İLERİ DÜZEY METOT İŞLEMLERİ

Atatürk Üniversitesi
Açıköğretim Fakültesi

İÇİNDEKİLER

- Metot Parametrelerinde Varsayılan Değer Atama İşlemi
- Parametrelerde "Ref" ve "Out" Anahtar Sözcüklerinin Kullanımı
- Metotlara Aşırı Yüklenme

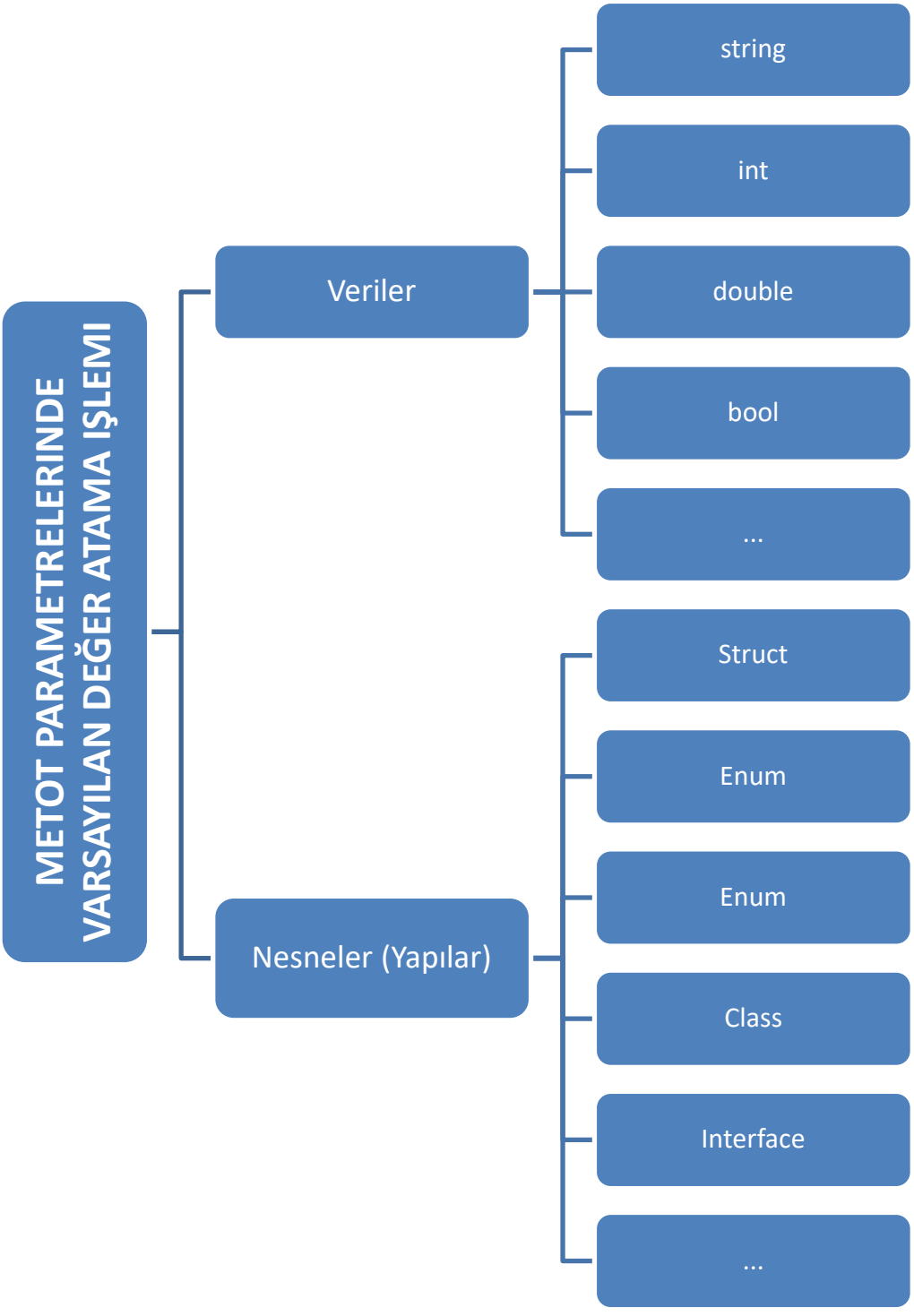
HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Metotların parametrelerinde varsayılan değerlerin kullanımı hakkında bilgi sahibi olabilecek,
  - "ref" ve "out" parametrelerinin kullanımına hakim olabilecek,
  - Metotların aşırı yüklenme sonrası kullanımları hakkında bilgi sahibi olabilecek,
  - Metotlarda parametrelerin kullanımları öğrenebileceksiniz.

NESNE TABANLI PROGRAMLAMA I

Dr. Öğr. Üyesi Muhammed Fatih ALAEDDİNOĞLU

ÜNİTE 10


METOT PARAMETRELERINDE VARSAYILAN DEĞER ATAMA IŞLEMI
Veriler
string
int
double
bool
...
Nesneler (Yapılar)
Struct
Enum
Enum
Class
Interface
...

GİRİŞ

Nesne tabanlı programlama kapsamında metotların temel kullanım amacı daha anlaşılır, fonksiyonel, kullanışlı ve basit çalışan kodlar üretmektir.

Metotlar kelime anlamı itibariyle yöntem, sistem, teknik, prosedür ve işlem gibi anlamlara gelmektedir. Nesne tabanlı programlama kapsamında metotların temel kullanım amacı daha anlaşılır, fonksiyonel, kullanışlı ve basit çalışan kodlar üretmektir.

