

**T.C.  
MİLLÎ EĞİTİM BAKANLIĞI**

# **BİLİŞİM TEKNOLOJİLERİ**

**NESNE TABANLI PROGRAMLAMADA  
METOTLAR  
482BK0162**

**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. METOTLAR .....	3
1.1. Metot Tanımlama .....	3
1.2. “return” İfadeleri Yazma .....	4
1.3. Metot Çağırma .....	4
UYGULAMA FAALİYETİ.....	6
ÖLÇME VE DEĞERLENDİRME.....	8
ÖĞRENME FAALİYETİ-2 .....	9
2. KAPSAM .....	9
2.1. Kapsam Uygulama .....	9
2.1.1. Yerel Kapsamı Tanımlama .....	9
2.1.2. Sınıf Kapsamını Tanımlama .....	10
2.2. Metotları Aşırı Yükleme .....	11
UYGULAMA FAALİYETİ.....	17
ÖLÇME VE DEĞERLENDİRME.....	19
MODÜL DEĞERLENDİRME .....	20
CEVAP ANAHTARLARI.....	22
KAYNAKÇA .....	23

# AÇIKLAMALAR

<b>KOD</b>	<b>482BK0162</b>
<b>ALAN</b>	<b>Bilişim Teknolojileri</b>
<b>DAL/MESLEK</b>	<b>Veritabanı Programcılığı</b>
<b>MODÜLÜN ADI</b>	<b>Nesne Tabanlı Programlamada Metotlar</b>
<b>MODÜLÜN TANIMI</b>	Nesne tabanlı programlamada metotların kullanımını anlatan bir öğrenme materyalidir.
<b>SÜRE</b>	40/32
<b>ÖN KOŞUL</b>	Nesne Tabanlı Programlamaya Giriş modülünü tamamlamış olmak
<b>YETERLİK</b>	Nesne tabanlı programlamada metotlar yazmak ve kapsam uygulamak
<b>MODÜLÜN AMACI</b>	<b>Genel Amaç:</b> Nesne tabanlı programlama ortamını kullanarak metot yazabilecek ve kapsam uygulayabileceksiniz. <b>Amaçlar:</b> <b>1.</b> Metot tanımlayabilecek ve bu metotları kullanabileceksiniz. <b>2.</b> Kapsam uygulayabilecek ve overload yöntemini kullanabileceksiniz.
<b>EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI</b>	<b>Ortam:</b> Nesne tabanlı programlama yazılımı <b>Donanım:</b> Bilgisayar
<b>ÖLÇME VE DEĞERLENDİRME</b>	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 kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

# GİRİŞ

## **Sevgili Öğrenci,**

Şu ana kadar yaptığımız örneklerde önceden hazırlanmış ReadLine(), WriteLine() vb. gibi metotları kullandık. Bu modülü bitirdiğimizde artık kendi metotlarımızı yapabiliriz.

Aslında bütün örneklerimizde birer metot yaratmıştık. O da çalışabilir her programda bulunması gereken Main metodu. Artık Main metodu gibi başka metotlar yaratıp programımızın içinde kullanabileceğiz.

Metotlar oluşturarak programımızı parçalara böler ve programımızın karmaşıklığını azaltırız. Ayrıca bazı kodları bir metot içine alıp aynı kodlara ihtiyacımız olduğunda bu metodu çağırabiliriz. Bu sayede de kod hamallığı yapmaktan kurtuluruz.



# ÖĞRENME FAALİYETİ-1

## AMAÇ

Metot tanımlayabilecek ve bu metotları kullanabileceksiniz.

## ARAŞTIRMA

- Nesne tabanlı programlama dilinde hazır olarak kullanılan metotların hangileri olduğunu araştırınız.
- Değişken tanımlama kurallarını hatırlayarak tartışınız.

## 1. METOTLAR

### 1.1. Metot Tanımlama

Metotlar, bir işlem yapmak üzere tasarlanmış kodlar topluluğudur. Program yazarken belli bir işlevi olan kod bloğunu birkaç kez kullanmak gerekebilir. Bu durumlarda aynı kodları program içerisinde sürekli yazmak yerine bu, bir metot olarak hazırlanabilir ve ihtiyaç duyulduğunda kullanılabilir. Temel metot yazım şekli aşağıdaki gibidir:

*dönüş\_türü metot\_adı (parametre listesi)*

- **dönüş\_türü**, bir veri türü adıdır ve metodun yaptığı işlem sonucunda döndüreceği veri türünü belirler. Bu *int* ya da *string* türü bir veri olabilir. Herhangi bir sonuç döndürmeyen bir metot yazılıyorsa **dönüş\_türü** yerine *void* anahtar sözcüğü kullanılmalıdır.
- **metot\_adı**, metodu çağırmak için kullanılan adıdır. Metot adları belirlenirken değişken adları tanımlanırken uygulanan kurallara uyulmalıdır. Örneğin, *Dort\_islem* geçerli bir metot adı olabilirken *Dort\$islem* tanımlaması yanlıştır.
- **parametre listesi**: İsteğe bağlıdır, metoda dışarıdan gönderilecek veri adlarını ve türlerini tanımlar. Parametreler önce tür adı, daha sonra parametre adı olacak şekilde parantez içerisinde değişken tanımlanmış gibi yazılır.

