Campus Nyköping

PROGRAMMERING I .NET C#1 - 12

Arv

Förra gången

- Mer om objektorienterad programmering
  - Klasser
  - Attribut
  - Properties
  - Metoder
  - Konstruktorn
- Uppgift Tjuv och Polis

Idag

- Demo
  - Klubb-Epidemin
  - Skogen
- Arv
  - Bas- och subclass
  - Virtual och Override

Objektorienterad programmering

- Tre pelare:
  - Encapsulation – Inkapsling
    - Inkapsling kan man se som ett skydd för en klass så att den inte kan användas på fel sätt. Det kan också handla om att inte visa/exponera för många saker, speciellt inte sånt som inte är relevant för den som ska använda klassen.
  - **Inheritance – Arv**
    - Ett arv innebär att vi kan skapa klasser som ärver attribut (variabler) och metoder från någon annan klass. Vi kan på så sätt utgå från en befintlig konstruktion och bygga vidare på den.
  - Polymorphism – Polymorfism
    - Ordet polymorfism används i olika sammanhang och beskriver situationer där något förekommer i flera olika former. I datavetenskapen beskriver den konceptet att objekt av olika typer kan nås via samma gränssnitt. Polymorfism innebär också att flera olika subklasser under en superklass kan hanteras som om de vore instanser av superklassen

Arv

- Syfet med arv är att minimera mängden kod, samt skapa struktur I vår kod.

Titta på de här tre klasserna:

```
class owl
{
    public int Weight { get; set; }
    public bool Nocturnal { get; set; }
    public int Wingspan { get; set; }
}
class dolphin
{
    public int Weight { get; set; }
    public bool Nocturnal { get; set; }
    public int FishPerDay { get; set; }
}
class horse
{
    public int Weight { get; set; }
    public bool Nocturnal { get; set; }
    public int HayPerDay { get; set; }
}
```

Alla tre djuren har properties för Weight och Nocturnal, samt varsin egen property.

Arv

- …då kan vi samla de gemensamma egenskaperna I en separat klass: Animal, och tala om att Owl, Dolphin och Horse ärver från Animal.

```
class Animal    // Basklass
    {
        public int Weight { get; set; }
        public bool Nocturnal { get; set; }
    }

    class Owl: Animal   // Subklass
    {
        public int Wingspan { get; set; }
    }

class Dolphin : Animal  // Subklass
{
    public int FishPerDay { get; set; }
}

class Horse : Animal   // Subklass
{
    public int HayPerDay { get; set; }
}
```

Metoder: virtual och override

- Subklasser kan innehålla properties, constructors och methods
- För att kunna använda en metod I en subclass ska den först definieras I basklassen, med nyckleordet **virtual**
- Därefter kan vi göra en override I subklassen.
- En virtuell metod är en metod som sen kan omdefinieras i ärvda subklassen. I C#har en virtuell metod en implementering i en basklass samt även i den ärvda subklassen. Den används när en metods grundläggande funktionalitet är densamma men ibland behövs mer funktionalitet i den härledda klassen.

- Basklassen:
  - `public virtual void myMethod()`
  - `{`
  - `..här händer något.`
  - `}`

- `Subklassen`
  - `public override void myMethod()`
  - `{`
    - `…här händer något annat…`
  - `}`
- Virtual Method in C# (c-sharpcorner.com)

Arv

```
class Animal       // Basklass
  {
        public int Weight { get; set; }
        public bool Nocturnal { get; set; }
  }
```

```
class Owl: Animal       // Subklass
{
    public int Wingspan { get; set; }
}

class Dolphin : Animal  // Subklass
{
    public int FishPerDay { get; set; }
}

class Horse : Animal   // Subklass
{
    public int HayPerDay { get; set; }
}
```

Properties I basklassen

- Properties I basklassen kan sättas på två sätt via subklassens constructor:

Basklassen:

```
class Animal
{
        public int Weight { get; set; }
        public bool Nocturnal { get; set; }
}
class Owl : Animal
{
        public int Wingspan { get; set; }
        public Owl(int weight, bool isNocturnal, int wingspan)
        {
            Weight = weight;
            Nocturnal = isNocturnal;
            Wingspan = wingspan;
        }
}
```

Properties I basklassen

- Eller Såhär:
  Basklassen

```
class Animal
{            protected int Weight { get; set; }
             protected bool Nocturnal { get; set; }
             public Animal(int weight, bool nocturnal)
             {
                          Weight = weight;
                          Nocturnal = nocturnal;
             }
}
class Owl : Animal
{
        public int Wingspan { get; set; }
        public Owl(int weight, bool isNocturnal, int wingspan) : base(weight, nocturnal)
        {
                          Wingspan = wingspan;
        }
}
```

Identifiera och definiera en subklass

- För att ta reda på om vad en instans är för subclass kan man skriva såhär:

```
if (animal is Horse)
{
   Console.WriteLine(“Hästar äter hö!");
}
```

- För att tala om vad en klass är för subclass, så castar man den till rätt subclass.