Günümüzde açık kaynak kodlu yazılımlar, her geçen gün yaygınlaşmaktadır. Bilişim konusunda büyük firmalar ve onların geliştirmiş oldukları program ve sınıflarda bütün kullanım standartları belli iken kişilerin açık kaynak kodlar ile geliştirmiş oldukları yazılımlarda birçok kısıt, kabul ve istisnalarda belirsizlikler vardır. Bu sebeple bir takım problemler çıkmaktadır. Bu ünitenin yazımındaki örneklerde kullanmış olduğumuz C# dili çok yaygın ve etkin kullanıldığı için hemen hemen bütün istisna, kısıt ve kabulleri kaynaklarda belirtilmiştir. *Genellikle dilin mantığı ve yapısına göre özelleşen bu bilgiler, yazılımın performansına, verilerin güvenli işlenmesine, index(adresleme) ve dosyalama yapısına göre değişkenlik göstermektedir. C# nesne tabanlı programlama dilinde veri ve nesne yapılarının ilk varsayılan değerlerini bilmek, metotları oluştururken hem hataların önüne geçmeye hem de doğru değerler ile doğru sonuçlara erişmek için önemlidir.*

Normal bir metot oluşturulurken giriş parametreleri işlemlerden etkilenmezken “ref” ve “out” anahtar sözcükleri sayesinde giriş parametreleri üzerinde işlemler yapmak mümkündür.

C# programlama dilinde metotlar yazılırken, özellikle metotların giriş parametreleri ile ilgili dikkat edilmesi gereken özel durumlar vardır. Bu durumların başında değişken ve nesnelerin başlangıçta atanan varsayılan değer bulunmakta ve bu varsayılanlarına hakim olmak yazılımdaki etkinlik açısından önem arz etmektedir. Ayrıca bu değerler bir takım esnek kullanımlarla çok daha etkili olabilmektedir. Bu ünitede bahsi geçen etkin kullanımlardan ikisi olan “ref” ve “out” anahtar sözcüğünden bahsedilecektir. Bu anahtar sözcükler, metotların giriş parametreleri üzerinde bir takım işlemleri gerçekleştirme için kullanılır. Yani “ref” ve “out” anahtar sözcüklerini kullanmaksızın normal bir metot oluşturulurken giriş parametreleri metot içerisindeki işlemlerden etkilenmezken “ref” ve “out” anahtar sözcükleri sayesinde giriş parametreleri üzerinde işlemler yapmak mümkündür. “ref” ve “out” anahtar sözcükleri, verinin hafızadaki referansları üzerinden gerçek değere erişip gerekli işlemleri yapmayı mümkün kılar.

METOT PARAMETRELERİNDE VARSAYILAN DEĞER ATAMA İŞLEMİ

Nesne tabanlı programlama için kullanılacak en etkin dillerden biri C# dilidir.

Yazılım geliştirilirken genellikle veriler üzerinde işlemler yapılmaktadır. C# programlama dilinde bu verilerle(değişken) işlem yapılmadan önce verilerin türleri (int, double, string vs.) belirtilmeli ve genellikle belirtildiği esnada değer atama işlemleri gerçekleşmelidir. Ancak C# programlama dilinde değişkenler tanımlanırken değer atama işlemlerinin gerçekleşmesi zorunlu değildir. Yazılımcı açısından çok büyük rahatlık sağlayan bu durumdan olumsuz etkilenmemek için değişkenlerin varsayılanlarını bilmek ve metotları buna göre oluşturmak gerekmektedir. Metotları daha kolay anlamaya yönelik yapılacak örnekler için öncelikle Visual Studio platformu ve C# programlama dilinde gerçekleştirilecektir. Daha önceki ünite(Nesne Tabanlı Programlama Ünite 8) de yeni bir proje

oluşturma adına Visual Studio platformu ve C# dili için gerekli ayarların seçilmesi ile ilgili işlemler anlatılmıştır.

Örnek

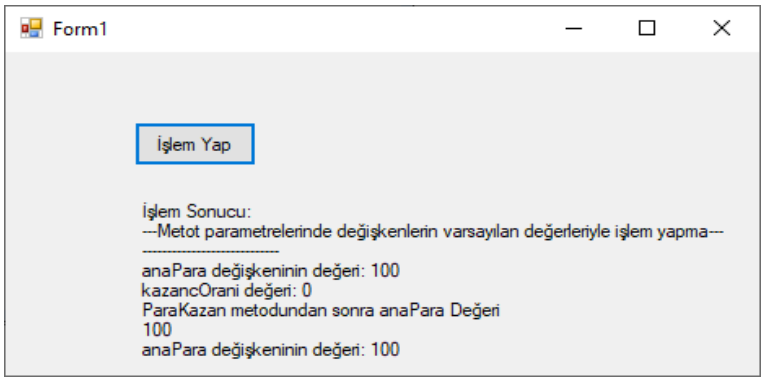
•Değişkenlerin varsayılan değerlerinin metot giriş parametrelerinde kullanımına ait basit bir örnek oluşturalım.

Nesne tabanlı programlama için kullanılacak en etkin dillerden biri C# dilidir.

Proje ilk oluşturulduğu zaman "Program.cs" adında bir sınıf otomatik oluşturulur.

```
private void IslemYap_Click(object sender, EventArgs e)
{
    double anaPara = 100; double kazancOrani=default(double);
        IslemSonucu.Text = "İşlem Sonucu:" + "\n" +
     "---Metot parametrelerinde değişkenlerin varsayılan değerleriyle işlem
     yapma---" + "\n" +
         "----------------------------" + "\n"+
     "anaPara değişkeninin değeri: " + anaPara+ "\n"+ "kazancOrani değeri: " +
     kazancOrani+ "\n"+
     "ParaKazan metodundan sonra anaPara Değeri" + "\n"+ParaKazan(anaPara, ka-
     zancOrani) + "\n";
     IslemSonucu.Text = IslemSonucu.Text + "anaPara değişkeninin değeri: " +
     anaPara;
}
public static double ParaKazan(double anaPara,double kazancOrani)
{
    anaPara = anaPara+anaPara*kazancOrani/100;
    return anaPara;
}
```

Yukarıdaki örnek uygulamada "kazancOrani" değişkenine ait herhangi bir değer belirtilmemiştir. Derleyici bu durumda "double" türündeki "kazancOrani" değişkenine bir değer atanmasını istemektedir. Aksi halde programın çalışmasını engellemektedir. Ancak değişkene değer atanmaması gereken durumlar olduğu zaman değişkene ait varsayılan(default) değerin verilmesi gerekmektedir. "double" veri türü için bu değer "default(double)" veya "default" şeklinde ifade edilmektedir. Ayrıca değişken genel(global) olarak tanımlanmış olsaydı "default" ifadesine de gerek kalmayacaktı. Ancak "kazancOrani" değişkenini yerel olarak yani metodun içinde tanımladığımız için "default" ifadesiyle varsayılan değer belirtilmelidir.



