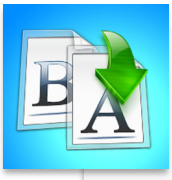
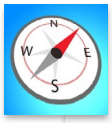
KURUCU VE STATİK METOTLAR



İÇİNDEKİLER

- Metotların Kullanım Çeşitleri
  - Yapıcı(Constructor) Metotlar
  - Kopyalayıcı Metotlar
  - Hazır Metotlar
  - Özyineleme(Recursive) Metotlar
- Metotlarda Özel İfade ve Yapılar
- Static Yapısı ve Kullanım Şekilleri



HEDEFLER

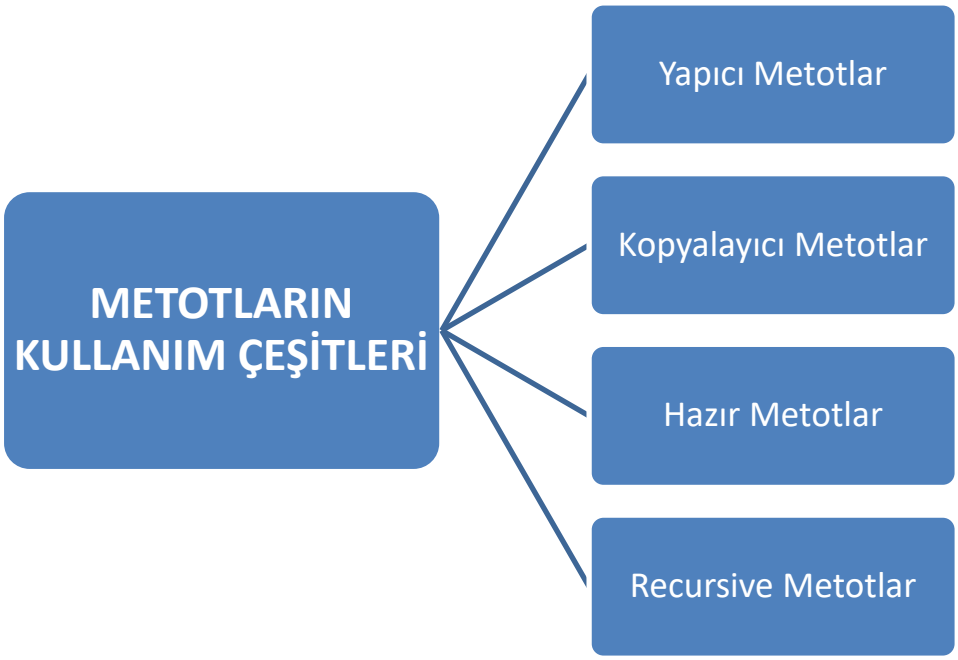
- Bu üniteyi çalıştıktan sonra;
  - Metotların kullanım çeşitleri hakkında bilgi sahibi olabilecek,
  - Yapıcı metodların nesne tabanlı programlamada kullanımına hakim olabilecek,
  - Metotların diğer kullanımları hakkında bilgi sahibi olabilecek,
  - Algoritma mantığında metotların kullanılmasının faydalarını anlayabilecek,
  - Statik metotların kullanımları hakkında detaylı bilgi sahibi olabileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

NESNE TABANLI PROGRAMLAMA I

Dr. Öğr. Üyesi
Muhammed Fatih ALAEDDİNOĞLU

ÜNİTE
9


METOTLARIN KULLANIM ÇEŞİTLERİ
Yapıcı Metotlar
Kopyalayıcı Metotlar
Hazır Metotlar
Recursive Metotlar

GİRİŞ

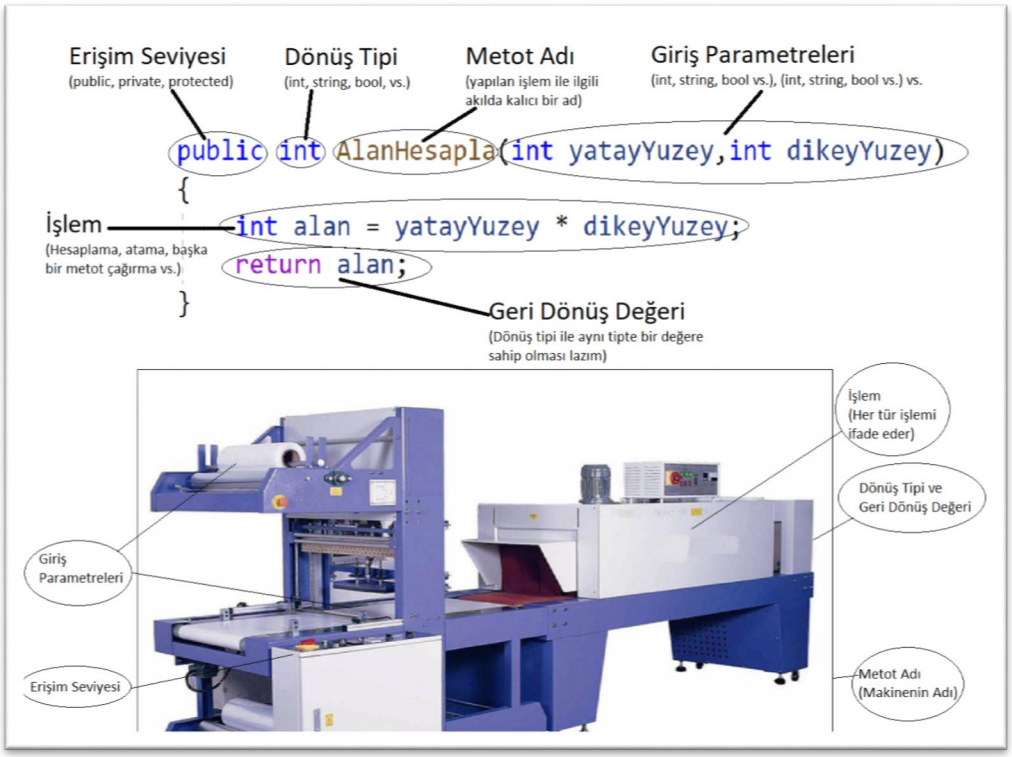
Yazılımlar, insanların zekâlarının ve hayallerinin erişebildikleri noktaları gerçekleştirdikleri yegâne araçlardır.

Teknolojinin gelişmesiyle birlikte donanımlar da gelişmektedir. Ancak bu gelişmede bir takım kısıtlar karşımıza çıkmaktadır. Ancak yazılımlar için bu kısıtlamalardan bahsetmek yanlış olur. Yazılımlar, insanların zekâlarının ve hayallerinin gidebildikleri bütün noktalara ulaşma yolunda çok daha hızlı ve etkin kullanılmaktadır. Bu hız ve etkinliğin en önemli faktörlerinden biri de üst seviye yazılımlardır. Üst seviye yazılımların da en etkin yöntemlerinden nesne tabanlı programlamanın en önemli öğelerinden biri metotlardır.

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

Metotlar yapısı gereği bir makine, bir alet veya bir fabrikaya benzemektedir.

Metotlar yapısı gereği bir makine, bir alet veya bir fabrikaya benzemektedir. Bu makine, alet veya fabrikada hammadde olarak içeriden veya dışarıdan verilen değer veya değerler belirli işlemlerden geçtikten sonra bir çıktı üretmektedir. Bu örnek Şekil 9.1'de metotlar ile sanayide kullanılan bir cihazın karşılaştırması özetlenmiştir.



Şekil 9.1. Metotları Anlatan Örnek Resim

Metotlar çeşitli kullanım alanlarına sahiptir. Bu alanlar hemen hemen bir yazılımın ihtiyacı olacak bütün alanları kapsamaktadır. *Metotlara yazılımların daha etkin daha basit ve daha anlaşılır olması için bir takım özellikler eklenmiştir.* C# programlama dilinde bu özellikler bir takım yapılara dönüştürülmüş ve ihtiyaçlara

Bu metot sınıfa ait bütün yapının bir kopyasının hafızada oluşmasını sağlayarak sınafa ait özelliklerin erişime bağlı kullanılmasını sağlamaktadır.

Static ifadesi metotlar oluşturulurken erişim belirleyicilerinin hemen sonrasında kullanılıp, metotların nesne tanımlanmadan direk kullanılmasını sağlamaktadır.

Metotların temel yapıları ile ilgili anlaşılmayan noktalar için nesne tabanlı programlamaya ait 8. Üniteye müracaat etmeleri uygun olacaktır.

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

göre yeniden güncellenmiştir. Bu yapıların genel olarak en sık kullanılanları aşağıdaki gibidir:

- Yapıcı metotlar
- Kopyalayıcı metotlar
- Hazır metotlar
- Özyinelemeli metotlar

Bu yapıların her biri metotların kullanımını ve faydasını artırmak için oluşturulmuştur. Örneğin yapıcı metotlar, C# programlama dilinde bir sınıf oluşturulduğu zaman bu sınıfa ait metotların direk kullanılabilmesi ve sınıftan nesne türetilebilmesi için arka planda otomatik olarak oluşturulmaktadır. *Benzer şekilde hazır metotlar olarak matematiksel, metinsel ve tarihsel işlemler metot yapılarında otomatik olarak projeye eklenmektedir.* Hemen hemen üst seviye bütün programlama dillerine eklenen bu ve benzeri özellikler ile yazılımcılar kodlarını daha hızlı yazmakta, aynı metot için aynı sonuçlara ulaşmakta ve başka bir programlama diline geçiş yaptığında yeni programlama diline kolayca adapte olabilmektedir.

Metotların kullanımında bir diğer önemli konu ise “static” ifadesidir. Bu ifade metotların erişim belirleyicilerinin hemen sonrasında kullanılıp, metotların nesne tanımlanmadan direk kullanılmasını sağlamaktadır. *Bu ifade aynı sınıf içinde kullanıldığı zaman sınıf ismine ihtiyaç duymazken başka sınıfta kullanımında erişim belirleyicisine bağlı olarak sınıf ismi ile birlikte direk kullanılabilmektedir.* Bu yapı sayesinde projenin hafızası daha etkin kullanılabilmektedir.

Kitabımızın bu bölümünün yani metotların kullanım çeşitlerinin ve static yapısının daha iyi anlaşılabilmesi için örnek uygulamalar, Visual Studio platformu C# programlama dilinde yapılacaktır. Ayrıca bu ünitenin daha iyi anlaşılabilmesi için metotların kullanımında özel ifade ve yapılar hakkında da bilgi vermek uygun olacaktır. Bu programlama dilinde metotlar çok aktif ve esnek şekillerde kullanılabilir. Yapılacak örnekler form ekranında yapılarak konunun daha iyi anlaşılması sağlanacaktır.