**Örnek:**

```
int alanHesaplama(int kısaKenar, int uzunKenar)
{
    //kodla
}
```

Yukarıdaki metot tanımında alanHesaplama metodun adını, metot adının başındaki int ifadesi metodun döndüreceği veri türünü, parantez içerisinde tanımlanan değişkenler parametre listesini göstermektedir.

Eğer tanımlanan metot dışarıdan değer alacak fakat değer döndürmeyecekse

```
void alanHesaplama(int kısaKenar, int uzunKenar)
{
    //kodlar
}
```

şeklinde tanımlanmalıdır.

## 1.2. “return” İfadeleri Yazma

Tanımlanan metodun bir değer döndürmesi isteniyorsa (başka bir deyişle dönüş türünün *void* olmasını istemiyorsanız) metot içinde bir *return* ifadesi yazılmalıdır. *return* anahtar sözcüğünden sonra, dönen değeri hesaplayan ifade ve sonunda noktalı virgül yer alır. Hesaplanan ifadenin türü, metot tarafından belirtilen dönüş türü ile aynı olmak zorundadır. Yani metot, *int* türünde bir değer döndürüyorsa *return* ifadesi de *int* türü döndürmelidir. Aksi takdirde program derlenmez.

**Örnek:**

```
int alanHesaplama(int kısaKenar, int uzunKenar)
{
    //kodlar
    return kısaKenar * uzunKenar;
}
```

*return* ifadesi, metodun sonlandırılmasına neden olduğundan genellikle sonda yer alır. *return* ifadesinden sonra yazılan herhangi bir ifade çalıştırılmaz (*return* sözcüğünden sonra herhangi bir ifade yazılırsa derleyici uyarır.).

## 1.3. Metot Çağırma

Tanımlanan metotları adları kullanılarak çağrılır. Metotların tanımlanma şekilleri, dönüş türleri ve parametre listelerini gördükten sonra aşağıdaki örnek incelenerek kod satırında bir metodun adıyla nasıl çağırıldığı görülsün.



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        void Main(string[] args)
        {
            int kısa_kenar, uzun_kenar, deger = 0;
            Console.WriteLine("Kısa kenar uzunluğunu giriniz:");
            kısa_kenar = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Uzun kenar uzunluğunu giriniz:");
            uzun_kenar = Convert.ToInt32(Console.ReadLine());
            deger = alan(kısa_kenar , uzun_kenar);
            Console.WriteLine("Dikdörtgenin alanı : " + deger + " dir");
        }

