Objektorienterad programmering
Föreläsning 12

Mahmud Al Hakim
mahmud@webacademy.se
www.webacademy.se

Agenda

- Introduktion till Arv
- Superklasser och Subklasser
- Dolda medlemmar (new och base)
- Statisk bindning
- Polymorfism
  (Dynamisk bindning och virtuella metoder)
- Konstruktorer vid arv

Arv

- Med hjälp av arv kan man skapa nya klasser genom att utgå från redan existerade klasser och utöka dem med ytterligare medlemmar.
- I C# åstadkommer man arv genom att skapa **subklasser** till klasser man redan har.
- OBS! Man kan inte skapa subklasser till structer.

Superklass

- Exempel
  Klassen Hus är en klass som beskriver hus i största allmänhet.

```
// Hus är en klass som beskriver hus i största allmänhet
class Hus
{
    protected double längd, bredd;

    protected int antalVåningar = 1;

    // senasteRenovering är ett årtal som avser yttre renovering
    protected int senasteRenovering;

    public double BeräknaYta()
    {
        return längd * bredd * antalVåningar;
    }
```


Klassen Hus fort.

```
    // Egenskaper
    public double Längd
    {
        get { return längd;  }
        set { längd = value; }
    }

    public double Bredd
    {
        get { return bredd;  }
        set { bredd = value; }
    }

    public int AntalVåningar
    {
        get { return antalVåningar;  }
        set { antalVåningar = value; }
    }

    public int SenasteRenovering
    {
        get { return senasteRenovering; }
        set { senasteRenovering = value; }
    }
}
```


Klassen Hus - Testprogram

```
static void Main(string[] args)
{
    // Ett hus
    Hus h = new Hus();
    h.Längd = 15;
    h.Bredd = 10;
    h.AntalVåningar = 2;
    h.SenasteRenovering = 2010;
    Console.WriteLine("Hus");
    Console.WriteLine("--------------------");
    Console.WriteLine("Längd: " + h.Längd);
    Console.WriteLine("Bredd: " + h.Bredd);
    Console.WriteLine("Antal våningar: " + h.AntalVåningar);
    Console.WriteLine("Senaste yttre renovering: " + h.SenasteRenovering);
    Console.WriteLine("Yta: " + h.BeräknaYta());
    Console.WriteLine("====================");
}
```

Subklass - Bostadshus

När man deklarerar en subklass skriver man ett kolon och efter detta anger man superklassens namn

```
// Bostadshus är subklass till Hus
class Bostadshus : Hus
{
    // Anger om huset är tilläggsisolerat
    bool tilläggsisolerat;

    public bool Tilläggsisolerat
    {
        get { return tilläggsisolerat;  }
        set { tilläggsisolerat = value; }
    }

    // Metoden gör det möjligt att isolera ett hus i efterhand
    public void Isolera()
    {
        tilläggsisolerat = true;
    }

}
```


Bostadshus - Testprogram

Ett objekt av klassen Bostadshus får nu fem instansvariabler

```
// Ett bostadshus
Bostadshus b = new Bostadshus();
b.Längd = 10;
b.Bredd = 20;
b.AntalVåningar = 3;
b.SenasteRenovering = 2012;
b.Isolera();
Console.WriteLine("Bostadshus");
Console.WriteLine("--------------------");
Console.WriteLine("Längd: " + b.Längd);
Console.WriteLine("Bredd: " + b.Bredd);
Console.WriteLine("Antal våningar: " + b.AntalVåningar);
Console.WriteLine("Senaste yttre renovering: " + b.SenasteRenovering);
Console.WriteLine("Yta: " + b.BeräknaYta());
if (b.Tilläggsisolerat)
    Console.WriteLine("Huset är tilläggsisolerat");
Console.WriteLine("====================");
```

Subklass - Flerfamiljshus

```
// Flerfamiljshus är subklass till Bostadshus
// Klassen beskriver bostadshus med flera lägenheter
class Flerfamiljshus : Bostadshus
{
    int antalLägenheter;

    public int AntalLägenheter
    {
        get { return antalLägenheter; }
        set { antalLägenheter = value; }
    }

    // hyraPerM2 är schablonhyra per kvadratmeter
    // Tips: const är konstant men blir static automatiskt
    const double hyraPerM2 = 1000;

    public double BeräknaHyresinkomst()
    {
        return BeräknaYta() * hyraPerM2;
    }

}
```


