Nesne Tabanlı Programlama

Ders Notu - 3

Dicle Üniversitesi
Mühendislik Fakültesi
Elektrik Elektronik Mühendisliği Bölümü

1

Sınıflar ve Fonksiyonlar

Tekrar kullanılabilir kodlar

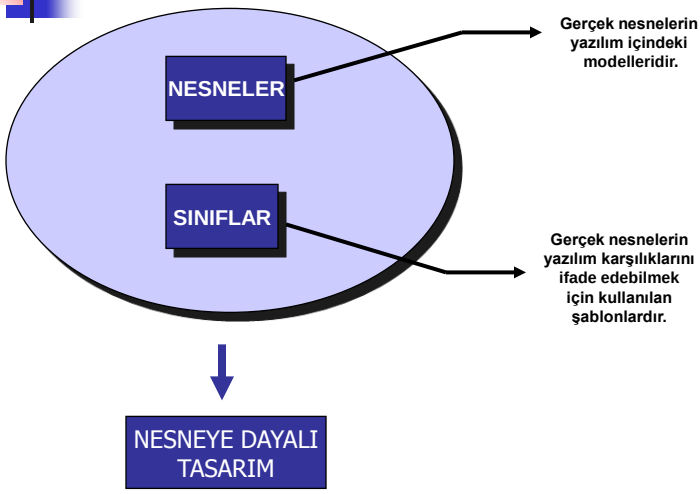
Nesne Tabanlı Tasarım

- Nesne tabanlı programlama kavramı 1960'larda ortaya çıkmıştır.
- 1972'de ortaya çıkan Smalltalk, gerçek anlamda ilk nesne tabanlı programlama dilidir.
- Smalltalk, nesne tabanlı programlamayı tanımlamış ve kurallarını koymuş bir dildir.
- C++'ın 1980'lerde çıkışı nesne tabanlı programlamayı, standart bir uygulama haline getirmiştir.
- C++ dilinin takipçileri olan C# ve Java dilleri yine nesne tabanlı diller olarak adlarını duyurmuşlardır.

NESNEYE DAYALI TASARIM

Kavramlar ve Tasarım Süreci

Nesneye Dayalı Tasarım Anahtar Kavramlar



Nesneler

Sınıflar

- Sınıf, gerçek hayattaki bir nesneyi istenen bir şekilde modelleyebilmek için gerekli bütün kod ve veriyi içeren yazılım birimidir.
- Modülaritenin yeni birimidir.
- Sınıf, hem veriyi hem de yöntemi içinde barındırır.
- Modelleme, yazılım üretiminin temelidir.

Nesne – Sınıf ?

- Sınıf nesnenin tanımıdır. Soyut bir kavramdır.
- Nesne sınıfın bir örneğidir. Somut bir niceliktir.
- Örneğin; İnsan sınıfı bir tanımlar kümesiyken, her birimiz İnsan sınıfının birer nesnesiyiz.

Sınıflar

- Bir sınıfın temel işlevi, nesnede kullanılan yöntemleri ve verileri tanımlamaktır.
- Sınıf tanımı aslında nesnenin soyut bir karşılığıdır.
- Nesneye dayalı programlamada, soyutlama tasarım sürecinin temelidir.

Sınıfın İsmi / Tip

Nesne

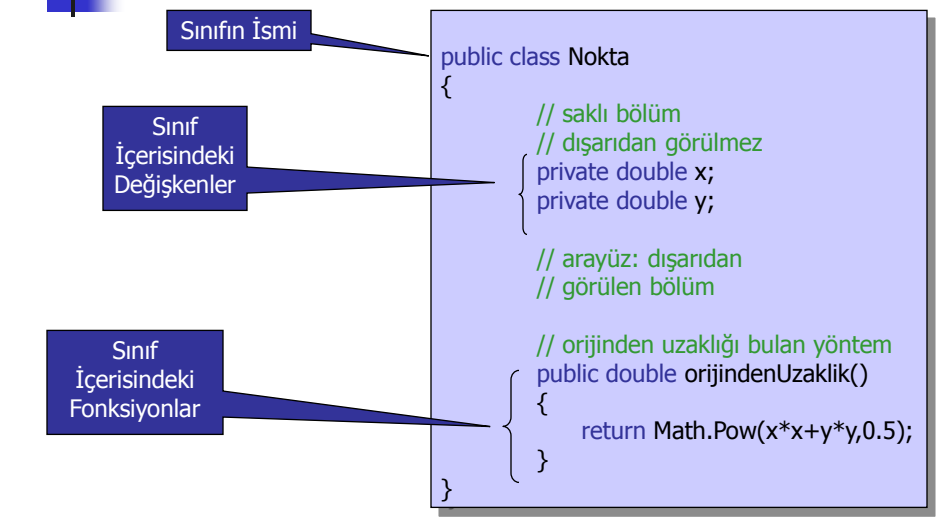
```
static void Main(string[] args)
{
    Araba a = new Araba(1999,"Honda");

    if(a.vitesGetir() != 0) {
        a.vitesiDegistir(0);
    }

    a.calistir();
    a.debriyajaBas(0);
    a.vitesDegistir(1);
    .....
    .....
}
```