METOTLARIN KULLANIM ÇEŞİTLERİ

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.

Yapıcı(Constructor) Metotlar

Nesne tabanlı programlama dillerinden C# dilinde bir sınıf oluşturulduğu zaman bu sınıfa ait metotların direk kullanılabilmesi ve sınıftan nesne türetilebilmesi için arka planda otomatik olarak oluşturulan özel metoda yapıcı metot denir. Bu metodun ismi sınıf ile aynı isimde olmak zorundadır. Bu metot sınıfa ait bütün yapının bir kopyasının hafızada oluşmasını sağlayarak sınıfa ait

bütün değişken ve metot gibi özelliklerin erişim seviyelerine bağlı olarak kullanılmasını sağlamaktadır. *Yapıcı metotların iki temel özelliği vardır. İlki bu metotların dönüş tipleri yoktur. Bu kısım dönüş tipi void olan yapıya benzemekle birlikte void kelimesinin yazılmasına ihtiyaç duymaz. İkincisi ise bu metotlar geri değer döndürmezler.* Yani bu metotların içerisinde istenilen işlemler gerçekleşir. Ancak return ifadesi gibi yöntemler ile dönüşüne bir değer almazlar. Yapıcı metotlar erişim belirleyicisi olarak standart olarak public değer alırlar. Ancak özellikle sınıftan bir nesnenin oluşturulması istenmediği durumlarda private olarak belirtilebilirler. Yapıcı metotlar için dikkat edilmesi gereken bir diğer husus ise kullanıcı standart olarak tanımlanan metot ile aynı isimde bir metot oluşturduğu esnada kullanıcının standart metot kullanılamaz olmaktadır. Bu metot yapısını daha iyi anlayabilmek için örnek uygulama yapmak uygun olacaktır.

Yapıcı metot, diğer metotlardan farklı olarak metot isminin sınıf ismi aynı olup dönüş değeri ve dönüş ifadelerini içermemiş olmasıdır.

Örnek

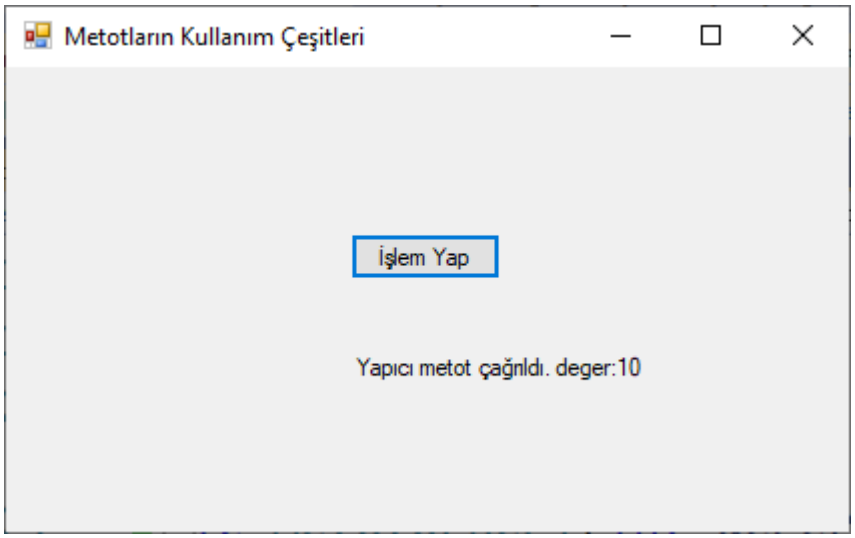
- Yapıcı metotların temel kullanımına ve çağrılmasına ait örnek kodlar aşağıda belirtilmiştir.

```
public partial class MetotOrnegi : Form
{
  public int deger;
  public MetotOrnegi()
  {
    InitializeComponent();
  }
  public MetotOrnegi(int deger)
  {
    this.deger = deger;
  }
  private void IslemYap_Click(object sender, EventArgs e)
  {
    MetotOrnegi metotOrnegi = new MetotOrnegi(10);
    IslemSonucu.Text = "Yapıcı metot çağrıldı";
  }
}
```

Yapıcı metotlar genel olarak bir parça karışık gelebilir. Ancak kullanımı zamanla anlaşılan özel bir yapı olduğundan bu örneğin anlaşılması çok önemlidir.

Yukarıdaki kodlarda görüldüğü gibi “MetotOrnegi” sınıfına ait bir nesne “IslemYap” butonuna tıklanıldığında tanımlanmıştır. Bu kısım, standart nesne çağırma ile aynı yapıdadır. Nesne çağrıldığı esnada yapıcı metot çalışmıştır. Daha sonra “MetotOrnegi” sınıfına ait “deger” değişkenin almış olduğu değer “IslemSonucu” alanına Resim 9.1.’de gösterildiği yazdırılmıştır. Ancak burada

dikkat edilmesi gereken kısım “deger” değişkeninin sahip olduğu “10” değerinin yapıcı metot içerisinde verilmiş olmasıdır. Yani örnekte tanımlanmış olan yapıcı metot nesne oluştuğu esnada içerisinde bulunan işlemler gerçekleşmiştir. *Normal şartlarda metodun tanımını yaptığımız zaman bir dönüş ifadesinin veya onun olmadığı durumlarda “void” kelimesinin olması gerekmektedir.* Ancak yapıcı metotlar özel bir metot olduğundan yapısının biraz farklı olduğunu görebiliyoruz. Bu özellikleriyle bu metodun diğer metotlardan farklı olarak metot isminin sınıf ismi aynı olup dönüş değeri ve dönüş ifadelerini içermemiş olmasıdır. Bu örnekte de görüldüğü gibi yapıcı metotlar en temel kullanım olarak kendi sınıfı içerisindeki değişkenler ile ilgili olarak işlemlerde kullanılabilir.



Resim 9.1. Yapıcı Metotların Temel Kullanımı

Örnek

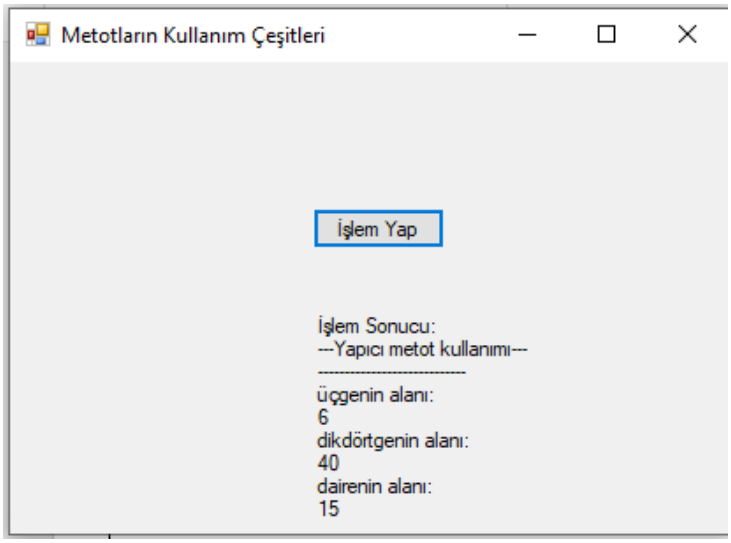
•Yapıcı metotların daha iyi anlaşılabilmesi için aşırı yüklemeler(overloading) ile bir uygulama yapmak uygun olacaktır. İlgili kodlar aşağıda belirtilmiştir.

```
    public MetotOrnegi ()
    {
        InitializeComponent();
      deger = 10;
    }
    public MetotOrnegi (int kenar1, int kenar2, double aci)
    {
      ucgeninAlani=AlanHesapla(kenar1, kenar2, aci);
    }
    public MetotOrnegi (int yatayKenar, int dikeyKenar)
    {
      dikdortgeninAlani=AlanHesapla(yatayKenar, dikeyKenar);
    }
    public MetotOrnegi (int yariCap)
    {
      daireninAlani=AlanHesapla(yariCap);
    }

    public double AlanHesapla(int kenar1, int kenar2, double aci) //üçgenin alanı
    {       double b = (aci * (Math.PI)) / 180;
      return ((double)kenar1 * (double)kenar2 * Math.Sin(b)) / 2;   }
    public int AlanHesapla(int yatayKenar, int dikeyKenar) //dikdörtgenin alanı
    {       return yatayKenar * dikeyKenar;      }
    public int AlanHesapla(int yariCap)//dairenin alanı
    {       return (int)(Math.PI * yariCap);      }

private void IslemYap_Click(object sender, EventArgs e)
    {
      MetotOrnegi metotOrnegi = new MetotOrnegi(10);
      IslemSonucu.Text = "İşlem Sonucu:" + "\n" +
      "---Yapıcı metot kullanımı---" + "\n" +
      "---------------------------" + "\n";
      MetotOrnegi MetotOrnegiUcgen = new MetotOrnegi(4, 6, 30.0);
      IslemSonucu.Text = IslemSonucu.Text + "üçgenin alanı:" + "\n" +
      MetotOrnegiUcgen.ucgeninAlani.ToString() + "\n" ;
      MetotOrnegi MetotOrnegiDikdortgen = new MetotOrnegi(5, 8);
      IslemSonucu.Text = IslemSonucu.Text + "dikdörtgenin alanı:" + "\n" +
      MetotOrnegiDikdortgen.dikdortgeninAlani.ToString() + "\n" ;
      MetotOrnegi MetotOrnegiDaire = new MetotOrnegi(5);
      IslemSonucu.Text = IslemSonucu.Text + "dairenin alanı:" + "\n" +
      MetotOrnegiDaire.daireninAlani.ToString();

  }
```



Resim 9.2. Yapıcı Metotların Aşırı Yüklemelerinin Çağrılması ve Çıktıları

Resim 9.2.'de gösterilen programda yapıcı metotların çeşitli kullanımlarına ait sonuç çıktılar gösterilmiştir. İlk olarak önceki ünitede örnek üzerinde anlatılan aşırı yükleme(overloading) olayının normal metotlar gibi yapıcı metotlar üzerinde de yapılabildiği bilgisini vermek gerekmektedir. Örnekte gösterildiği gibi farklı değişkenler ile oluşturulmuş 4 adet yapıcı metot bulunmaktadır. *Bu metotların hepsinin ismi sınıf ismi ile aynıdır. Ancak her binin görevi farklı görevleri yapan metotlar çağırarak işlemlerin sonuçlarının "MetotOrnegi" sınıfına ait "deger, ucgeninAlani, dikdortgeninAlani, daireninAlani" değişkenlerine atanmasını sağlamaktır.* Örnekte her bir yapıcı metot giriş parametreleri olarak verilen değerler ile birlikte kullanılarak nesne tanımlandığı anda yapıcı metot içerisindeki işlemler gerçekleşerek sınıfa ait değişkenlere ilgili değerlerin atanması sağlanmıştır. Daha sonra yapıcı sınıfındaki "IslemYap" butonunun tıklanmasıyla örnekte gösterildiği gibi nesne oluşturulmuş ve Resim 9.2.'de gösterildiği gibi sonuçlara ulaşılmıştır.