        int alan(int a, int b)
        {
            return a * b;
        }
    }
}

```

**Resim 1.1: Metot çağırma**

**NOT: Console.ReadLine** ifadesi kullanıcıdan değer olarak bir değişkene atamak için kullanılır. İfadenin döndüreceği değer **string** olacağı için **Convert.ToInt32** yöntemiyle **int** türüne dönüştürülmüştür.

Program kısa ve uzun kenar değerleri kullanıcı tarafından girilen bir dikdörtgenin alanını hesaplamaktadır. Tanımlanan *alan* adlı metoda kısa ve uzun kenar değerleri gönderilmekte, metot içerisinde gerekli hesaplamalar yapıp sonuç değeri *return* ifadesiyle geri döndürülmektedir.

Aşağıda görüldüğü gibi *alan* adlı metot, farklı şekillerde çağrılırsa farklı hatalarla karşılaşılabilir.

alan;	//derleme zamanı hatası, parantez yok
alan();	//derleme zamanı hatası, yetersiz bağımsız değişken
alan(kısa_kenar);	//derleme zamanı hatası, yetersiz bağımsız değişken
alan("kısa_kenar","uzun_kenar");	//derleme zamanı hatası, yanlış türler

## UYGULAMA FAALİYETİ

Metot tanımlayarak bu metotları kullanınız.

İşlem Basamakları	Öneriler
➤ Yeni bir proje oluşturunuz.	➤ Ctrl+N, File-New Project ya da Recent Project seçeneklerini kullanabilirsiniz(bk. Resim 1.2).
➤ WPF uygulamasını seçiniz.	➤ Uygulamayı oluştururken dili seçmeyi unutmayınız.
➤ Oluşan forma üç tane label nesnesi ve karşılıklarına üç tane textBox nesnesi ekleyiniz.	➤ ToolBox panelini kullanınız.
➤ Label nesnelerinin isimlerini “1.sayı, 2.sayı ve Sonuç” olarak değiştiriniz.	➤ Properties panelindeki Content özelliğini kullanınız.
➤ Formunuza buton nesnesi ekleyerek ismini “hesapla” olarak değiştiriniz.	➤ Properties panelindeki Content özelliğini kullanınız.
➤ Buton nesnesinin “click” özelliğine işlem(); ifadesini ekleyiniz.	➤ Click özelliği için buton üzerine çift tıklayınız.
➤ Buton nesnesinin click özelliğinin sonlandırıldığı {( küme parantezi) nin hemen altına public void işlem() metodunu tanımlayınız.	➤ Yazdığımız metot için küme parantezleri kullanmayı unutmayınız.
➤ Tanımladığımız metot içerisinde textBox1 ve textBox2 nesnelerini toplayarak textBox3 nesnesine aktarınız.	➤ Gerekli tür dönüşümlerini yapmayı unutmayınız.
➤ Programınızı çalıştırınız.	➤ F5 kısayolunu 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. WPF uygulamasını seçtiniz mi?		
3. Programlama dilini seçtiniz mi?		
4. Formunuza Label, Button ve textBox nesnelerini eklediniz mi?		
5. Button nesnesinin Click özelliğini kullandınız mı?		
6. Metot tanımlaması yaptınız mı?		
7. Tanımladığınız metodu çağırdınız mı?		

## DEĞERLENDİRME

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



# ÖĞRENME FAALİYETİ-2

## AMAÇ

Kapsam uygulayabilecek ve overload yöntemini kullanabileceksiniz.

## ARAŞTIRMA

- Nesne tabanlı programlamada sınıf kavramını araştırınız.
- Local ve global değişken kavramlarını araştırınız.

## 2. KAPSAM

### 2.1. Kapsam Uygulama

Program içerisinde değişkenlerin nasıl tanımlandığı görüldü. Değişkenler tanımlandıkları yerde oluşturulur ve daha sonra istenilen şekilde kullanılabilir yani tanımlandıkları yer program gövdesi ya da herhangi bir metodun içerisi olabilir. Program ya da metod sonlandığında ise değişken artık kullanılamaz.

Bir değişken belirli bir yerde kullanılabilirse değişken o konumda kapsam (scope) içerisindedir. Farklı bir biçimde ifade etmek gerekirse bir değişkenin kapsamı, değişkenin o program içerisinde kullanıldığı bölgedir. Bu kural değişkenler için olduğu gibi tanımlanan metotlar için de geçerlidir.

#### 2.1.1. Yerel Kapsamı Tanımlama

Bir metodun gövdesini oluşturan, açılan ve kapatılan küme parantezleri, kapsamı tanımlar. Metodun gövdesi içinde tanımlanan bütün değişkenler, o metodun kapsamına eklenir. Metod sona erdiğinde geçerliliklerini yitirir ve sadece tanımlandıkları metod üzerinden erişilebilir. Bu tür değişkenler sadece tanımlandıkları metod içerisinde kullanılabilirliğinden yerel(local) değişkenler olarak adlandırılır. Resim 2.1'deki örnek incelenmelidir.

```
void bolme()
{
    int a = Int16.Parse(textBox1.Text);
    int b = Int16.Parse(textBox2.Text);

    textBox3.Text = (a / b).ToString();
}
void carpma()
{
    a = Int16.Parse(textBox1.Text);
    b = Int16.Parse(textBox2.Text);

    textBox3.Text = (a * b).ToString();
}
```

**Resim 2.1: Yerel deęişkenler**

Resim 2.1’de görüldüğü gibi bolme ve carpma adlı iki metot tanımlanmıştır. bolme adlı metot içerisinde ise int türünde a ve b deęişkenleri tanımlıdır. Aynı deęişkenler carpma adlı metot içerisinde tanımlanmadan kullanılamaz. Çünkü deęişkenler bolme metodu kapsamında tanımlanmıştır. carpma metodunun içerisinde kullanılmaya çalışıldığında yazılan kodların derlenmesi başarısızlıkla sonuçlanır ve program hata verir. Hata ile karşılaşmamak için carpma metodunda deęişkenler yeniden tanımlanmalıdır.

### 2.1.2. Sınıf Kapsamını Tanımlama

Bir sınıfın gövdesini oluşturmak için açılan ve kapatılan küme parantezleri de bir kapsam oluşturur. Sınıf gövdesi içinde tanımlanan deęişkenler (metot içinde deęil), o sınıfın kapsamındadır. Sınıf içerisinde tanımlanmış deęişkenler *field(alan)* olarak adlandırılır. Yerel deęişkenlerin aksine bu tür deęişkenleri (alanları), metotlar arasında bilgi paylaşımı için kullanılabilir. Resim 2.2 incelenmelidir.

