Nesne Yönelimli Programlama

7.Hafta

Konu: Kalıtım, Protected Erişim Belirteci, Örnek Proje (Devam)

Dr. Öğr. Üyesi Güncel SARIMAN
E-posta: guncelsariman@mu.edu.tr

Kapsülleme Örneği

Makina" isimli bir sınıf içerisinde "isi" isimli bir "properties" i bulunmakta. "isi" özelliğini kapsüllenecektir. Aynı zamanda ısı o'dan küçük ise veya 100' den büyükse hata fırlatarak ana programda bu hatanın yakalanması sağlanacaktır.

```
class Makina
   {
      int isi;
      public int Isi
      {
           get { return isi; }
           set?
       }
}
```

Kapsülleme Örneği

```
set
        {
          if (value < = 100 && value>=0)
          {
            isi = value;
          }
          else if (value < 0)
          {
            isi = 0;
            throw new Exception("Fazla soğuk");
          }
          else
          {
           // Console.WriteLine("Aşırı ısınma engellendi"); // Bu şeklide sınıfımızı Windows formda
            //çalıştırmak istediğimizde çalışmaz.Bunun yerine
            //aşağıdaki şekilde Hata fırlatabiliriz.

             isi = 100;
            throw new Exception("Aşırı derecede ısınma var");

          }
        }
```

Kalıtım

Inheritance (miras alma, kalıtım), bir nesnenin özelliklerinin farklı nesneler tarafından da kullanılabilmesine olanak sağlayan OOP özelliğidir.

Yazılan bir sınıf bir başka sınıf tarafından miras alınabilir. Bu işlem yapıldığı zaman temel alınan sınıfın tüm özellikleri yeni sınıfa aktarılır. Bir sınıf diğer bir sınıftan türediği zaman, türediği sınıfın bütün özelliklerini içerir. Bunun yanında kendine has özellikler de barındırabilir.

Örneğin

İnsan – memeli ilişkisinde, insanın memeli sınıfını miras aldığı söylenebilir. Bu sayede insan sınıfını yazarken memelilerin özelliklerini tekrar yazmamıza gerek kalmaz.

Veya

bir taşıt sınıfı varsa; otomobil, kamyon, motosiklet gibi alt sınıfları üretmek çok daha az çaba gerektirir.

Kalıtım

```
public class Canli
{
   public int ayak;     // Canlının kaç ayağı olduğunu tutan field
    public int el;       // Canlının kaç eli olduğunu tutan field
    public string tur;    // Ne tür bir canlı olduğunu belirleyeceğimiz field.
}
```

İki adet daha sınıf oluşturarak Canli sınıfından türetilsin:

Kalıtım

```
class Insan : Canli
{
   public string Konus()
   {
      string değer=string.format("Selam benim {0} adet elim ve {1} adet ayağım vardır.",el,ayak);
      return değer;
   }
}
```

```
class Kopek : Canli
{
   public return Havla()
   {
      return "Hav";
   }
}
```

```
Insan ins=new Insan();
Kopek kop=new Kopek();

ins.el=2;
ins.ayak=2;
kop.ayak=4;
ins.Konus();
kop.Havla();
```

Kalıtım

```
class Ev
{
    public string sahip;
    public Ev()
    {
        Console.WriteLine("Ev Kurucu.");
    }

    public void EvSahibiniYaz(string sahip)
    {
        Console.WriteLine("Ev 'in sahibi : {0}", sahip);

    }
}
```

```
class ApartmanKatı : Ev
{
    string apartmanYöneticisi;
    ApartmanKatı()
    {
        Console.WriteLine("ApartmanKatı Kurucu.");
    }
    void AptYöneticiniYaz(string s)
    {
        Console.WriteLine("Apartman Yöneticis : {0}", s);
    }
```

```
class Uygulama
{
    public static void Main()
    {
        ApartmanKatı apt = new ApartmanKatı();

        Console.WriteLine("Evin sahibi kim?  ");
        apt.sahip = Console.ReadLine();
        apt.EvSahibiniYaz(apt.sahip);

        Console.WriteLine("Apartman Yöneticisi kim?  ");
        apt.apartmanYöneticisi = Console.ReadLine();
        apt.AptYöneticiniYaz(apt.apartmanYöneticisi);
    }
}
```

Kalıtım

Ev Sınıfı bütün evleri temsil ederken, ApartmanKatı ise Ev Sınıfının bazı niteliklerini taşır, ama kendine has niteliklere de sahiptir. Örneğin ev sınıfında salon, mutfak, oda sayısı, ısıtma sistemi, emlak vergisi ve fiyatı gibi değişken ve metotlar varsa, onlar apartman katı için de geçerlidir. Evin kaçıncı katta olduğu, apartman yöneticisi, kapıcı her evde olması gerekmeyen nitelikler olabilir.

Kalıtım

Örneğin : Bir pastane için yazılımı yapılmaktadır. Bir pastanın türü farklılık gösterebilir. Fakat tüm pastaların ortak kullandığı malzemeler veya özellikler olabilir.

Örneğin : Muzlu Pasta ile Kakaolu Pasta arasında bulunan ortak özellikler var. Fiyat, imal tarihi, son kullanma tarihi, Malzemeler vb. Bu saydığım özellikler tüm pastalarda bulunur ! Her pastanın bir fiyatı vardır.

Üretim tarihi vs her pastada mevcut olmalıdır. Bu durumda ben her pasta için ortak özellikleri yeniden yazmak yerine bir kez yazmayı tercih etmeliyim.