Kopyalayıcı Metotlar

Yapıcı metotların aşırı yükelemeler ile giriş parametresi olarak kendi sınıfından oluşturulmuş bir nesne alması durumuna kopyalayıcı metotlar denir. Bu metotların anlaşılması biraz zor olduğu için yapıcı metotlardan farklı bir başlık olarak anlatılması uygun görülmüştür. Konunun daha iyi anlaşılabilmesi için bir örnek üzerinde göstermek daha uygun olacaktır.

Kopyalayıcı metotların kullanım alanları giderek arttığı ve anlaşılması biraz zor olduğu için ayrı başlıkta belirtilmiştir.

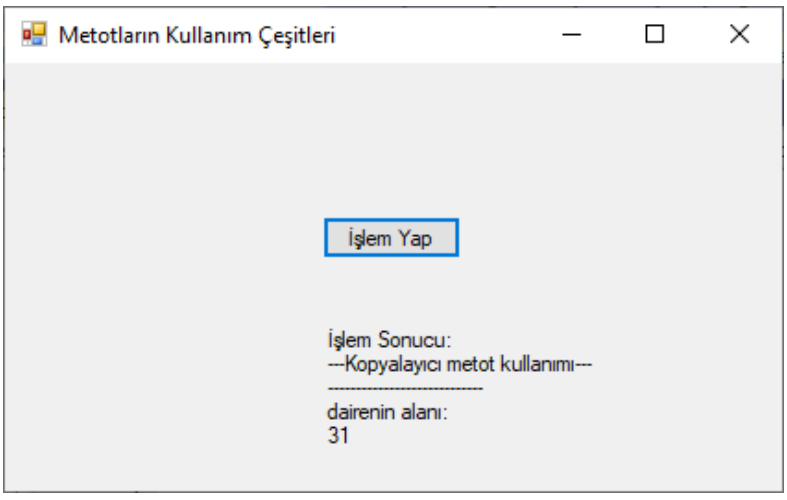
Örnek

- Kopyalayıcı metotların daha iyi anlaşılabilmesi için bir uygulama yapmak uygun olacaktır. İlgili kodlar aşağıda belirtilmiştir.

```
public partial class MetotOrnegi : Form
{
  public int deger;
  public double ucgeninAlani;
  public int dikdortgeninAlani;
  public int daireninAlani;

  public MetotOrnegi()
  {
    InitializeComponent();
    deger=10;
  }
  private void IslemYap_Click(object sender, EventArgs e)
  {
    MetotOrnegi metotOrnegi = new MetotOrnegi();
    IslemSonucu.Text = "İşlem Sonucu:" + "\n" +
    "---Kopyalayıcı metot kullanımı---" + "\n" +
    "--------------------------" + "\n";
    MetotOrnegi MetotOrnegiDaire = new MetotOrnegi(metotOrnegi);
    IslemSonucu.Text = IslemSonucu.Text + "dairenin alanı:" + "\n" +
    MetotOrnegiDaire.daireninAlani.ToString();

  }
  public MetotOrnegi(MetotOrnegi yariCap)
  {
    daireninAlani = AlanHesapla(yariCap.deger);
  }
  public int AlanHesapla(int yariCap)//dairenin alanı
  {
    return (int)(Math.PI * yariCap);
  }
}
```



Resim 9.3. Kopyalayıcı Metotların Çağrılması ve Çıktıları

Örnekte gösterildiği gibi "MetotOrnegi" sınıfına ait aşırı yükleme ile oluşturulmuş "MetotOrnegi" metodu giriş parametresi olarak yapıcı metot türünden başka bir yapıcı metot kullanmaktadır. Bu durumda ilk olarak "deger" değişkenine 10 değerinin atanacağı yapıcıyı ifade eden metottan bir nesne oluşturulmuştur. Daha sonra bu metot diğer bir yapıcı metot için giriş parametresi olarak kullanılmıştır. Bu yapı bir parça karışık gelebilir ancak kullanıldıkça çok işe yarayan bir yapı olduğu anlaşılacaktır. Resim 9.3.'de sonuç çıktıları belirtilmitir.

Hazır Metotlar

Hazır metotlar, programlamayı ve hatırlamayı kolaylaştırmaktadır.

C# programlama diline ait önceden oluşturulmuş "System" kütüphanesi içinde bazı hazır metotlar bulunmaktadır. Bu metotlar sayesinde hızlı, etkili ve sonuç açısından tutarlı kodlama yapmak mümkündür. *Bu metotlardan en önemli ve en çok kullanılanları metinsel, matematiksel ve tarihsel metotlardır.* Bu ünite kapsamında bu metotlar anlatılacaktır.

Metinsel (String) Metotlar

C# Programlama dilinde metinsel ifadelerin gerçekleşmesi için "string" sınıfı oluşturulmuştur. Bu sınıf altında oldukça kullanışlı ve hemen her yazılım için olmazsa olmaz metotlar bulunmakta ve çok fazla kolaylık sağlamaktadır. Bu metotlardan en sık kullanılanları Compare, Concat, Copy, Format, IsNullOrEmpty, CompareTo, Contains, EndsWith, IndexOf, PadLeft, Remove, Trim, Replace, Split, Substring, ToLower, ToUpper metotları olup bunların bir kısmı "String" sınıfı altında çağrılırken diğer bir kısmı ise string değere sahip değişkenler ile birlikte kullanılır. Bu farkı her bir metot için örnekler üzerinden öğrenmek uygun olacaktır. Yapacağımız örnekte belirttiğimiz metinsel metotların hepsinin kullanımını bir örnek uygulama üzerinde göstereceğiz. *Ancak bilinmesi gereken çok önemli bir durum ise bu metotların hemen hemen hepsinin aşırı yükleme ile oluşturulmuş çok fazla sayıda metotları vardır.* Biz sadece örnek olması için birer tanesini kullanacağız.

Örnek

- Aşağıda verilen kodlarda Compare, Concat, Copy, Format, IsNullOrEmpty, CompareTo, Contains, EndsWith, IndexOf, PadLeft, Remove, Trim, Replace, Split, Substring, ToLower, ToUpper metotlarının kullanımı aşağıda görülmektedir.

```
private void IslemYap_Click(object sender, EventArgs e)
  {
    IslemSonucu.Text = "İşlem Sonucu:" + "\n" +
     "---Hazır metinsel metot kullanımı---" + "\n" +
     "---------------------------" + "\n";
        string MetinselMetotlarinKullanimi = "Atatürk Üniversitesi Açıköğretim Fakültesi";
   int compare = String.Compare(MetinselMetotlarinKullanimi, "Atatürk Üniversitesi Açıköğretim");
        IslemSonucu.Text = IslemSonucu.Text + "Compare metot kullanımı : " + compare + "\n";
   string concat = String.Concat(MetinselMetotlarinKullanimi, " çok başarılıdır.");
        IslemSonucu.Text = IslemSonucu.Text + "Concat metot kullanımı : " + concat + "\n";
   string copy = String.Copy(MetinselMetotlarinKullanimi);
        IslemSonucu.Text = IslemSonucu.Text + "Copy metot kullanımı : " + copy + "\n";
   string format = String.Format("{0,50}", MetinselMetotlarinKullanimi);
        IslemSonucu.Text = IslemSonucu.Text + "Format metot kullanımı : " + format + "\n";
   bool isNullOrEmpty = String.IsNullOrEmpty(MetinselMetotlarinKullanimi);
        IslemSonucu.Text =IslemSonucu.Text + "IsNullOrEmpty metot kullanımı : " + isNullOrEmpty +
"\n";
   int compareTo = MetinselMetotlarinKullanimi.CompareTo("Atatürk Üniversitesi Açıköğretim");
        IslemSonucu.Text =IslemSonucu.Text + "CompareTo metot kullanımı : " + compareTo + "\n";
   bool contains = MetinselMetotlarinKullanimi.Contains("Atatürk Üniversitesi Açıköğretim");
        IslemSonucu.Text =IslemSonucu.Text + "Contains metot kullanımı : " + contains + "\n";
   bool endsWith = MetinselMetotlarinKullanimi.EndsWith("Atatürk Üniversitesi Açıköğretim");
        IslemSonucu.Text =IslemSonucu.Text + "EndsWith metot kullanımı : " + endsWith + "\n";
   int indexOf = MetinselMetotlarinKullanimi.IndexOf("Atatürk Üniversitesi Açıköğretim");
        IslemSonucu.Text =IslemSonucu.Text + "IndexOf metot kullanımı : " + indexOf + "\n";
        string padLeft = MetinselMetotlarinKullanimi.PadLeft(60);
        IslemSonucu.Text =IslemSonucu.Text + "PadLeft metot kullanımı : " + padLeft + "\n";
        string remove = MetinselMetotlarinKullanimi.Remove(32);
        IslemSonucu.Text =IslemSonucu.Text + "Remove metot kullanımı : " + remove + "\n";
        string trim = MetinselMetotlarinKullanimi.Trim('i');
        IslemSonucu.Text =IslemSonucu.Text + "Trim metot kullanımı : " + trim + "\n";
        string replace = MetinselMetotlarinKullanimi.Replace("Fakültesi", "Fakültesi çok büyük bir
ailedir.");
        IslemSonucu.Text =IslemSonucu.Text + "Replace metot kullanımı : " + replace + "\n";
        string[] split = MetinselMetotlarinKullanimi.Split(' ');
        IslemSonucu.Text =IslemSonucu.Text + "Split metot kullanımı : " + split[0] + "-" + split[1] + "-" +
split[2] + "-" + split[3] + "\n";
        string substring = MetinselMetotlarinKullanimi.Substring(20);
        IslemSonucu.Text = IslemSonucu.Text + "Substring metot kullanımı : " + substring + "\n";
        string toLower = MetinselMetotlarinKullanimi.ToLower();
        IslemSonucu.Text = IslemSonucu.Text + "ToLower metot kullanımı : " + toLower + "\n";
        string toUpper = MetinselMetotlarinKullanimi.ToUpper();
        IslemSonucu.Text = IslemSonucu.Text + "ToUpper metot kullanımı : " + toUpper + "\n";
      }
```

Yukarıda verilen örnekte metinsel metotların kullanımları görülmektedir. Örnekte kullanılan metinsel metotların kullanımları ve işlevleri şu şekildedir:

Compare metodu: "String" sınıfına ait giriş parametresi olarak almış olduğu iki değeri birebir karşılaştırır ve eşit olması durumunda sonuç 0(sıfır) döner aksi

halde 1(bir) veya -1(eksi bir) sonucu döner. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Concat metodu: "String" sınıfına ait giriş parametresi olarak almış olduğu iki değeri art arda olacak şekilde birleştirilir Sonuç olarak iki değişken metinsel olarak birleşmiştir. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Copy metodu: "String" sınıfına ait giriş parametresi olarak almış olduğu değeri kopyalayarak yeni bir değere atamaktadır. Sonuç olarak değişkendeki değerin aynısı yeni değişkene aktarılmış olacaktır. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Format metodu: "String" *sınıfına ait giriş parametreleri olarak iki değer almaktadır. Bu parametrelerden ilki formatı belirleyen ifadeyi ikicisi ise string ifadeyi yani format verilecek değişkeni temsil eder. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.* Sonuç olarak ise string değişken "{0,50}" formatına yani 51 karakter uzunluğunda bir alan olup ekranın sağ tarafına yaslı şekilde duracağını ifade eder. Bu metodun çok fazla değişik yapılarda kullanımı mevcuttur.

