## Nesneye Dayalı Programlama (Object Oriented Programming)

Bu dersimizde Object Oriented Programlamaya başlıyoruz. İlk olarak sınıfları (class), nesneleri (object) ve bunlar hakkında temel bilgileri öğreneceğiz. Ardından diğer derslerimizde yapıcılar(constructor), erişim belirleyiciler(Access modifiers), özellikler (properties), method aşırı yüklemesi (method overloading) ve static metotlar konularına geçeceğiz.

#### Sınıf Kavramı

Bir veri tipini modellemek için kullandığımız yapılara sınıf denir. Bir sınıf, kapsüllenmiş verileri bu veriler üzerinde işlem yapacak metotları ve bu verilere ait özellikleri içerebilir. Örneğin .Net Framework Class Library içinde bulunan String sınıfını inceleyelim. Bu sınıf karakterlerden oluşan bir dizi barındırır, bu kısma data denir, barındırdığı data üzerinde işlemler yapabileceğimiz ToLowerCase(), Trim(),SubString() v.b gibi metotlar içerir. Ayrıca barındırdığı dataya ait özellikler içerir. Örn stringin uzunluğunu bulan Lenght özelliği gibi.

C# da bir sınıfın tanımlanması basitçe aşağıdaki gibidir.

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4  5 | class YeniClass  {      //sınıf ıcınde verileri tutacak alanlar,      //metotlar ve özellikler bu kısma yazılıyor  } |

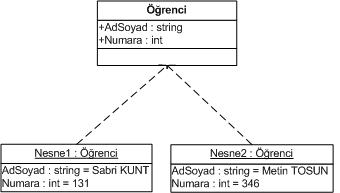
### Nesneler(Object)

Sınıflar bir veri modelidir, nesneler ise bu veri modeline bağlı kalarak verileri tutan ve model içerisinde tanımlanmış metotları ve özellikleri kullanmamızı sağlayan modele ait bir referanstır.

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1 | YeniClass yeniNesne = new YeniClass() |

Yukarıdaki kod ile YeniClass sınıfı türünde bir yeniNesne ismiyle bir nesne oluşturmuş olduk. Bu nesne ile sınıfın içerdiği tüm modeli kullanabiliriz.



#### Fields

Fieldler sınıf içerisinde veri barındıran alanlardır. ,Fieldler referans veya değer tiplerinden herhangi birisi olabilir, Örn başka bir sınıfa referans gösteren nesnelerde bir sınıf içerisinde field olarak kullanılabilir.  
  
Bir öğrencinin bilgilerini ve notlarını tutan basit bir sınıf yazalım.

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | class Ogrenci  {      //sınıfa ait fieldler      string adSoyad;      int yas;      int numara;      int matematikNotu = 0; //ilk deper verilmiş..      int ingilizceNotu;      int dersOrtalama;  } |

Yukarıdaki kodda matematikkodu isimli değişkende olduğu gibi sınıflara ait fieldlere tanımlama esnasında değer ataması yapabilirsiniz. Eğer değer ataması yapmazsanız değişkenler default değerlerine set edilir. Değişken tiplerine göre default değerleri aşağıdaki tablodaki gibidir.

|  |  |
| --- | --- |
| **Veri Tipi** | **Default Değeri** |
| int | 0 |
| long | 0 |
| float | 0.0 |
| double | 0.0 |
| bool | FALSE |
| char | \0' (null karakteri |
| string | "" (boş metin) |
| Object | null |

#### Metotlar

Metotlar, verileri kullanarak belirli işlemler yapan yapılardır. Metotlar parametre olarak bazı değerler alabildiği gibi gerçekleştirdiği işlem sonucunda bazı değerleri de geri döndürürler. Bir metodun yapısı aşağıdaki gibidir.

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4 | <dönüş değiş. tipi><metodun ismi>(<parametre tipi> <parametre ismi>,..)  {      //Metodun gerçekleştireceği işlemler için kod kısmı  } |

Örnek bir metot;

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int ToplamBul(int sayi1, int sayi2)  {      int toplam = sayi1 + sayi2;      return toplam;  }    int MatematikNotunuDondur()  {      return matematikNotu;  } |

Yukarıdaki örnekte iki sayıyı parametre olarak alan ve bunları topladıktan sonra toplamlarını döndüren basit bir metot tanımladık. Metotlar dönüş işlemlerini return kelimesi ile yaparlar. Eğer tanımladığımız metot bir dönüş değerine sahip değilse dönüş tipi olarak void yazmalıyız. Dönüş tipi olarak belirlediğimiz değişken tipi ile return ile döndürdüğümüz değerin değişken tipi aynı olmak zorundadır.  Dönüş değeri olmayan ve herhangi bir parametre almayan basit bir örnek;

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4  5 | void TarihiYaz()  {      Console.WriteLine("Şuanki Tarih ve Saat : "  + DateTime.Now.ToString("dd.MM.yyyy HH:mm"));  } |

#### Nesne Oluşturulması ve Nesneye Değer Ataması

Bir sınıfı referans gösterecek bir nesne oluşturmak için new anahtar kelimesi kullanılır.

