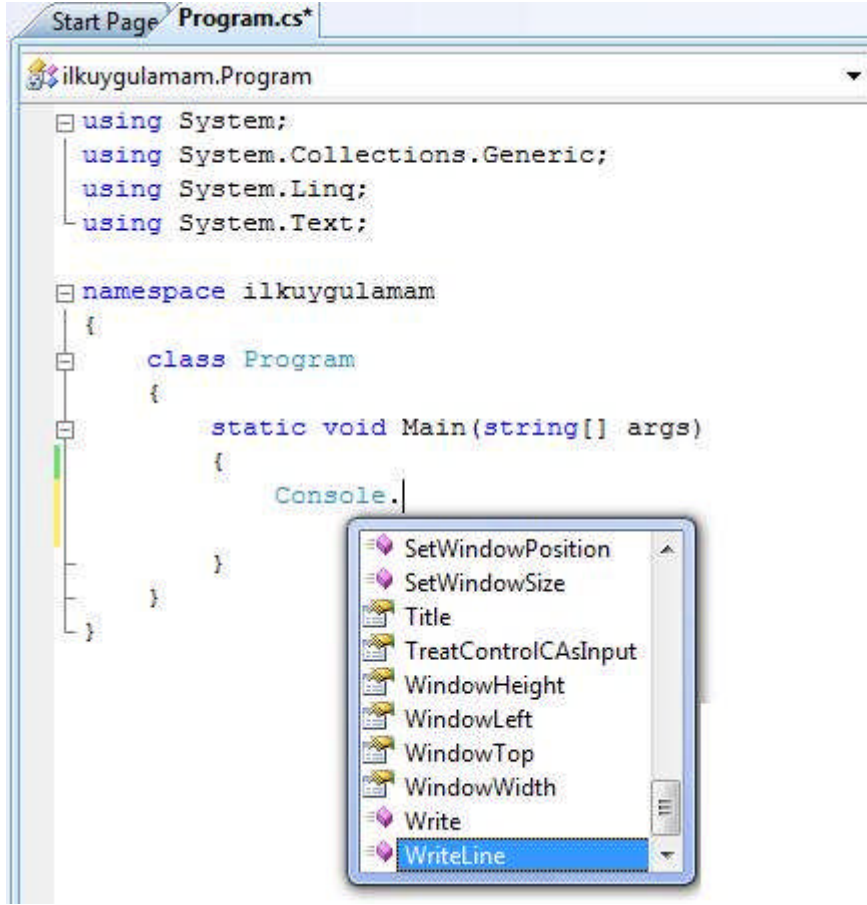
IntelliSense, nesne tabanlı programlama yazılımının kod tamamlama özelliğidir. Bu özellik daha hızlı ve hatasız kod yazma olanağı sağlar. Bir intellisense önerisini kabul etmek için açılan listeden ok tuşları yardımıyla seçtikten sonra Tab, Enter ya da boşluk tuşlarından birisine basılarak kullanılabilir. Bu işlem fare ile de yapılabilir.

NOT: Herhangi bir durumda intellisense öneri listesini görmek için **Ctrl + Boşluk** tuş kombinasyonu kullanılabilir.

Console.WriteLine ifadesini yazarken **Intellisense** özelliğinden yararlanarak nasıl yazıldığı Resim 1.7’de görülmektedir.



Resim 1.7: Intellisense kod tamamlama özelliği



Resim 1.8: Intellisense kod tamamlama özelliği

UYGULAMA FAALİYETİ

Nesne tabanlı programlama ortamını kullanınız.

İşlem Basamakları	Öneriler
➤ Yeni bir proje oluşturunuz.	➤ Ctrl+N, File-New Project ya da Recent Project seçeneklerini kullanabilirsiniz(bk. Resim 1.2).
➤ Konsol uygulamasını seçiniz.	➤ Uygulamayı oluştururken programlama dilini seçmeyi unutmayınız(bk. Resim 1.3).
➤ Program satırları içerisinde kodları yazacağınız küme parantezlerini bulunuz.	➤ Doğru parantezler içerisine yazılmazsa program çalışmaz(bk. Resim 1.5).
➤ Kodları yazarken programın intellisense özelliğini kullanınız.	➤ Kod yazarken size kolaylık sağlayacaktır(bk. Resim 1.7).
➤ Haftanın günlerini alt alta yazan program kodlarını yazınız. Her bir gün için ayrı kod satırı kullanınız.	➤ Console.WriteLine ifadesi yan yana yazacaktır. Alt alta yazmak için Console.Write ifadesini kullanınız.
➤ Yazdığınız kodları kontrol ederek satır sonlarına noktalı virgül (;) koyunuz.	➤ Noktalı virgül olmayan satırlarda program hata verecektir.
➤ Uygulamayı çalıştırınız.	➤ Ctrl+F5' i kullanabilirsiniz.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadıklarınız için **Hayır** kutucuklarına (X) işareti koyarak öğrendiklerinizi kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Yeni bir proje oluşturduunuz mu?		
2. Konsol uygulamasını seçtiniz mi?		
3. Uygun programlama dilini seçtiniz mi?		
4. Intellisense özelliğini kullandınız mı?		
5. Console.WriteLine ifadesini kullandınız mı?		
6. Console.Write ifadesini kullandınız mı?		
7. Start Debugging seçeneğini ve butonunu kullandınız mı?		
8. Ctrl+F5 kısayol tuşunu kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise “Ölçme ve Değerlendirme” ye geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

Nesne tabanlı programlama da isim uzaylarını kullanabilecek, grafik ekranda form tasarımı yapabilecek, form üzerine nesne ekleyebilecek ve bu nesnelere içerisine kod yazarak kullanıcı arayüzü oluşturabileceksiniz.

ARAŞTIRMA

- Programlama da kütüphane ve sınıf kavramlarını araştırınız.
- İsim uzayı kavramını araştırınız.
- Grafikselle arayüz kavramını araştırınız.

2. İSİM UZAYLARI

2.1. Namespace

Namespaceler(isim uzayları) nesneye dayalı, hiyerarşik ve birleşik, içinde çok sayıda sınıf, arayüz ve yapı bulundurabilen kütüphaneler olarak tanımlanabilir.

İsim uzayları program yazmanın en önemli parçasıdır ve daha çok yazılan kodların tekrar kullanılabilirliğini artırmak için geliştirilmiştir. Yazılan kodlarla program büyüdükçe iki sorun ortaya çıkar. Birincisi büyük programları anlamak ve bakımını yapmak küçük program parçacıklarına göre çok daha zordur. İkincisi ise daha fazla kod, ad, metod, sınıf demektir. Programda kullanılan ad sayısı arttıkça iki ya da daha fazla adın çakışma ihtimali, dolayısıyla projenin hata verme olasılığı da artar.

İsim uzayları(ad alanları) bu problemlerin çözümünde bize büyük kolaylıklar sağlarlar. Örneğin, aynı adla iki sınıf, farklı isim uzaylarında yer alıyorsa birbirine karıştırılmaz.

Örnek: Toplama ve bölme işlemleri yapan iki farklı ad alan (*toplama*, *bolme*) olsun. İki program parçacığının içinde de “*Islem*” adlı sınıf olsun.

```
namespace toplama
{
    class Islem
    {
        .....
    }
}

namespace bolme
{
    class Islem
    {
        .....
    }
}
```

Şimdi projenin içerisinde, farklı isim uzayları(ad alanları) içerisinde kullanılan aynı sınıfların, birbirine karıştırılmadan nasıl kullanıldığı görülsün.

“**toplama**” adlı isim uzayı içerisindeki “**Islem**” adlı sınıfı kullanmak için yazım şeklimiz,

toplama.Islem

“**bolme**” adlı isim uzayı içerisinde ki “**Islem**” adlı sınıfı kullanmak için yazım şeklimiz;

bolme.Islem

olmalıdır.

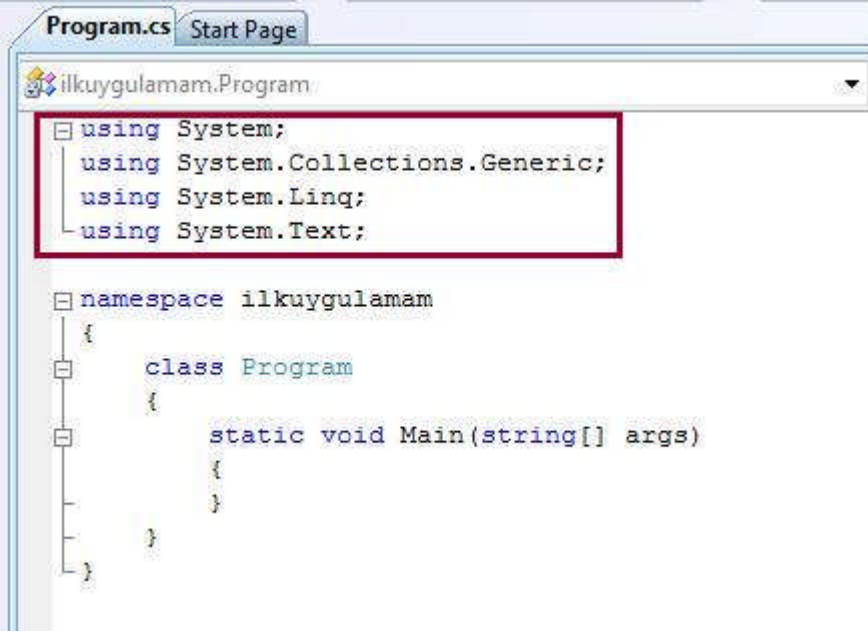
Görüldüğü gibi projenin herhangi bir yerinde toplama işlemi yapmak için “***toplama.Islem***” bölme işlemi yapmak için “***bolme.Islem***” ifadelerini kullanmak yeterli olacaktır. İki isim uzayı içerisindeki sınıf adları aynı olsa da işlevleri farklıdır ve bu durum nesne tabanlı programlama yazılımı tarafından ayırt edilebilmektedir.

