Görsel Programlama

Ders 4

Dr. Öğr. Üyesi Mehmet Dinçer Erbaş
Bolu Abant İzzet Baysal Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Kalıtım

- Programlama dillerinde kalıtım sınıflandırma ile ilgili bir kavramdır.
  - Sınıflar arasındaki ilişkiyi tanımlar.
- Gerçek hayattan bir örnek: Hayvanlar sınıfı
  - Hayvanların ortak özellikleri vardır. Örneğin hepsi nefes alır, sıcakkanlıdır, yavrularını beslerler.
  - Bununla birlikte her hayvan türünün farklı özellikleri vardır. Örneğin atların ayakları vardır, balinaların ise yüzgeçleri.

Kalıtım

- Hayvanlar sınıfı
  - Bir program içerisinde hayvanları modellemek istediğimizi düşünelim.
    - Bunun için her bir hayvan türü için ayrı sınıflar yaratabiliriz.
      - Örneğin at sınıfı, balina sınıfı.
    - Bu sınıfların her birinde o hayvan türüne özel fonksiyonlar bulunur.
      - Örneğin at için yürümek, balina için yüzmek gibi.
    - Bununla birlikte birçok özellik ve fonksiyon her oluşturulan sınıfta ortak olacaktır.
      - Örneğin nefes almak, yavrasunu beslemek gibi.
    - Her sınıfa tekrar tekrar ortak özellikleri yazabiliriz.
      - Ancak bu yöntem hem oldukça zahmetli olacak, hem de programın bakımı zorlaşacaktır.

Kalıtım

- Hayvanlar sınıfı
  - C# gibi nesne tabanlı programlama dilleri oluşturduğumuz sınıfları organize edebilmemiz için kalıtım yöntemi içerir.
  - Hayvanların ayrı ayrı sınıflarını oluştururken bu sınıfların ortak özelliklerini içeren hayanlar isimli bir üst sınıf oluştururuz.
  - Bunun sonrasında oluşturulacak at, balina, insan, karıncayiyen gibi sınıflar, üst sınıftan türer ve üst sınıfın her türlü özelliğini alır.
    - Sınıfa özel özellik ve fonksiyonlar ise ilgili alt sınıflara eklenebilir.
      - At sınıfına yürümek, balina sınıfına ise yüzmek eklenebilir.
    - NefesAlmak fonksiyonu ortak olacaktır.
      - Bu fonksiyonda bir değişiklik yapacaksanız sadece hayvanlar sınıfındaki fonksiyonu değiştirmeniz yeterli olacaktır.

Kalıtım

- Bir sınıfın diğer bir sınıftan türediğini gösteren söz dizimi şu şekildedir.

```
class DerivedClass : BaseClass {
    ...
}
```

- Bu şekilde belirtildiğinde türetilmiş sınıf (DerivedClass) temel sınıftan (BaseClass) türemiş olur.
- Bir sınıf birden fazla sınıftan türeyemez.
- DerivedClass sealed olarak belirtilmediği sürece, türetilmiş sınıftan başka sınıflar türetilebilir.

```
class DerivedSubClass : DerivedClass {
    ...
}
```

Kalıtım

- Daha önce belirttiğimiz hayvanlar sınıfını şu şekilde tanımlarız.

```
class Mammal
{
    public void Breathe()
    {
        ...
    }

    public void SuckleYoung()
    {
        ...
    }
    ...
}
```

Kalıtım

- Daha önce belirttiğimiz hayvanlar sınıfını şu şekilde tanımlayabiliriz.

```
class Horse : Mammal
{
    ...
    public void Trot()
    {
        ...
    }

}
```

```
class Whale : Mammal
{
    ...
    public void Swim()
    {
        ...
    }

}
```

Kalıtım

- Bir üretilmiş sınıf otomatik olarak temel sınıfın tüm alanlarını içerir.
- Nesne yaratıldığında tüm alanların başlatılması gerekir.
  - Bunu her sınıfın sahip olduğu kurucular ile yaparız.
- Türetilmiş sınıflar genellikle temel sınıfın sahip olduğu kurucuyu çağırırlar.
  - Türetilmiş bir sınıftan temel sınıf kurucusunu çağırmak için base anahtar sözcüğünü kullanırız.
- Türetilmiş sınıf kurucusu içersinde temel sınıf açık şekilde çağırılmıyorsa, derleyici bunu otomatik olarak yapar.
  - Ancak bu durumda temel sınıfta public bir kurucunun olmasına dikkat edilmelidir.