```

public partial class Window1 : Window
{
    void bolme ()
    {
        a = Int16.Parse(textBox1.Text);
        b = Int16.Parse(textBox2.Text);

        textBox3.Text = (a / b).ToString();
    }
    void carpma ()
    {
        a = Int16.Parse(textBox1.Text);
        b = Int16.Parse(textBox2.Text);

        textBox3.Text = (a * b).ToString();
    }
    int a;
    int b;

    public Window1 ()
    {
        InitializeComponent();
    }
}

```

**Resim 2.2: Sınıf (global) kapsamı**

Resim 2.2’de görüldüğü gibi *a* ve *b* değişkenleri herhangi bir metod içinde değil *Window1* adlı sınıf içinde tanımlanmıştır. Dolayısıyla aynı değişkenler, *bolme* ve *carpma* adlı metotlar içerisinde tekrar tanımlanmaksızın kullanılabilir. Sınıf içindeki diğer metotlar da *a* ve *b* değişkenlerini aynı şekilde kullanabilir. Çünkü bu değişkenler, sınıf kapsamındadır.

Diğer bir önemli nokta ise *a* ve *b* değişkenlerinin metotlardan sonra tanımlanmalarına rağmen metotlar içerisinde kullanılabilirler. Bir metotta kullanmadan önce değişkeni tanımlamak gerekir. Sınıf kapsamındaki değişkenler (alanlar) biraz farklıdır. Bir metod, alanı tanımlayan ifadeden önce, o alanı kullanabilir. Bu noktada derleyici devreye girer ve bu durumu sizin için düzenler.

## 2.2. Metotları Aşırı Yükleme

Metotların belirli bir işi gerçekleştirebilen, isteğe bağlı değerler döndürebilen program parçacıkları olduğu biliniyor. Tanımlanan metotlar tek bir iş için tanımlansa da ilerleyen zamanlarda metoda yeni özellikler ekleyip işlevselliği artırılmak istenebilir. Metotların aynı adla yeniden tanımlanmaları iki şartla mümkündür. Eğer metodun aldığı parametre türü veya sayısı değiştirilirse metodu aynı adla tekrar tekrar tanımlanabilir. Bu işleme *metodun aşırı yüklenmesi (overload)* adı verilir. Metod aşırı yüklenirken sadece geri dönüş türünü

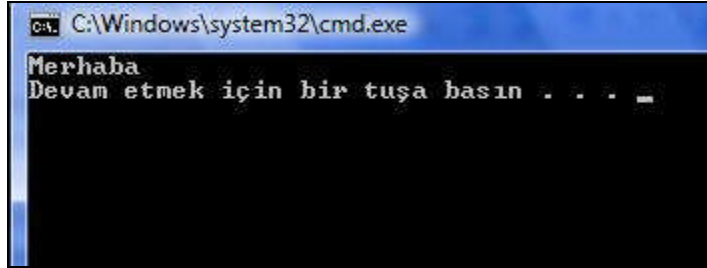
değiřtirmek iře yaramaz. Bu derleyici için kabul edilebilir bir durum deęildir. Program hata verir.

**Örnek:** Bir konsol uygulaması oluşturarak ekrana “Merhaba” mesajını yazan program kodları yazılsın.

```
namespace overload
{
    class Program
    {
        static void Main(string[] args)
        {
            yaz("Merhaba");
        }
        static void yaz(string metin)
        {
            Console.WriteLine(metin);
        }
    }
}
```

Resim 2.3: Metot tanımlama

Resim 2.3'te *yaz* adlı bir metot tanımlanarak “Merhaba” mesajının ekrana yazılması sağlanmıştır. Metot *string* türünde bir parametre almakta ama geriye deęer döndürmemektedir. Ekran çıktısı Resim 2.4'teki gibidir.



Resim 2.4: Ekran çıktısı

Aynı isimde bir metot yazarak *overload* (aşırı yükleme) yöntemiyle mesajın ekrana büyük ya da küçük harflerle nasıl yazıldığı görülsün.



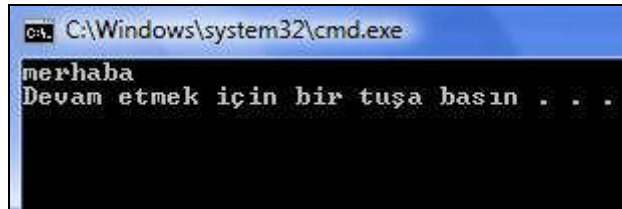
```