IsNullOrEmpty metodu: "String" sınıfına ait giriş parametresi olarak almış olduğu değerin boş veya değer atanmamış olması durumunda "false" aksi halde "true" değeri döndüren metottur. Sonuç olarak değişkendeki değerin varlığını kontrol eder. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

CompareTo metodu: Metinsel (string) bir ifadenin başka bir metinsel ifade ile karşılaştırılması için kullanılan bir metottur. Bu metot metinsel ifadeyi compareTo metodunun giriş parametresi ile birebir karşılaştırır ve eşit olması durumunda sonuç 0(sıfır) döner aksi halde 1(bir) veya -1(eksi bir) sonucu döner. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Contains metodu: Metinsel (string) bir ifadenin içinde Contains metoduna ait giriş parametresi ile karşılaştırır. Bu metodun giriş parametresi metinsel ifadenin içinde geçen bir kısım olması durumunda sonuç "true" döner aksi halde "false" sonucu döner. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

EndsWith metodu: Metinsel (string) bir ifadenin son kısmında EndsWith metodunun giriş parametresinin olup olmadığını kontrol eden metottur. Bu metodun giriş parametresinin tamamı metinsel ifadenin son kısmı ile eşleşiyor olması durumunda sonuç "true" döner aksi halde "false" sonucu döner. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Bir metin ile ilgili işlem yapmak için metinsel metotlar bir arada kullanılabilir.

IndexOf metodu: Metinsel (string) bir ifadenin içinde IndexOf metodunun giriş parametresi olup olmadığını kontrol eden metottur. Bu metodun giriş parametresindeki ifadenin metinsel ifadenin içinde geçmesi durumunda sonuç giriş parametresinin bulunduğu index değeri döner aksi halde -1(eksi bir) sonucu döner. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

PadLeft metodu: Metinsel (string) bir ifadenin PadLeft metodunun giriş parametresinin değeri kadar ayrılmış bir alanın sol tarafına yaslı şekilde

konumlandırılmasını sağlar. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Remove metodu: Metinsel (string) bir ifade için kullanılan Remove metodu, bu metot giriş parametresinin değeri, metinsel ifadenin bu değere sahip index değerinden başlayarak metinsel ifadenin sonuna kadar silinmesini sağlar. Bu metoda ait çeşitli kullanımlar yani aşırı yükleme ile oluşturulmuş metotlar vardır. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Trim metodu: Metinsel (string) bir ifadede Trim metodunun kullanılması, *bu metodun giriş parametresinin değerin eğer metinsel ifadenin son kısmında bulunuyorsa bu giriş parametrenin silinmesini sağlar.* Bu metoda ait çeşitli kullanımlar yani aşırı yükleme ile oluşturulmuş metotlar vardır. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Replace metodu: Metinsel (string) bir ifadede, Replace metodunun kullanılması, bu metodun ilk giriş parametresinin değeri eğer metinsel ifadenin içinde bulunuyorsa bu ifade girilen ikinci değer ile değiştirilir. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Split metodu: Metinsel (string) bir ifadede, Split metodunun kullanılması durumunda bu metodun giriş parametresinin değeri eğer metinsel ifadenin içinde bulunuyorsa bu ifadenin bulunduğu yerlerinden itibaren metinsel ifade bölünür. Oluşan yeni öğe ise "string[]"(string dizi) şekline dönüşür. Bu metot çok değişik ve çeşitli işlemler için kullanışlı bir yapıdır. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Substring metodu metin içerisinden istenilen metin parçasını almak için sıkça kullanılan bir yapıdır.

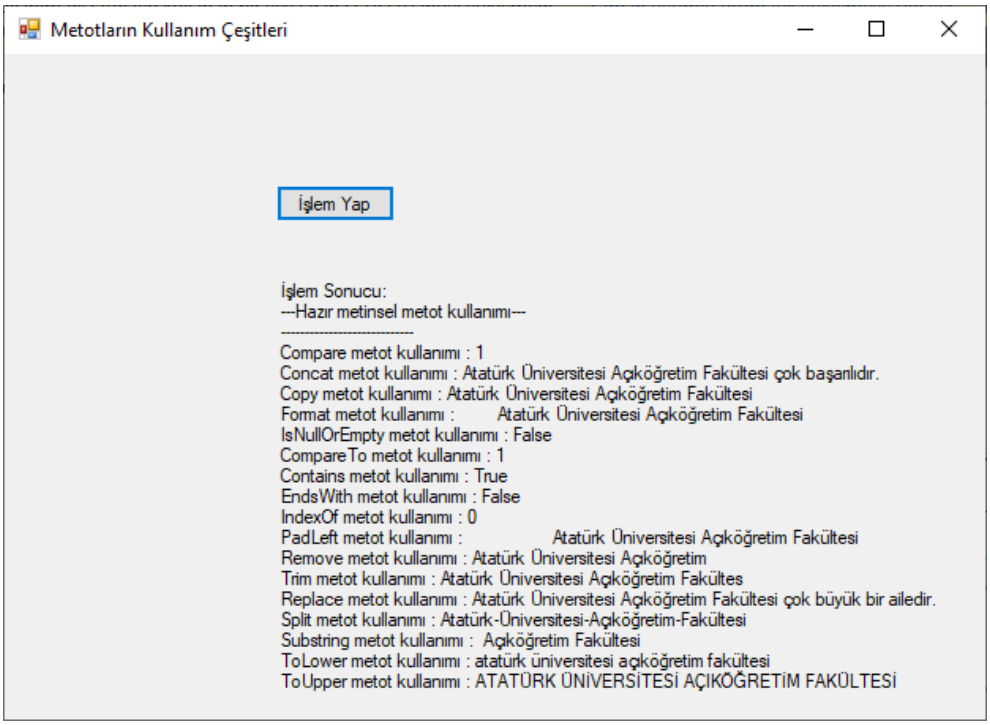
Substring metodu: Metinsel (string) bir ifadede, Substring metodunun kullanılması durumunda bu metoda ait iki giriş parametresi vardır. Bunların ilki metinsel ifadeden alınacak alt metin için başlangıç indexi olup diğeri ise index değerinden başlayarak kaç karakter olacağını ifade eder. *Yani metinsel ifadenin içerisinden belirli bir kısmın alınarak yeni bir değere atanmasını sağlar. Oluşan yeni öğe ise yeni bir metinsel ifadeye dönüşür.* Bu metoda ait çeşitli kullanımlar yani aşırı yükleme ile oluşturulmuş metotlar vardır. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

ToLower metodu: Metinsel (string) bir ifadede, ToLower metodunun kullanılması durumunda metinsel ifadede bulunan bütün karakterlerin küçük harfe dönüştürülmesi sağlanır. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

ToUpper metodu: Metinsel (string) bir ifadede, ToUpper metodunun kullanılması durumunda metinsel ifadede bulunan bütün karakterlerin büyük harfe dönüştürülmesi sağlanır. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Metinsel(string) metotların kullanımlarını yukarıdaki örnekte gösterdik. Bu kullanımların sonuçlarını ise Resim 9.4.'de görebilirsiniz. *Bu metotlar bir yazılım açısından oldukça fazla önemli olmaktadır. Hemen hemen her yazılımcı metinsel metotları çok değişik amaçlar için değişik şekillerde aşırı yüklemeleriyle birlikte kullanmaktadır.* Bu ünitede gösterilmeyen çok daha fazla metinsel metot

bulunmaktadır. *Bir yazılım geliştirilirken metinsel bir işlem yapacaksanız bu işleme ait hazır bir metot olup olmadığı kontrol etmenizi öneririz.*



Resim 9.4. Metinsel Metotların Kullanımlarına Ait Sonuçlar

Matematiksel Metotlar

Matematiksel metotlar sayesinde Cos, Sin gibi özel hesaplama isteyen işlemler kolaylıkla tanımlanır.

C# Programlama dilinde matematiksel işlemlerin gerçekleşmesi için "Math" sınıfı oluşturulmuştur. Bu sınıf altında oldukça kullanışlı ve hemen her yazılım için olmazsa olmaz metotlar bulunmakta ve çok fazla kolaylık sağlamaktadır. *Ayrıca matematiksel işlemlerde kullanıcı tarafından yazılan metotlarda yuvarlama ve değişken türünü çevirme işlemleri sonucunda bazı değerlerde değişiklikler olmakta ve değişiklikler ise gerçekleşen yazılımların sonuçlarında uyuşmazlıklara yol açmaktadır. Bu durumların kısmen de olsa ortadan kalkması için matematiksel işlemlerde hazır metotların kullanılması daha uygun olacaktır.* Bu metotlardan en sık kullanılanları Abs, Ceiling, Max, Min, Pow, Round, Sign, Sqrt, Cos, Sin metotlarıdır. Bu metotların kullanımını örnekler üzerinde görmek uygun olacaktır.

Aşağıda verilen örnekte bazı matematiksel metotların kullanımı görülmektedir.