Sınıflar

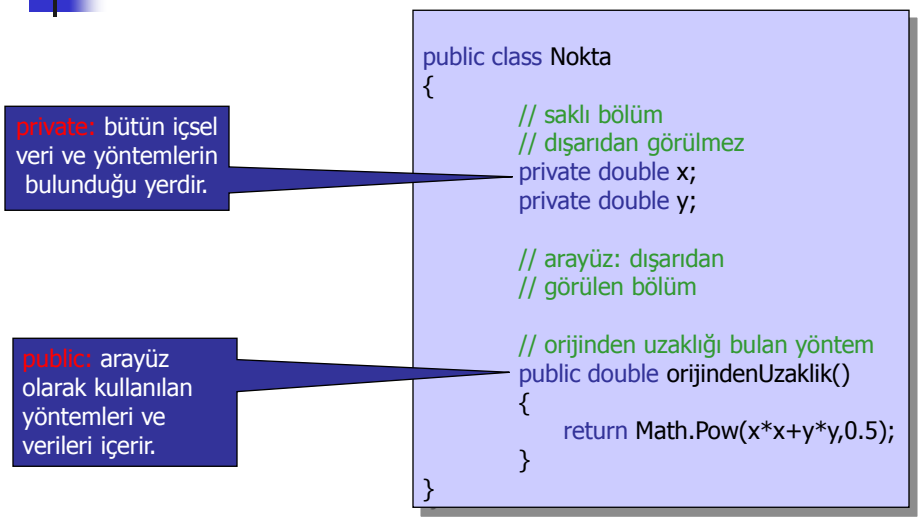
- C# sınıfı yeni bir TİP tanımlar.
- Nesne, sınıfın hafızada oluşturulmuş bir örneğidir.
- Sınıfın arayüzü ve uygulama bölümleri vardır. Arayüz dışarıdan nasıl gözüktüğü, uygulama bölümü sınıfın ne yaptığıdır.
- Sınıflarda arayüz iyi tanımlanmalıdır.
- Sınıflar, "private" anahtar kelimesiyle veriyi dışarıdan saklar.
- "public" anahtar kelimesi arayüzü belirler.

```
public class Nokta
{
        // saklı bölüm
        // dışarıdan görülmez
        private double x;
        private double y;

        // arayüz: dışarıdan
        // görülen bölüm

        // orijinden uzaklığı bulan yöntem
        public double orijindenUzaklik()
        {
            return Math.Pow(x*x+y*y,0.5);
        }
}
```

SINIFLAR: Erişim Denetimi



SINIFLAR: Erişim Denetimi



SINIFLAR: Erişim Denetimi

Sınıf içindeki üye değişkenlere ve yöntemlere erişimin denetlenmesi üç farklı C# anahtar kelimesiyle sağlanır.