namespace overload
{
    class Program
    {
        static void Main(string[] args)
        {
            yaz("Merhaba", false);
        }
        static void yaz(string metin)
        {
            Console.WriteLine(metin);
        }
        static void yaz(string metin, bool tercih)
        {
            if (tercih)
            {
                Console.WriteLine(metin.ToUpper());
            }
            else
            {
                Console.WriteLine(metin.ToLower());
            }
        }
    }
}

```

**Resim 2.5: Metodu aşırı yükleme (overload)**

Resim 2.5'te görüldüğü gibi *yaz* isimli iki metod tanımlanmıştır. Sonradan tanımlanan metod birincisinden farklı olarak *bool* türünde *tercih* adlı fazladan bir parametre daha almaktadır. Bu parametre program ana gövdesinden *false* olarak gönderilmiştir. Böylelikle ikinci metodun aşırı yüklenerek farklı bir işlevi yerine getirmesi sağlanmıştır. Kullanılan if yapısı daha sonraki modüllerde öğrenilecek. Ekran çıktısı Resim 2.6'daki gibidir.



**Resim 2.6: Ekran çıktısı**

Resim 2.6'da görüldüğü gibi metnin tamamı, küçük harflerle yazılmıştır. Program gövdesinden gönderilen *tercih* değeri *true* olsaydı mesajın tamamı büyük harflerle yazılacaktı.