Yine nesne yönelimli programlamanın en önemli özelliklerinden birisi olan yeniden kullanılabilirlik (Reusability Bir işi bir kez yap, bir çok kez kullan) içinde son derece önemlidir.

Kalıtım

Genel olarak sahip olduğum sorunu parçalara bölmem gerekiyor. Bir pasta ihtiyacım var. Bu pasta herhangi bir nesneye kalıtıldığında ek malzemeleri olmasa bile öz niteliğinde pasta olarak bilinmeli. Böylece ilk olarak Pasta adında bir class ihtiyacımız vardır. Pasta nesnesi için örnek aşağıda belirtilmiştir.

```
public class Cake
{
    public decimal Price { get; set; }
    public string Name { get; set; }
    public DateTime ProductionDate { get; set; }
    public DateTime ExpirationDate { get; set; }
    public string[] Supplies { get; set; }
}
```

public class BananaCake : Cake { }

Kalıtım

Şimdi pastaya bir nitelik katmamız gerekiyor. Yani ben muzlu bir pasta istediğim zaman farklı bir sınıf kullanmak kaydıyla pastamı yapabilirim.

```
public class BananaCake : Cake

{

 public string BananaQuantity { get; set; }

}
```

Kalıtımla birlikte Cake sınıfı içerisinde bulunan tüm üye elemanlar BananaCake sınıfı içerisine geçti.

BananaCake içerisine Muz Miktarı adında bir özellik ekledim.

public class BananaCake : Cake { }

Erişim Belirteçleri

Türetilmiş bir sınıfa, temel sınıftaki elemanlardan sadece public ve protected olarak tanımlananlar aktarılır. Erişim belirteci private olan elemanlar kalıtım ile aktarılmaz.

Erişim Belirleyicisi: Metot ve özelliklere olan erişimin sınırlarını belirtmektedir.

Private: Sadece tanımlandığı sınıf içerisinden erişilebilir. (Kalıtım ile aktarılmaz.)

Public: Her yerden erişilebilir. (Kalıtım ile aktarılır.)

Internal: Sadece bulunduğu projede erişilebilir. (Kalıtım ile aktarılır.)

Protected: Tanımlandığı sınıfta ve o sınıfı miras (kalıtım) alan sınıflardan erişilebilir. (Kalıtım ile aktarılır.)

public class BananaCake : Cake { }

Erişim Belirteçleri

```
public class TemelSinif
{
   protected string degisken_1 = "Temel Sınıf İçerisinde";
   public string degisken_2 = "Temel Sınıf İçerisinde";
   private string degisken_3 = "Temel Sınıf İçerisinde";
}

 public class TuretilmisSinif : TemelSinif
{
   public void Metot_1()
   {
      degisken_1 = "Türetilmiş Sınıf İçerisinde";
   }

   public void Metot_2()
   {
      degisken_2 = "Türetilmiş Sınıf İçerisinde";
   }
}
```

public class BananaCake : Cake { }

Erişim Belirteçleri

public ile protected belirteçleri arasındaki fark nedir?

TemelSinif'dan *TuretilmisSinif*'a kalıtım yoluyla aktardığımız "*degisken_1 (protected)*" ve "*degisken_2 (public)*" elemanlarını *TuretilmisSinif* içerisinde kullanabiliriz. Ancak *TuretilmisSinif*'dan oluşturduğumuz bir nesne üzerinden sadece "*degisken_2*" elemanına erişebiliriz. Erişim belirteci *protected* olan *degisken_1* elemanına başka bir sınıftan erişemeyiz. Bunun nedeni; Erişim belirteci protected olan elemanlar tanımlandıkları sınıfta ve kalıtım yolu ile aktarıldıkları sınıflar da private elemanların özelliklerini gösterirler.

public class BananaCake : Cake { }

Örnek Çalışma: Laboratuvar Envanter Takip Sistemi

İş Tanımı

1-Üniversite bünyesindeki tüm laboratuvarlardaki malzemelerin sisteme kaydedilmesi, güncellenmesi, silinmesi ve okul bazında rapor alınabilmesini sağlayan bir sistem geliştirilecektir.
2-Sistem MSKÜ'deki fakülte, yüksekokul bölümlerinin lablarını sisteme kaydetmektedir.
3-Kaydedilen labların içlerine türlerine göre malzemeler kaydedilmektedir.
4-Lab türleri bölümlere göre değişmektedir. Bilgisayar, Kimya, Biyoloji, Elektrik vb.
5-Her bölümün malzemeleri farkı olsa da girilmesi gereken bilgiler ortaktır.
6-Malzeme Adı, Markası, Ne zaman Alındığı, Kaç adet olduğu, Tipi(Demirbaş veya Sarf Malzemesi vb.)
7-Sistem de okul bölüm ve sorumlu kullanıcı sisteme tanıtılabilecektir.
8-Kullanıcılar sistemi kullanırken malzemeler hakkında tüm bilgilere erişebilmektedir.

public class BananaCake : Cake { }

Örnek Çalışma: Laboratuvar Envanter Takip Sistemi

1-Okul

2-Bölüm

3-Program

4-Laboratuvarlar

5-Lab Türleri

6-Malzemeler

7-Kullanıcılar

public class BananaCake : Cake { }

Örnek Çalışma: Laboratuvar Envanter Takip Sistemi

Okul
*Id OkulKodu Adi Adresi Telefon
OkulBilgisiGetir() OkulEkle()

Bölüm:Okul
*Id BölümKodu
BölümBilgisiGetir() BölümEkle()

Program:Okul
*Id ProgramKodu
ProgramBilgisiGetir() ProgramEkle()