

C#- Arv och polymorfi

Arv och polymorfi

Dessa två begrepp är väldigt viktiga inom objektorienterad programmering och är väldigt viktiga att lära sig. Arv (inheritance på engelska) är ett sätt för en klass att ärva från en annan och polymorfi (betyder många former på latin!) handlar om hur vi kan ge ärvda metoder (som exempel) ny mening eller form. Vi kommer till en vettigare förklaring.

Arv

Arv använder vi oss av när vi som sagt behöver ärva egenskaper från en annan klass. Vad menas då med att ärva egenskaper och varför vill vi skapa arv? Som exempel skapar vi en klass vi döper till Animal. I denna klass specificerar vi egenskaper som är generella för djur. För enkelhetens skull ger vi klassen en sträng för namn och en metod vi kallar "Talk()" som skriver ut en textrad till konsolen. Nu ser klassen Animal ut så här:

```
class Animal
{
    // Protected betyder att bara parent-klassen och
    // dess child-klasser kan använda sig av variabeln.
    protected string name;

    // Konstruktor för klassen
    public Animal (string name)
    {
        this.name = name;
    }

    // Metod som skriver ut en rad text
    public void Talk ()
    {
        Console.WriteLine("Hi from animal class!");
    }
}
```

Vi kan nu skapa ett objekt av denna klass och definiera ett generellt djur. Men vi vill vara mer specifika, vi vill skapa en hund. Då en hund är ett djur så är det logiskt att vi vill ärva egenskaper från klassen animal då den redan har egenskaper som en hund har (i detta fall har vi bara egenskapen namn och en metod för att prata, använd er fantasi så ser klassen mer fantastisk ut).

Ärva från klassen Animal

Nu ska vi skapa klassen Dog som vi vill ska ärva egenskaperna från klassen Animal. Man ärver genom att i klassfilen lägger till "": arvclass" efter klassnamnet där arvclass är den klass vi vill ärva ifrån.

```
class Dog : Animal
{
    // Lagra en bool!
    private bool isCute;

    // Konstruktör för Dog, tar in namn och ett bool-värde.
    // ": base (name)" betyder att vi skickar inkommande
    // namn variabel till basklassen (Animal).
    public Dog (string name, bool isCute) : base (name)
    {
        // Tilldela klassen det inkommande bool-värdet
        this.isCute = isCute;
    }

    // Skriver ut hundinfo
    public void printDogInfo ()
    {
        Console.WriteLine("The name of the dog is {0} and it is {1} that {2} is cute", name, isCute, name);
    }
}
```

Längst upp ser vi "Dog : Animal" som indikerar på att klassen Dog ärver av klassen Animal. I klassen skapar vi en klassspecifik variabel för att hålla ett boolvärde. Därefter skapade vi vår konstruktör som tar in två värden, en sträng för namn och ett boolvärde. Efter deklarationen av vår konstruktör ser ni "": base (name)" som betyder att vi skickar variabeln name till vår basklass (Animal). I konstruktören tilldelar vi det inkommande boolvärdet till isCute.

Eftersom vi ärver från klassen animal betyder det att vi har tillgång till alla metoder och variabler som finns i Animal. Det ser ni i metoden "printDogInfo()" som skriver ut en sträng där vi inkluderar name-variabeln som finns i basklassen Animal samt isCute som är specifik för Dog-klassen.

Nu går vi in i program och skapar ett objekt av klassen Dog och anropar funktionerna.

```
class Program
{
    static void Main(string[] args)
    {
        // Skapa hundobjekt med namn Lassie och ett bool-värde
        Dog myDog = new Dog("Lassie", true);

        // Anropa printDogInfo från klassen Dog
        myDog.printDogInfo();
        // Anropa Talk-metoden som klassen Dog ärvt
        // från Animal klassen
        myDog.Talk();

        Console.ReadLine();
    }
}
```

Som ni ser kan vi anropa två funktioner, printDogInfo() och Talk(). printDogInfo() finns i Dog-klassen och Talk() finns i Animal klassen som blir tillgänglig för Dog på grund av arv.