Flerfamiljshus - Testprogram

```
// Ett Flerfamiljshus
Flerfamiljshus f = new Flerfamiljshus();
f.Längd = 10;
f.Bredd = 20;
f.AntalVåningar = 3;
f.SenasteRenovering = 2013;
f.Isolera();
f.AntalLägenheter = 6;
Console.WriteLine("Flerfamiljshus");
Console.WriteLine("--------------------");
Console.WriteLine("Längd: " + f.Längd);
Console.WriteLine("Bredd: " + f.Bredd);
Console.WriteLine("Antal våningar: " + f.AntalVåningar);
Console.WriteLine("Senaste yttre renovering: " + f.SenasteRenovering);
Console.WriteLine("Yta: " + f.BeräknaYta());
if (f.Tilläggsisolerat)
    Console.WriteLine("Huset är tilläggsisolerat");
Console.WriteLine("Antal lägenheter: " + f.AntalLägenheter);
Console.WriteLine("Hyresinkomster: " + f.BeräknaHyresinkomst() );
Console.WriteLine("====================");
```

Dolda medlemmar

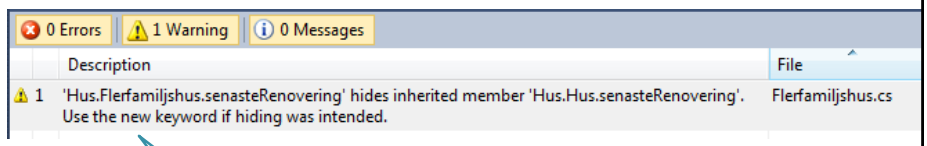
- Vad händer om man deklarerar en medlem i en subklass och det finns en motsvarande medlem med samma namn i superklassen?

```
// senasteRenovering anger när huset senast renoverades invändigt
new int senasteRenovering;

public new int SenasteRenovering
{
    get { return senasteRenovering; }
    set { senasteRenovering = value; }
}
```

senasteRenovering finns ju i superklassen Hus men **döljs** i subklassen Flerfamiljshus Med ordet **new** döljer man variabeln som finns i superklassen

Vad händer om man inte skriver ordet new?



Man får en varning från kompilatorn!
Använd nyckelordet **new** om du vill dölja medlemmar som redan finns i superklassen

Hur kommer man åt dolda medlemmar (inuti klassen)?

Med nyckelordet **base** kommer vi åt medlemmar i superklassen

```
// Ny metod med samma namn i superklassen
// Metoden i superklassen döljs
public new double BeräknaYta()
{
    return base.BeräknaYta() * 0.95; // ta bort 5% trappuppgånger
}
```

Ny metod i klassen Flerfamiljshus

Här anropas den nya metoden BeräknaYta() som finns i Flerfamiljshus

```
Console.WriteLine("Yta: " + f.BeräknaYta());
```


base – Exempel 2

Ny metod i klassen Flerfamiljshus

```
// En metod som anger när yttre respektive inre renovering utfördes
public void Renovera(int yttre, int inre)
{
    base.SenasteRenovering = yttre;
    senasteRenovering = inre;
}
```

Med ordet **base** kommer vi åt medlemmar i superklassen

Statisk bindning

- Hur kommer man åt dolda medlemmar (utanför klassen)?
- **Referensvariabelns typ** avgör vilken medlem ska anropas.
- hTemp refererar nu till samma objekt men den är av typen Hus

```
f.Renovera(2013, 2014);

Hus hTemp = f;

Console.WriteLine("Flerfamiljshus, senaste yttre renovering: " + hTemp.SenasteRenovering);

Console.WriteLine("Flerfamiljshus, senaste inre renovering:  " + f.SenasteRenovering);
```

senasteRenovering hämtas här från klassen Hus

Detta kallas **statisk bindning** eftersom det redan vid kompileringen kan bestämmas exakt vilken medlem som menas.