2.2. Using İfadesi

Tüm sınıfların bir isim uzayı içerisinde yer almasının sağladığı kolaylıklardan, bir önceki konuda söz edilmişti. Nesne tabanlı programlama yazılımı ortamının etkin bir şekilde kullandığı yazılım geliştirme paketi .NET Platformu içinde her bir sınıf, bir ad alanı içinde yer alır. Örneğin, **Console** sınıfı **System** ad alanı içerisinde yer alır. Bu, sınıfın tam adının aslında **System.Console** olması demektir.

Ancak **Console** adlı sınıfın her kullanımında **System** önekinin eklenmesi isim uzayı(ad alanı) kavramının anlamsızlaşmasına yol açacaktır. Neyse ki bu problem programlarda “**using**” yönergesi kullanılarak çözülebilir.

Bir önceki uygulama faaliyetinde hazırlanan **ilkuygulamam** adlı projenin Program.cs dosyasındaki kodlara bakılacak olunursa en üstte Resim 2.1’deki işaretlenmiş ifadeler görülür.



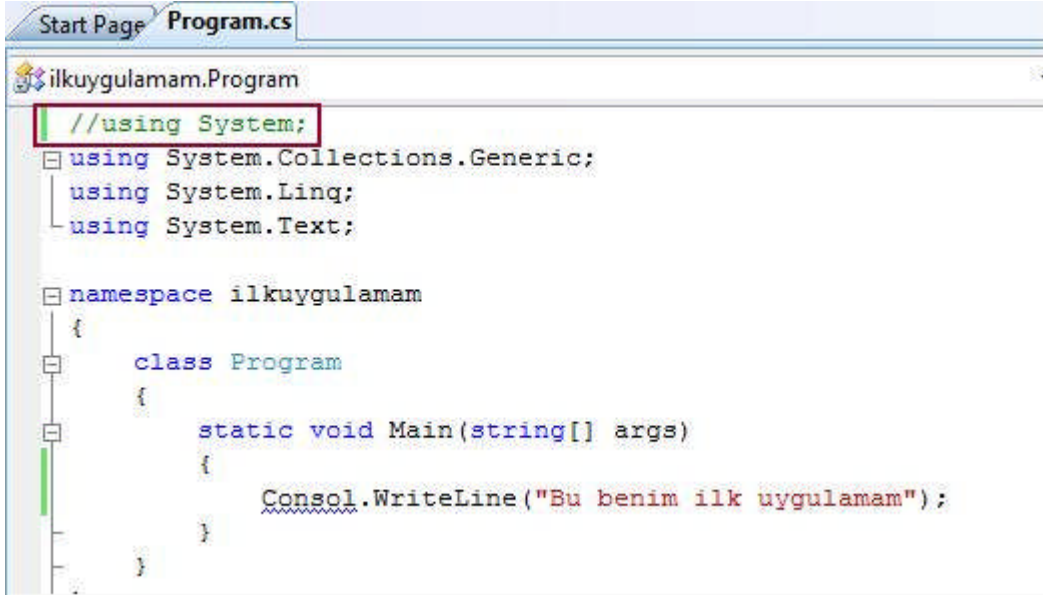
```
Program.cs Start Page
ilkuygulamam.Program
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ilkuygulamam
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Resim 2.1: Using İfadesi

Using ifadesi, kapsama ad alanı getirir yani kullanılacak nesnelere ve sınıfları, artık ait oldukları ad alanları ile nitelendirmek gerekmez. İşaretlenmiş alandaki dört ad alanı, her yeni proje oluşturulduğunda nesne tabanlı programlama yazılımı tarafından otomatik olarak eklenir ve oldukça sık kullanılan sınıfları kapsar.

Bunu daha iyi anlamak için aşağıdaki uygulama adım adım gerçekleştirilsin.

- Resim 2.2’de görüldüğü gibi Program.cs dosyasındaki ilk **using** yönergesini başına // eklenerek yorum satırı hâline getirilsin.



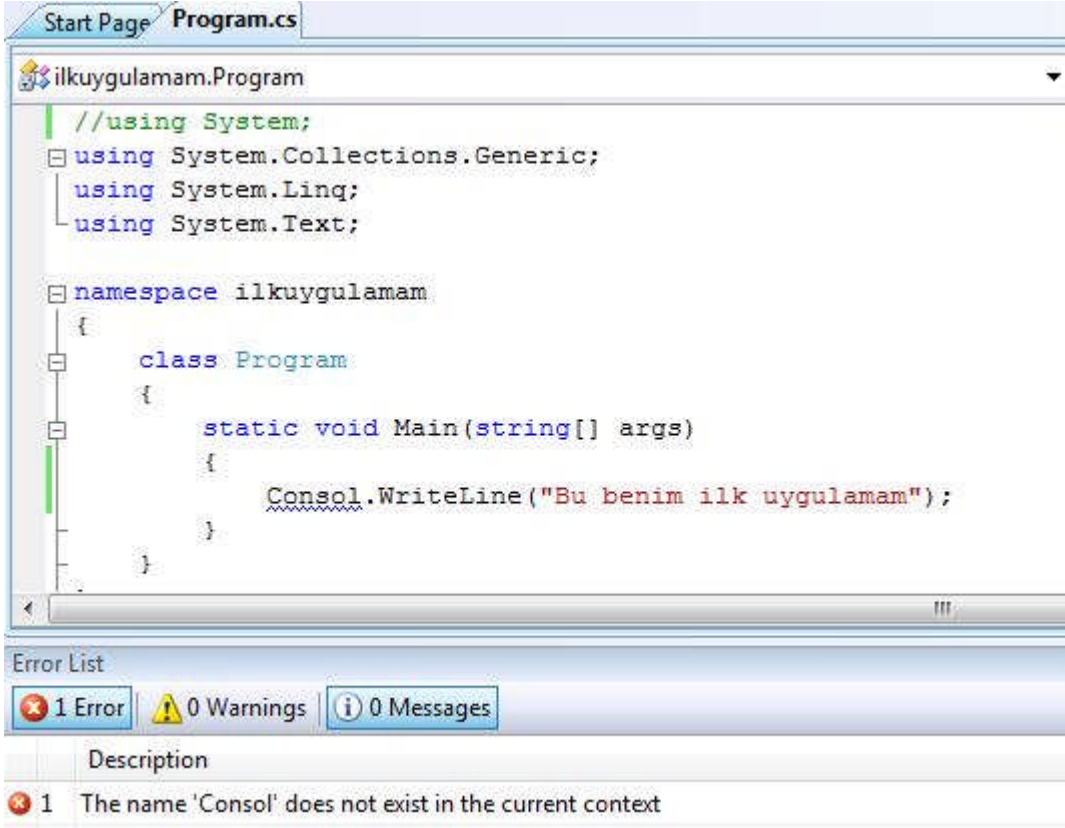
```
Start Page Program.cs
ilkuygulamam.Program
//using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ilkuygulamam
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Bu benim ilk uygulamam");
        }
    }
}
```

Resim 2.2: Using uygulaması

NOT: // ifadesi herhangi bir kod satırının başına eklendiğinde artık kod satırı devre dışı kalır ve program tarafından işletilmez. Bu satırlara artık yorum satırı adı verilir.

- Program F5 ile test edilsin. Resim 2.3'teki ekranla karşılaştırılsın.



Resim 2.3: Hata ekranı

Resim 2.3'te görüldüğü gibi **System** isim uzayı devre dışı bırakılıp yorum satırına dönüştürüldüğünde programın hata listesinde **Console** nesnesinin tanınmadığı hatası ile karşılaşılacaktır.

Bu sorunu çözmenin iki yolu vardır: Birincisi, yorum satırı hâline getirilen **using System** satırı, başındaki // ifadesi silinerek eski hâline getirmek ya da ekranda mesaj vermek için yazılan kod satırını, **System.Console.WriteLine** şeklinde yazmaktır.

Birinci yol her zaman tercih edilir. Çünkü **using** ifadesi ad alanındaki öğeleri kolayca programda kullanılabilir hâle getirir ve yazılan kodlarda sınıf adlarını uzun olarak yazmaya gerek kalmaz. Bu durum isim uzaylarının amacını açıklar.

2.3. Grafikselle Arayüz

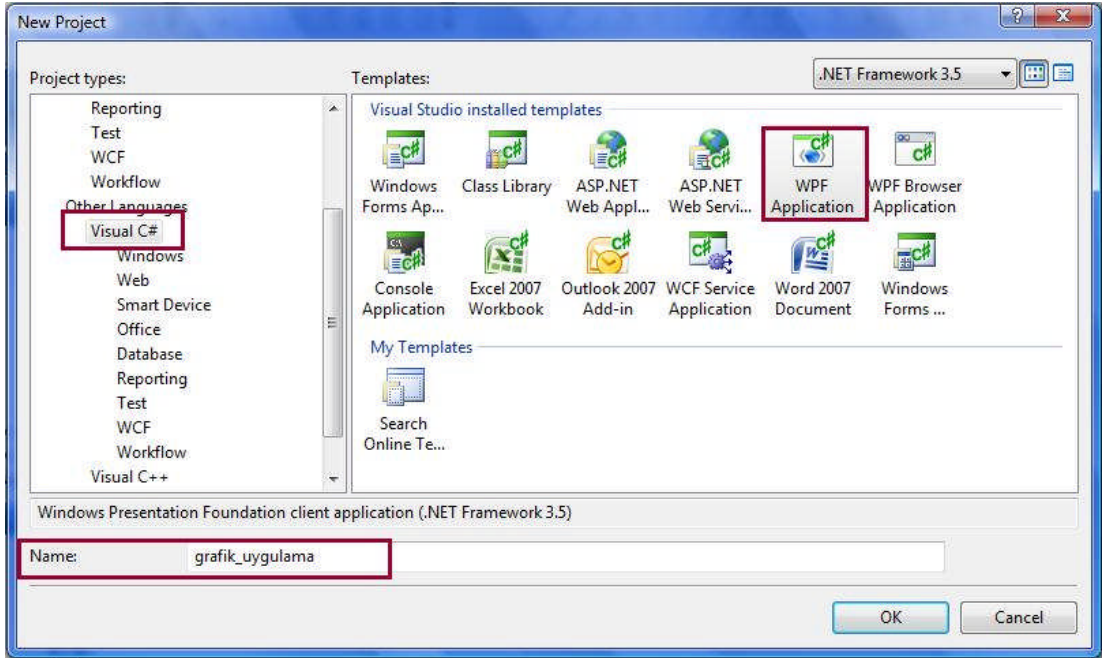
Şimdiye kadar basit konsol uygulaması oluşturuldu ve çalıştırıldı. Nesne tabanlı programlama yazılımı ortamı, aynı zamanda grafikselle windows tabanlı uygulamalar oluşturmak için ihtiyaç duyulan her şeyi kapsar. Windows tabanlı uygulamanın forma dayalı kullanıcı arayüzü etkileşimli olarak tasarlanabilir.

Nesne tabanlı programlama yazılımı grafikselle uygulamanın iki görünümünü sağlar: Tasarım görünümü ve kod görünümüdür.

Adım adım grafiksel bir uygulamanın nasıl oluşturulduğu görülsün. Oluşturulacak program, ad girilebilecek bir metin kutusu ve basıldığında ileti kutusunda bir selamlama iletisi gösterecek olan bir düğme içeren, basit bir form uygulaması olacaktır.

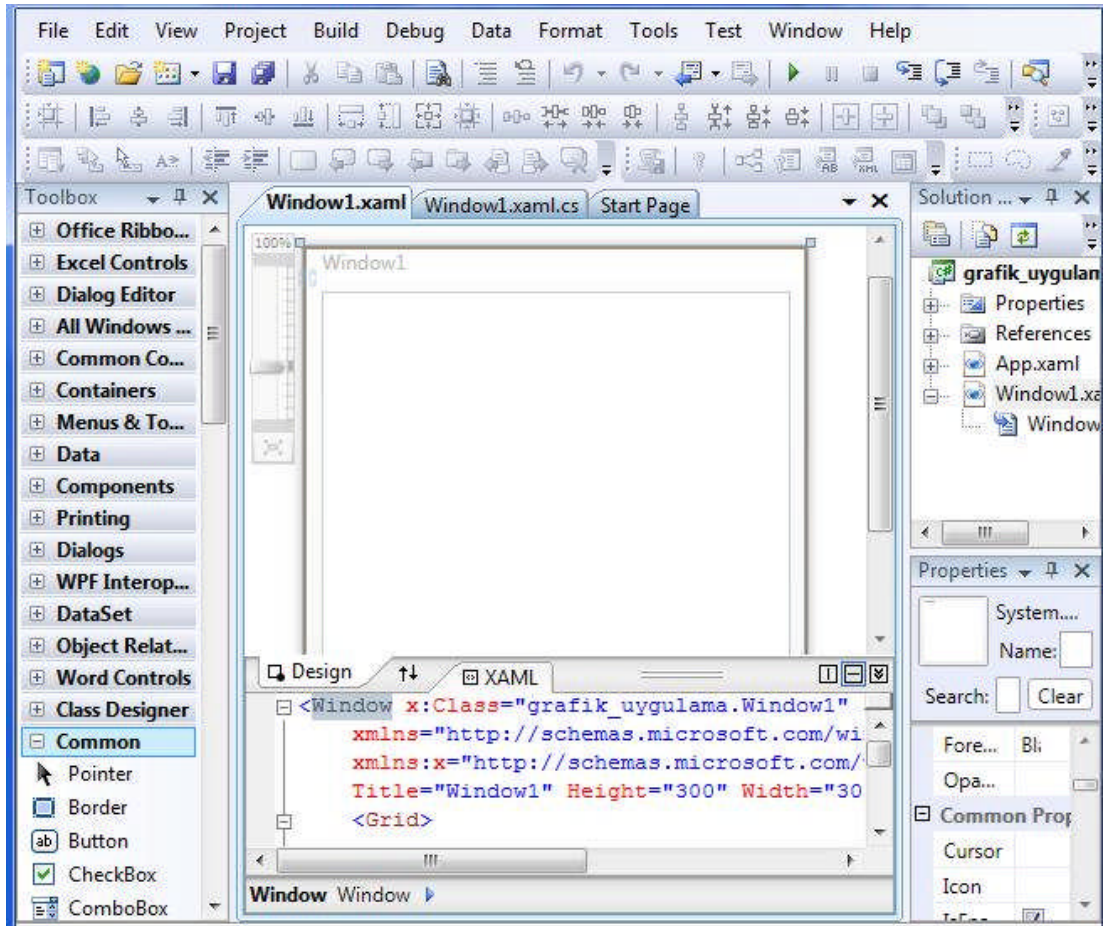
NOT: Nesne tabanlı programlama yazılımı grafiksel uygulamalar için iki şablon sağlar: Windows Form Application şablonu ve WPF Application şablonudur. WPF daha geliştirilmiş bir teknoloji olduğu için birçok ek özellik ve kapasite sağlar. Bu yüzden yeni yazılım geliştirmeleri için tercih edilir.

- “**File**” menüsünden “New Project” seçeneği seçiliyor. Açılan New Project penceresi daha önce de görülmüştü.
- Project types **ilgili programlama dili**, Templates ise **WPF Application** olarak seçilmelidir. Yeni projenin adını **grafik_uygulama** olarak verilsin. Ardından “OK” tıklansın. Resim 2.4’te seçimler işaretlenmiştir.



Resim 2.4: New Project penceresi

- Oluşturulan grafik uygulaması Resim 2.5’te görüntülenmektedir.