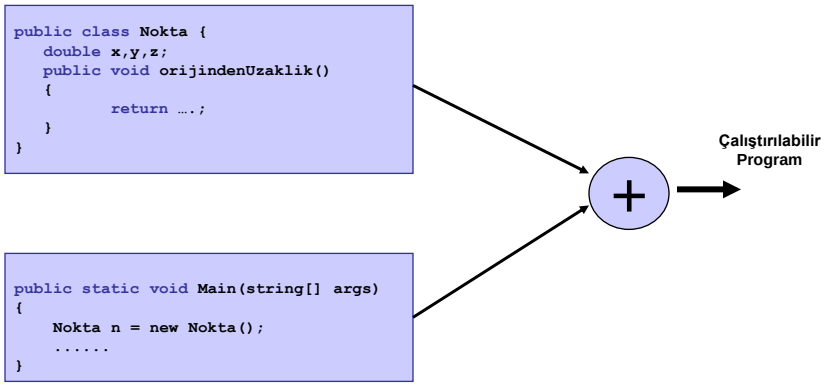
- **private:** sınıf dışından görünmeyen değişkenlerin ve yöntemlerin tutulduğu bölümü işaretler
- **public:** sınıf dışından görünen ve erişilebilen değişkenlerin ve yöntemlerin bulunduğu bölümü tanımlamak için kullanılır
- **protected:** Türetilmiş sınıflardan erişilebilen değişkenlerin ve yöntemlerin bulunduğu bölümdür. Dışarıdan görülmez ama türetilmiş sınıflardan erişilir.

SINIFLAR: Erişim Denetimi

- Sınıf tanımında varsayılan erişim denetimi private'dır.
- Sınıf tanımına başlarken, hiç bir erişim denetimi anahtar kelimesi yazılmazsa, private olarak alınır.

```
public class Nokta
{
        // saklı bölüm
        // dışarıdan görülmez
        double x;
        double y;

        // arayüz: dışarıdan
        // görülen bölüm

        // orijinden uzaklığı bulan yöntem
        public double orijindenUzaklik()
        {
            return Math.Pow(x*x+y*y,0.5);
        }
}
```

Bir C# Programının Yapısı



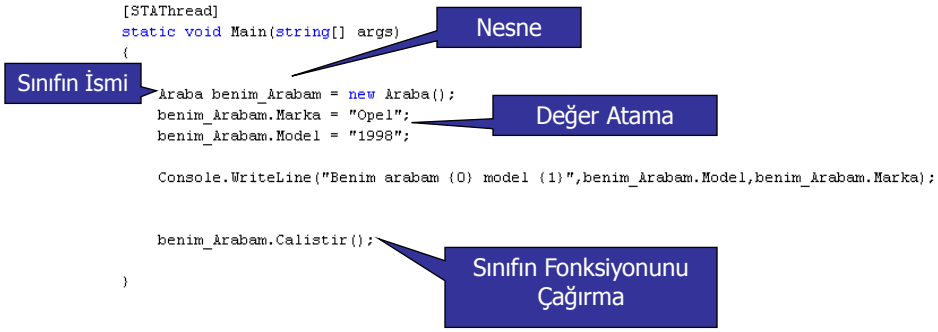
SINIFLAR: Kullanım

Bir Araba sınıfı oluşturalım. Sınıfın Marka ve Model isminde iki public değişkeni ve Calistir isminde bir public Fonksiyonu olsun. Calistir fonksiyonu ekrana yazı yazsın.

```
public class Araba
{
    public string Marka;
    public string Model;
    public void Calistir()
    {
        Console.WriteLine("hrnnnnn... rnnnnnnn");
    }
}
```

SINIFLAR: Kullanım

Araba sınıfı programın başka bir yerinde kullanılmak istendiğinde öncelikle Araba tipinde bir nesne oluşturulmalıdır. Daha sonra sınıfın public üyelerine isteğe göre değere atanabilir, çağrılabilir.



Sınıf Çalışması

- İnsan sınıfı yazınız.
- Özellikler
  - Adı
  - Soyadı
  - Yaşı
- Main fonksiyonu içerisinde yazdığınız sınıfın bir nesnesini oluşturarak değer atayınız.
- Atanan değerleri ekrana yazdırınız.

SINIFLAR - Fonksiyonlar

- Nesne Tabanlı Mimaride sınıflar için tanımlanmış daha bir çok özellik mevcuttur.
  - Sarma(Encapsulation)
  - Polimorfizm(Polymorphism)
  - Kalıt(Inheritance)
- Bu özellikleri incelemeden önce sınıfların içerisinde tanımlanan fonksiyonların üzerinde duralım.

Fonksiyonlar (Yöntemler)

- Fonksiyonlar belirli bir işi gerçekleştirmek amacı ile yazılmış değişkenler ve ifadeler kümeleridir.
- Bilirli tipte parametreler alarak belirli tipte sonuçlar döndürürler.
- Programın diğer bölümlerinden tekrar tekrar çağırılabilirler.