Resim 10.1. Metot Parametrelerinde Değişkenlerin Varsayılan Değerleriyle İşlem Yapma

Resim 10.1.'de görüldüğü gibi "kazancOrani" değişkenin varsayılan değeri "0(sıfır)" olduğundan örnek metodun sonucu değişmemektedir. Benzer şekilde diğer veri türleri içinde varsayılan değerlerin atanması gereken durumlarda varsayılan değer atamalarını örnek üzerinde görmek uygun olacaktır.

Örnek

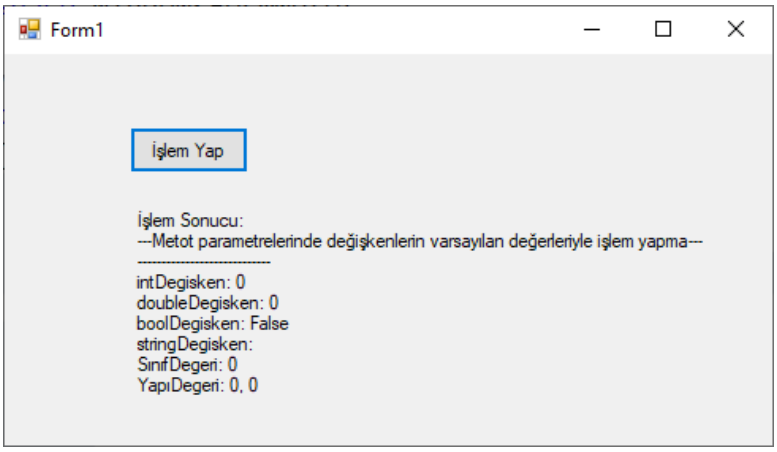
- Metotlarda giriş parametresi olarak kullanılan veri ve nesnelerin bazılarının varsayılan değerlerinin gösterimlerini örnek üzerinde görelim.

```
public double kar;
int intDegisken; double doubleDegisken;
bool boolDegisken; string stringDegisken;
public Form1()
{
    InitializeComponent();
}
public Form1(double kar)
{
}
private void IslemYap_Click(object sender, EventArgs e)
{  Form1 form1 = new Form1(kar);
    Yapi yapi = new Yapi();

        IslemSonucu.Text = "İşlem Sonucu:" + "\n" +
        "---Metot parametrelerinde değişkenlerin varsayılan değerleriyle
        işlem yapma---" + "\n" +
         "----------------------------" + "\n"+
            "intDegisken: " + intDegisken + "\n"+ "doubleDegisken: " + double-
     Degisken + "\n"+
     "boolDegisken: " + boolDegisken+ "\n" + "stringDegisken: " + string-
     Degisken + "\n"+
      "SınıfDegeri: " + VarsayilanSinifDegeri(form1) + "\n"+
      "YapıDegeri: " + yapi.X+ ", "+yapi.Y + "\n";
}

public static double VarsayilanSinifDegeri(Form1 form1)
{
    return form1.kar;
}
public struct Yapi
{
    public Yapi(double x, double y)
    {
        X = x;
        Y = y;
    }
    public double X { get; }
    public double Y { get; }
    public override string ToString() => $"({X}, {Y})";
}
```

Yukarıdaki örneklerde görüldüğü gibi değişken ve yapılardan oluşan birçok durum herhangi bir değer ataması yapmaksızın çağrılıp kullanılabilir.



Resim 10.2. Metotlarda Giriş Parametresi Olarak Kullanılan Veri Ve Nesnelerin Bazılarının Varsayılan Değerlerinin Gösterimlerimi

Resim 10.2.'de gösterildiği gibi değişkenler(int, double, bool, string) ve nesneler(sınıf, yapı) varsayılan değerleri kullanılabilmektedir. İlk olarak örnekte gösterildiği gibi "int" türünde bir değişken genel(global) yani sınıfın altında tanımlanmıştır. Bu durumda "int" veri türünün varsayılan değeri olan "0(sıfır)" Resim 10.2.'de gösterildiği gibi atanmıştır. Örnekte gösterilen diğer değişkenler olan "double" veri türündeki "doubleDegisken" "0(sıfır)" değerini, "bool" veri türündeki "boolDegisken" "false" değerini "string" veri türündeki "stringDegisken" "boş değer yani null" değerini varsayılan olarak almıştır. Değişkenlerde varsayılan değerler kullanıldığı gibi "Sınıf" türündeki ve "VarsayilanSinifDegeri" metodunun giriş parametresi olarak tanımlanmış "form1" nesnesi bir değer almaksızın altındaki "kar" değişkenini varsayılan değeriyle kullanmaya izin vermektedir. Benzer şekilde "Nesne" adındaki yapılar (struct) da herhangi bir değer almaksızın altında bulunan değişkenlerin varsayılan değerleriyle çağrılıp kullanılabilmektedir.

PARAMETRELERDE "REF" VE "OUT" ANAHTAR SÖZCÜKLERİNİN KULLANIMI

C# nesne tabanlı programlamada ise nesne mantığına uygun, sınıf ve metotların kullanımını kolaylaştıracak yapılara rastlamak mümkündür.

Hemen hemen her programlama dilinin, kullanıcıların işlerini kolaylaştırıcı kendine özgü özellikleri vardır. Bu özellikler ilgili dilin veri yapısı(veriyi çağırma, hafızada tutma, adresleme ve veriler arası ilişkiler kurma) ve mantıksal işleyişine göre şekillenmektedir. C# nesne tabanlı programlamada ise nesne mantığına uygun, sınıf ve metotların kullanımını kolaylaştıracak yapılara rastlamak mümkündür. Bu yapılardan iki tanesi de metot giriş parametrelerindeki verilerin gerektiğinde değişmesini sağlayan "ref" ve "out" anahtar sözcükleridir. *Bu anahtar sözcükleri, değişkenin (class, struct, enum gibi nesne belirten yapılar da olabilir.) hafızadaki yerine giderek değişkenin bizzat kendisi ile işlem yapılmasını sağlar.* Böylece değişken üzerine yapılan işlem sonrasında değişikliği görmek mümkün olur. Normal şartlarda metotlara dışarıdan giriş parametresi olarak verilen değişken, işlem sonrasında değeri korumaktadır. Çünkü giriş parametresi olarak verilen değişkenin bir kopyası oluşturulmakta ve kullanılmaktadır. Sonuç olarak asıl değişkenin değerinde herhangi bir değişiklik olmamaktadır.