Resim 2.5: Grafik uygulaması

Resim 2.5'te görüldüğü gibi ekranda artık boş bir WPF formu bulunmaktadır. Formun alt tarafında ise “**Design**” ve “**XAML**” olmak üzere iki sekme mevcuttur.

Design sekmesinde formun grafiksel tasarımı kolaylıkla yapılır ve ilgili kodlar nesne tabanlı programlama yazılımı tarafından otomatik olarak oluşturulur.

XAML ise “Genişletilebilir Uygulama İşaretleme Dili” (Extensible Application Markup Language) anlamına gelir ve XML benzeri bir dildir. Oluşturulan forma ve daha sonra form üzerine yerleştirilecek nesnelere XAML ile müdahale edilebilir. İstenilen değişiklikler kodlarla da yapılabilir.

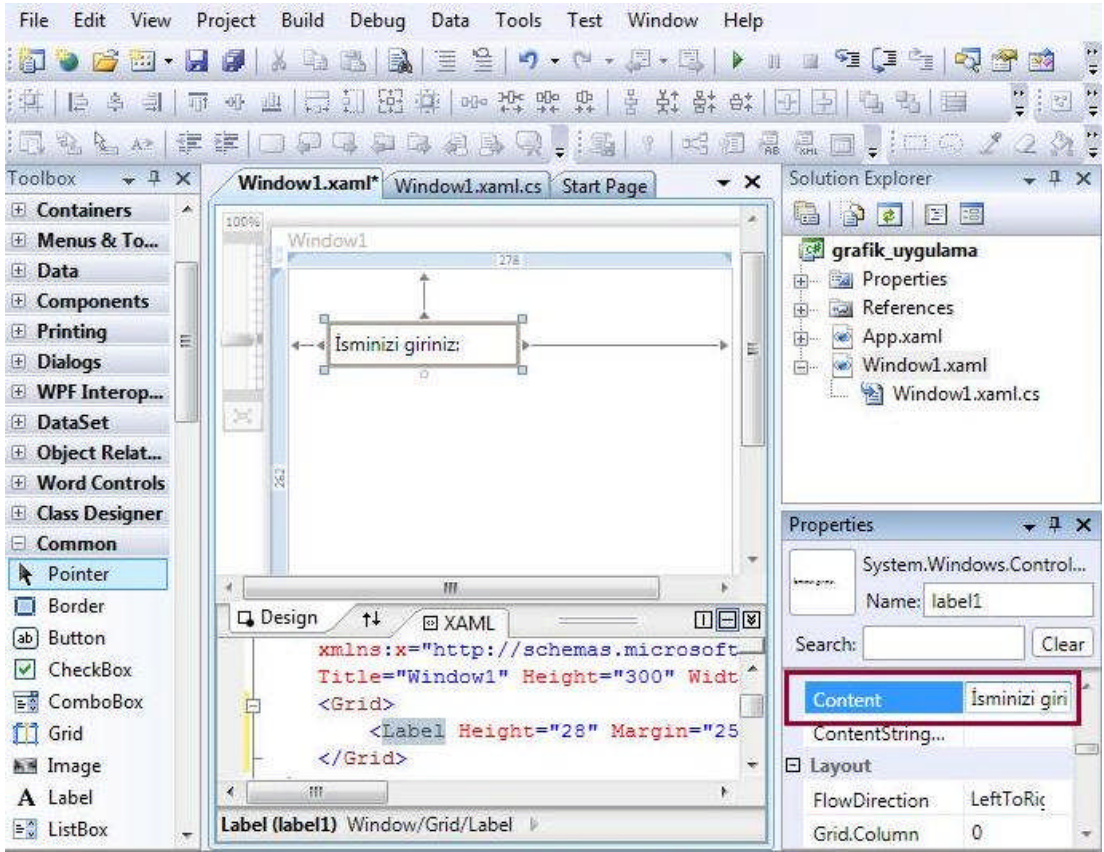
Programın sol yanın da bulunan “**Toolbox**” formda ve ileri uygulamalarda kullanılacak nesnelere sağlar (Buton , Label , TextBox vb).

Sağ alt köşede bulunan “**Properties**” penceresi ise seçili olan herhangi bir nesnenin gelişmiş özelliklerini gösterir ve değiştirmeyi sağlar(Form , Label, TextBox , Buton vb.).

2.4. Nesne Ekleme

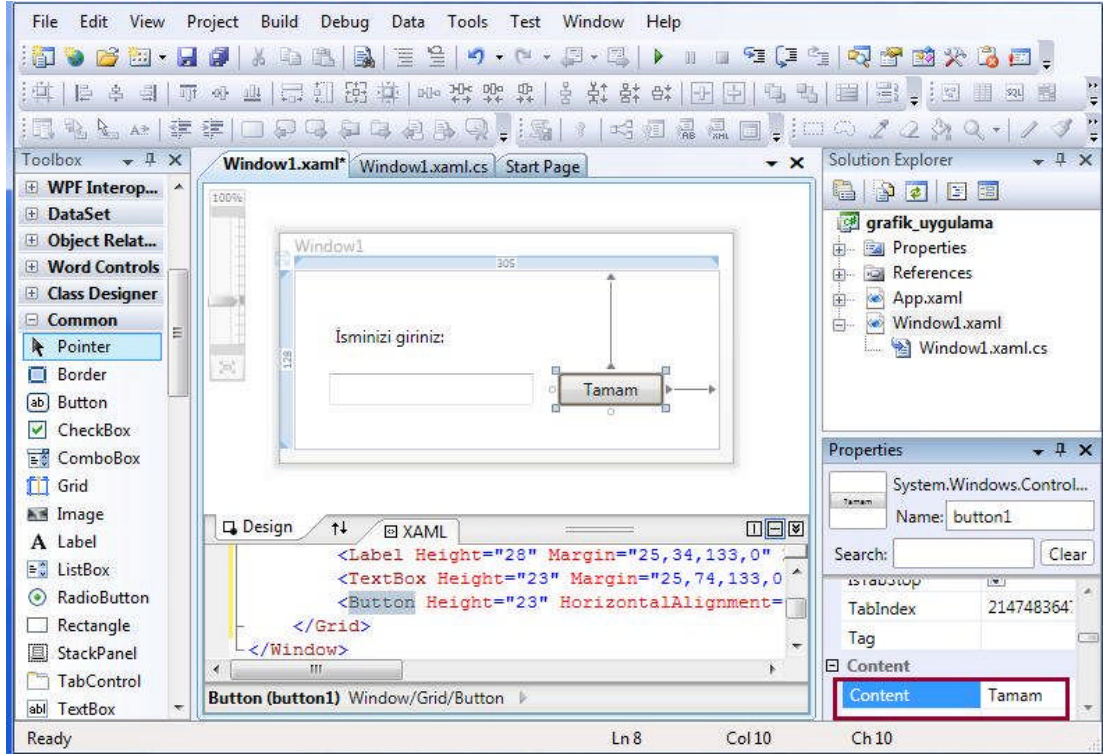
Oluşturulan **grafik_uygulama** isimli projeyi kullanarak isim girilen bir metin kutusu ile basıldığında ileti kutusunda selamlama iletisi gösterilecek bir düğme içeren arayüz oluşturulacak.

- Programın sol tarafında bulunan Toolbox sekmesinin Common bölümündeki Label nesnesine tıklanılıp formun herhangi bir yerine tıklanarak “Label” nesnesini eklensin. Resim 2.6’da görüldüğü gibi forma eklenen “Label” nesnesi seçiliyken Properties sekmesindeki Content yazan yere şu mesaj yazılsın. **“İsminizi giriniz:”**



Resim 2.6: Properties sekmesi ve Label nesnesi özellikleri

- Toolbox sekmesinin Common bölümündeki TextBox nesnesine tıklanıp daha önce forma eklenen Label nesnesinin alt tarafına tıklanarak forma eklensin.
- Toolbox sekmesinin Common bölümündeki Button nesnesine tıklanıp formdaki TextBox nesnesinin yan tarafına tıklanarak buton eklensin. Resim 2.7’de görüldüğü gibi Buton nesnesi seçiliyken Properties sekmesinin Content bölümüne “Tamam” yazıp buton üzerindeki metin değiştirilsin.

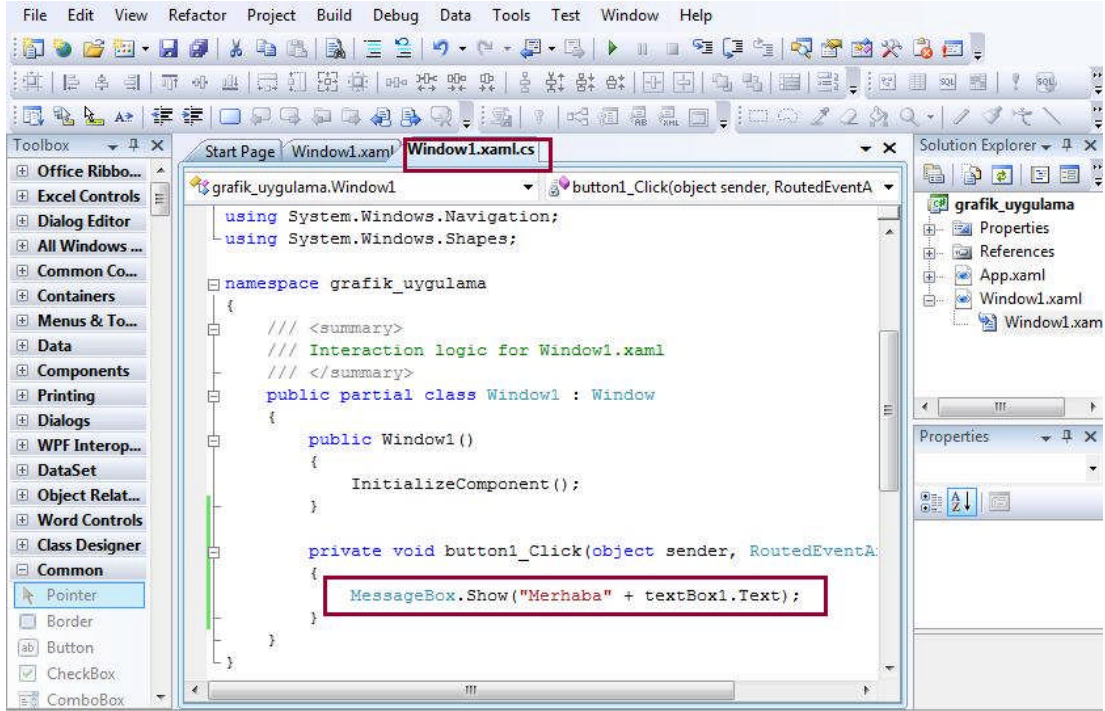


Resim 2.7: Properties sekmesi ve Button nesnesi özellikleri

Oluşturulan form tasarımı Resim 2.7’de görülmektedir.

2.5. Nesneye Kod Yazma

- Eklenen buton nesnesi üzerine çift tıklandığında Resim 2.8’de görülen **Window1.xaml.cs** dosyası açılacaktır. Oluşturulan grafiksel form uygulamasında bulunan tüm nesnelere ait kodlar bu dosya içerisinde tutulur.



Resim 2.8: Window1.xaml.cs ile buton nesnesinin Click özelliği

- Resim 2.8’de işaretlenmiş olarak görülen kod satırı ilgili yere yazılısın.

MessageBox.Show("Merhaba" + textBox1.Text);

Bu, kullanıcı “Tamam” butonuna tıkladığında çalışacak koddur. “**MessageBox.Show**” ifadesinin, “Merhaba” metni ile birlikte, kullanıcının forma girdiği adı içeren ileti kutusunu görüntüleyecektir.

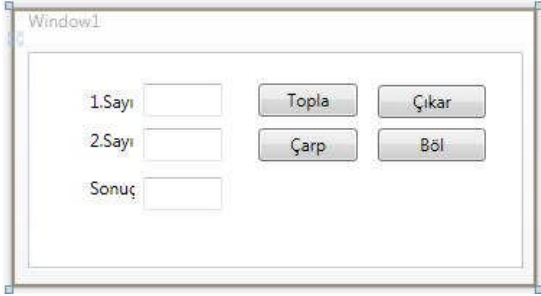
- Artık program **F5** tuşu ile test edilebilir. Program çıktısı Resim 2.9’da görüldüğü gibi olmalıdır. Metin kutusu içerisine isim yazıp “Tamam” butonuna tıkladığında ekranda mesaj iletisi görüntülenecektir.



Resim 2.9: Program çıktısı

UYGULAMA FAALİYETİ

Kod içerisinde isim uzaylarını(namespace) kullanınız.

İşlem Basamakları	Öneriler
➤ Yeni bir proje oluşturunuz.	➤ Ctrl+N, File-New Project ya da Recent Project seçeneklerini kullanabilirsiniz.
➤ WPF Application'ı seçiniz. Name bölümüne "hesapmakinesi" adını veriniz.	➤ Uygulamayı oluştururken uygun programlama dilini seçmeyi unutmayınız.
➤ Oluşan forma Label nesnesi ekleyiniz. Adını 1.Sayı olarak değiştiriniz.	➤ Properties panelindeki Content seçeneğini kullanınız.
➤ Forma Label nesnesi ekleyiniz. Adını 2.Sayı olarak değiştiriniz.	➤ Properties panelindeki Content seçeneğini kullanınız.
➤ Forma Label nesnesi ekleyiniz. Adını Sonuç olarak değiştiriniz.	➤ Properties panelindeki Content seçeneğini kullanınız.
➤ Forma üç adet TextBox nesnesi ekleyiniz. Her birini daha önce eklediğiniz Label nesnelerinin karşısına yerleştiriniz.	➤ ToolBox panelini kullanarak TextBox nesnelerini ekleyiniz.
➤ Forma Button nesnesi ekleyiniz. Adını Topla olarak değiştiriniz.	➤ Properties panelindeki Content seçeneğini kullanınız.
➤ Forma Button nesnesi ekleyiniz. Adını Çıkar olarak değiştiriniz.	➤ Properties panelindeki Content seçeneğini kullanınız.
➤ Forma Button nesnesi ekleyiniz. Adını Çarp olarak değiştiriniz.	➤ Properties panelindeki Content seçeneğini kullanınız.