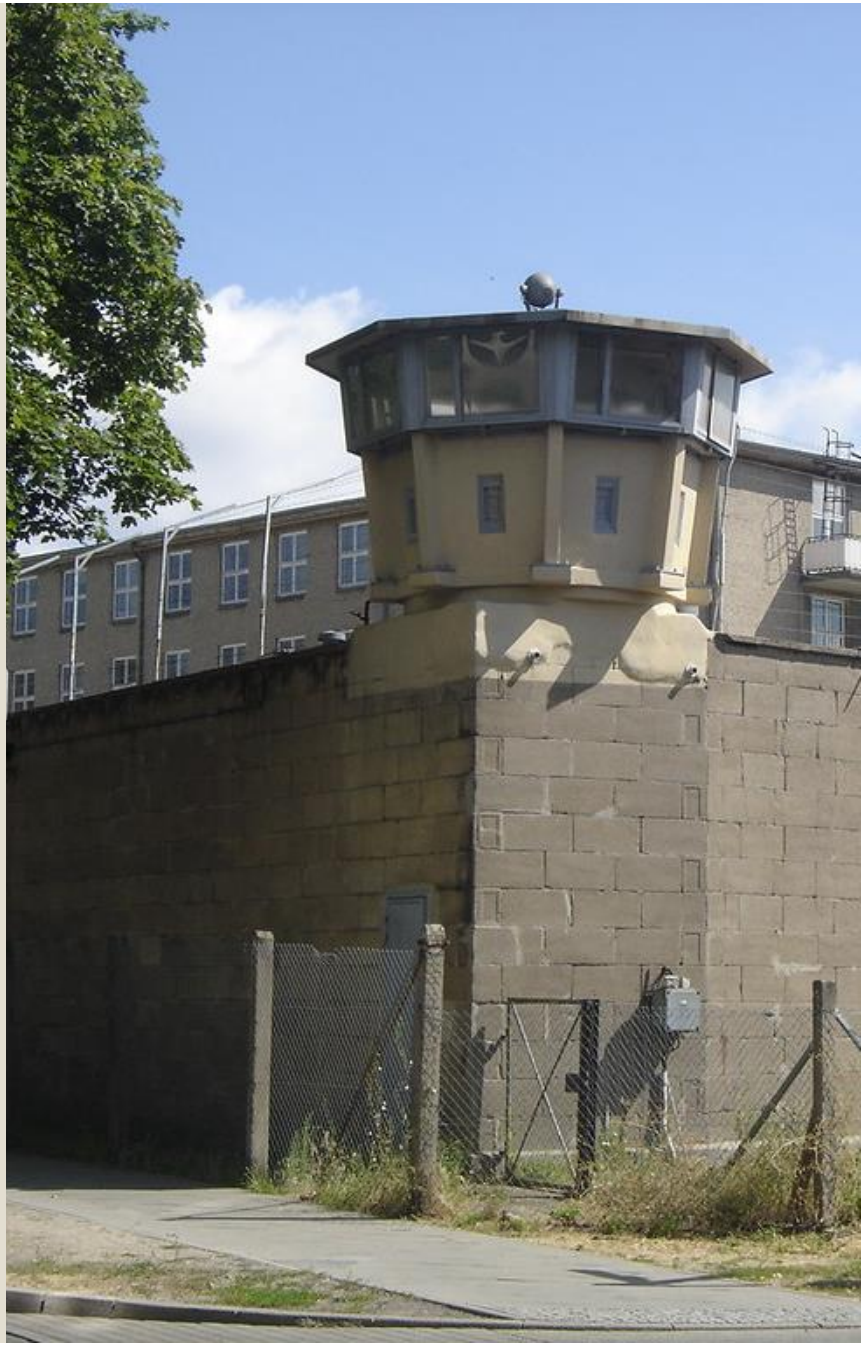
```
if (a is Horse)
{
    Console.WriteLine("Hö per dag: " + ((Horse)a).HayPerDay + " kg");
}
```

CODE – ALONG THE FORREST

Kod: TheForrest2022

Tjuv och Polis - Tillägg

- I staden ska det finnas ett fängelse, med en yta på 10x10 rutor.
- Så fort en tjuv blir anhållen ska tjuven förpassas till fängelset.
- Tjuven blir bara anhållen då denne har tjuvgods, annars inte.
- Fängelset ska visas nedanför staden, I konsollen, och tjuvarna ska fortsätta sina rörelser (upp-ner-höger-vänster-osv) inom fängelsets murar.
- En text ska visa hur många tjuvar som är I fängelset, och hur många som är på fri fot.
- Valfri extrauppgift:
  - Beroende på hur många saker tjuven har stulit så är denne fängslad olika tid.
    - En tjuv som stulit t ex 4 saker ska vara I fängelset 40 sekunder.
    - Därefter blir tjuven frisläppt.



Länkar

- C# Inheritance (w3schools.com)
- C# Polymorphism (w3schools.com)