Örnek

- Aşağıda verilen kodlarda Abs, Ceiling, Max, Min, Pow, Round, Sign, Sqrt, Cos, Sin metotlarının kullanımı aşağıda görülmektedir.

```
        double MatematikselMetotlarinKullanimi1 = 520.06302;
        double MatematikselMetotlarinKullanimi2 = -220.2133;

        double abs = Math.Abs(MatematikselMetotlarinKullanimi2);
        IslemSonucu.Text = IslemSonucu.Text + "Abs metot kullanımı:"+ abs + "\n";
        double ceiling = Math.Ceiling(MatematikselMetotlarinKullanimi1);
        IslemSonucu.Text = IslemSonucu.Text + "Ceiling metot kullanımı:"+ ceiling + "\n";
        double max = Math.Max(MatematikselMetotlarinKullanimi1,
MatematikselMetotlarinKullanimi2);
        IslemSonucu.Text = IslemSonucu.Text + "Max metot kullanımı:"+ max + "\n";
        double min = Math.Min(MatematikselMetotlarinKullanimi1,
MatematikselMetotlarinKullanimi2);
        IslemSonucu.Text = IslemSonucu.Text + "Min metot kullanımı:"+ min + "\n";
        double pow = Math.Pow(MatematikselMetotlarinKullanimi1, 2);
        IslemSonucu.Text = IslemSonucu.Text + "Pow metot kullanımı:"+ pow + "\n";
        double round = Math.Round(MatematikselMetotlarinKullanimi1);
        IslemSonucu.Text = IslemSonucu.Text + "Round metot kullanımı:"+ round + "\n";
        int sign = Math.Sign(MatematikselMetotlarinKullanimi1);
        IslemSonucu.Text = IslemSonucu.Text + "Sign metot kullanımı:"+ sign + "\n";
        double sqrt = Math.Sqrt(MatematikselMetotlarinKullanimi1);
        IslemSonucu.Text = IslemSonucu.Text + "Sqrt metot kullanımı:"+ sqrt + "\n";
        double cos = Math.Cos(MatematikselMetotlarinKullanimi1);
        IslemSonucu.Text = IslemSonucu.Text + "Cos metot kullanımı:"+ cos + "\n";
        double sin = Math.Sin(MatematikselMetotlarinKullanimi1);
        IslemSonucu.Text = IslemSonucu.Text + "Sin metot kullanımı:"+ sin + "\n";
```

Abs Metodu: Bu metot giriş parametresi olarak aldığı değerin mutlak değerini yani sıfıra olan yakınlığını ifade eden değerdir. Çok farklı türde giriş parametrelerini kullanarak yine giriş parametresi türünde değer döndürür. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Ceiling Metodu: Bu metot giriş parametresi olarak aldığı değerin tam sayı değerini bir artırarak yeni bir değer oluşturan metottur. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Max Metodu: Bu metot giriş parametresi olarak aldığı iki değerden büyüğünü geri döndüren metottur. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Min Metodu: Bu metot giriş parametresi olarak aldığı iki değerden küçüğünü geri döndüren metottur. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Pow Metodu: Bu metot giriş parametresi olarak aldığı iki değerden ilkinin ikincisi kadar üssünü geri döndüren metottur. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

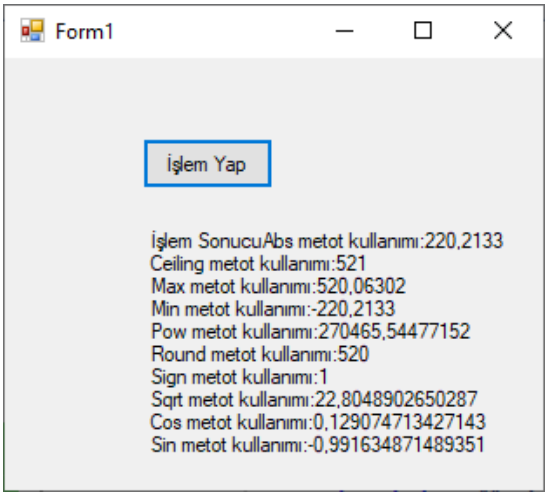
Round Metodu: Bu metot giriş parametresi olarak aldığı değeri en yakın tam sayıya dönüştüren metottur. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Sign Metodu: Bu metot giriş parametresi olarak aldığı değerin işaretini döndüren metottur. Sayı pozitif ise 1(bir), negatif ise -1(eksi bir) ve 0'a eşit ise 0(sıfır) değerini alır. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Sqrt Metodu: Bu metot giriş parametresi olarak aldığı değerin karakökünü döndüren metottur. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Cos Metodu: Bu metot giriş parametresi olarak aldığı değerin radyan açı değerinin kosinüs değerini veren metottur. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.

Sin Metodu: Bu metot giriş parametresi olarak aldığı değerin radyan açı değerinin sinüs değerini veren metottur. Yukarıdaki örnek uygulamada kullanımı bulunmaktadır.



Resim 9.5. Matematiksel Metotların Sonuçları

Tarihsel Metotlar

C# Programlama dilinde tarihsel işlemlerin gerçekleşmesi için "DateTime" sınıfı oluşturulmuştur. *Bu sınıf altında oldukça kullanışlı ve hemen hemen her yazılım için olmazsa olmaz metotlar bulunmakta ve çok fazla kolaylık sağlamaktadır.* Ayrıca tarihsel işlemlerde kullanıcı tarafından yazılan metotlarda formatlar ile ilgili bazı değerlerde değişiklikler olmakta ve değişiklikler ise gerçekleşen yazılımların sonuçlarında uyuşmazlıklara yol açmaktadır. Bu durumların kısmen de olsa ortadan kalkması için tarihsel işlemlerde hazır metotların kullanılması daha uygun olacaktır. Bu metotlardan en sık kullanılanları Compare, DaysInMonth, IsLeapYear, Parse, Subtract, AddDays, AddMonths, AddYears, AddHours, AddMinutes, AddSeconds, AddMilliseconds metotlarıdır. Ayrıca "Datetime" sınıfına ait Today, Now, Month, Day, Year, DayOfWeek, DayOfYear, TimeOfDay, Hour, Minute, Second, Millisecond değişkenleri hakkında da genel bilgiler verilecektir. Bu metotları ve değişkenlerin kullanımını örnekler üzerinde görmek uygun olacaktır. Yapacağımız örnekte belirttiğimiz tarihsel metotların bir kısmının kullanımını bir örnek uygulama üzerinde göstereceğiz.

Tarihsel metotların hazır olarak kullanılması uluslararası tarih ve saat bilgilerinin tutarlılığı açısından çok önemlidir.

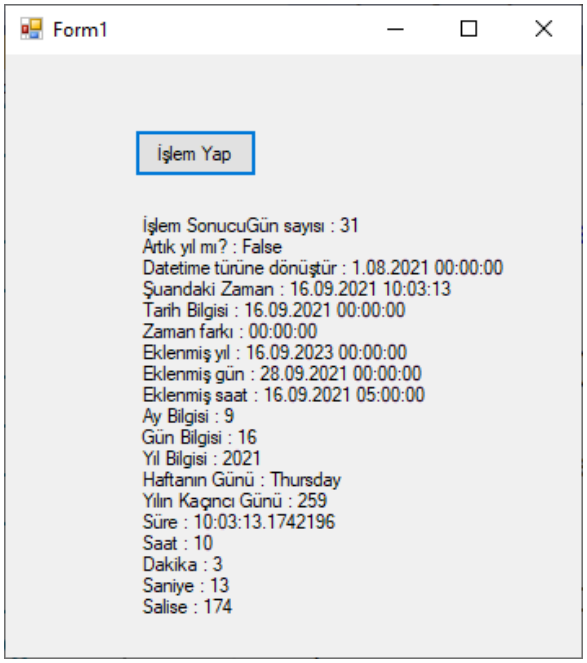
Örnek

•Aşağıda verilen kodlarda Compare, DaysInMonth, IsLeapYear, Parse, Subtract, AddDays, AddMonths, AddYears, AddHours, AddMinutes, AddSeconds, AddMilliseconds metotları ve Ayrıca "Datetime" sınıfına ait Today, Now, Month, Day, Year, DayOfWeek, DayOfYear, TimeOfDay, Hour, Minute, Second, Millisecond kullanımları aşağıda görülmektedir.

```
int gunSayisi = DateTime.DaysInMonth(2021, 3);
IslemSonucu.Text = IslemSonucu.Text + "Gün sayısı : "+ gunSayisi + "\n";
bool artikYilMi = DateTime.IsLeapYear(2021);
IslemSonucu.Text = IslemSonucu.Text + "Artık yıl mı? : "+ artikYilMi + "\n";
DateTime tarih1 = DateTime.Parse("Ağustos/2021");
IslemSonucu.Text = IslemSonucu.Text + "Datetime türüne dönüştür : "+ tarih1 + "\n";
DateTime zaman = DateTime.Now;
IslemSonucu.Text = IslemSonucu.Text + "Şuandaki Zaman : "+ zaman + "\n";
DateTime tarih = zaman.Date;
IslemSonucu.Text = IslemSonucu.Text + "Tarih Bilgisi : "+ tarih + "\n";
TimeSpan yeniTarih = tarih.Subtract(tarih);
IslemSonucu.Text = IslemSonucu.Text + "Zaman farkı : "+ yeniTarih + "\n";
DateTime eklenmisYil = tarih.AddYears(2);
IslemSonucu.Text = IslemSonucu.Text + "Eklenmiş yıl : "+ eklenmisYil + "\n";
DateTime eklenmisGun = tarih.AddDays(12);
IslemSonucu.Text = IslemSonucu.Text + "Eklenmiş gün : "+ eklenmisGun + "\n";
DateTime eklenmisSaat = tarih.AddHours(5);
IslemSonucu.Text = IslemSonucu.Text + "Eklenmiş saat : "+ eklenmisSaat + "\n";
int ay = zaman.Month;
IslemSonucu.Text = IslemSonucu.Text + "Ay Bilgisi : "+ ay + "\n";
int gun = zaman.Day;
IslemSonucu.Text = IslemSonucu.Text + "Gün Bilgisi : "+ gun + "\n";
int yil = zaman.Year;
IslemSonucu.Text = IslemSonucu.Text + "Yıl Bilgisi : "+ yil + "\n";
DayOfWeek haftaninGunu = zaman.DayOfWeek;
IslemSonucu.Text = IslemSonucu.Text + "Haftanın Günü : "+ haftaninGunu + "\n";
int yilinKacinciGunu = zaman.DayOfYear;
IslemSonucu.Text = IslemSonucu.Text + "Yılın Kaçıncı Günü : "+ yilinKacinciGunu + "\n";
TimeSpan sure = zaman.TimeOfDay;
IslemSonucu.Text = IslemSonucu.Text + "Süre : "+ sure + "\n";
int saat = zaman.Hour;
IslemSonucu.Text = IslemSonucu.Text + "Saat : "+ saat + "\n";
int dakika = zaman.Minute;
IslemSonucu.Text = IslemSonucu.Text + "Dakika : "+ dakika + "\n";
int saniye = zaman.Second;
IslemSonucu.Text = IslemSonucu.Text + "Saniye : "+ saniye + "\n";
int salise = zaman.Millisecond;
IslemSonucu.Text = IslemSonucu.Text + "Salise : "+ salise + "\n";
```