Polymorfi

Då vi ärver från en klass får vi som tidigare nämnt alla egenskaper som basklassen har, det vill säga de variabler och metoder klassen har. När vi använder metoden `Talk()` som finns i basklassen skrivs "Hi from animal class!" ut vilket är lite dumt eftersom det inte är något en hund säger. För att göra metoden `Talk()` mer hundspecifik måste vi använda oss av polymorfi. Med hjälp av polymorfi ("många former" på latin!) kan vi ändra på metoden `Talk()` i hundklassen utan att metoden `Talk()` i `animal` ändras, detta gör vi med hjälp av nyckelorden "virtual" och "override".

För att vi ska tillåtas ändra på metoden i en subklass måste vi lägga till nyckelordet `virtual` i basklassens metod. Så istället för "public void .." skriver vi "public virtual void ..".

```
class Animal
{
    // Protected betyder att bara parent-klassen och
    // dess child-klasser kan använda sig av variabeln.
    protected string name;

    // Konstruktor för klassen
    public Animal (string name)
    {
        this.name = name;
    }

    // Metod som skriver ut en rad text
    // Nyckelordet virtual tillåter oss att i en
    // subklass överskrida metoden med override
    public virtual void Talk ()
    {
        Console.WriteLine("Hi from animal class!");
    }
}
```

Nu kan vi fritt ändra på metoden Talk() från basklassen i alla subklasser som följer. Detta gör vi genom att lägga till nyckelordet "override".

```
class Dog : Animal
{
    // Lagra en bool!
    private bool isCute;

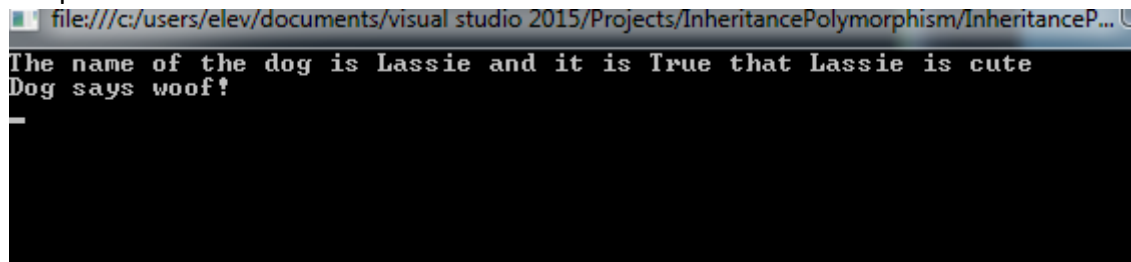
    // Konstruktor för Dog, tar in namn och ett bool-värde.
    // ": base (name)" betyder att vi skickar inkommande
    // namn variabel till basklassen(Animal).
    public Dog (string name, bool isCute) : base (name)
    {
        // Tilldela klassen det inkommande bool-värdet
        this.isCute = isCute;
    }

    // Överskrid basklassens Talk() metod
    public override void Talk()
    {
        Console.WriteLine("Dog says woof!");
    }

    // Skriver ut hundinfo
    public void printDogInfo ()
    {
        Console.WriteLine("The name of the dog is {0} and it is {1} that {2} is cute", name, isCute, name);
    }
}
```

Nu när vi anropar metoden Talk() via vårt Dog objekt så anropas override-metoden och skriver ut "Dog says woof!".

Output:



```
file:///c:/users/elev/documents/visual studio 2015/Projects/InheritancePolymorphism/InheritanceP...
The name of the dog is Lassie and it is True that Lassie is cute
Dog says woof!
```

När man pratar om arv så ger man klasserna som ärver från basklassen en "is a"-relation. Det vill säga Dog "is a" Animal.

Mer exempel på det är ifall vi skapar en basklass vi kallas för "Item" och sedan skapar vi en klass som heter "Potion" som ärver från "Items". Då blir relationen Potion "is a" Item.