➤ Forma Button nesnesi ekleyiniz. Adını Böl olarak değiştiriniz.	➤ Properties panelindeki Content seçeneğini kullanınız.
➤ Oluşturduğunuz form tasarımını resimdeki haliyle kontrol ediniz.	
➤ Projenizi 4.öğrenme faaliyetinde kullanmak üzere kaydediniz.	➤ Ctrl+Shift+S kısayol tuşunu ya da File-Save All seçeneğini kullanınız. Projenizi 4.öğrenme faaliyetinin uygulama faaliyetinde kullanmak üzere saklayınız.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadıklarınız için **Hayır** kutucuklarına (X) işareti koyarak öğrendiklerinizi kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Grafik uygulaması oluşturduunuz mu?		
2. Forma Label nesnesi eklediniz mi?		
3. Forma Button nesnesi eklediniz mi?		
4. Forma textBox nesnesi eklediniz mi?		
5. Properties panelini kullandınız mı?		
6. Label nesnesinin Content özelliğini değiştirdiniz mi?		
7. Button nesnesinin Content özelliğini değiştirdiniz mi?		
8. textBox nesnesinin Content özelliğini değiştirdiniz mi?		
9. Button nesnesinin “Click” özelliğine kod yazdınız mı?		
10. Design penceresini kullandınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise Ölçme ve Değerlendirme’ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlede boş bırakılan yere getirilecek bilginin bulunduğu seçeneği işaretleyiniz.

1. “..... nesneye dayalı, hiyerarşik ve birleşik, içinde çok sayıda sınıf, arayüz ve yapı bulundurabilen kütüphaneler olarak tanımlanabilir.”
A) System
B) Namespace
C) Grafik arayüz
D) Konsol uygulamaları

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

2. Aşağıdakilerden hangisi bir kod satırının başına eklendiğinde kod satırını yorum satırına dönüştürür?
A) { }
B) []
C) ” ”
D) //
3. Aşağıdakilerden hangisi görsel tasarım yapılan grafiksel arayüz ifadesinin karşılığıdır?
A) Console Application
B) WCF Service Application
C) WPF Application
D) Class Library
4. Üzerinde, form tasarımında kullanılacak nesnelere bulunduran panel aşağıdakilerden hangisidir?
A) ToolBox
B) Properties
C) Solution Explorer
D) Start Page
5. Form tasarımında kullanılan nesnelere ait özelliklerin değiştirildiği panel aşağıdakilerden hangisidir?
A) ToolBox
B) Properties
C) Solution Explorer
D) Start Page
6. Form tasarımında kullanılan nesnelere ait kodlar aşağıdaki dosyalardan hangisinde tutulur?
A) App.xaml
B) Program.cs
C) App.xml
D) Window1.xaml.cs

7. Aşağıdaki ifadelerden hangisi parantez içerisinde yazılan metni ekranda ileti olarak gösterir?
A) MsgBox.Show
B) MessageBox
C) MessageBox.Show
D) MsgBox
8. Aşağıdaki nesnelere hangisinin “Click” özelliği vardır?
A) Label
B) Button
C) TextBox
D) Form

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-3

AMAÇ

Değişken tanımlama kurallarını öğrenerek değişken tanımlayabilecek ve veri türlerini kullanabileceksiniz.

ARAŞTIRMA

- Veri kavramını araştırarak günlük hayatta kullanılan veri türlerini yazınız.
- Programlamada değişken kavramını araştırınız.

3. DEĞİŞKENLER VE VERİ TÜRLERİ

3.1. Değişken Tanımlama Kuralları

Değişken tanımlarken (Aynı zamanda ad alanları, yöntemler ve sınıflar tanımlanırken de bu kurallara uyulmalıdır.) aşağıdaki söz dizimi kurallarına uyulmalıdır.

- Değişken adlarında yalnızca harf (büyük ve küçük), rakam ve alt çizgi(_) karakteri kullanılabilir, boşluk içermez.
- Değişken adı, bir harf ile (Alt çizgi harf olarak kabul edilir.) başlamak zorundadır, rakamlarla başlayamaz.
- Türkçe karakterler kullanılabilir ama önerilmez(ç,ğ,ş,ö,ü,ı,i vb.).
- Programlama dili bazı tanımlayıcı kelimeleri kendisi kullanmak için ayırmıştır. Anahtar sözcükler olarak adlandırılır ve her biri özel bir anlama sahiptir(class, enum, base, const, decimal vb.).

Geçerli tanımlama biçimleri: Soyad, _fiyat, dogumTarihi, Okul_Nu., Adres2

Geçersiz tanımlama biçimleri: Ucret%, fiyat\$, 2.Adres, class

NOT: Bazı programlama dilleri büyük küçük harfe duyarlıdır. Örneğin C# dilinde dogumTarihi ile DogumTarihi aynı anlama gelen isimler değildir.

3.2. Değişken Tanımlama

Değişkenler değerleri tutar. Programlama dili depolayabileceği ve işleyebileceği çok sayıda farklı değer türüne sahiptir(Tam sayı, kayan nokta sayısı ve karakterler dizesi gibi). Bir değişken bildirildiğinde içinde ne tür bir değişken tutacağı da belirtilmelidir.

Örneğin, aşağıdaki ifade “*yas*” adında ve “*int*” (tamsayı) değerler taşıyan bir değişkeni bildirir. İfade sonunda noktalı virgül yer alır.