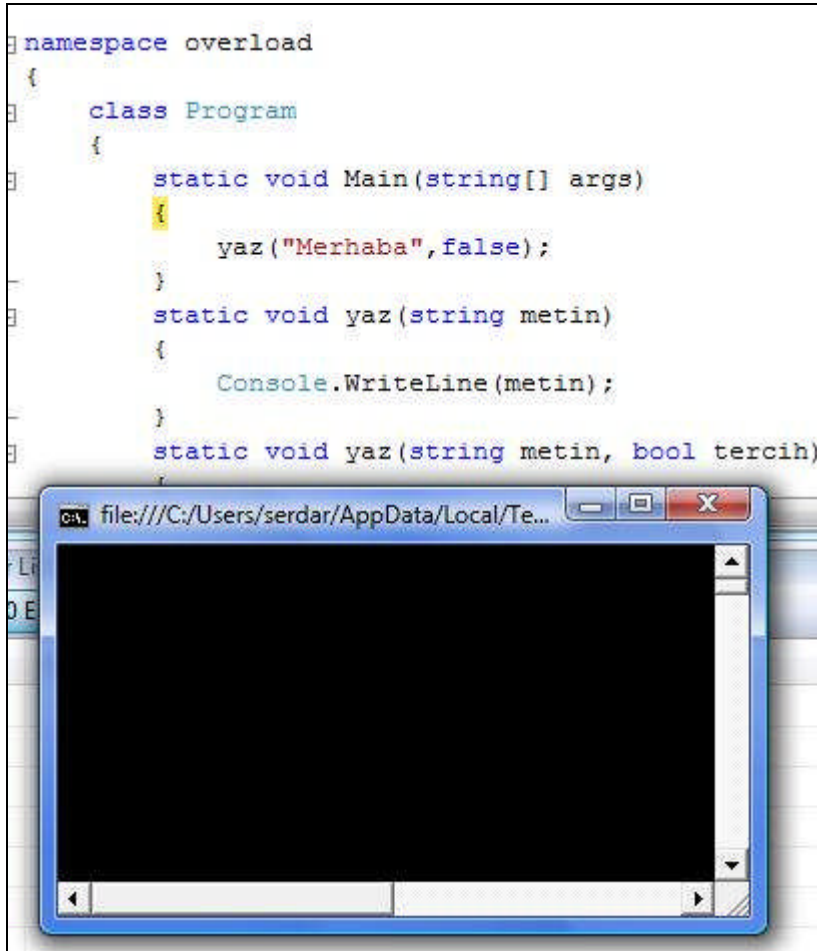
Programı yavaş biçim de çalıştırmak için *Nesne tabanlı programlama yazılımı hata ayıklayıcısını* kullanılsın. Her bir metodun ne zaman çağırıldığı (*stepping into the method-*

*yönetimin içine girmek*), her bir return ifadesinin denetiminin nasıl aktarıldığı (*stepping out of the method-yönetimin dışına çıkmak*) görülecektir. Metotların içine girerken ve metotlardan çıkarken **Debug menüsü** veya **Standart araç çubuğu** kullanılacaktır. Buna programı adım adım çalıştırmak da denebilir.



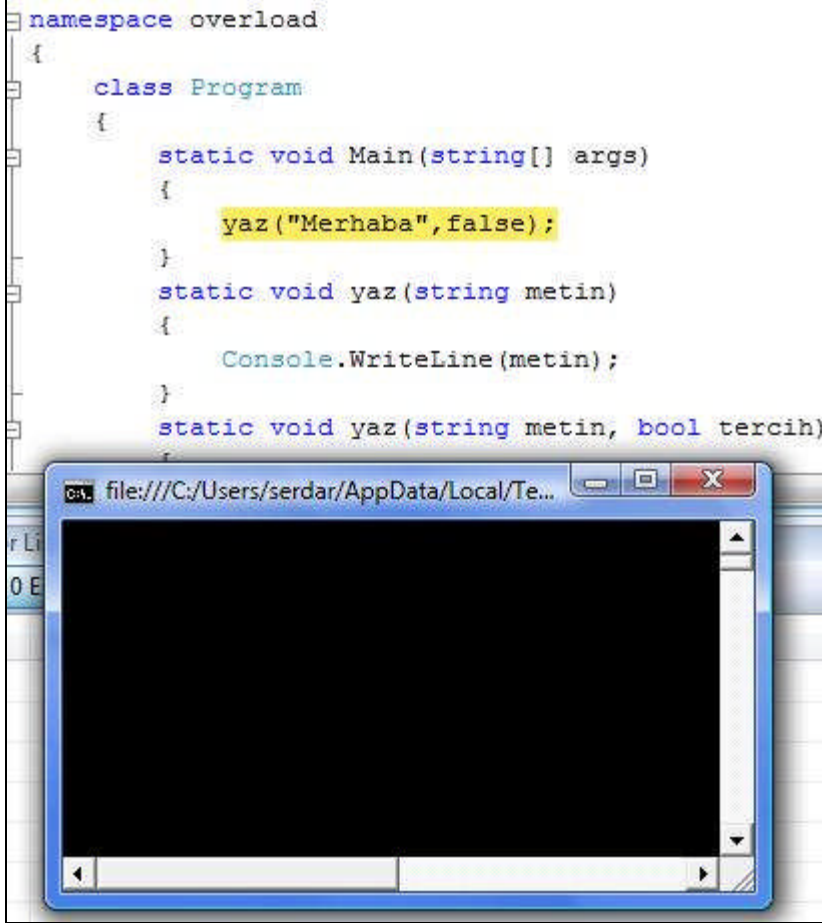
Adım adım uygulaması:

- Fareyle program satırının başına tıklanır. Araç çubuğundaki **Step Into** butonuna tıklanır. Main metodunun ilk satırındaki küme parantezinin zemininin sarı renkle boyandığı görülecektir. Bu arada konsol ekranında çalışmaya başladığı görülecektir. Resim 2.7’de gösterilmiştir.



Resim 2.7: Step Into

- Tekrar Step Into'ya tıklanır, metod adının yazılı olduğu sonraki satıra ilerlediği görülecektir. Resim 2.8'de görülebilir.

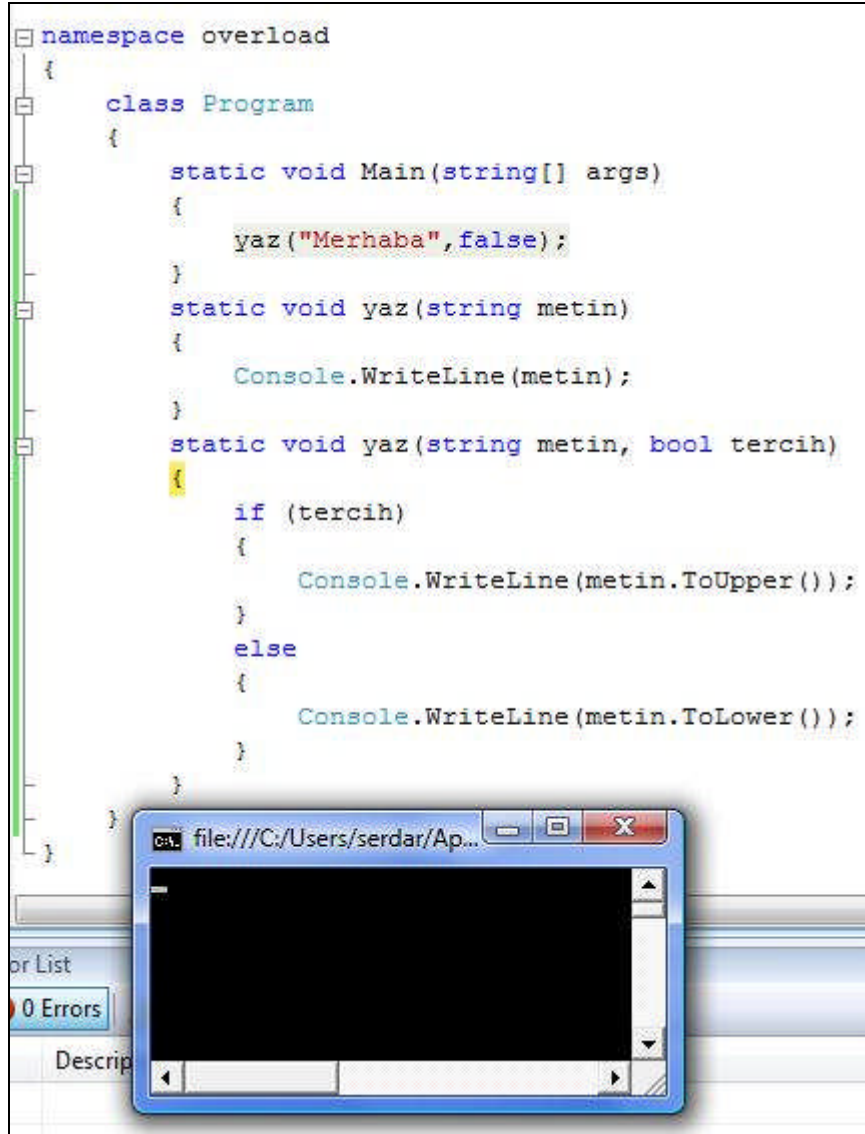