"ref" ve "out" anahtar sözcüklerinin daha iyi anlaşılabilmesi için örnek üzerinde görmek uygun olacaktır.

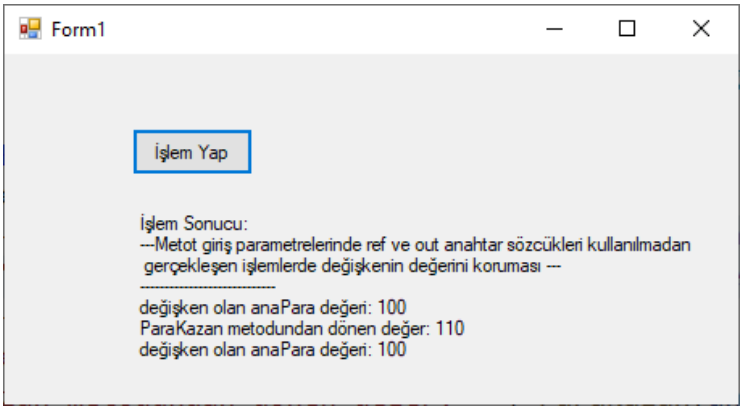
Örnek

- Metotlarda giriş parametrelerinin "ref" ve "out" anahtar sözcükleri ile ifade edilmesi ve kullanım özelliklerini örnek üzerinde görelim.

```
private void IslemYap_Click(object sender, EventArgs e)
{
   double anaPara = 100;
   IslemSonucu.Text = "İşlem Sonucu:" + "\n" +
    "---Metot giriş parametrelerinde ref ve out anahtar sözcükleri kullanılmadan "
    + "\n" +" gerçekleşen işlemlerde değişkenin değerini koruması ---" + "\n" +
    "----------------------------" + "\n" +
    "değişken olan anaPara değeri: " + anaPara + "\n" +
    "ParaKazan metodundan dönen değer: " + ParaKazan(anaPara, 10) + "\n";

   IslemSonucu.Text = IslemSonucu.Text + "değişken olan anaPara değeri: " + anaPara;
}
public static double ParaKazan(double anaPara, double kazancOrani)
{
  anaPara = anaPara + anaPara * kazancOrani / 100;
   return anaPara;
}
```

Yukarıdaki örnekte ilk olarak "ref" ve "out" ifadeleri kullanılmadan "anaPara" değişkeninin değeri ile ilgili işlem yapılmıştır. Bu örnekte tanımlanan "anaPara" değişkenine ilk olarak 100 değeri atanmıştır. Daha sonra bu değişken "ParaKazan" metoduna giriş parametresi olarak verilmiştir.



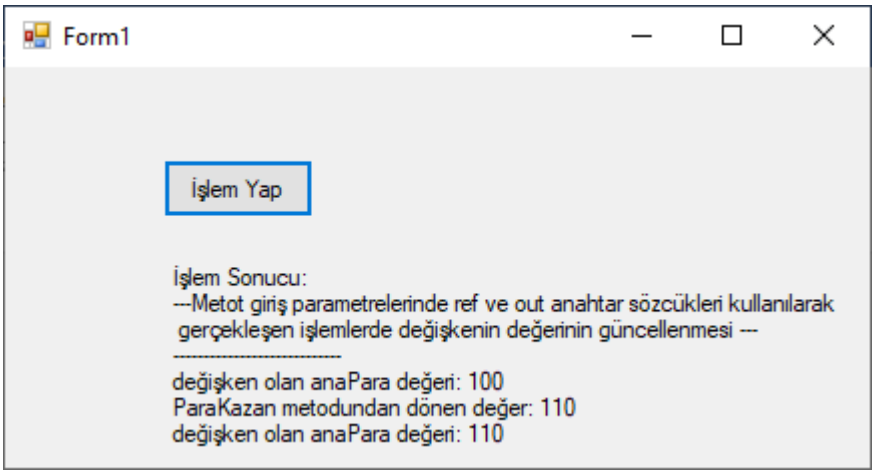
Resim 10.3. Metot Giriş Parametrelerinde "Ref" ve "Out" Anahtar Sözcükleri Kullanılmadan Gerçekleşen İşlemlerde Değişkenin Değerini Koruması

Metot içerisinde işlem sonuçlarında görüldüğü gibi metot içerisinde ilgili işlemler gerçekleşirken “anaPara” değişkenine %10 oranında kar eklenmiş ve “anaPara” değişkenine atanmıştır. Ancak bu işlemler esnasında metot içerisinde “anaPara” değişkeninin bir kopyası oluşturulmuş ve bu kopyaya yeni değer atanmıştır. Bu sonuç “ParaKazan” metoduna ait dönüş değerinde görülmektedir. Daha sonra tekrardan asıl “anaPara” değişkeninin değeri gösterilmek istenmiştir. Resim 10.3.’de gösterildiği gibi “anaPara” değişkenine ilk olarak atanan 100 değeri, metot içerisinde işlem gördükten sonra “ParaKazan” metodunun dönüşünde 110 değeri olarak geri dönmüştür. Ancak buradaki değişken, mantık olarak metottan sonra yeni değeri ile işlem görmesi gerekirken gerçek “anaPara” değişkeninin bir kopyası olduğundan asıl “anaPara” değişkenini etkilememekte ve ilk atanan 100 değerini korumaktadır. Bu durum, “ref” ve “out” anahtar sözcükleriyle kolay bir şekilde “anaPara” değişkeninin güncellemesi ile sonuçlanabilir. Aksi halde her işlemden değişkenin değerine yeniden atama işlemi yapmak gerekmektedir. Şimdi ise ref ve “out” ifadeleriyle değişkenin değerinin metot içerisinde güncellenmesini aşağıdaki örnek kodlar üzerinde görmek uygun olacaktır.

```
private void IslemYap_Click(object sender, EventArgs e)
{
   double anaPara = 100;
   IslemSonucu.Text = "İşlem Sonucu:" + "\n" +
    "---Metot giriş parametrelerinde ref ve out anahtar sözcükleri kullanılarak "
    + "\n" +" gerçekleşen işlemlerde değişkenin değerinin güncellenmesi ---" + "\n" +
    "----------------------------" + "\n" +
    "değişken olan anaPara değeri: " + anaPara + "\n" +
"ParaKazan metodundan dönen değer: " + ParaKazan(ref anaPara, 10) + "\n";

   IslemSonucu.Text = IslemSonucu.Text + "değişken olan anaPara değeri: " + anaPara;
}
public static double ParaKazan(ref double anaPara, double kazancOrani)
{
  anaPara = anaPara + anaPara * kazancOrani / 100;
   return anaPara;
}
```