```
int yas;
```

Değişken bildirdikten sonra değişkene değer atanabilir. Aşağıdaki deyim *yas* değişkenine 30 değerini atar.

```
yas = 30;
```

Ayrıca, bildirim ve atama işlemleri aynı anda yapılabilir.

```
int yas = 30;
```

Eşittir işareti(=), sağ taraftaki değeri, sol taraftaki değişkene atayan bir **atama** işlecidir. Bu atamadan sonra **yas** değişkeni tuttuğu değere başvurmak için kod da kullanılabilir.

Aşağıdaki deyim, konsola **yas** değişkeninin değeri olarak 30 yazar:

```
Console.WriteLine(yas);
```

İPUCU: Nesne tabanlı programlama yazılımı kod düzenleme penceresinde fare işaretçisini bir değişkenin üzerinde tutarsanız değişkenin türünü gösteren bir ekran ipucu belirir.

3.3. Temel Veri Türleri

Programlama dili temel veri türleri olarak adlandırılan birkaç yerleşik türe sahiptir. Aşağıdaki tabloda, kullanılan programlama dilinde en çok kullanılan temel veri türleri ve her birinde depolanabilecek değer oranı listelenmiştir.

Veri Türü	Açıklama	Boyut(bit)	Aralık	Örnek Kullanım
int	Tam sayılar	32	-2^{31} 'den $2^{31}-1$ ' e kadar	int numara; numara = 30;
long	Tam sayılar (daha büyük aralık)	64	-2^{63} 'den $2^{63}-1$ ' e kadar	long saniye; saniye = 22L;
float	Kayan noktalı sayılar	32	$\pm 1,5 \times 10^{45}$,ten $\pm 3,4 \times 10^{38}$, e kadar	float olcum; olcum = 0.52F;
double	İki kat daha hassas(daha fazla kesin) kayan noktalı sayılar	64	$\pm 5,0 \times 10^{324}$,ten $\pm 1,7 \times 10^{308}$, e kadar	double miktar; miktar = 0.15;
decimal	Parasal değerler	128	28 basamak	decimal fiyat; fiyat = 25M;
string	Karakter sıraları	Karakter başına 16 bit	Geçersiz	string ad; ad = "Ali";
char	Tek karakter	16	0'dan $2^{16}-1$ 'e kadar	char secim; secim = '*';
bool	Boolean	8	Doğru yada Yanlış	Bool tercih; tercih = true;

Tablo 3.1: Sık kullanılan veri türleri

Atanmamış Yerel Değişkenler

Bir değişken bildirildiğinde, değişken değer atanana kadar rastgele bir değer içerir. Bu, bir değişkenin oluşturulduğu ve değişkene bir değer verilmeden önce yanlışlıkla bilgi kaynağı olarak kullanıldığı, bazı programlama dillerinde zengin bir hata kaynağıydı. Nesne tabanlı programlama dili, değer atanmamış değişken kullanılmasına izin vermez. Bir değişken kullanılmadan önce, değişkene bir değer atanmak zorunluluğu vardır. Aksi takdirde program derlenmeyebilir. Örneğin, aşağıdaki ifadeler yas değişkenine değer atanmadığı için derleme hatası verecektir.