```
namespace overload
{
    class Program
    {
        static void Main(string[] args)
        {
            yaz("Merhaba", false);
        }
        static void yaz(string metin)
        {
            Console.WriteLine(metin);
        }
        static void yaz(string metin, bool tercih)
    }
}
```

The image shows a code editor window with a C# program. The code defines a namespace 'overload' containing a class 'Program'. Inside 'Program', there is a 'Main' method that calls 'yaz("Merhaba", false)'. Below 'Main', there are two overloaded 'yaz' methods: one taking a 'string metin' and another taking 'string metin' and 'bool tercih'. A debugger window is overlaid on the code, showing a black console output area. The title bar of the debugger window indicates the file path: 'file:///C:/Users/serdar/AppData/Local/Te...'. The 'yaz("Merhaba", false);' line in the code is highlighted in yellow.

**Resim 2.8: Step Into**

- Step Into'ya tıklanır, seçili alanın metodun içerisine geçtiği görülecektir. Resim 2.9'da görülebilir.



**Resim 2.9: Step Into**

- Step Over butonu kullanılırsa hata ayıklama yapmadan (metodun içine girmeden) bir sonraki ifadeye geçildiği görülecektir.
- Step Out butonuna tıklanırsa geçerli metodun kesintiye uğramaksızın yani adım adım değil de sonuna kadar çalışmasına neden olduğu görülecektir.
- Programın sonuna gelindiğinde uygulama tamamlanır ve çalışması sonlandırılır. Bu arada ekran çıktısı da adım adım takip edilebilir.

## UYGULAMA FAALİYETİ

Kapsam uygulayabilecek ve overload yöntemini kullanınız.

İşlem Basamakları	Öneriler
➤ Yeni bir proje oluşturunuz.	➤ Ctrl+N, File-New Project ya da Recent Project seçeneklerini kullanabilirsiniz.
➤ Konsol uygulamasını seçiniz.	➤ Uygulamayı oluştururken dili seçmeyi unutmayınız.
➤ Program gövdesi Main metodunun içine Console.WriteLine(kareal(5)); Console.WriteLine(kareal(5.1)); Console.WriteLine(kareal("3")); Console.Read(); ifadelerini yazınız.	➤ Büyük küçük harf ayırımına ve noktalama işaretlerine dikkat ediniz. Kodları Main bloğundaki küme parantezleri içerisine yazınız.
➤ Bir int parametre alan ve int değer döndüren "kareal" adında metod tanımlayınız.	➤ int kareal(int x) { }
➤ Metodun aldığı x parametresini kendisiyle çarparak return ifadesiyle geri gönderiniz.	➤ return x*x;
➤ Bir double parametre alan ve double değer döndüren "kareal" adında metod tanımlayınız.	➤ double kareal(double x) { }
➤ Metodun aldığı x parametresini kendisiyle çarparak return ifadesiyle geri gönderiniz.	➤ return x*x;
➤ Bir string parametre alan ve int değer döndüren "kareal" adında metod tanımlayınız.	➤ int kareal(string x) { }
➤ Metodun aldığı x parametresini int türüne çevirip kendisiyle çarparak return ifadesiyle geri gönderiniz.	➤ return (int32.Parse(x)*int32.Parse(x));
➤ Programınızı çalıştırarak test ediniz.	➤ F5 kısayol tuşunu kullanabilirsiniz.
➤ Nesne tabanlı programlama yazılımı hata ayıklayıcısını kullanarak çalıştırma adımlarını tekrarlayınız.	➤ Araç çubukları üzerindeki Step Into, Step Over, Step Out kısayol tuşlarını kullanabilirsiniz veya Debug menüsünden yararlanabilirsiniz.

## 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. Konsol uygulaması oluşturduunuz mu?		
2. Gerekli metotları program gövdesine yazdınız mı?		
3. Metotları tanımladınız mı?		
4. Return ifadelerini yazdınız mı?		
5. Step Into araç çubuğunu kullandınız mı?		
6. Step Over araç çubuğunu kullandınız mı?		
7. Step Out araç çubuğunu kullandınız mı?		
8. Ekran çıktısını adım adım incelediniz 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.

## ÖLÇME VE DEĞERLENDİRME

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

1. ( ) Aynı sınıf kapsamında birden fazla metot tanımlanamaz.