Temel Yazım :

```
dönen_tip FonksiyonAdı (Parametreler)
```

Fonksiyonlar (Yöntemler)

- Fonksiyonlar bir sınıf içerisinde bulunmalıdır.
- Bir sınıfın içerisinde bulunmayan, kendi başına bir fonksiyon yazılamaz.

```
public class benimSınıfım
{
    public void fonksiyon()
    {
        birşeyleryap;
    }
}
```

Fonksiyonlar

- Şu ana kadar, sadece tek bir fonksiyon yazarak programlarımızı yazdık. Bu fonksiyon da static void Main(string[] args) fonksiyonuydu.
- Yazımı şöyleydi:

```
static void Main(string[] args)
{
    /* tanımlar    */
    /* ifadeler    */
}
```

- `void` kelimesi, fonksiyonun herhangi bir değer döndürmeyeceğini belirtmektedir.

Fonksiyonlar

- `()` parantezleri `Main`'in bir fonksiyon olduğunu belirtmektedir. Bu parantezlerin içine, fonksiyona gönderilecek parametreler konulabilir. `Main(string[] args)` fonksiyonu içine string[] tipinde değerler gönderilebileceğini anlatan parametre kullanılmıştır.
- Çoğunlukla fonksiyonlar parametre listesi isterler. Bu parametrelere göre işlem yaparak, bir iş yaparlar ve sonuç döndürürler.

Fonksiyonlar

- İki farklı deyimle, yazılan fonksiyon kullanılabilir hale getirilebilir. Bunlar
  - Fonksiyonun kendisi (ilk satır + içindeki deyimler)
  - Fonksiyonu çağıran deyim.

Fonksiyon Tanımı

- Fonksiyon tanımının ilk satırına ; (noktalı virgül konulmaz)

```
private float toplam(int a,int b,int c)
```

- Burada, `toplam` ismindeki fonksiyon `a`, `b` ve `c` isminde üç parametre istemektedir. Fonksiyondan dönen değer `float` tipindedir.
- `a`, `b` ve `c` değişkenlerine, bu fonksiyonu çağıran deyim, değer yükler.

Fonksiyon Tanımı

- Fonksiyonun tanımı, tanımlayıcı ilk satır ve bundan sonra gelen küme parantezleriyle çerçevelenmiş bir dizi deyimden oluşur:

```
private float toplam( int a, int b, int c)
{
      float t;

      t = a + b + c ;
      return t;
}
```

- Not: Küme parantezleri fonksiyonun başlangıç ve bitişini belirler.

Fonksiyon Tanımı

```
private float toplam( int a, int b, int c)
{
    float t;
    t = a + b + c ;
    return t;
}
```

- `a, b` ve `c` değişkenleri sadece `toplam` fonksiyonu içinde geçerlidir.
- Fonksiyon içinde tanımlanan `t` değişkeni de sadece bu fonksiyon içinde oluşturulmuştur ve çıkışta silinir.

Fonksiyon Tanımı

- Fonksiyon `return` anahtar kelimesiyle bir değer döndürür. Eğer dönen tipi `void` olursa, `return` deyimi genelde kullanılmaz.
- return anahtar kelimesinin döndürdüğü değişken veya değerin tipi fonksiyonun tanımında verilen tiple aynı olmalıdır.

```
private float toplam( int a, int b, int c)
{
      float t;

       t = a + b + c ;
       return t;
}
```

Fonksiyonun Çağrılması

- Fonksiyonun çağrılmasında, fonksiyon tanımı dikkate alınır. Sadece veri tipleri yazılmaz ve değerler sabitler yada değişkenler olabilir. Genellikle, fonksiyondan gelen değer bir değişkene atanır.