```
int yas;
Console.WriteLine(yas); //derleme zamanı hatası
```

NOT: .NET Platform yazılımı, dizeler olarak tutulan değerler üzerinde aritmetik hesaplamalar yapmak gerekirse dize değerini tam sayı değerine dönüştürmek için kullanılacak *Int32.Parse* adlı bir yöntem sağlar. Örneğin,

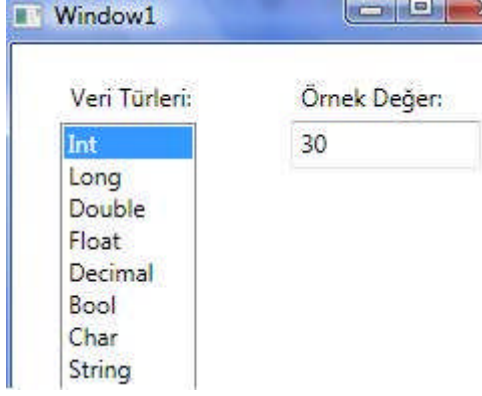
```
string a = "2";
```

ifadesindeki *a* değişkeni *string*(dize) olarak tanımlanmıştır. Bu değeri matematiksel işlemlerde kullanılamaz. Bunun için değişkenin değerini *int* türüne çevirmek gerekir.

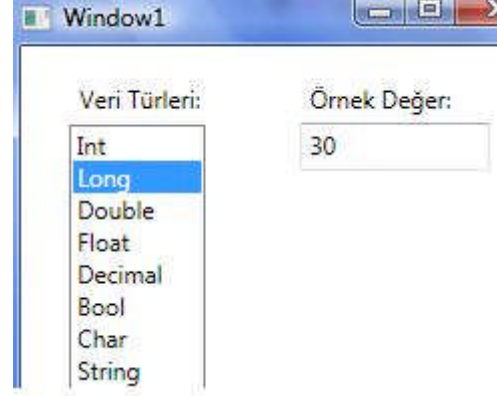
```
a*50; ifadesi hata verecektir.
```

```
int32.Parse(a)*50; şeklinde yazılabilir.
```

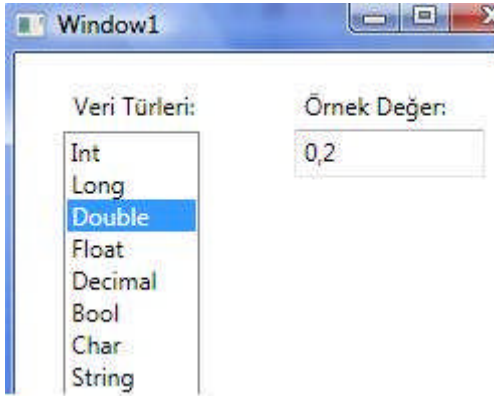
Veri türlerini, bu türlerde değişkenlerin nasıl tanımlandığı ve kod yazımları hazırlanan bir uygulama yardımıyla görüntülensin.



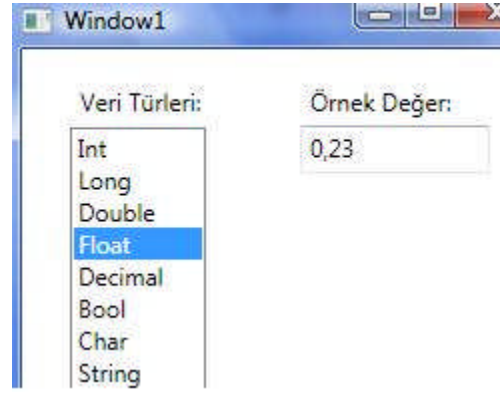
Resim 3.1: (a) Int veri türü ve örnek değer



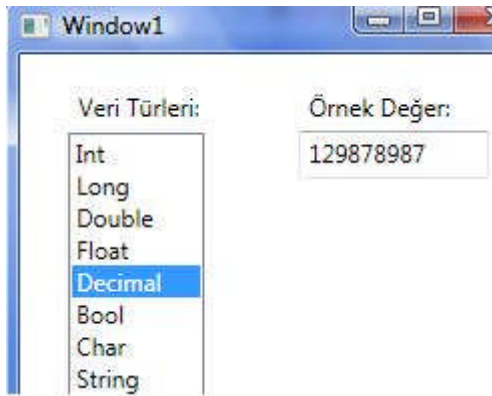
(b) Long veri türü ve örnek değer



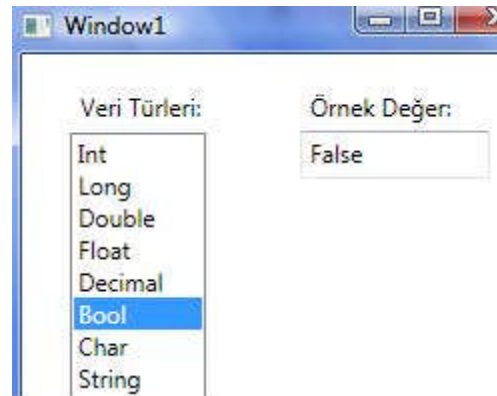
(c) Double veri türü ve örnek değer



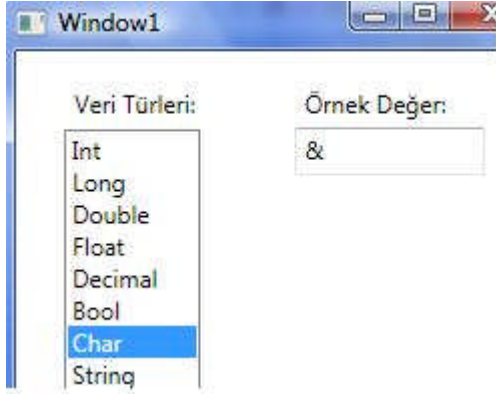
(d) Float veri türü ve örnek değer



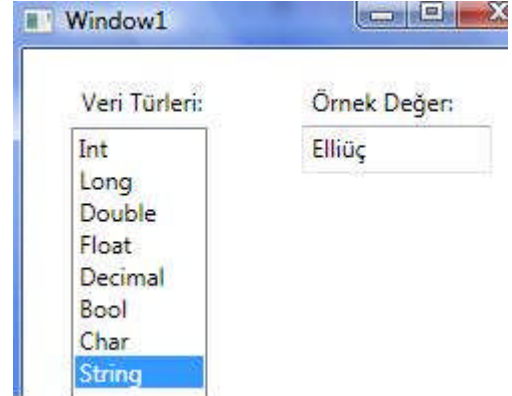
(e) Decimal veri türü ve örnek değer



(f) Bool veri türü ve örnek değer



(g) Char veri türü ve örnek değer



(h) String veri türü ve örnek değer

Resim 3.1'de görülen uygulamaya ait kodlar aşağıda görüntülenmiştir. Kodları incelerken görülecek ToString metodu sayı türü değişkenlerin metin türüne dönüştürülmesi için kullanılır. textBox1 nesnesinin Text özelliği metin türü değişken saklayabileceği için bu metoda başvurulmuştur.

```

private void intgoster()
{   int i = 30;
    textBox1.Text = i.ToString();
}
private void Longgoster()
{   long saniye = 30L;
    textBox1.Text = saniye.ToString();
}
private void Doublegoster()
{   double i = 0.20;
    textBox1.Text = i.ToString();
}
private void Floatgoster()
{   float i = 0.23F;
    textBox1.Text = i.ToString();
}
private void Decimalgoster()
{   decimal i = 129878987M;
    textBox1.Text = i.ToString();
}
private void Boolgoster()
{   bool i = false;
    textBox1.Text = i.ToString();
}
private void Chargoster()
{   char i = '&';
    textBox1.Text = i.ToString(); ;
}
private void Stringgoster()
{   string i = "Elliüç";
    textBox1.Text = i;|
}

```

Resim 3.2: Uygulamaya ait kodlar

UYGULAMA FAALİYETİ

Nesne tabanlı programlama ortamında değişkenleri kullanınız.

İşlem Basamakları	Öneriler
<p>➤ Yeni bir proje oluşturunuz.</p>	<p>➤ File-NewProject komutunu kullanınız.</p>
<p>➤ Oluşan formunuzun üzerine TextBox nesnesi ekleyiniz.</p>	<p>➤ Sol taraftaki Toolbox'ı kullanınız.</p>
<p>➤ Form Load kısmına</p> <pre>privatevoid Form1_Load(object sender, EventArgs e) { int i = 30; textBox1.Text = i.ToString(); }</pre> <p>kodunu yazınız.</p>	<p>➤ Properties penceresini kullanınız.</p>
<p>➤ Değişken türünü ve değerini değiştirerek uygulamayı çalıştırınız.</p>	<p>➤ Parantezler ve noktalama işaretlerine dikkat ediniz.</p>

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. () Değişken tanımlarken, değişken adının başında olmamak şartıyla rakam kullanılabilir.
2. () Herhangi bir değişken **Veri_turleri** şeklinde adlandırılabilir.
3. () **yuzde%** adlı bir değişken tanımlanabilir.
4. () Programda **int** adlı bir değişken tanımlanabilir.
5. () `int a = 30*5;` şeklinde bir tanımlama yapılabilir.
6. () Değer atanmamış herhangi bir değişken kodlar içerisinde kullanılabilir.
7. () `float sayac = textBox1.Text;` ifadesi doğru bir tanımlamadır.
8. () `Int32.Parse` ifadesi bir dize değerini **int** değerine dönüştürmeyi sağlar.
9. () `Tostring` ifadesi rakamsal bir değeri string değere dönüştürmek için kullanılır.
10. () **char** tipi değişkenler 16 karaktere kadar veri saklayabilir.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ-4

AMAÇ

Aritmetiksel operatörleri kullanabileceksiniz.

ARAŞTIRMA

- Matematiksel işlemlerde işlem önceliklerini araştırınız.

4. OPERATÖRLER

Programlama dillerinde tanımlanmış sabit ve değişkenler üzerinde işlemler yapmayı sağlayan karakter ya da karakter topluluklarına **operatör** denir.

4.1. Aritmetiksel Operatörler

Programlama dili bilinen dört işlemin aritmetik işlemlerini destekler: *Toplama(+)*, *çıkarma(-)*, *çarpma(*)* *bölme(/)*.+, -, * ve / işaretleri dört işlemi temsil ettiğinden *işlemler* olarak adlandırılır.

Örnek:

```
long ucret;  
ucret = 50*30;
```

NOT: İşlemin üzerinde işlem yaptığı değerler işlenen(operands) olarak adlandırılır. * işleç, 50 ve 30'da işlenendir.

4.2. İşlemler ve Türler

Tüm işlemler tüm veri türlerine uygulanamaz. Bir değer üzerinde kullanılacak işlemler, değer türüne bağlıdır. Örneğin, *char*, *int*, *long*, *float*, *double*, ya da *decimal* veri türlerinde tüm aritmetik işlemleri kullanılabilir. Bununla birlikte *string* ya da *bool* türü değişkenler üzerinde artı(+) işleci dışında aritmetik işlemler kullanılamaz.

Bu nedenle *string* türü çıkarma işlemini desteklemediğinden aşağıdaki ifade derlenemez(Bir karakter kümesinin diğer bir karakter kümesinden çıkarılması anlamsızdır.).