SenasteRenovering hämtas här från klassen Flerfamiljshus

15

Polymorfism

- Om man deklarerar en medlem i en subklass och det finns en motsvarande medlem med **samma namn och parametrar** i superklassen.
- Och man önskar vid metodanrop att objektets typ (inte referensens) skall styra vilken metod som skall anropas.
- Då måste man använda sig av s.k. **virtuella metoder**.
- Detta kallas **polymorfism** (månformighet).
- Objekt som tillhör polymorfa klasser kan ha operationer med samma namn som logiskt sett utför samma operation på de olika objekten, men operationerna kan ändå utföra olika ting beroende på vilket slags objekt det är fråga om.

16

Virtuella metoder - Exempel

Metoden BeräknaYta()
i klassen Hus är nu **virtuell**

```
public virtual double BeräknaYta()
{
    return längd * bredd * antalVåningar;
}
```

Vi byter ut new mot **override** i metoden
BeräknaYta()
i klassen Flerfamiljshus
Man brukar säga att man åsidosätter
(överskuggar) metoden

```
public override double BeräknaYta()
{
    return base.BeräknaYta() * 0.95; // ta bort 5% trappuppgånger
}
```


Virtuella metoder – Exempel fort.

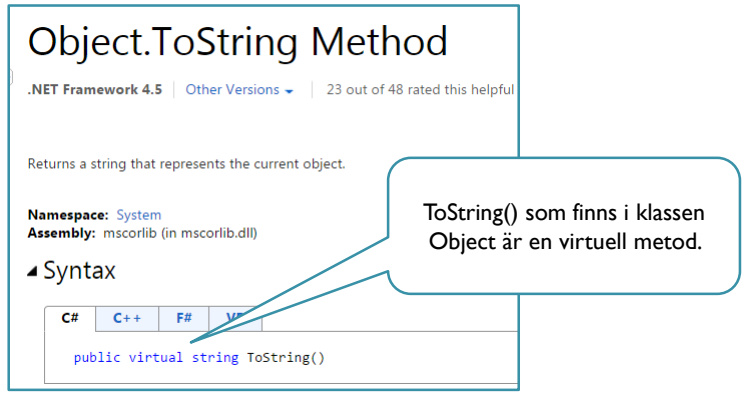
```
Hus hv = new Hus();
hv.Längd = 10; hv.Bredd = 10;
Flerfamiljshus fv = new Flerfamiljshus();
fv.Längd = 20; fv.Bredd = 20;
Console.WriteLine("Hus: "+ hv.BeräknaYta());
Console.WriteLine("Flerfamiljshus: " + fv.BeräknaYta());

hv = fv; // Nu pekar referensen hv på samma objekt som fv pekar på
Console.WriteLine("Hus: " + hv.BeräknaYta());
```

Vilken metod anropas?
Vad händer om man ändrar
override till new i subklassen
Flerfamiljshus?

Metoden ToString() är virtuell

- Läs gärna på MSDN



Dynamisk bindning

- En metod som har markerats med **override** i en subklass och som har samma namn och parametrar som en metod som markerats som **virtual** (eller override) i en superklass blir **en alternativ version** av metoden i superklassen.
- Objektets typ (klassen) styr vilken metod som skall anropas.
- Typen på referensen saknar betydelse.
- Sökning efter första metod som passar sker nerifrån och uppåt i klasshierarkin.

Konstruktorer vid arv (Klassen Hus)

```
// En parameterlös konstruktor
public Hus() { }

// En konstruktorer med 3 parametrar
public Hus(double längd, double bredd, int antalVåningar)
{
    this.längd = längd;
    this.bredd = bredd;
    this.antalVåningar = antalVåningar;
}
```


Konstruktorer vid arv (Klassen Bostadshus)

```
// En parameterlös konstruktor
public Bostadshus()
{
    tilläggsisolerat = true;
}

// En konstruktorer med 1 parameter
public Bostadshus(bool tilläggsisolerat)
{
    this.tilläggsisolerat = tilläggsisolerat;
}
```

Konstruktorer vid arv (Klassen Bostadshus – del 2)

```
// En konstruktorer med 4 parametrar

public Bostadshus(double längd, double bredd, int antalVåningar, bool tilläggsisolerat)

    : base(längd, bredd , antalVåningar)

{
    this.tilläggsisolerat = tilläggsisolerat;
}
```

En **konstruktorinitierare** läggs efter parameterlistan.
: base anropar superklassens konstruktor.