2. ( ) Overload yöntemi metotların işlevselliğini artırmak için kullanılır.
3. ( ) Overload yönetiminde metot isimlerinin aynı olması zorunlu değildir.
4. ( ) Bir metodun aşırı yüklenmesi geri döndürdüğü veri türüne bağlıdır.
5. ( ) Bir sınıf içerisinde tanımlanan değişken o sınıf içindeki tüm metotlarda kullanılabilir.
6. ( ) Bir programın çalışması tüm adımlarıyla izlenmek isteniyorsa hata ayıklama uygulaması kullanılabilir.
7. ( ) Hata ayıklama uygulamasındaki Step Out butonu tüm programı sonlandırmak için kullanılır.
8. ( ) Metotlar her zaman bir sınıf içerisinde yazılmak zorundadır.
9. ( ) Overload işlemi yapabilmek için tanımlanan metot isimleri aynı, metoda gönderilen veri türü ya da parametre sayıları farklı olmalıdır.
10. ( ) Tanımlanan metot adlarının aynı olması overload işlemi yapıldığı anlamına gelmez.

## 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 “Modül Değerlendirme”ye geçiniz.

# MODÜL DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Aşağıdaki metot tanımlamalarından hangisi doğrudur?  
A) int hesapla(int a,b)  
B) void işlem()  
C) int sonuc(int x; double y)  
D) void (int x,y, bool a)
2. Metotların aşırı yüklenmesine ne ad verilir?  
A) Local Variable  
B) Global Variable  
C) OverLoad  
D) Step Over
3. Metotlarla ilgili aşağıdakilerden hangisi yanlıştır?  
A) Overload işlemi tekrarlı kod satırlarını önleyebilir.  
B) Main metodu içinde sınırsız sayıda metot çağrılabilir.  
C) Metot eğer geriye bir değer döndürmüyorsa return kullanılmayabilir.  
D) Metotların geri dönüş türü ayırt edici özelliklerdendir.
4. Değişkenlerle ilgili olarak aşağıdakilerden hangisi yanlıştır?  
A) Kapsam alanı dışındaki bir yerden değişkene ulaşamaz.  
B) Sınıf içinde tanımlanmış bir değişkene, metot içinde farklı bir veri türü atanabilir.  
C) Local değişkenler yalnızca tanımlandıkları metot içinde kullanılabilir  
D) Bir değişken birden çok yerde kullanılacaksa sınıf düzeyinde tanımlanmalıdır.
5. Aşağıdaki metot tanımlamalarından hangisinde değer döndürülmemiştir?  
A) int deneme (int yas, string ad)  
B) bool tercih (bool secim)  
C) void toplam(int a, int b)  
D) double bolme ( )
6. Aşağıdaki yazımlardan hangisi yanlıştır?  
A) return(int a=0);  
B) return a+b;  
C) return a\*b  
D) return a;
7. Hata ayıklaması yapmadan (metodun içine girmeden) bir sonraki ifadenin çalıştırılmasını sağlayan Debug seçeneği aşağıdakilerden hangisidir?  
A) Step Into                      B) Step Over                      C) Step Out                      D) Step On



8. İki tanımlayıcı aynı isme sahipse ve aynı kapsam içinde bildirilirse aşağıdakilerden hangisi söylenemez?
- A) Parametre sayıları farklı olmalıdır.
  - B) Parametre türleri farklı olmalıdır.
  - C) Geri dönüş türü farklı olmalıdır.
  - D) Overload yapılmış olabilir.
9. Bir sınıf kapsamında tanımlanmış değişkenlere ne ad verilir?
- A) Field
  - B) Variable
  - C) Metot
  - D) Debug
10. Return ifadesiyle ilgili olarak aşağıdakilerden hangisi yanlıştır?
- A) return ifadesinin döndürdüğü veri türü ile metodun döndürdüğü veri türü aynı olmalıdır.
  - B) return ifadesi metodun sonunda bulunmak zorundadır.
  - C) return ifadesinden sonra noktalı virgül konulmalıdır.
  - D) Her metoda return ifadesi bulunmak zorundadır.

## 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 öğretmeninize başvurunuz.

# CEVAP ANAHTARLARI

## ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

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

## ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

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

## MODÜL DEĞERLENDİRMENİN CEVAP ANAHTARI

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

# KAYNAKÇA

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