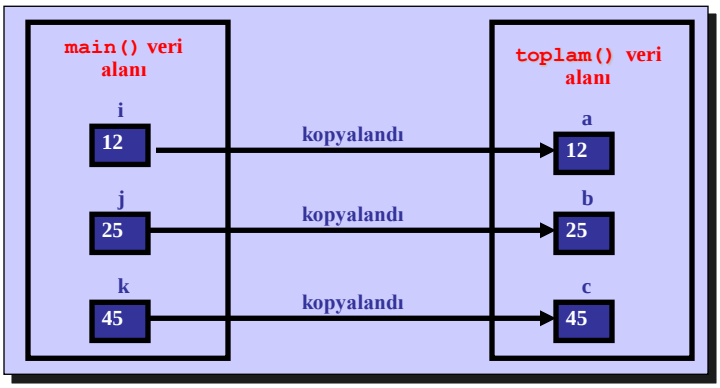
```
deger = toplam( i, j, k ) ;
```

veya

```
deger = toplam( 5, x, 2.7 ) ;
```

Fonksiyonun Çağrılması

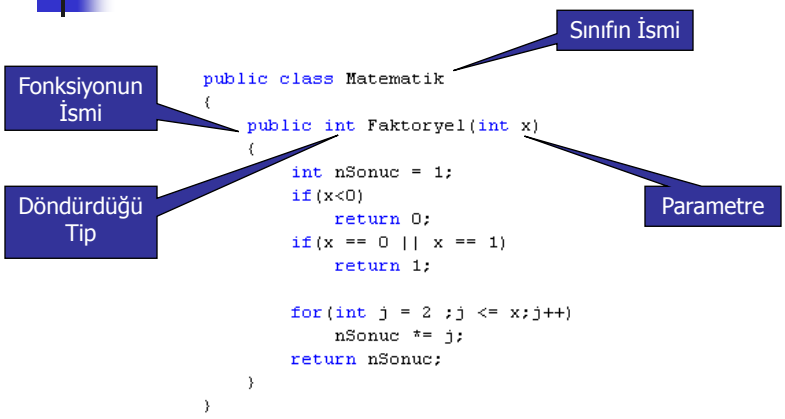
main() fonksiyonu içinden **toplam()** fonksiyonu çağrıldığında, **main()**'in veri alanında bulunan değerler, **toplam()** fonksiyonunun veri alanına kopyalanır.



Alıştırma: Faktoryel Fonksiyonu

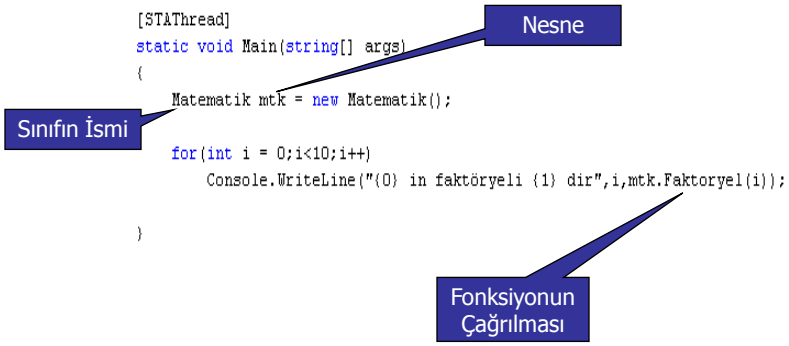
- Bir sayının faktöryelini alan fonksiyonu ve bulunduğu sınıfı yazınız. Fonksiyon 0'dan küçük değerler için 0 döndürürken, 1 ve 0 için 1 değerini döndürecektir. 2'den büyük sayılar için faktöryeli bir döngü kullanarak hesaplayınız.
- Ana programda1'den 10'a kadar olan sayıların faktöryelini ekrana yazınız.
- Fonksiyonun bulunduğu sınıfın nesnesini kullanmanız gerektiğini unutmayınız.

Örnek: Faktoryel Fonksiyonu




```
public class Matematik
{
    public int Faktoryel(int x)
    {
        int nSonuc = 1;
        if(x<0)
            return 0;
        if(x == 0 || x == 1)
            return 1;

        for(int j = 2 ;j <= x;j++)
            nSonuc *= j;
        return nSonuc;
    }
}
```

Örnek: Faktoryel Ana Programı



Sınıf Çalışması

- İki parametre alan, bu değerlerin farkının faktöryelini döndüren fonksiyonu yazınız.
- Fonksiyonun ismi şudur:

```
int faktoryel_fark(int a, int b)
```

- Bu fonksiyon önceki slaytlarda bulunan `faktoryel()` fonksiyonunu çağırabilir.
- Fonksiyonun bulunduğu sınıf olarak **`faktoryel()` fonksiyonunun bulunduğu sınıfı kullanınız.**

Sınıf Çalışması: Ana Program

Yazılacak sınıfın kullanımı aşağıdaki gibi olacaktır.

```
[STAThread]
static void Main(string[] args)
{
    Matematik mtk = new Matematik();

    int a = 5;
    int b = 3;
    int sonuc = mtk.Faktoryel_Fark(a,b);
    Console.WriteLine(sonuc);
}
```