Yukarıda önceki kod örneğinden farklı olarak sadece metot içerisinde kullanılan giriş parametrelerinden önce “ref” sözcüğü kullanılmıştır. Aynı zamanda kullanılan metodun tanımlanması esnasında da “ref” sözcüğü kullanılması gerekmektedir. Resim 10.4.’da görüldüğü gibi “ParaKazan” metodundan sonra “anaPara” değişkeni 100 değerinden 110 değerine güncellenmiştir.



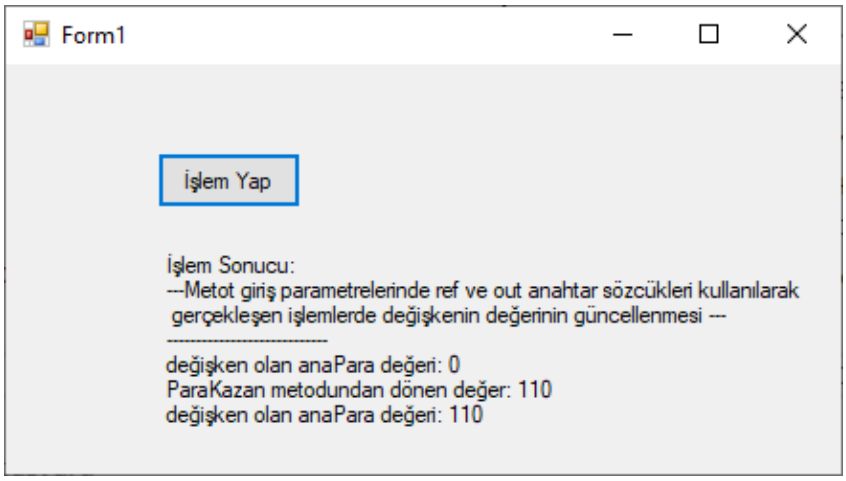
Resim 10.4. Metot Giriş Parametrelerinde "Ref "ve "Out" Anahtar Sözcükleri Kullanılarak Gerçekleşen İşlemlerde Değişkenin Değerinin Güncellenmesi"

Örnekte "ref" anahtar sözcüğü kullanılmıştır. Ancak "ref" yerine "out" sözcüğü de kullanılabilir. *Aralarındaki en temel fark "out" anahtar sözcüğü kullanıldığı zaman giriş parametresi olan değişkene başlangıçta değer verme zorunluluğumuz olmamasıdır.* Yani "out" anahtar sözcüğü kullanılmak istenildiği zaman değişkenin değeri metodun içerisinde tanımlanabilir. Aşağıdaki kod örneğinde bu durumu anlatan örnek belirtilmiştir.

```
private void IslemYap_Click(object sender, EventArgs e)
{
   double anaPara=default;
   IslemSonucu.Text = "İşlem Sonucu:" + "\n" +
    "---Metot giriş parametrelerinde ref ve out anahtar sözcükleri kullanılarak "
    + "\n" +" gerçekleşen işlemlerde değişkenin değerinin güncellenmesi ---" + "\n" +
    "----------" + "\n" +
    "değişken olan anaPara değeri: " + anaPara + "\n" +
"ParaKazan metodundan dönen değer: " + ParaKazan(out anaPara, 10) + "\n";

   IslemSonucu.Text = IslemSonucu.Text + "değişken olan anaPara değeri: " + anaPara;
}
public static double ParaKazan(out double anaPara, double kazancOrani)
{
   anaPara = 100;
  anaPara = anaPara + anaPara * kazancOrani / 100;
   return anaPara;
}
```

Aşırı yüklemeler C# dilinde çok yaygın ve etkin kullanıma sahiptir.



Resim 10.5. Metot Giriş Parametrelerinde "Ref "ve "Out" Anahtar Sözcükleri Kullanılarak Gerçekleşen İşlemlerde Değişkenin Değerinin Güncellenmesi"

"ref" ve "out" anahtar sözcüklerinin bir diğer kullanım amacı da; metotlar sadece 1 geri dönüş değeri döndüren metotlardan birden fazla değeri güncellemektir.

Yukarıdaki kod örneğinde metoda ait giriş parametresi "out" anahtar sözcüğü ile kullanılmıştır. Böylece parametrenin değerini metot içerisinde verebilme imkânı doğmuştur. Resim 10.5.'de "anaPara" değişkeninin değeri ile ilgili değişimleri görmek mümkündür.