Yukarıdaki örnekte gösterildiği gibi tarihsel metot ve değerleri hazır yapılar sayesinde kullanmak mümkündür. Bu bölümde her bir metot ve değişken ayrı ayrı anlatılmayacaktır. *Çünkü genellikle her biri tek amaç ve değer için oluşturulmuştur. Bu sebeple çok geniş kullanıma sahip tarihsel metot ve değerleri bir arada anlatmak uygun görülmüştür.*

Örnekte gösterildiği gibi bir tarih değerinin gün sayısını, artık yıl olup olmadığını, şu anki zaman bilgisini, zamanlar arasındaki farkı, yıl eklenmiş halini, gün eklenmiş halini, saat eklenmiş halini, ay bilgisini, gün bilgisini, yıl bilgisini, haftanın hangi gününde olduğu bilgisi, yılın kaçıncı gününde olduğu bilgisi, saat bilgisini, dakika bilgisini, saniye bilgisini ve milisaniye bilgisini çok kolay bir şekilde alabiliriz. Bu bilgilerin sonuçlarını Resim 9.6.'da görebiliriz. *Tarihsel metot ve değerlerin kullanımları sayesinde yazılım esnasında tarihsel bilgi olarak kullanılması mümkün olan hemen hemen bütün bilgilere ulaşılmış olunacaktır.*



Resim 9.6. Tarihsel Metot ve Değişkenlerin Sonuçları

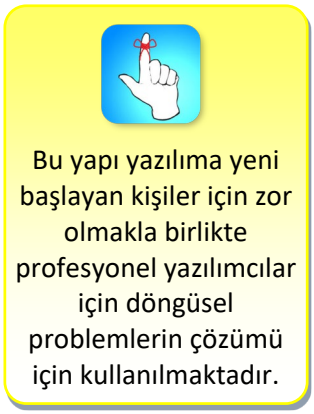
Özyinelemeli(Recursive) Metotlar

Özyinelemeli metotlar, çoğu yazılımcının kurtarıcı metotlarından biri olmaktadır.

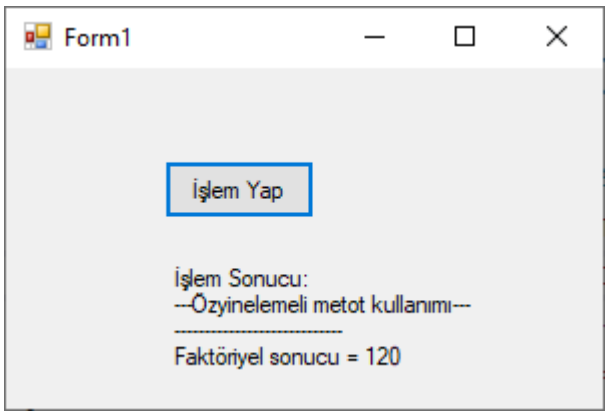
Bu metot kendini çağıran metot olarak isimlendirilmekte ve ileri yazılım geliştirme tekniklerinde çoğu problemin çözümünde kullanılmaktadır. Bu metodun iyi anlaşılması çok önemlidir. Bu sebeple en sık kullanılan ve anlaşılması en kolay faktöriyel örneği üzerinden işlemi anlatmak gerekmektedir. İlk olarak faktöriyel hesabı; verilen sayıdan başlayarak 1 e kadar giderek küçülen sayıların çarpımı olarak ifade edilebilir. Yani 5!(Faktöriyel 5)'in işlemi 5*4*3*2*1=120 olarak sonuçlanmıştır. Bu örnekte görüldüğü gibi işlem sürekli olarak bir alttaki sayı 1 oluncaya kadar çarpılarak sonuca kavuşulmaktadır. Bu işlemi örnek bir uygulama olarak görmek uygun olacaktır.

```
private void IslemYap_Click(object sender, EventArgs e)
{
   IslemSonucu.Text ="İşlem Sonucu:" + "\n" +
    "---Özyinelemeli metot kullanımı---" + "\n" +
    "---------------------------" + "\n"
    + "Faktöriyel sonucu = "+ Faktoriyel(5);
}
public static long Faktoriyel(long sayi)
{
   if (sayi <= 1)
   {
      return 1;
   }
   else
   {
      long deger = sayi * Faktoriyel(sayi - 1);
      return deger;
   }
}
```

Yukarıdaki örnekte gösterildiği gibi "Faktöriyel" isimli bir metot oluşturulmuştur. *Ancak bu metot içinde dikkat edilmesi gereken çok önemli bir nokta bulunmaktadır. "IslemYap" butonu tıklandığında "Faktöriyel" isimli metodun çağrıldığı görülmektedir.* Bu yapı yazılıma yeni başlayan kişiler için anlaşılması zor bir özellik olmakla birlikte yazılımda profesyonel çözüm arayan kişiler için çoğu döngüsel problemlerin çözümü için kullanımı uygun olacaktır. Örnekte "Faktöriyel" metodu çağrıldığı zaman metot içerisinde bulunan kodlar çalışacaktır. Bu aşamada metot içindeki "else" şartı sağlandığı sürece kendini çağırmaya devam edecektir. *İlk olarak giriş parametresi 5 olarak verilen değer 1 oluncaya kadar eksilerek herhangi bir işlem yapmaksızın döngüye girecektir. Daha sonra en alttan başlayarak değerler çarpılarak geri dönüş değerine eklenecektir.* Böylece Resim 9.7.'de gösterildiği gibi faktöriyel sonucu ortaya çıkacaktır.



Bu yapı yazılıma yeni başlayan kişiler için zor olmakla birlikte profesyonel yazılımcılar için döngüsel problemlerin çözümü için kullanılmaktadır.



Resim 9.7. Özyinelemeli Metoda Ait İşlemin Sonucu

METOTLARDA ÖZEL İFADE VE YAPILAR

This Anahtar Sözcüğü

This anahtar sözcüğü global (genel) olarak tanımlanmış değişkenlerin referans ile metot içerisinde kullanılmasını sağlar. Yani sınıf içerisinde tanımlanan değişkenin sınıfa ait olduğunu ifade eden bir referans yapısı olarak tanımlanır. Böylece metot içerisinde benzer isimde tanımlanan değişkenler ayrılır.

Örnek

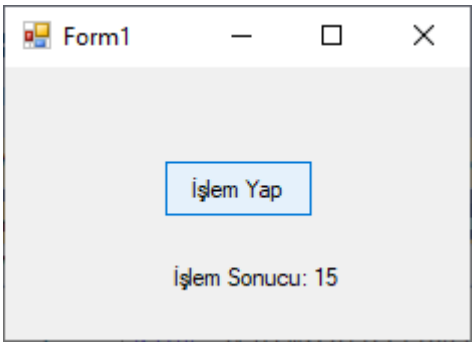
- Global olarak tanımlanmış değişkenin referans şeklinde "this" anahtar sözcüğü ile tanımlanması örneğini aşağıda görebilirsiniz.

```
public partial class Form1 : Form
{
    int sonuc = 0;

    public int SonucEkle(int sonuc)
    {
      this.sonuc = this.sonuc + sonuc;
      return this.sonuc;
    }

    private void IslemYap_Click(object sender, EventArgs e)
    {
      IslemSonucu.Text="İşlem Sonucu: "+ SonucEkle(15).ToString();
    }
}
```

Örnekte gösterildiği gibi sınıf içerisinde global olarak tanımlanan “sonuc” değişkeni “SonucEkle” metodu içerisinde “this” anahtar kelimesi ile birlikte kullanılmıştır. Aynı zamanda “SonucEkle” metodunun giriş parametresi de “sonuc” kelimesi olarak ifade edilmiştir. Böyle durumlarda oluşturulan değişken ismi program için genel olarak mı yoksa metodun giriş parametresi olarak mı ifade edilmiş karışacağı için “this” anahtar sözcüğü kullanılır. Bu sözcük genel yani global olarak tanımlanmış değişkeni kullanabilmek için eklenir. Resim 9.8.’de gösterildiği gibi butona tıklandığı zaman “SonucEkle” metodu için verilen yerel giriş parametresi (15 değeri), global olarak ifade edilen “sonuc” değerine atanır. Daha sonra bu değer işlem sonucuna atanır.



Resim 9.8. "This" Anahtar Sözcüğünün Kullanım Şekli

Main Metodu

Bu metot projenin yani C# uygulamasının başlangıç noktasıdır. Yani geliştirilen program derlenmeye başladığı anda Main metodunu bularak bu kısımdan başlar. Program içerisinde yalnızca bir tane *Main metodu vardır. Main metoduna ait aşırı yüklemeler (overloading) yapılabilir. Yani Main metodu argüman (giriş parametreleri) verilmeden, erişim seviyesi değiştirilerek (standartta private iken public veya protected olabilir) işlem yapılabilir.* Ayrıca geri dönüş değeri verilecek şekilde güncellenebilir. Ancak bu metot kesinlikle statik şekilde tanımlanmalıdır.

Main metodu projenin yani C# uygulamasının başlangıç noktasıdır.

Program Sınıfı

Bu sınıf Visual Studio platformunda C# programı oluşturulurken otomatik oluşturulan bir sınıftır. Projenin ya da programın ilk olarak başladığı sınıf olarak genellikle içerisinde Main ana metoduyla birlikte otomatik oluşturulur.

Void ifadesi oluşturulan metodun geri değer döndürmeyeceği anlamını taşır.

Void İfadesi

Bu ifade oluşturulan metodun geri değer döndürmeyeceği anlamını taşır. Yani oluşturulan metot bir takım işlemler yaptıktan sonra çıkan sonuçlardan herhangi birinin metodun bir çıktısı olarak ifade edilmemesi olayıdır. *Ünitenin başında belirttiğimiz gibi metodu bir fabrika içindeki makinelere benzetirsek, üretilen mamuller fabrika içerisinde başka birimlerin veya makinelerin kullanımına sunulduktan sonra çıkış ünitesinden herhangi bir ürün çıkmaması gibi yapılan programda da oluşturulan metodun program içerisinde bir takım değerleri oluşturup veya kontrolleri sağladıktan sonra çıkış değeri olarak herhangi bir sonuç geri döndürmemesi gibi anlaşılabilir.*

Return İfadesi

Bu ifade nesne tabanlı programlamada sıkça kullanılan ve yapılan işlemlerden sonra istenilen değerin geri gönderilmesi anlamında kullanılmaktadır. Özellikle metotlarda kullanılan bu ifade metot içerisinde bir takım işlemlerden sonra istenilen değerin -metodun veri tipi ile de uyumlu olacak şekilde- metodun sonucu olarak çıktı değer vermesi şeklinde kullanılabilir. Bu ifadeyi bir örnek ile açıklamak uygun olacaktır.