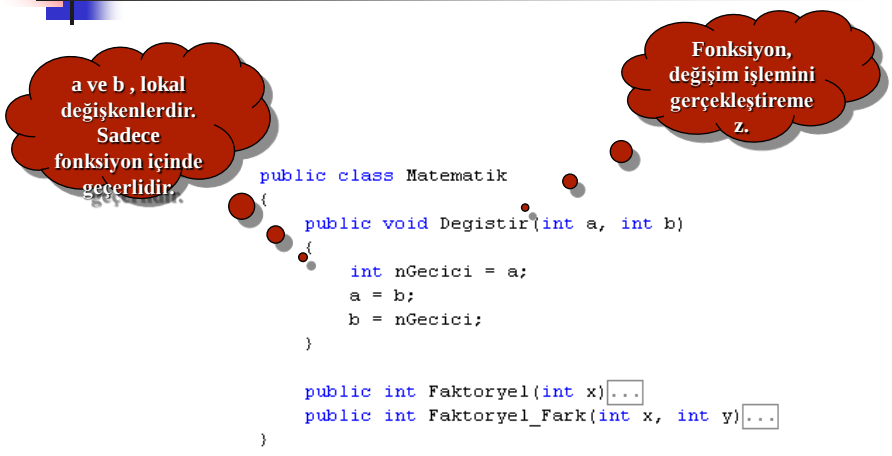
Örnek

Gönderilen iki değişkenin değerini değiştiren bir fonksiyon yazınız.

```
[STAThread]
static void Main(string[] args)
{
    Matematik mtk = new Matematik();

    int a = 5;
    int b = 3;
    Console.WriteLine("{0} {1}",a,b);
    mtk.Degistir(a,b);
    Console.WriteLine("{0} {1}",a,b);
}
```

İlk çıktıda **"5 3"** yazarken ikinci çıktıda **"3 5"** yazacaktır.

Örnek

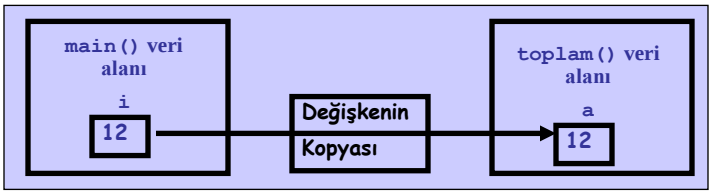


Neden Çalışmadı?

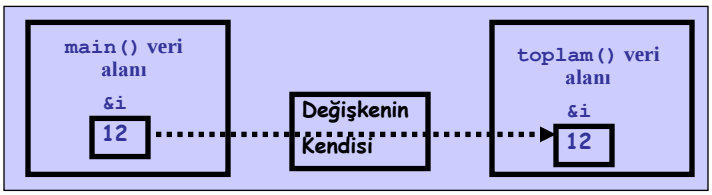
- Çünkü, fonksiyonlara iki şekilde değer gönderilebilir.
  - değer
  - referans
- Değerle: Fonksiyon çağrılırken gönderilen değişkenlerin değerlerin kopyası çıkartılır ve sadece değerleri gönderilir.
- Referansla: Çağrılırken, değişkenlerin adresleri gönderilir. Fonksiyon içinde o adresteki değerler değiştirildiğinde, ana programdaki değişkenlerin değerleri de değişmiş olur.

Değerler - Referanslar

Değer



Referans



Refaranslar

- Bir fonksiyona referans olarak değer göndermek istendiğinde ref anahtar kelimesi kullanılır.
- Fonksiyon tanımında referans olarak kullanılacak değişkenin tipinin önüne ref anahtar kelimesi konulur.
- Fonksiyon çağrılırken referans değişkeninin önüne yine ref anahtar kelimesi konulur.