Kalıtım

```
class Mammal // temel sınıf
{
    public Mammal(string name)  // temel sınıf için kurucu
    {
        ...
    }
    ...
}

class Horse : Mammal // türetilmiş sınıf
{
    public Horse(string name)
           : base(name) // Mammal(name) yöntemini çağırır
    {
        ...
    }
    ...
}
```

Kalıtım

- Türetilmiş sınıflar arası ilişkiler konusunda örnekler.

```
class Mammal
{
    ...
}
class Horse : Mammal
{
    ...
}

class Whale : Mammal
{
    ...
}
...
Horse myHorse = new Horse("Neddy");  // önceki gösterilen kurucu bir ad bekler!
Whale myWhale = myHorse;             // hata – farklı türler


Horse myHorse = new Horse("Neddy");
Mammal myMammal = myHorse; // doğru, Mammal Horse'un temel sınıfıdır
```

Kalıtım

- Türetilmiş sınıflar arasındaki ilişkiler konusunda örnekler

```
Horse myHorse = new Horse("Neddy");
Mammal myMammal = myHorse;
myMammal.Breathe();          // Tamam - Breathe Mammal sınıfının parçasıdır
myMammal.Trot();             // hata - Trot Mammal sınıfının parçası değildir
```

```
Mammal myMammal = myMammal("Mammalia");
Horse myHorse = myMammal;    // hata
```

Kalıtım

- Oluşturduğunuz programlarda anlamlı tanımlayıcılar bulmanız gerekir.
- Temel bir sınıf ile üretilmiş bir sınıf aynı imzaya sahip iki farklı yöntem bildirirse, derleme hatası oluşturur.
  - Bir yöntemin imzası, o yöntemin adı ve parametrelerinin türü ile sayısıdır.
  - Farklı dönüş tiplerine sahip iki fonksiyon aynı imzaya sahip olabilir.
- Türetilmiş bir sınıfta, temel sınıfta bulunan yöntem ile aynı imzaya sahip bir yöntem tanımlarsanız, temel sınıftaki yöntem gölgelenir.
  - Bu durumda derleyici uyarı verir.

Kalıtım

```
class Mammal
{
    ...
    public void Talk() // tüm memeliler konuşur
    {
        ...
    }
}

class Horse : Mammal
{
    ...
    public void Talk()  // atlar diğer memelilerden farklı yolla konuşur!
    {
        ...
    }
}
```

Kalıtım

- Bazı durumlarda bir yöntemin temel sınıfta bulunan versiyonunun gölgelenmesi istenebilir.
  - Örnek: Object sınıfındaki ToString metodu
  - Verilen objenin metin olarak gösterilmesini sağlar
  - Ancak bizim hayvanlar sınıfımızı düşünürsek bu pek anlamlı olmaz.
  - Bu sebeple bu fonksiyon aşağıda belirtilen şekilde tanımlanmıştır.

```
namespace System
{
    class Object
    {
       public virtual string ToString()
        {
            ...


        }
        ...
    }
    ...
}
```

Kalıtım

- Temel sınıfta virtual olarak tanımlanan fonksiyonlar override anahtar sözcüğü ile gölgelenebilir.

```
class Horse : Mammal
{
    ...
    public override string ToString()
    {
        ...
    }
}
```

- Türetilmiş sınıflar, base anahtar sözcüğü ile temel sınıftaki orjinal uygulamayı çağırabilir.

```
public override string ToString()
{
    base.ToString();
    ...
}
```

Kalıtım

- virtual ve override kelimeleri ile özel bir yöntem bildiremezsiniz.
- İki yöntemin imzaları aynı olmalıdır.
- Iki yöntem aynı erişim hakkına sahip olmalıdır.
- Yalnızca sanal yöntemleri geçersiz hale getirebilirsiniz.

Kalıtım

```
class Mammal
{
    ...
    public virtual string GetTypeName()
    {
        return "This is a mammal";
    }
}

class Horse : Mammal
{
    ...
    public override string GetTypeName()
    {
        return "This is a horse";
    }
}

class Whale : Mammal
{
    ...
    public override string GetTypeName ()
    {
        return "This is a whale";
    }
}

class Aardvark : Mammal
{
    ...
}
```

```
Mammal myMammal;
Horse myHorse = new Horse(...);
Whale myWhale = new Whale(...);
Aardvark myAardvark = new Aardvark(...);

myMammal = myHorse;
Console.WriteLine(myMammal.GetTypeName()); // Horse
myMammal = myWhale;
Console.WriteLine(myMammal.GetTypeName()); // Whale
myMammal = myAardvark;
Console.WriteLine(myMammal.GetTypeName()); // Aardvark
```

Kalıtım

- protected anahtar kelimesi
  - public ve private alan ve yöntemler için kural bellidir.
    - public alan ve yöntemlere herkesin erişimine açıktır.
    - private alan ve yöntemler sadece sınıfın kendisine açıktır.
  - Birbirinden türeyen sınıflar arası ilişkileri kontrol etmek için protected anahtar kelimesi kullanılabilir.
    - A sınıfı B sınıfından türemişse, B sınıfının protected (korumalı) alan ve yöntemlerine erişebilir.
    - A sınıfı B sınıfından türememişse, B sınıfının korulmalı alan ve yöntemlerine erişemez.