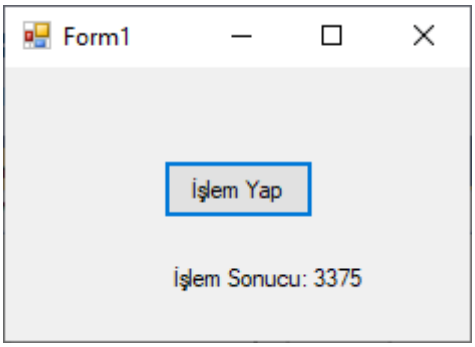
Örnek

•Return ifadesinin kullanımını bir örnek üzerinde gösterim. Bu ifade sayesinde metottan dönen değer, metodun çıktısı şeklinde kullanılacaktır.

```
public int ReturnSonuc(int sonuc)
{
  return sonuc*sonuc*sonuc;
}

private void IslemYap_Click(object sender, EventArgs e)
 {
  IslemSonucu.Text="İşlem Sonucu: "+ ReturnSonuc(15).ToString();
 }
```

Örnekte gösterildiği gibi "ReturnSonuc" metodundan dönen sonuç değeri "IslemSonucu" alanına atanmıştır. Daha sonra bu değişken Resim 9.9.'daki gibi form ekranına değer olarak yazdırılmıştır.



Resim 9.9. Return İfadesinin Kullanımına Ait Çıktı

Aşırı Yüklemeler(Overloading)

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

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.

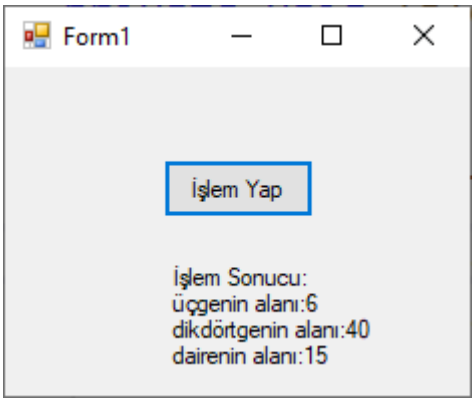
```
public double AlanHesapla (int kenar1, int kenar2, double aci)
{
  double b = (aci * (Math.PI)) / 180;
  return ((double)kenar1 * (double)kenar2 * Math.Sin(b)) / 2;
 }

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 9.10.’da gösterildiği gibi bir sonuç sayfasına ulaşılmaktadır.

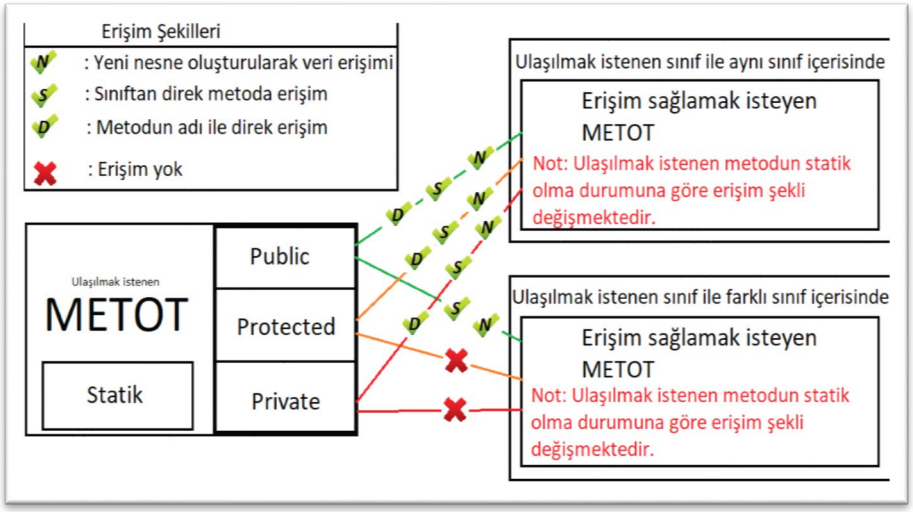


Resim 9.10. Aşırı Yükleme (Overloading) ile Aynı İsme Sahip Farklı Metotlara Ait Sonuçlar

STATIC YAPISI VE KULLANIM ŞEKİLLERİ

Bir metot statik olarak tanımlandıktan sonra erişim seviyesine bağlı olarak kendi sınıfının içinde direk çağrılır.

C# dilinde sıkça karşımıza çıkan bu ifade genellikle tanımlı olduğu sınıfa ait bir değeri veya sonucu vermesi için kullanılır. Yani "static" olarak oluşturulan bir metot veya bir değer sınıfa ait nesne tanımlanmaksızın direk olarak sınıfın altında çağrılarak kullanılır. Bir metot statik olarak tanımlandıktan sonra erişim seviyesine bağlı olarak kendi sınıfının içinde direk çağrılır. Ancak farklı bir sınıf içerisinde erişim seviyesine bağlı olarak (Erişim seviyesi "public" ise bütün sınıflarda, "protected" ise kendi sınıfının altında oluşturulan sınıflarda, "private" ise sadece kendi sınıfında kullanılır.) kullanılır. *Statik ifadesinin kullanım amacı ise sistemin hafızasını ve performansını en iyi şekilde kullanarak daha efektif yazılımlar gerçekleştirmektir.* Static yapısının daha iyi anlaşılabilmesi için Şekil 9.2.'de belirtilmiştir.



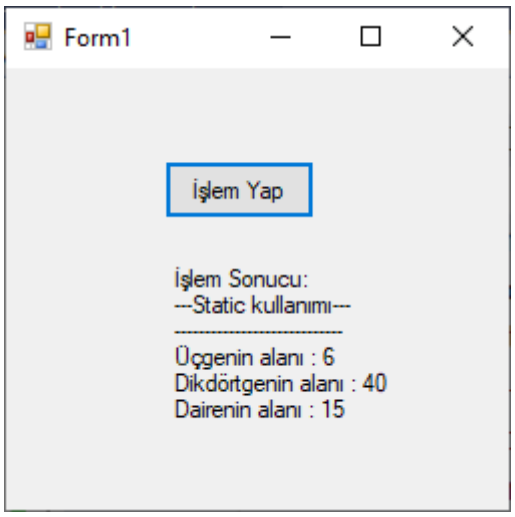
Şekil 9.2. Static İfadesine Bağlı Olarak Erişim Belirleyicilerinin Kullanımı

Konunun daha iyi anlaşılması için örnek bir uygulama yapmak uygun olacaktır.

Örnek

•Static yapısının metotlarda kullanım şekillerinin daha iyi anlaşılması için örnek kodlar aşağıda gösterilmiştir.

```
private void IslemYap_Click(object sender, EventArgs e)
{
   Form1 form1 = new Form1();
   IslemSonucu.Text ="İşlem Sonucu:" + "\n" +
    "---Static kullanımı---" + "\n" +
    "--------------------------" + "\n"
    +"Üçgenin alanı : "+ form1.UcgenAlanHesapla(4,6,30.0) + "\n"
    +"Dikdörtgenin alanı : "+ Form1.DikdortgenAlanHesapla(5,8) + "\n"
    +"Dairenin alanı : "+ DaireAlanHesapla(5) + "\n";
}
public double UcgenAlanHesapla(int kenar1, int kenar2, double aci)//üçgenin alanı
{
   double b = (aci * (Math.PI)) / 180;
   return ((double)kenar1 * (double)kenar2 * Math.Sin(b)) / 2;
}
public static int DikdortgenAlanHesapla(int yatayKenar, int dikeyKenar)//dikdörtgenin alanı
{
   return yatayKenar * dikeyKenar;
}
private static int DaireAlanHesapla(int yariCap)//dairenin alanı
{
   return (int)(Math.PI * yariCap);
}
```



Resim 9.11. Static İfadesinin Kullanımına Ait Sonuçlar

Yukarıdaki örnekte görüldüğü gibi statik(static) ifadesi metotlar için kullanılabilen bir ifadedir. Bu ifadenin kullanım amacı nesne tanımlamaya ihtiyaç

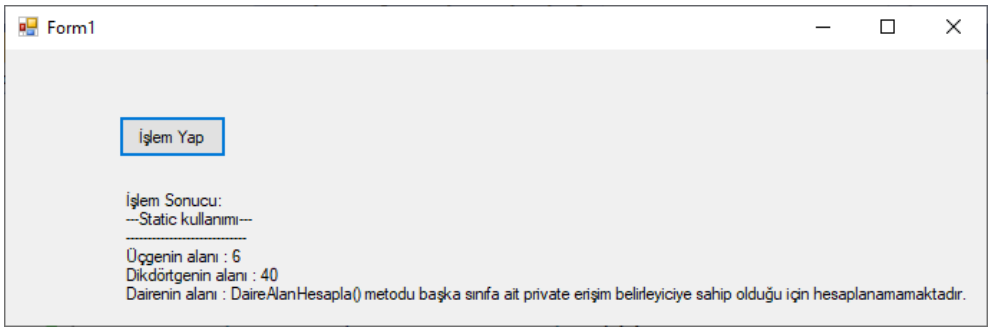
duyulmadan metodun kullanımını sağlamaktır. Yani örnekte "UcgenAlanHesapla" metodunda static ifadesi kullanılmadığı için bu metot nesne tanımlanmadan kullanılamamaktadır. Diğer yandan örnekte oluşturulan "DikdortgenAlanHesapla" metodunda "static" ifadesi olduğundan ve çağrılan sınıf ile aynı sınıfta olduğundan nesne tanımlamadan sınıfın altında çağırmak mümkündür. Benzer şekilde yine örnekte gösterildiği gibi "DaireAlanHesapla" metodu nesne tanımlanmadan direk kullanılmıştır. Resim 9.11.'de sonuçlar gösterilmiştir. Benzer bir örneği de farklı sınıflarda oluşturulan ve statik ifadesinin kullanımına örnek olacak bir uygulama ile göstermek uygun olacaktır.