Örnek - Düzeltme

Degistir fonksiyonunun parametrelerinin önüne ref anahtar kelimesi koyarak parametreleri referans tipine çevirelim.

```
public class Matematik
{
    public void Degistir(ref int a, ref int b)
    {
        int nGecici = a;
        a = b;
        b = nGecici;
    }

    public int Faktoryel(int x)...
    public int Faktoryel_Fark(int x, int y)...
}
```

Örnek - Düzeltme

Ana programda fonksiyona değişkenleri ref anahtar kelimesi ile aktardığımızda, programın doğru çalıştığını göreceksiniz.

```
[STAThread]
static void Main(string[] args)
{
    Matematik mtk = new Matematik();

    int a = 5;
    int b = 3;
    Console.WriteLine("{0} {1}",a,b);
    mtk.Degistir(ref a, ref b);
    Console.WriteLine("{0} {1}",a,b);
}
```

SINIFLAR: Kurucu Fonksiyon(Constructor)

- Kurucu fonksiyonlar, nesne hafızada oluşturulurken çağrılan ve nesneyi kurma işlemini yapan fonksiyonlardır.
- Kurucu fonksiyon, sınıfla aynı isme sahiptir. Kurucular hiç bir şey döndürmezler (void bile)
- Birden fazla kurucu olabilir. Her birinin argümanları farklı olmalıdır (imzaları farklı).
- Kurucu tanımlanmayan sınıflarda, varsayılan kurucu otomatik olarak kullanılır.

```
public class Nokta
{
    // dışarıdan görülmez
    private double x;
    private double y;

    // arayüz
    public Nokta()      // varsayılan kurucu
    {
        x = y = 0.0;
    }

    // alternatif kurucu
    public Nokta(double xx,double yy)
    {
        x = xx;
        y = yy;
    }
    // orijinden uzaklığı bulan yöntem
    public double orijindenUzaklik()
    {
        return Math.Pow(x*x+y*y,0.5);
    }
}
```

SINIFLAR: Kurucu Fonksiyon(Constructor)

- Sadece bir tane varsayılan (argümansız) kurucu fonksiyon olabilir.
- new anahtar kelimesi ile nesne oluşturulurken Constructor çağrılır.
- Basit tipli değişkenler kurucu fonksiyon kullanılmadan otomatik olarak varsayılan değerler alırlar.

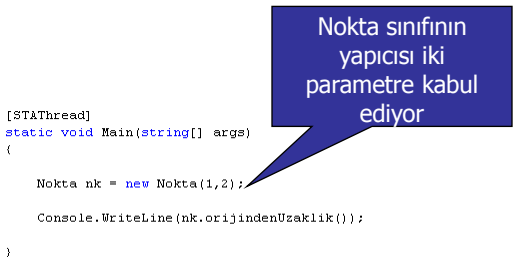
```
public static void Main(string[] args)
{
    Nokta n = new Nokta();
    ......
}
```

Constructor çağrıldı

```
Numerik (int,long…) -> 0
Bool                -> false
Char                -> '\0'
Referans            -> null
```

SINIFLAR: Kurucu Fonksiyon(Constructor)

- Sınıflar oluşturulurken yapıcılarındaki parametrelere göre yazılırlar.




```
[STAThread]
static void Main(string[] args)
{

    Nokta nk = new Nokta(1,2);

    Console.WriteLine(nk.orijindenUzaklik());

}
```

Sınıf Çalışması

Zaman sınıfı yazınız.

Sınıfın 2 adet kurucu fonksiyonu

Saati yazdıran bir fonksiyonu

Ve değer eklemeye yarayan bir fonksiyonu olacaktır.

Yazdığınız sınıfın bütün metodlarını bir örnekte kullanınız.

```
public class Zaman
{
    public  int saat;
    public  int dakika;
    public  int saniye;

    public Zaman(int st, int dk, int sn)
    {
        ///// CODE
    }

    public Zaman(Zaman z)
    {
        ///// CODE
    }

    public void SaatiYazdır()
    {
        ///// CODE
    }

    public void ZamanEkle(int st, int dk, int sn)
    {
        ///// CODE
    }
}
```

Sınıflar: Statik Üyeler

- Sınıf içinde bulunan statik veri üyeleri, o sınıftan türetilen bütün nesneler tarafından erişilir.
- Statik üyenin, bütün nesnelerce paylaşılan hafızada tek bir kopyası vardır. Birinin değiştirdiğini diğeri görebilir.
- Statik üyeler sınıflar arasında bilgi taşıyabilir.
- Statik üyeler static anahtar kelimesi ile oluşturulur.

