Metotlar ve Fonksiyonlar

- *Metot Nedir?*
- *Metot bildirimi*
- *Metot Parametresi olarak diziler*
- *Değer ve Referans Parametreleri*
- *Ref ve out Anahtar sözcükleri*
- *Recursive (Özyineli) metotlar*
- *Main, Math Sınıfı*

Main Metodu

Main metodunu diğer metotlardan temel olarak farkı yoktur, tek farkı programın çalışmaya başladığı nokta olmasıdır.

```
static void Main(string[ ] args)
…
foreach (string ss in args)
  Console.WriteLine(ss);
 if (args[0].Equals("topla"))
     topla(Convert.ToInt16(args[1]),Convert.ToInt16(args[2]));
 Console.ReadKey();
```

System.Math sınıfı ve metotları

.Net sınıf kütüphanesinde belirli matematiksel işlemleri yapmak için System isim alanında bulunan Math sınıfı bulunmaktadır.

Static tanımlı olduğu için yeni bir nesne oluşturmaya gerek yoktur.

```
double pi=Math.PI;
Console.WriteLine("Pi değeri={0}",pi);
```

System.Math sınıfı ve metotları

Abs(x)	Mutlak değerini alır. (x double)
Cos(x),Sin(x)	Cosinus, Sinus değerini verir(x radyan)
Ceiling(x)	x'ten büyük ilk tamsayıya
Floor(x)	x'ten küçük ilk tamsayıya
Max(x,y) Min(x,y)	Maksimumu verir.(2 parametre alır) Minimumu verir.(2 parametre alır)
Pow(x,y)	x üssü y
Sqrt(x)	Karekök değeri.(<0 için NaN değeri)
Log(x)	e tabanında logaritma
Log10(x)	10 tabanında logaritma
Exp(x)	e^x değerini verir

Sınıflar, Yapılar ve Numaralandırmalar(Enumeration)

- *Sınıflar, bildirimleri*
- *Yapıcı Metotlar*
- *Yıkıcı Metotlar*
- *Statik üye Elemanlar*
- *const ve readonly elemanları*
- *Yapılar*
- *Numaralandırmalar (Enumeration)*

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

- **Nesne Yönelimli Programlama(NYP) nedir?**

  Çözülmesi istenen problemi çeşitli parçalara ayırıp her bir parça arasındaki ilişkiyi gerçeğine uygun şekilde belirleme tekniğine NYP denir.

  NYP de her şeyin nesnelere dayalı olması gerekir. C# 'ta herşey nesne olduğu için %100 nesne yönelimlidir.

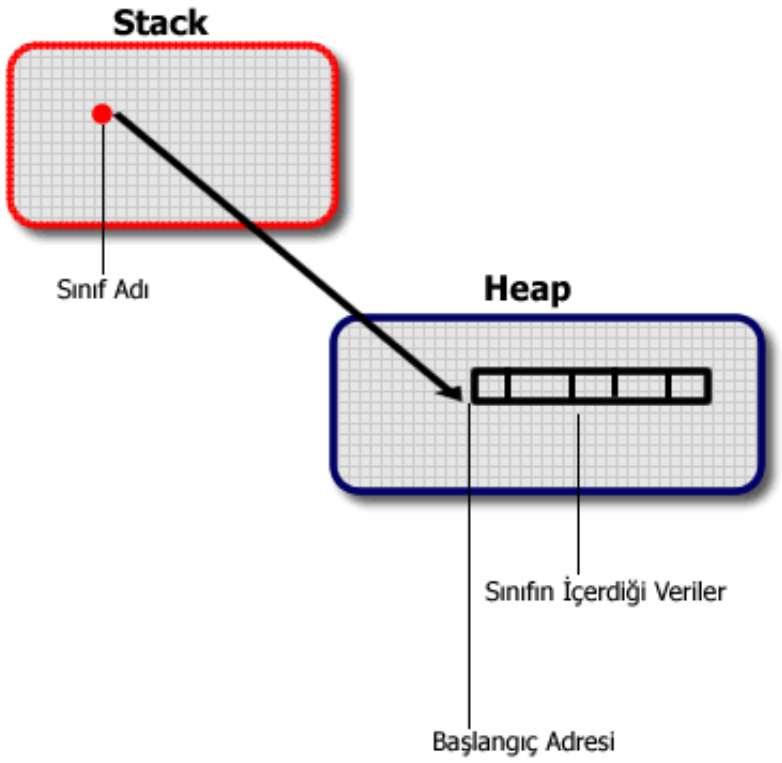
Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