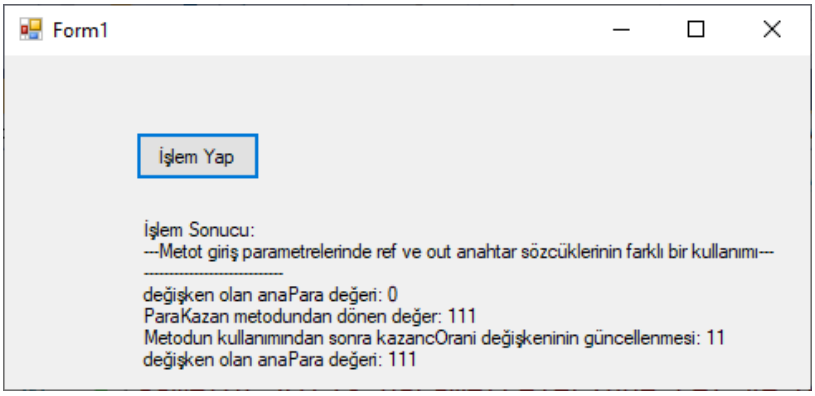
"ref" ve "out" anahtar sözcüklerinin bir diğer kullanım amacı da; metotlar sadece 1 geri dönüş değeri döndüren metotlardan birden fazla değeri güncellemektir. *Ancak bazı metotlarda birden fazla değerin hesaplanarak değişkenlere atanması gerekebilir. Böyle durumlarda da "ref" ve "out" anahtar sözcüklerini kullanmak ve bu kullanımı örnek üzerinde görmek uygun olacaktır.*

Örnek

- Metotlarda giriş parametrelerinin "ref" ve "out" anahtar sözcükleri ile birden fazla dönüş değeri döndürmesini sağlayan örneği gösterelim.

```
private void IslemYap_Click(object sender, EventArgs e)
{
   double anaPara=default;
   IslemSonucu.Text = "İşlem Sonucu:" + "\n" +
    "---Metot giriş parametrelerinde ref ve out anahtar sözcükleri kullanılarak "
    + "\n" +" gerçekleşen işlemlerde değişkenin değerinin güncellenmesi ---" + "\n" +
    "----------" + "\n" +
    "değişken olan anaPara değeri: " + anaPara + "\n" +
"ParaKazan metodundan dönen değer: " + ParaKazan(out anaPara, 10) + "\n";

   IslemSonucu.Text = IslemSonucu.Text + "değişken olan anaPara değeri: " + anaPara;
}
public static double ParaKazan(out double anaPara, double kazancOrani)
{
   anaPara = 100;
  anaPara = anaPara + anaPara * kazancOrani / 100;
   return anaPara;
}
```



Resim 10.6. Metot Giriş Parametrelerinde Ref ve Out Anahtar Sözcüklerinin Farklı Bir Kullanımı

Aşırı yüklemeler C# dilinde çok yaygın ve etkin kullanıma sahiptir.

Yukarıdaki örnekte ve Resim 10.9.'da gösterildiği gibi "ref" ve "out" anahtar sözcükleriyle birden fazla değişkenin değeri güncellenmiştir. Bu örnekte bir giriş parametre değişkeni "ref", diğer giriş parametre değişkeni ise "out" anahtar sözcüğüyle ile ifade edilmiştir. *Yukarıda da bahsedildiği gibi "ref" ve "out" arasında fark ise giriş parametresinin başlangıçta atanması veya atanmamasıdır. Bu sebeple bu sözcükler birbiri yerine genellikle kullanılabilir.*

Aşırı yüklemeler oluşturulan metotlarda giriş parametreleri farklı tip veya sayıda olduğundan aynı isimde tanımlanmaktadır.

AŞIRI YÜKLEMELER(OVERLOADING)

Çoğu nesne tabanlı programlama dilinde oldukça fazla avantaj sunan ve C# dilinde ise çok önemli bir özellik olan bu yapı aynı isimde ancak farklı giriş parametreler ile tanımlanmış metotlardır. Yani aynı isme sahip birden fazla metot aynı sınıf içinde farklı görevlere hizmet edecek şekillerde ifade edilebilir. Ancak dikkat edilmesi gereken çok önemli bir husus ise aynı isimde tanımlanan bu metotların bütün parametreleri aynı sıra ve aynı veri tiplerinde olamayacağıdır. Buna göre en az bir giriş parametresi diğerlerinden farklı olmalıdır. Bu durumu bir örnek ile açıklamak daha uygun olacaktır.

Örnek

- Aşırı yüklemeler(overloading) konusunun anlaşılması adına örnek bir uygulamayı göstermek uygun olacaktır. Bu örnekte bir metodun aynı isimde farklı görevlerde kullanılması gösterilmektedir.

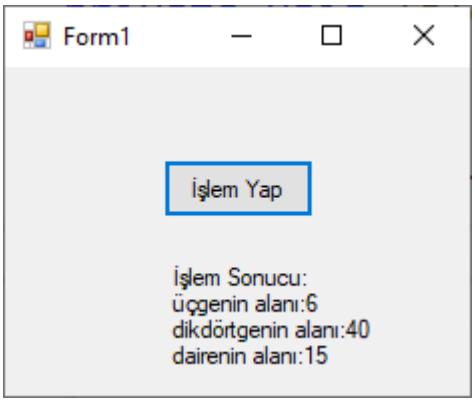
```
public double AlanHesapla (int kenar1, int kenar2, double aci)
{
  double b = (aci * (Math.PI)) / 180;
  return ((double)kenar1 * (double)kenar2 * Math.Sin(b)) / 2;
 }

public int AlanHesapla (int yatayKenar, int dikeyKenar)
{
  return yatayKenar * dikeyKenar;
}

public int AlanHesapla (int yariCap)
{
 return (int)(Math.PI * yariCap);
}

private void IslemYap_Click(object sender, EventArgs e)
{
  IslemSonucu.Text="İşlem Sonucu: "+"\n" +
    "üçgenin alanı:" +AlanHesapla (4, 6, 30.0)+"\n"
    +"dikdörtgenin alanı:"+ AlanHesapla (5, 8) + "\n"
    +"dairenin alanı:"+AlanHesapla (5);
}
```

Örnekte gösterildiği gibi “AlanHesapla” metodu üç defa tanımlanmıştır. Ancak her tanımlamada giriş parametreleri farklı tip veya sayıda olduğundan aynı isimde tanımlanmaktadır. Bu durum C# programlama dilinde “Overloading” olarak tanımlanmakta ve hatasız bir şekilde çalışmaktadır. Resim 10.7.’de gösterildiği gibi bir sonuç sayfasına ulaşılmaktadır.