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4 | Ogrenci ogrenci = new Ogrenci();  //İlk olarak nesneyi tanımlayıp daha sonra oluşturabilirsiniz..  Ogrenci ogrenci2;  ogrenci2 = new Ogrenci(); |

Yukarıdaki örnekte 2. Yöntem ile tanımlama yaparken Ogrenci ogrenci2; kod parçacığı Ogrenci tipinde bir referans oluşturur, fakat bu referans herhangi bir nesne içermez. Bu yüzden bu referansın değeri nulldur. ogrenci2 = new Ogrenci();dediğimizde ise heap de bir nesne oluşturulur ve referans değişkenimiz bu alana işaret eder.

#### Bir Sınıfın Üyelerine Erişmek

Bir sınıfın üyelerine (field, metot veya properties gibi) erişmek için sınıfa ait nesneden sonra nokta operatörünü kullanırız.

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Ogrenci ogrenci = new Ogrenci();  ogrenci.adSoyad = "Mehmet Sabri KUNT";  ogrenci.yas = 27;  ogrenci.matematikNotu = 89;  ogrenci.ToplamBul(13, 43);  int matnot = ogrenci.MatematikNotunuDondur(); |

#### Erişim Belirleyiciler ve Erişilebilirlilik Seviyeleri

Bir sınıfa ait üyeler için erişilebilirlilik konusu önemlidir. Peki, Erişilebilirlilik ne demektir? Erişim belirleyiciler kullanarak bir üyenin güvenliğini sağlamış oluruz. Eğer bir üyeye sınıfımız içindeki metotlar dışında herhangi bir erişim yapılmasını istemiyorsak veya projemiz dışında herhangi bir yerden bu üyeye erişim izni vermek istemiyorsak erişim belirleyicileri kullanmak zorundayız.  
  
Peki, erişimi engellemenin mantığı ne? Bir web servis yazdık ve bu web servisimiz içinde tanımladığımız bir sınıf var. Bu sınıfa ait bir üyenin sadece web servis içinde tanımlanmış ve ya web servis projesi içinde tanımlanmış üyeler tarafından erişilmesini istiyoruz. Web servisi kullanan insanların bu üyeye ait değerleri set etmesi veya get etmesi işimize gelmiyorsa, bu üyenin erişilebilirliliğini kısıtlamamız gerekir. C# da kullanabileceğimiz toplam 5 farklı erişim seviyesi bulunuyor. Bunları liste halinde aşağıda verdim.

|  |  |
| --- | --- |
| **Erişim Belirleyici** | **Açıklama** |
| private | Erişim seviyesi private olarak ayarlanan üyelere sadece bu üyenin bulunduğu sınıf içerisinden erişilebilir. |
| internal | Sadece üyenin bulunduğu proje içerisinden erişilebilir. |
| protected | Sadece üyenin bulunduğu sınıf ve bu sınıftan türemiş alt sınıflardan erişilebilir. |
| protected internal | Sadece üyenin bulunduğu proje içerisinden ayrıca üyenin sınıfından türemiş alt sınıflar tarafından erişilebilir. |
| public | public üyelerin erişiminde herhangi bir kısıtlama yapılmaz. İstenilen her yerden üyeye erişim açıktır. |

Object Oriented programlamada tüm sınıf fieldlerinin private yapılması ve bu fieldlere erişimin metotlar ve/veya Özellikler(properties) üzerinden sağlanması tavsiye edilir. Bu sayede fieldlerimize dışarıdan verilecek değerler kontrol edilebilir. Aksi takdirde dışarıdan fielde verilecek değerleri kontrol etmemiz imkânsızdır.  
  
Üye tanımlarken eğer bir erişim belirleyici kullanmazsak C# default olarak üyemizin erişim seviyesini private yapacaktır.  
  
Namespaceler hariç tüm veri tipleri için erişim belirleyici kullanabilirsiniz. Fakat üst seviye tipler için (sınıf, struct, enum) için sadece public ve internal erişim seviyelerini kullanabilirsiniz.

#### Özellikler(Properties)

Eğer sınıftaki tüm fieldlerin erişim seviyelerini private yaparsak o zaman bu fieldlere nasıl erişebiliriz diye düşünmüşsünüzdür. Cevap tabi ki, properties kullanarak. C# properties özelliğini destekleyen ilk programlama dilidir. Java ve C++ gibi dillerde private bir fielde değer atamam veya bu fieldin değerini almak için public tanımlanmış metotlar kullanılır. Private bir field tanımlayalım ve bu fielde erişimi metotlar ve properties ler üzerinden nasıl yapacağımıza bakalım.