```
//derleme zamanı hatası
```

```
Console.WriteLine("İstanbul"- "Ankara");
```


string türü değerler üzerinde + işleci kullanıldığında dize değerleri birleştirir. Aşağıdaki ifade ekrana “578” (“65 “ değil) yazar.

```
Console.WriteLine(“57”+”8”);
```

Aritmetik işlem sonucundaki değer türü, kullanılan işlenenler türüne bağlıdır. Örneğin, 5.0/2.0 ifadesinin değeri 2.5’tir. Her iki işlenenin türü **double**’dir bu nedenle sonuç türü de **double**’dir. 5/2 ifadesinde her iki işlenen de **int** türünde ise sonuçta **int** türünde olacak ve sonuç 2 olacaktır. Bu gibi durumlarda her zaman değerleri aşağıya yuvarlanır.

Yüzde işareti ile gösterilen kalan ya da mod işleci. $x \% y$ ifadesinin sonucu x ’in y ’ye bölünmesi sonucunda kalan değerdir. Örneğin, 9’un 2’ye bölünmesi sonucu bölüm 4, kalan da 1 olduğundan, $9\%2$ sonucu 1’dir.

NOT: Mod işleci, tüm sayısal türlerde geçerlidir ve sonucun tam sayı olması gerekmez. Örneğin, $7.0\%2.4$ ifadesinin sonucu 2.2’dir.

ToString yöntemi

.NET platform yazılımındaki her sınıf, **ToString** yöntemine sahiptir. **ToString** yönteminin amacı, bir nesneyi dize gösterimine dönüştürmektir. Sayısal türleri metne dönüştürmek için kullanılır. Örneğin,

```
int i = 123;
```

```
string a = i.ToString();
```

Yukarıdaki ifadede **int** türünde bir değişken bildirilmiş ve daha sonra bu değişken **ToString** metoduyla metne dönüştürülerek **string** türündeki a değişkenine atanmıştır.

4.3. İşlem Önceliği

Öncelik bir deyimdeki işleçlerin gerçekleştirilme sırasınıdır. + ve * işleçlerini kullanan aşağıdaki deyim düşünülün.

```
4 + 5 * 2
```

Bu ifade belirsiz olmaya yatkındır. Önce toplama işlemi mi yoksa çarpma işlemi mi yapılmalıdır? Diğer bir deyişle 5 değeri soldaki + işlecine mi yoksa sağdaki * işlecine mi bağlıdır? Burada işleçlerin sırası önemlidir çünkü sonucu değiştirir:

- İlk olarak toplama işlemi yapılırsa toplama sonucu (4+5), * işlecinin sol tarafındaki işleneni oluşturur ve tüm ifadenin sonucu $9*2$ yani 18 olur.
- İlk olarak çarpma işlemi yapılırsa çarpım sonucu (5*2), + işlecinin sağ tarafındaki işleneni oluşturur ve tüm ifadenin sonucu 4+10 yani 14 olur.

En Yüksek
()
!
*, /, %
+, -
<, >
=, !=
&&
En Düşük

İşlem öncelik sırası yukarıdaki tabloda en yüksekten en düşüğe doğru sıralanmıştır.

4.4. Birleşim Özelliği

Programlamada çarpma işlemi, bölme işlemi ve mod hesaplama(%) işlemleri kendi aralarında eşit önceliklere sahip olmakla birlikte toplama ve çıkarma işlemlerine göre önceliklidir. Toplama ve çıkarma işlemleri de kendi aralarında eşit önceliklere sahiptir.

Parantezler ise çarpma, bölme, toplama ve çıkarma işlemlerine göre önceliklidir. O zaman daha önce yapılması istenen işlemler parantez içerisine alınmalıdır.

() parantez, { } küme parantezi, [] köşeli parantez olarak adlandırılır.

ÖNEMLİ: Kendi aralarında eşit önceliklere sahip işlemlerde işlemler, soldan başlanarak sağa doğru yapılır. Buna birleşim özelliği adı verilir.

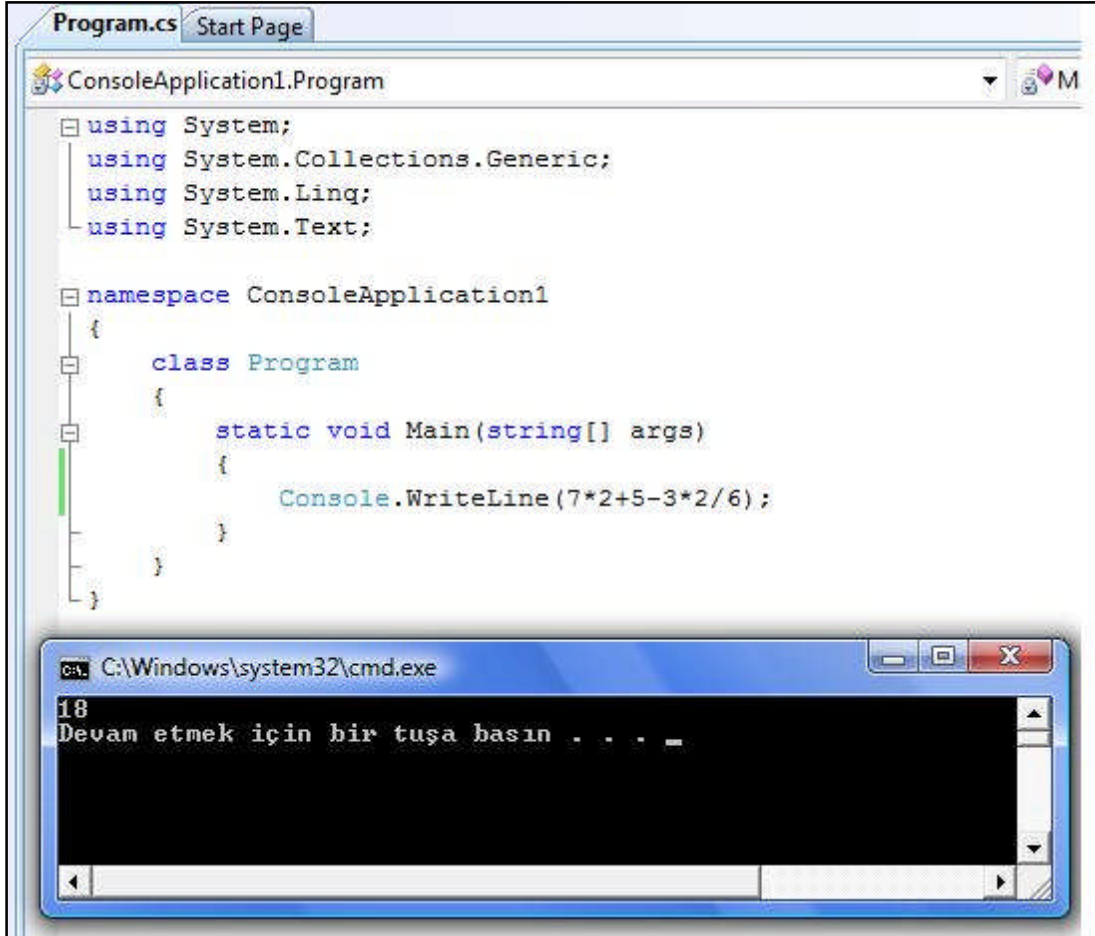
Örnek:

$$7 * 2 + 5 - 3 * 2 / 6 = ?$$

Yukarıdaki ifade de önce $7*2$ işleminin sonucu hesaplanır ve 14 bulunur. Daha sonra $3*2/6$ ifadesi soldan itibaren hesaplanarak $6/6=1$ bulunur. Artık ifade şu şekildedir:

$$14 + 5 - 1$$

Yine soldan itibaren işlemler gerçekleştirilir ve sonuç 18 olarak bulunur. Konsol uygulaması ve sonucunu Resim 4.1'de görülebilir.



Resim 4.1: İşlem önceliği ve birleşim özelliği

4.5. Birleşim ve Atama Operatörü

Programlama dilinde eşittir işareti (=) bir işleçtir. Tüm işleçler, işlenenlerine bağlı olarak bir değer döndürür. Atama işleci (=)'de farklı değildir. İki işlenen alır, sağ tarafındaki işlenen değerlendirilir ve daha sonra sol taraftaki işlenende depolanır. Atama işlecinin değeri, sol işlenene atanmış olan değerdir. Aşağıdaki atama ifadesinde, atama işleci tarafından döndürülen değer 20'dir. Aynı zaman da **yas** değişkenine atanan değerdir.

```
int yas;
yas = 20; // atama ifadesinin değeri 20
```

Atama işleci bir değer döndürdüğünden, atama ifadesinin başka bir oluşumunda da aynı değer kullanılabilir:

```
int yas1;  
int yas2;  
int yas3;  
int yas4;  
yas1 = yas2 = yas3 = yas4 = 20;
```

Bu örnekte atama ifadesi, her iki değişkene de aynı değeri atar. Bu oldukça kullanışlı bir yöntemdir. Bu ifadeden, atama işlecinin sağdan sola doğru birleşim özelliğine sahip olduğu çıkarılabilir. İlk olarak sağ taraftaki atama gerçekleşir ve atanan değer sağdan sola doğru değişkenler boyunca yayılır. Herhangi bir değişken daha önce bir değere sahipse atanan yeni değer üzerine yazdırılır.

4.6. Artırma ve Azaltma Operatörleri

Bir değişkene 1 eklemek istenirse + işleci kullanılabilir:

```
sayac = sayac + 1;
```

Bununla birlikte, bir değişkeni 1 artırmak oldukça yaygındır. Sadece bu amaç için ayrı bir işleç olan ++ işleci de kullanılabilir. “sayac” değişkenini 1 artırmak için aşağıdaki ifade kullanılabilir:

```
sayac++;
```

Benzer şekilde bir değişkenden 1 azaltmak için – işleci de kullanılabilir.

```
sayac--;
```

Bu işleçler, değişkenden önce ve sonra kullanılabilir.