```
public partial class Form1:Form
{
    private void IslemYap_Click(object sender, EventArgs e)
    {
        Hesapla hesapla = new Hesapla();
        IslemSonucu.Text ="İşlem Sonucu:" + "\n" +
         "---Static kullanımı---" + "\n" +
         "--------------------------" + "\n"
         +"Üçgenin alanı : "+ hesapla.UcgenAlanHesapla(4,6,30.0) + "\n"
         +"Dikdörtgenin alanı : "+ Hesapla.DikdortgenAlanHesapla(5,8) + "\n"
         +"Dairenin alanı : "+ DaireAlanHesapla(5) + "\n";
//"DaireAlanHesapla() metodu başka sınıfa ait private erişim
//belirleyiciye sahip olduğu için hesaplanamamaktadır."
    }
}
public class Hesapla {
    public double UcgenAlanHesapla(int kenar1, int kenar2, double aci)//üçgenin alanı
    {
        double b = (aci * (Math.PI)) / 180;
        return ((double)kenar1 * (double)kenar2 * Math.Sin(b)) / 2;
    }
    public static int DikdortgenAlanHesapla(int yatayKenar, int dikeyKenar)//dikdörtgenin alanı
    {
        return yatayKenar * dikeyKenar;
    }
    private static int DaireAlanHesapla(int yariCap)//dairenin alanı
    {
        return (int)(Math.PI * yariCap);
    }
}
```

Örnekte görüldüğü gibi statik(static) ifadesi farklı sınıflardaki metotlarda belirtilerek kullanılabilen bir ifadedir. Bu ifadenin kullanım amacı farklı sınıflarda bile nesne tanımlamaya ihtiyaç duyulmadan metodun kullanımını sağlamaktır.

Yani yukarıdaki örnekte "UcgenAlanHesapla" metodunda static ifadesi kullanılmadığı için bu metot nesne tanımlanmadan kullanılamamaktadır. Diğer yandan "DikdortgenAlanHesapla" ve "DaireAlanHesapla" metotlarında "static" ifadesi olduğundan "DikdortgenAlanHesapla" sınıf ismiyle kullanılabilirken "DaireAlanHesapla" metodu private erişim belirleyicinin başka sınıfta

kullanılmamasından dolayı kullanılamamıştır. Resim 9.12.'de metotlara ait sonuçlar gösterilmiştir.



Resim 9.12. Statik İfadesinin Kullanımına Ait Sonuçlar

Bireysel Etkinlik

- Hazır metotların olmadığını düşündüğünüz zaman ne gibi problemler ile karşılaşılırdı?

Özet

- Metotlar, yazılımların okunurluluğu, algoritmik yapılar, anlaşılırlık ve güncellemeler ile ilgili işlemlerde kullanılmaktadır. Özellikle metotların isimlerinin doğru ve amacını belirtir şekillerde ifade edilmeleri çok önemlidir. Metotlar, sistemlerin, projelerin ve otomasyonların etkin ve doğru şekillerde gerçekleşmesi için çok önemlidir. 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. Tam bu esnada ortaya koyulan algoritmalar için kodları basitleştirmek, kodları azaltmak, güvenli ve düzenli kod yazımını sağlamak, isimlendirmeler ile anlaşılır hale getirmek, erişim belirleyicileri ile yetkilendirme işlemlerini rahat bir şekilde yapmak çok büyük bir önem arz etmektedir. Bunun için ise nesne tabanlı programlama, sınıf yapısı ve metotlar devreye girmektedir. Metotlar algoritmaların adım adım işlenebilmesi ve gerektiğinde kolay bir şekilde ten noktadan müdahalenin yapılabildiği özel yapılardır. Metotlar her ne kadar kalıplaşmış bir yapıda görünse de çok esnek kullanımlara sahiptirler. Metotlar amaçlarına ve işlevlerine göre çok çeşitli şekillerde kullanılabilmektedir. Metotları daha rahat kullanabilmek için yapılacak işlemleri tem bir amaç gözeterek gerçekleştirmek uygun olacaktır. Metotlardaki isimlendirmelerindeler de bu amaç doğrultusunda belirtildiği takdirde yazılımın anlaşılırlığı oldukça fazla olacaktır.
- Metotlar kullanım amaçlarına göre geri dönüş değerlerine sahip olabilmektedir. Yani metodun yapması gereken işlemi gerçekleştirmesinden sonra return ifadesi ile metodun sonucu olarak bir değer döndürmesi işlemidir. Metot eğer bir değer döndürmüyorsa void ifadesi kullanılarak metodun yalnızca metodun parantezleri içinde işlem yaptığı ve sonuç olarak herhangi bir değere sahip olmadığı anlamı çıkar.
- Metotların kullanım amaçlarına göre static ifadesi ile birlikte hem daha güvenli hem de kullanım amacına daha hizmet etmesi açısından önemlidir. Public, protected ve private erişim belirleyicileri ve static ifadesi metodun kullanım amacına göre değişkenlik göstermektedir. Public erişime sahip olan metotlara hem kendi sınıfları hem de başka sınıflardan erişmek mümkündür. Public ifadesi kullanılan metotlara diğer metot veya sınıflardan erişirken ya sınıftan bir nesne oluşturulur ya da statik ifadesi eklenerek hem doğrudan (aynı sınıfta ise) hem de sınıf üzerinden erişmek mümkün olmaktadır. Protected erişim belirleyicilere sahip bir metoda erişmek için aynı sınıfta olduğu zaman public erişim ile aynı iken farklı sınıftan erişmek isteyen sınıfın, kullanılmak istenen protected metoda sahip sınıftan türetilmesi gerekmektedir. Son olarak ise private erişim belirleyiciye sahip metotların kullanımından bahsetmek gerekmektedir. Bu kullanım ise metodun yalnızca kendi sınıfı tarafından kullanımına izin vermekte başka sınıfların erişimini ise kısıtlamaktadır. Metotların kullanımlarında dikkat edilmesi gereken diğer bir durum ise statik ifadesinin kullanımıdır. Bu ifade metodun sınıf üzerinde direk erişilebilir hale gelmesi yani herhangi bir nesne tanımlamadan kullanımına izin vermektedir.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi özyinelemeli metotları en iyi tanımlayan cümledir?
   a) Bu metotlar başka sınıflardaki metotları çağırmakta ve giriş parametrelerini geri döndürmektedir.
   b) Bu metotlar bir sınıfta yalnızca bir kez tanımlanmakta ve başka sınıflar tarafından kullanılmaktadır.
   c) Bu metotlar sınıf ismi ile aynı olduğundan başka sınıflar tarafından kullanılmaktadır.
   d) Bu metotlar kendini çağıran metot olarak isimlendirilmekte ve ileri yazılım geliştirme tekniklerinde çoğu problemin çözümünde kullanılmaktadır.
   e) Bu metotlarda giriş parametreleri bulunmamaktadır.

2. Aşağıdakilerden hangisi static ifadesinin kullanımını açıklayan cümledir?
   a) Nesne tabanlı programlamada metot oluşturulurken static ifadesi asla kullanılmaz.
   b) Private ifadesi ile static ifadesi bir arada kullanıldığı zaman dönüş değeri string olur.
   c) Sistemin hafızasını ve performansını en iyi şekilde kullanarak daha efektif yazılımlar gerçekleştirmek için kullanılır.
   d) Sınıfa ait bütün yapının bir kopyasının hafızada oluşmasını sağlayarak sınıfa ait bütün değişken ve metot gibi özelliklerin erişim seviyelerine bağlı olarak kullanılmasını sağlamaktadır.
   e) Public ve static ifadeleri bir arada kullanılamaz.

3. Aşağıdakilerden hangisi metot kelimesi ile benzer anlamda kullanılmaktadır?
   a) Yöntem
   b) Öz erişim
   c) Sınıf
   d) Kalıtım
   e) Substring

4. Aşağıdakilerden hangisi kopyalayıcı metotları en iyi şekilde açıklar?
   a) Kodların anlaşılması kolaylaşır ve sınıftaki bütün kodları birebir kopyalar.
   b) Kopyalayıcı metotlar, sadece string giriş parametresi alır.
   c) Kopyalayıcı metotlar sayesinde kopyalanan kodlar başka sınıflar tarafından kullanılamaz.
   d) Kopyalayıcı metotlar, sadece string giriş parametresi alır.
   e) Kopyalayıcı metotlar, yapıcı metotların aşırı yükelemeler ile giriş parametresi olarak kendi sınıfından oluşturulmuş bir nesne alması durumudur.

5. Aşağıdakilerden hangisi yapıcı metotların kullanım amaçlarını en iyi şekilde tanımlar?
   a) Bu metotlar zorunlu olmadıkça kullanılmamalıdır.
   b) Bu metotlar sınıfa ait bütün yapının bir kopyasının hafızada oluşmasını sağlayarak sınıfa ait bütün değişken ve metot gibi özelliklerin erişim seviyelerine bağlı olarak kullanılmasını sağlamaktadır.
   c) Bu metotlarda aşırı yükleme kesinlikle yapılamaz. Erişim belirleyici olarak ise sadece private ifadesiyle birlikte kullanılır.
   d) Bu metotların normal metot yapısından hiçbir farkı yoktur.
   e) Bu metotlarda erişim seviyesinin ifade edilmediği durumlarda erişim seviyesi protected olarak kabul edilir.

6. Aşağıdaki açıklamalardan hangisi private erişim belirleyicilerinin en önemli özelliğidir?
   a) Private ile public aynı erişim seviyesini ifade etmektedir.
   b) Protected ile public erişim belirleyicilerinin ortak noktasıdır.
   c) Bir metodun kesinlikle kullanılmasına izin vermez.
   d) Private ile tanımlanmış bir metodun başka sınıflarda direk kullanımına izin verilmez.
   e) Bir sınıfta yalnızca bir adet private ifadesi kullanılır.

7. Aşağıdakilerden hangisi tarihsel işlemler için oluşturulmuş sınıfın adıdır?
   a) String
   b) DateTime
   c) Public
   d) Math
   e) bool

8. Math sınıfına ait “Cos metodu” kullanımı ile ilgili en uygun açıklama aşağıdakilerden hangisidir?
   a) Bu metot giriş parametresi olarak aldığı değerin radyan açı değerinin kosinüs değerini veren metottur.
   b) Bu metot giriş parametresi olarak aldığı iki değerden büyüğünü geri döndüren metottur.
   c) Bu metot giriş parametresi olarak aldığı iki değerden ilkinin ikincisi kadar üssünü geri döndüren metottur.
   d) Bu metot giriş parametresi olarak aldığı değerin işaretini döndüren metottur.
   e) Bu metot giriş parametresi olarak aldığı değerin radyan açı değerinin sinüs değerini veren metottur.

9. Metinsel metotlar genellikle hangi veri tipi ile ilgili işlemlerde kullanılır?
   a) Integer
   b) Double
   c) String
   d) Boolean
   e) Char

10. Aşağıdakilerden hangisi hazır metotların kullanımı en iyi şekilde açıklayan cümledir?
   a) Hazır metotlar sadece string sınıfından oluşur.
   b) Hazır metotlar, benzer metotların yazılmasını engeller.
   c) Yazılımcı hazır metotları hiçbir şekilde kullanmaması gerekir.
   d) Bir sınıf içinde hazır metotlardan en fazla iki tane kullanılır.
   e) Yazılımcının daha hızlı ve etkin kodlama yapmasını sağlayarak zaman kazandırır.

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

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