#### Metot kullanarak:

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | private int dersNotu;  public void SetDersNotu(int not)  {      dersNotu = not;  }    public int GetDersNotu()  {      return dersNotu;  } |

#### Properties kullanarak

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | private int dersNotu;  public int DersNotu  {      get { return dersNotu; }      set { dersNotu = value; }  } |

Yukarıdaki kodları incelerseniz properties kullanarak private olan bir fielde erişmenin ne kadar kolay olduğunu görürsünüz. Ayrıca Visual Studio bir field için properties tanımlamayı otomatikleştiriyor. Bunun için kodunuzdaki fieldi sağ tıklayın açılan menüden **Refactor->Encapsulate** **Field**menüsünü seçin, yukarıdaki get ve set metotları sizin için otomatik olarak oluşturulacaktır. Properties içinde set veya get metotlarında en az birini tanımlamanız gerekiyor. Eğer fieldinizin sadece readonly olmasını istiyorsanız set özelliğini silerek bunu yapabilirsiniz.  
  
Eğer tanımladığınız field alanının sadece belirli bir aralıkta değer alması gerekiyorsa, ‘örneğin ders notu 0 ile 100 arası olmaz zorunda’ set metodu içine bu kontrolü aşağıdaki gibi ekleyebilirsiniz. Böylece dersNotu isimli fieldinizin alabileceği değerleri kontrol etmiş olursunuz. Bu kontrolü yapabilmek için fieldinize ait erişim seviyesini private yapmalı ve aşağıdaki gibi bir properties kullanmanız gerekiyor. Sanırım bu kontrol ile bir fielde ait erişim seviyesinin neden private yapılması gerektiğini anlamışsınızdır.

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | private int dersNotu;  public int DersNotu  {      get { return dersNotu; }      set      {          if (dersNotu >= 0 && dersNotu <= 100)              dersNotu = value;      }  } |

Basit bir örnekle yukarıda anlattıklarımız pekiştirelim.

|  |
| --- |
| using System;  namespace OOPOrnek  {     class Program      {          static void Main(string[] args)          {              Dortgen dortgen = new Dortgen();              dortgen.KısaKenarUzunlugu = 200;              dortgen.UzunKenarUzunlugu = 200;              Console.WriteLine("Kenar Uzunlukları = {0}x{1}",                dortgen.KısaKenarUzunlugu, dortgen.UzunKenarUzunlugu);              Console.WriteLine("Dörtgen Tipi : {0}", dortgen.DortgenTipi);          }      }      class Dortgen      {          private int kısaKenarUzunlugu;          private int uzunKenarUzunlugu;          public int KısaKenarUzunlugu          {              get { return kısaKenarUzunlugu; }              set { kısaKenarUzunlugu = value; }          }            public int UzunKenarUzunlugu          {              get { return uzunKenarUzunlugu; }              set { uzunKenarUzunlugu = value; }          }            public string DortgenTipi          {              get              {                  string tip = "";                  if (kısaKenarUzunlugu == uzunKenarUzunlugu)                      tip = "Kare";                  else                      tip = "Dikdörtgen";                  return tip;             }          }      }  } |

Yukarıdaki kodu inceleyecek olursak, basit bir sınıf tanımladık. Sınıfımız basit bir dörtgen yapısını barındırıyor. Tabi ki bu sınıfı daha detaylı hale getirebiliriz. İlerleyen derslerimizde bu sınıfa yeni ilaveler yapacağız.  Yeni bir console uygulaması açıp yukarıdaki kodu Program.cs dosyasının içine yapıştırırsanız sorunsuz çalışır. Çalıştırdığınız zaman aşağıdaki çıktıyı elde edeceksiniz.

Kenar Uzunlukları = 200x200

Dörtgen Tipi : Kare

Devam etmek için bir tuşa basın . . .

#### Sınıfların Static Üyeleri

Bir sınıda ait üyeye erişebilmek için o sınıfa referans eden bir nesne oluşturmanız gerekir. Bu kural static üyeler için geçerli değildir. Yani yukarıdaki örneğimizde Dörtgen sınıfında ürettiğimiz bir dortgen nesnesini kullanarak kisaKenarUzunluğu üyesine erişebiliriz. Eğer sınıfımıza ait bir üye varsa ve bu üye bu sınıftan türüyen tüm nesnelerde ayı değere sahip olacaksa static metod tanımlamak mantıklıdır. Yukarıdaki sınıfımıza basit bir static üye ekleyelim  ve main fonksiyonu içinden çağıralım.

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | class Dortgen      {          public static int kenarAcisi = 90;      …      …      }      static void Main(string[] args)  {       Console.WriteLine(Dortgen.kenarAcisi);  } |

Yukarıda gördüğünz gibi Dortgen sınıfının adını yazdıktan sonra direk olarak static üyemizin ismini yazarak kullandık.  Static üye tanımlarken dikkat etmeniz gereken nokta şudur. Eğer bir methodu static tanımladıysanız, bu metot içinde sınıfın üyelerinden static olmayan bir üye kullanamazsınız. Tersi durumda yani static olmayan bir üye içinde static bir üye kullanımında ise herhangi bir sorun yoktur.

[?](http://www.yazilimmutfagi.com/10184/dersler/cSharp/yazilim-mutfagi-dersleri-cSharp-ders-5-nesneye-dayali-programlamaya-giris.aspx)

|  |  |
| --- | --- |
| 1 | An object reference is required for the non-static field, method, or property |

Yukarıdaki gibi bir hata aldıysanız static bir metor içinde static olmayan bir metot kullanmaya çalışıyorsunuz demektir.