Resim 10.7. Aşırı Yükleme (Overloading) ile Aynı İsme Sahip Farklı Metotlara Ait Sonuçlar

Aşırı yüklemeler metotlarda sıkça başvurulan bir yapıdır.

Aşırı yüklemeler metotlarda sıkça başvurulan bir yapıdır. Aynı işi yapan fakat farklı parametreler ile farklı durumlar için sonuçlar üreten metotlar geliştirebilmek yazılımcı açısından oldukça esneklik sağlamaktadır. *C# programlama dilinde yazılımcıların hizmetine sunulmuş çok fazla aşırı yükleme ile oluşturulmuş hazır sınıf ve metot bulunmaktadır.* Bunlardan bir tanesi "String" sınıfına ait "Format" metodu için oluşturulmuş aşırı yüklemelere ait örnek bir kullanım gösterilmiştir.

```
public static String Format(String format, object arg0);
public static String Format(String format, object arg0, object arg1, object
      arg2);
public static String Format(String format, params object[] args);
public static String Format(String format, object arg0, object arg1);
public static String Format(IFormatProvider provider, String format, object
      arg0, object arg1, object arg2);
public static String Format(IFormatProvider provider, String format, params
      object[] args);
public static String Format(IFormatProvider provider, String format, object
      arg0, object arg1);
public static String Format(IFormatProvider provider, String format, object
      arg0);
```

Bireysel Etkinlik

- Değişkenlere ait default değerlerin olmasının hafızayı etkin kullanma açısından önemi nedir?
- "ref" ve "out" gibi yazılımcının işini kolaylaştıran başka anahtar sözcükleri var mıdır?

Özet

- Nesne tabanlı programlama için metotlar, oldukça sık başvurulan, esneklik sağlayan ve algoritmik açıdan çok büyük önem arz eder. Metotlar, sistemlerin, projelerin ve otomasyonların etkin ve doğru şekillerde gerçekleşmesi için kullanılır. Her geçen gün yaygınlığı ve etkinliği artan yazılımlar problemin karmaşıklığına bağlı olarak çok kolay veya zor algoritmalar içerdiğini söylemek mümkündür. Ancak algoritmalar zorlaştıkça yapıya hakimiyet ve müdahale de oldukça zorlaşmakta hatta imkansız hale gelmektedir. Bu sebeple yazılımcılar tarafından kullanılan metotlardaki esneklik, problemleri çözmek için kolaylıklar sağlar. Benzer şekilde metotlarda da giriş parametreleri yani değişkenlerin kullanımlarındaki esneklikler oldukça önemlidir. Her geçen gün önemi artan yazılım geliştirme, hemen hemen her iş için değişkenlik göstermekte ve farklı yapılar ile kolaylaştırılmaya çalışılmaktadır.
- C# dilini geliştiren kişiler ise yazılımcıların işlerini kolaylaştırmak, gerekli kontroller sağlamak ve hafızayı etkin kullanmak için değişkenlerin, sınıfların ve giriş parametresi olabilecek herhangi bir yapının kullanımını en basit hale getirmek için bazı teknikler geliştirmektedirler. Bu bağlamda C# nesne tabanlı programlamada, nesne mantığına uygun, sınıf ve metotların kullanımını kolaylaştıracak yapılar bulunmaktadır. Bunlardan bir tanesi, değişken veya nesnelere ait default yani varsayılan değerleri arka planda tanımlamaktır. Bir diğeri ise metot giriş parametrelerindeki verilerin gerektiğinde değişmesini sağlayan “ref” ve “out” anahtar sözcükleridir. Bu anahtar sözcükleri değişkenin hafızadaki yerine giderek değişkenin bizzat kendisi ile işlem yapılmasını sağlar. Böylece değişken üzerine yapılan işlem sonrasında değişikliği görmek mümkün olur. Normal şartlarda metotlara dışarıdan giriş parametresi olarak verilen değişken, işlem sonrasında önceki değeri korumaktadır. Çünkü giriş parametresi olarak verilen değişkenin bir kopyası oluşturulmakta ve asıl değişkene dokunmamaktadır. Sonuç olarak asıl değişkenin değerinde herhangi bir değişiklik olmamaktadır.
- Metotlar kullanım amaçlarına göre geri dönüş değerlerine sahip olabilmektedir. Ancak bir metodun tekbir geri dönüş değeri olduğu için "ref" ve "out" anahtar sözcükleriyle dönüş değerlerini artırmak mümkün olabilmektedir. Yani metodun yapması gereken işlemi gerçekleştirmesinden sonra return ifadesi ile metodun sonucu olarak bir değer döndürmesi işlemidir.
- Çoğu nesne tabanlı programlama dilinde oldukça fazla avantaj sunan ve C# dilinde ise çok önemli bir özellik olan bu yapı aynı isimde ancak farklı giriş parametreler ile tanımlanmış metotlardır. Yani aynı isme sahip birden fazla metot aynı sınıf içinde farklı görevlere hizmet edecek şekillerde ifade edilebilir. Ancak dikkat edilmesi gereken çok önemli bir husus ise aynı isimde tanımlanan bu metotların bütün parametreleri aynı sıra ve aynı veri tiplerinde olamayacağıdır. Buna göre en az bir giriş parametresi diğerlerinden farklı olmalıdır.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi metotlarda kullanılan giriş parametre türlerinden biri değildir?
   a) string
   b) struct
   c) class
   d) double
   e) static

2. Aşağıdakilerden hangisi metot için giriş parametresi olan string türündeki bir değişkenin varsayılan değerini almasını sağlamaz?
   a) `string deger= default;`
   b) `string deger= default(string);`
   c) `string deger= " ";`
   d) `string deger;`
   e) `string deger= null;`

3. Aşağıdakilerden hangisi nesne tabanlı programlamadaki metot ile benzer anlamda kullanılmamaktadır?
   a) Yöntem
   b) Fonksiyon
   c) Prosedür
   d) Teknik
   e) Sınıf

4. Aşağıdakilerden hangisi “ref” ve “out” anahtar sözcüklerini en iyi şekilde tanımlar?
   a) Değişkenin (class, struct, enum gibi nesne belirten yapılar da olabilir.) başka bir kopyasını oluşturarak işlem yapılmasını sağlar.
   b) Değişkenin (class, struct, enum gibi nesne belirten yapılar da olabilir.) hafızadaki yerine giderek değişkenin bizzat kendisi ile işlem yapılmasını sağlar.
   c) Değişkenin (class, struct, enum gibi nesne belirten yapılar da olabilir.) değerini geçici olarak değiştirir sonra eski değerini verir.
   d) Değişkenin (class, struct, enum gibi nesne belirten yapılar da olabilir.) ilk değerinin verilmesini sağlar.
   e) Değişkenin (class, struct, enum gibi nesne belirten yapılar da olabilir.) başka metotlar tarafından kopyalanmasını sağlar.