Sınıflar: Statik Üyeler

Bir bilgisayar oyunu yazdığımızı düşünelim. Oyunda Martian isminde uzaylıyı modelleyelim. Martianlar en az 3 kişi oldukarında saldırıyorlar, eğer sayıları 3 kişiden azsa saldırmaya cesaret edemiyorlar. Her bir Martian çevresinde kaç Martian olduğunu nasıl bilecek?

```
public class Martian
{
    public static int count;
    public Martian()
    {
        count++;
    }
    public void Attack()
    {
        if(count<3)
            Console.WriteLine("nooo!");
        else
            Console.WriteLine("arrggh!");

    }
}
```

```
[STAThread]
static void Main(string[] args)
{
    Martian ebu = new Martian();
    ebu.Attack();

    Martian abu = new Martian();
    abu.Attack();

    Martian ubu = new Martian();
    ubu.Attack();

    abu.Attack();
}
```

Sınıflar: Statik Üyeler

- Sınıflar aynı zamanda statik fonksiyonlar içerebilirler.

```
public class Math
{
    public static double PI = 3.14;
    public static double AlanHesapla(int radius)
    {
        return PI * radius * radius;
    }
}
```

Statik metod

```
[STAThread]
static void Main(string[] args)
{
    Console.WriteLine(Math.AlanHesapla(2));
}
```

Nesne oluşturulmadan doğrudan çağrılmış

Sınıflar: Statik Üyeler

- Statik fonksiyonları kullanmak için nesneye ihtiyaç yoktur.

```
[STAThread]
static void Main(string[] args)
{
    string sYazi = "135";
    int nSayi = Convert.ToInt32(sYazi);
}
```

Convert sınıfından bir nesne oluşturulmadan ToInt32 fonksiyonu çağrılmış

Sınıf Çalışması

- Sıcaklık birimleri arasında dönüşüm yapan SicaklikDonustur isimli sınıf yazınız.
- Sınıf Celcius, Fahrenheit ve Kelvin arasında dönüşüm yapacaktır.
- Dönüşümü yapacak olan fonksiyonlar statik olacaktır. Yani dönüşüm yapmak için nesneye ihtiyaç bulunmayacaktır.
- Ek bilgi:
  - Tc=(5/9)*(Tf-32)
  - Tc = Tk - 273.18

const ve readonly üyeler

- C# dili programcılara programın çalışması esnasında değerleri değiştirilemeyecek olan değişken tipleri sunar.
- Sınıf içerisinde sabit değerli bir üye oluşturmak için değişkenin isminin başına const veya readonly anahtar kelimeleri konulur.
- const değişkenlerin değerleri tanımlama aşamasında verilmelidir.
- readonly değişkenlerin değerleri tanımlama anında veya sınıfın kurucu fonksiyonunda verilebilir.
- Bu aşamalardan sonra const ve readonly tipindeki değişkenlerin değerleri bir daha değiştirilemez.

const ve readonly üyeler

```
public class Daire
{
    int radius;
    public const double PI = 3.14;
    public Daire(int r)
    {
        radius = r;
        PI = 3.1459;
    }
    public double CevreHesapla()
    {
        return 2 * PI * radius;
    }
}
```

```
[STAThread]
static void Main(string[] args)
{
    Daire d = new Daire(4);

    System.Console.WriteLine( d.CevreHesapla());
}
```

Hata!
const olarak tanımlanan değişkenin değeri değiştirilmeye çalışılıyor

Numaralı Tipler

- Kendi istediğimiz veri tipini oluşturma şansını C dili bize vermektedir.
- Kullanımı programların okunabilirliğini arttırabilir.
- Örneğin, haftanın günlerini içeren bir veri tipi şöyle oluşturabiliriz:

```
enum Gunler { Pazartesi, Sali,
              Carsamba,  Persembe,
              Cuma, Cumartesi,
              Pazar };
```

Numaralı Tipler

- Oluşturduğumuz tipten değişkenler oluşturabiliriz.

```
enum Gunler gun = Gunler.Sali;
..
...
switch(gun)
{
       case Gunler.Pazartesi:
              printf("Pazartesi"); break;
       case Gunler.Sali:
              printf("Sali"); break;
       ......
}
```

Sınıf Çalışması

- Üniversitede bulunan insanları enum kullanarak sıralayınız. (Öğretim Üyesi, Öğrenci, Memur, İşçi)
- Genel bir insan sınıfı yazınız.
- Sınıfın içerisinde oluşturduğunuz enum tipindeki bir değişken ile nesnenin hangi pozisyonda olacağını tutunuz.
- Yine sınıf içerisine enum değerine göre kendi özelliğini ekrana yazan bir fonksiyon yazınız.