- **Sınıf bildirimi**

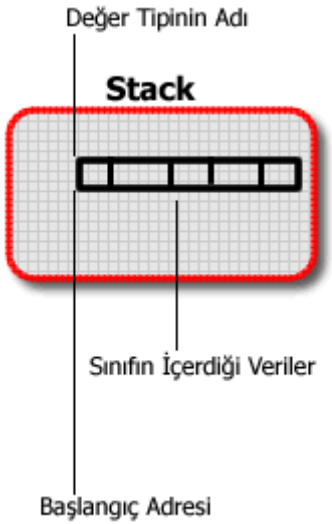
  Sınıf bildirimi class anahtar kelimesi ile gerçekleştirilir. Sınıf isminden sonra sınıf içindeki üye eleman ve metotlar tanımlanır.

```
class sinif_ismi
{
erişim veri_tipi değişken;
erişim dönüş_değeri metot_ismi(parametreler)
  {
  }
}
```

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)



Referans Tiplerinin Bellekte Tutuluş Şekli



Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

C# dilinde 5 tane erişim belirleyici vardır. Public, private (varsayılan), protected,internal ve private protected anahtar sözcükleri ile belirlenir.

Örneğin: Sistem içindeki Array sınıfı public tanımlı olduğudan Sort metodunu kullanabiliyoruz.

Diğer bir erişim belirleyicisi olan protected ise private gibidir, kalıtım konusunda tekrar bahsi geçecektir.

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

public : public olarak tanımlanan öğe, kod bloğunun içinde ve dışında tamamen erişilebilirdir. Yani, hiçbir kısıtlama yoktur.

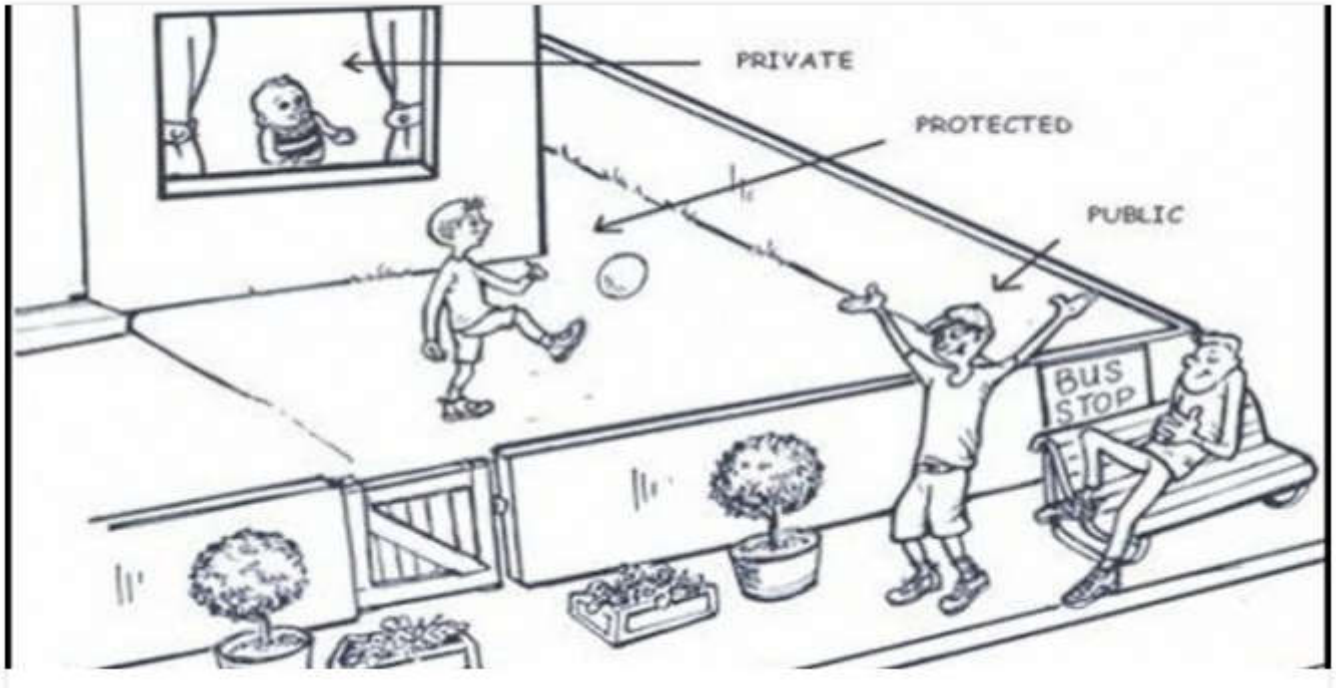
protected : protected olarak tanımlanan öğe, sadece tanımlandığı class'ın içinde ve o class'tan türetilmiş diğer class'ların içinde erişilebilirdir.

internal : internal olarak tanımlanan öğe, bulunduğu assembly'nin (Dll veya Exe dosyası) içinde erişilebilirdir. Dll veya Exe dosyasının içerisinde erişim için kısıtlama yoktur, ama dışarıdan erişilemez.

protected internal : protected internal erişim belirleyicisi, protected ve internal erişim belirleyicilerinin VEYA (OR) işlemiyle birleştirilmiş halidir. protected internal olarak tanımlanmış öğe, tanımlandığı class'ın içinde ve o class'tan türetilmiş diğer class'ların içinde erişilebilir. Ayrıca, aynı assembly içinde olmasalar dahi, tanımlandığı class'tan türetilmiş diğer class'ların içinde de erişilebilirdir.

private : private olarak tanımlanan öğe, sadece tanımlandığı class'ın içerisinde erişilebilirdir. En katı erişim belirleyicidir.