5. Aşağıdakilerden hangisi “ref” ve “out” anahtar sözcüklerinin arasındaki farkı en iyi şekilde tanımlar?
   a) “out” anahtar sözcüğünün kullanımında giriş parametresinin başlangıç değerinin verilmesini zorunlu koşarken “ref” için zorunlu değildir.
   b) “ref” ve “out” anahtar sözcüklerinin arasında hiçbir fark yoktur.
   c) “ref” anahtar sözcüğünün kullanımında giriş parametresinin başlangıç değerinin verilmesini zorunlu koşarken “out” için zorunlu değildir.
   d) “ref” anahtar sözcüğü sınıflarda kullanılırken ve “out” anahtar sözcüğü değişkenlerde kullanılır.
   e) “ref” ve “out” anahtar sözcükleri tamamen farklı olup birbirlerinin yerine asla kullanılmazlar.

6. Aşağıdakilerden hangisi “out” anahtar sözcüğünün metotlarda kullanımına uygundur?
   a) `public void outKullanimi (out double deger){deger ++;}`
   b) `public void outKullanimi (double out deger){deger ++;}`
   c) `public void outKullanimi (out deger double){deger ++;}`
   d) `public void outKullanimi (deger out double){deger ++;}`
   e) `public void outKullanimi (out deger){deger ++;`

7. Aşağıdakilerden hangisi aşırı yüklemeler(overloading) özelliğini açıklayan cümledir?
   a) Aşırı yüklenerek nesne tabanlı programın kasılmasını sağlayan özelliktir.
   b) Aynı sınıf içinde aynı isim ve aynı giriş parametreleri ile sonsuz sayıda metot tanımlanabilir.
   c) Aşırı yüklemeler özelliğinin kullanımı her geçen gün yok olmakta ve programın bozulmasına sebep olmaktadır.
   d) Aynı sınıf içinde aynı isimde fakat farklı giriş parametreleri ile tanımlanan metotlar farklı görevleri yapabilmektedir.
   e) Aşırı yüklemeler sadece metotlarda değil aynı zamanda değişkenlerde de kullanılabilir.

8. Aşağıdakilerden hangisi “ref” ve “out” anahtar sözcüklerinin sağladığı avantajlardan biri değildir?
   a) Metotların kullanımını kolaylaştırma birlikte çok esnek kullanımlar sunmaktadır.
   b) Metotlara ait giriş parametrelerini kopyasını oluşturarak asıl değişkenin değerini korur.
   c) Metotlara ait giriş parametrelerinin metodun çağrılması sonucunda güncellenmesini sağlayarak daha etkin kullanımlar sunar.
   d) Metotların dönüş değerlerinde birden fazla değerin güncellenmesini sağlar.
   e) Metotlara ait giriş parametrelerini referans alarak asıl değişkenin değeri üzerinde işlem yapılmasını sağlar.

9. Aşağıdakilerden hangisi standart bir metot oluşturulurken zorunlu yapılardan biri değildir?
   a) Giriş parametresi
   b) Metot ismi
   c) “{}”(Süslü parantezler)
   d) “()”(Normal parantezler)
   e) Dönüş değeri(void ifadesi de dönüş ifadesidir)

10. Aşağıdaki metotlardan hangi ikili aşırı yüklenmiş metotlara uygun örnek olamaz?

a)
```
public static String Format(String format, object arg0);
public static String Format(String format, object arg0, object arg1);
```

b)
```
public static String Format(String format, object arg0);
public static String Format(String format, object arg0);
```

c)
```
public static String Format(String format, object arg0);
public static String Format(String format, string arg0);
```

d)
```
public static String Format(String format, object arg0);
public static String Format(object arg0, double arg1);
```

e)
```
public static String Format(String format, object arg0);
public static String Format();
```

Cevap Anahtarı
1.e, 2.c, 3.e, 4.b, 5.c, 6.a, 7.d, 8.b, 9.a, 10.b

YARARLANILAN KAYNAKLAR

Akbuğa, M.(2016). NESNE TABANLI PROGRAMLAMA-I Sınıflar. Erişim Adresi: https://docplayer.biz.tr/26099291-Unite-nesne-tabanli-programlama-i-okt-mustafa-akbuga-icindekiler-hedefler-siniflar.html. Erişim Tarihi:27.08.2021

Herbert, S. C. H. I. L. D. T. (2005). Herkes için C#. Alfa Yayınevi, 1.

Millî Eğitim Bakanlığı, (2017). Bilişim Teknolojileri-Metotlar. Ankara. Erişim Adresi: https://www.mku.edu.tr%2Ffiles%2F1874-8073aa76-3303-473a-9f28-2ac8face3cc8.pdf&usg=AOvVaw3FuYVZ0ibTbIaoDez-sSpa, Erişim Tarihi: 27.08.2021

Static (C# Başvurusu) (2020, 25 Eylül), Erişim Adresi: https://docs.microsoft.com/tr-tr/dotnet/csharp/language-reference/keywords/static, Erişim Tarihi:18.08.2021

Ünal, G. (2014)., C# “static” Kullanımı, Erişim Adresi: http://kod5.org/c-static-kullanimi/, Erişim Tarihi:27.08.2021

Yanık, M. (2009). C# a Başlangıç Kitabı. Erişim Adresi: https://docplayer.biz.tr/1951106-C-a-baslangic-kitabi.html Erişim Tarihi:29.08.2021