```
sayac++; //sonek artırma  
++sayac; //önek artırma  
sayac--; //sonek azaltma  
--sayac; //önek azaltma
```

İşleçlerin değişkenden önce ve sonra kullanılması sonucu değiştirmez. Her durumda değişkenin değeri 1 artırılır ya da 1 azaltılır ama işlem öncelik sırasından kaynaklanan farklılıklar doğabilir. Örnekle şöyle açıklanır:

```
int a;  
a = 20;  
Console.WriteLine(a++); // a'nın son değeri 21 olmasına karşı ekrana 20 yazılır.  
  
a = 20;  
Console.WriteLine(++a); // a'nın son değeri 21'dir ve ekrana 21 yazılır.
```

4.7. “var” Değişken Türü

Programlama dilinde değişken tanımlanırken değişkenin adının ve veri türünün belirtilmesi gerektiğini görmüştür. Aşağıdaki örnekte olduğu gibi

```
int yas;
```

Ayrıca değişken tanımlanırken bir başlangıç değeri atanabileceği de belirtilmişti.

```
int yas = 20;
```

Hatta herhangi bir işlem yaparak bir değişken aşağıdaki gibi de tanımlanabilir:

```
int yas = fark*10;
```

Bir değişkene atanan değer, değişken ile aynı veri türünde olması gerektiği bilgisi hatırlanmalıdır. Örneğin, **int** türünde bir değişkene sadece **int** türü değer atanabilir.

Derleyici, bir değişkene başlangıç değeri atamak için kullanılan ifadenin türünü hızlı bir şekilde ortaya çıkarabilir ve değişkenin türü ile aynı değilse geri bildirebilir. Ayrıca, derleyiciden bir ifadedeki değişkenin türünü bulması ve aşağıdaki gibi tür yerine *var* anahtar sözcüğünü kullanılarak değişken bildirildiğinde, bulunduğu türü kullanması istenebilir:

```
var fiyat = 100;  
var soyad = “Yılmaz”;
```

fiyat ve **soyad** değişkenleri, **kapalı türde değişkenler** olarak adlandırılır. “*var*” anahtar sözcüğü, derleyicinin değişkenlerin türünü, değişkene başlangıç değeri atamak için kullanılan ifadeden çıkarmasını sağlar. Bu tarz bir kullanımda, değişkene mutlaka başlangıç değeri atanması gerekir. Aşağıdaki gibi bir kullanım hatalıdır ve derleme hatasına neden olur.

```
var fiyat;
```

Ayrıca bu bir kolaylıktır ve değişken bildirildikten sonra değişkene sadece derleyicinin ortaya çıkardığı değer atanabilir. Yani **fiyat** değişkeni **int** türünde bir değişkendir ve **float**, **double** ya da **string** değerler atanamaz.

UYGULAMA FAALİYETİ

Aritmetiksel operatörleri kullanınız.

İşlem Basamakları	Öneriler
<p>➤ İkinci öğrenme faaliyetinde oluşturduğumuz “hesap makinesi” adlı uygulamayı açınız.</p>	<p>➤ File-Open Project komutunu kullanınız.</p>
<p>➤ Topla butonuna çift tıklayarak Click özelliğine aşağıdaki kodu yazınız.</p> <pre>int a=Int16.Parse(textBox1.Text); int b=Int16.Parse(textBox2.Text); textBox3.Text = (a+b).ToString();</pre>	<p>➤ Parantezler ve noktalama işaretlerine dikkat ediniz.</p>
<p>➤ Çıkar butonuna çift tıklayarak Click özelliğine aşağıdaki kodu yazınız.</p> <pre>int a=Int16.Parse(textBox1.Text); int b=Int16.Parse(textBox2.Text); textBox3.Text = (a-b).ToString();</pre>	<p>➤ Parantezler ve noktalama işaretlerine dikkat ediniz.</p>
<p>➤ Çarp butonuna çift tıklayarak Click özelliğine aşağıdaki kodu yazınız.</p> <pre>int a=Int16.Parse(textBox1.Text); int b=Int16.Parse(textBox2.Text); textBox3.Text = (a*b).ToString();</pre>	<p>➤ Parantezler ve noktalama işaretlerine dikkat ediniz.</p>
<p>➤ Böl butonuna çift tıklayarak Click özelliğine aşağıdaki kodu yazınız.</p> <pre>int a=Int16.Parse(textBox1.Text); int b=Int16.Parse(textBox2.Text); textBox3.Text = (a/b).ToString();</pre>	<p>➤ Parantezler ve noktalama işaretlerine dikkat ediniz.</p>
<p>➤ Uygulamayı çalıştırınız.</p>	<p>➤ F5’i kullanabilirsiniz.</p>
<p>➤ Uygulamanızı test ediniz.</p>	<p>➤ Metin kutularına değerler girerek dört işlem için butonları kullanınız.</p>

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadıklarınız için **Hayır** kutucuklarına (X) işareti koyarak kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Daha önce kaydettiğiniz uygulamayı açtınız mı?		
2. İlgili butona çift tıklayıp toplama işlemi için gerekli kodları yazdınız mı?		
3. İlgili butona çift tıklayıp çıkarma işlemi için gerekli kodları yazdınız mı?		
4. İlgili butona çift tıklayıp çarpma işlemi için gerekli kodları yazdınız mı?		
5. İlgili butona çift tıklayıp bölme işlemi için gerekli kodları yazdınız mı?		
6. Uygulamanızı çalıştırdınız mı?		
7. Dört işlem için uygulamanızı test ettiniz mi?		

DEĞERLENDİRME



Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise Ölçme ve Değerlendirme’ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Üzerine buton, metin kutusu vb. nesnelere yerleştirilen arabirime ne ad verilir?
A) Solution Explorer
B) Form
C) Properties
D) ToolBox
2. Buton, metinkutusu, label vb. nesnelere özelliklerini değiştirmek için kullanılan panel aşağıdakilerden hangisidir?
A) Solution Explorer
B) ToolBox
C) Properties
D) Form
3. Nesne tabanlı programlama yazılım ortamında hazırladığımız uygulamayı çalıştırmak için hangi kısayol tuşu kullanılır?
A) F7
B) F6
C) F3
D) F5
4. Form üzerine nesne yerleştirmek için hangi panel kullanılır?
A) Form
B) ToolBox
C) Properties
D) Solution Explorer
5. *Label.Content = "Adınız";* kod satırının anlamı aşağıdakilerden hangisidir?
A) Adınız yazısını mesaj penceresinde gösterir.
B) Adınız yazısını buton nesnesinde gösterir.
C) Adınız yazısını label nesnesinde gösterir.
D) Adınız yazısını TextBox nesnesinde gösterir.
6. *MessageBox.Show(a);* kod satırının açıklaması aşağıdakilerden hangisinde doğru olarak verilmiştir?
A) a değişkeninin değerini mesaj penceresinde görüntüler.
B) a değişkeninin değerini label nesnesinde görüntüler.
C) a ifadesini mesaj penceresinde görüntüler.
D) Çalışmaz hata verir.

7. `c = a.ToString();` kod satırının açıklaması hangi seçenekte doğru olarak verilmiştir?
A) a değişkeninin değerini string türüne dönüştürerek c değişkenine aktarır.
B) c değişkeninin değerini string türüne dönüştürerek a değişkenine aktarır.
C) a değişkeninin değerini integer türüne dönüştürerek c değişkenine aktarır.
D) c değişkeninin değerini integer türüne dönüştürerek a değişkenine aktarır.
8. Herhangi bir kod satırını yorum satırına dönüştürmek için hangi karakter kullanılır?
A) { }
B) ()
C) //
D) ""
9. Program içerisinde farklı değerler alabilen ifadelere ne ad verilir?
A) Sabit
B) Sınıf
C) Değişken
D) Hiçbiri
10. `sayac++;` kod satırının işlevi aşağıdakilerden hangisinde doğru olarak verilmiştir?
A) sayac değişkeninin değerini 2 artırır.
B) sayac değişkeninin değerini 1 artırır.
C) sayac değişkeninin değerini 2 azaltır.
D) sayac değişkeninin değerini azaltmadan önce kendisiyle toplar.
11. Matematiksel mod alma işlemi için hangi operatör kullanılır?
A) +
B) #
C) *
D) %
12. Aşağıdakilerden hangisi veri türü değildir?
A) int
B) goto
C) string
D) bool
13. Aşağıdakilerden hangisi bool türü bir değişkenin değeri olabilir?
A) "No"
B) 132
C) &
D) False
14. 2(iki) sayısını hafızada saklamak üzere aşağıdaki veri türlerinden hangisi kullanılamaz?
A) bool
B) int
C) string
D) char

15. İşlem önceliği kavramına göre aşağıdakilerden hangisi doğrudur?
A) $7+14/7=3$
B) $2+7*3=27$
C) $12+3/3=13$
D) $15-5/1=11$
16.  resimdeki nesne aşağıdakilerden hangisidir?
A) Label
B) textBox
C) Buton
D) ListBox
17.  resimdeki nesne aşağıdakilerden hangisidir?
A) Label
B) textBox
C) Buton
D) ListBox

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmenize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1	D
2	C
3	B
4	A
5	B

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1	B
2	D
3	C
4	A
5	B
6	D
7	C
8	B

ÖĞRENME FAALİYETİ-3'ÜN CEVAP ANAHTARI

1	Doğru
2	Doğru
3	Yanlış
4	Yanlış
5	Doğru
6	Yanlış
7	Yanlış
8	Doğru
9	Doğru
10	Yanlış

ÖĞRENME FAALİYETİ-4'ÜN CEVAP ANAHTARI

1	B
2	D
3	C
4	A
5	B
6	A

MODÜL DEĞERLENDİRMENİN CEVAP ANAHTARI

1	B
2	C
3	D
4	B
5	C
6	A
7	A
8	C
9	C
10	B
11	D
12	B
13	D
14	A
15	C
16	C
17	A

KAYNAKÇA

- SHARP John(Çeviri:Ümit TEZCAN), **Adım Adım Microsoft Visual C# 2008**, Arkadaş Yayınevi, Ankara, 2008.