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)



Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

Sınıf bir veri yapısıdır. Gerçek hayattaki bir nesneye yönelik modelleyebiliriz. Örneğin bir kredi kartı hesabı için bir çok özellik vardır, hesap no,limit,kart sahibi. Bu bizim için gerçek hayattaki bir modeli temsil eden bir yapıdır.

```
class KrediHesabi
{
public int HesapNo;
public double Limit;
public string KartSahibi;
}
```

Sınıf bildirimleri için bellekte yer tahsis edilmez.

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

Sınıf Nesneleri Tanımlama

Sınıf türünden nesnelerin tanımlanması new anahtar sözcüğü ile yapılmaktadır. Sınıfı da bir veri türü gibi düşünebiliriz. Aşağıdaki şekilde bir tanımda yapılabilir.

```
KrediHesabi hesap1;
```

Bu bildirim ile hesap1 değişkeni bildiriliyor. Fakat henüz bellekte üye elemanları tutmak için herhangi bir yer tahsis edilmesi. Yer tahsisatı yapabilmek için new kullanılır.

```
KrediHesabi a=new KrediHesabı();
```

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

Sınıf Nesneleri Tanımlama

```
class KrediHesabi
    {
    int HesapNo;
    private double Limit;
    public string KartSahibi;
    }
KrediHesabi hesap=new KrediHesabı();
```

hesap.HesapNo=12; //yanlış, erişim private

hesap.Limit=200000 //yanlış, erişim private

hesap.KartSahibi="deneme" //doğru, public

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

Birden fazla sınıf nesnesi tanımlama

Aynı sınıf türünden birden fazla nesneyi tanımlayabiliriz. Ancak tanımladığımız nesneleri birbirlerine atarken önemli birtakım noktalar vardır.

Referans ve değer türlerinin aktarımı. Değer tipleri birbirine atanırken bitsel olarak kopyalanır. Referans türleri böyle değildir.

İki nesneyi birbirine atadığınızda nesnelerin elemanları tek tek kopyalanmaz. Atanan değerler dinamik bellek bölgesindeki referans olan adreslerdir. Sonuçta atama yapılan nesneler üzerindeki değişliklerden etkilenirler.

```
class KrediHesabı
  {
     public int HesapNo;
     private double Limi;
     public string KartSahibi;
  }

class Program
  {
     static void Main(string[] args)
     {
        KrediHesabı hesap1=new KrediHesabı();
        KrediHesabı hesap2;
        hesap1.HesapNo = 1234567;
        Console.WriteLine(hesap1.HesapNo);
        hesap2 = hesap1;
        Console.WriteLine(hesap2.HesapNo);
        hesap2.HesapNo = 7654321;
        Console.WriteLine(hesap1.HesapNo);
        Console.ReadKey();
     }
  }
```

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

Sınıflara Metot Ekleme

Bir sınıf değişken ve metotlardan oluşmakta idi. Bir metodun bildirimi ve kullanımı ile ilgili bir örnek düşünürsek;

- Dörtgen isimli bir sınıf tanımı yapılsın
- Alan() ve Giris() isimli iki metot olsun
- DortgenYaz() isimli metot ile de bilgiler yazdırılsın

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

Sınıflara Metot Ekleme

Sınıfın tanımını yaparsak;

```
class Dortgen
{
 int En;
 int Boy;
 int Alani;
  public void Alan()
      {
       Alani=En*Boy;
      }
public void Giris(int en,int boy)
{
  En=en;
  Boy=boy;
}
public void DortgenYaz()
{
Console.WriteLine(“en”+En);
Console.WriteLine(“boy”+Boy);
Console.WriteLine(“Alan”+Alani);
}
static void Main()
{
```

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

Sınıflara Metot Ekleme

Sınıf tanımını yaptıktan sonra ana programın olduğu sınıfı tanımlarsak;(bu sınıfları ayrı kaynak kodu olarak derleyebiliriz)

```
class Ana
{
 static void Main()
{
 Dortgen d=new Dortgen();
 d.Giris(21,22); d.Alan();
d.DortgenYaz();
}
}
```

Sınıflar, Yapılar ve Numaralandırmalar(Enumaration)

this anahtar sözcüğü

this anahtar sözcüğü ilgili nesnenin referansını belirtir.

```
public void enboy(int En,int Boy)
{
 if (En<0 || Boy<0)
{
En=0; Boy=0;
}
else {
En=En;
Boy=Boy;
}
```

Else kısmında

```
this.En=En;
this.Boy=Boy;
```

Şeklinde düzeltilirse prob. halledilecektir.

Sonraki haftaya nesneler ve sınıflar (devam)