Nedarvning/ inheritance

Det er muligt at vedarve felter og metoder fra en class til en anden med nedarvningskonceptet.
Man kan dele konceptet op i to grupper:

1 Parent Class - Der skal nedarves fra.
1 Child Class – Der skal nedarve fra en anden.

Syntaksen for at nedarve er ":"

```
1 reference
class Meal  // parent class
{
    public string type = "Pizza";  // Meal field
    1 reference
    public void yummy()             // Meal method
    {
        Console.WriteLine("Nam, nam!");
    }
}
```

```
2 references
class Dinner : Meal  // child class
{
    public string typePizza = "Kebab";  // Dinner field
}
0 references
class Program
```

```
Microsoft Visual Studio Debug Console
Nam, nam!
Pizza Kebab
```

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        // Create a myDinner object
        Dinner myDinner = new Dinner();

        // Call the yummy() method (From the Meal class) on the myDinner object
        myDinner.yummy();

        // Display the value of the type field (from the Meal class) and the value of the typePizza from the Meal class
        Console.WriteLine(myDinner.type + " " + myDinner.typePizza);
    }
}
```

Brug af nedarvning er nødigt i forbindelse med genbrugskode af felter og "Methods" ved dannelsen af nye "Classes".

Nedarvning med Override

Når man nedarver flere ”Child classes” fra en ”Parent class” kan man genbruge ”Method” navne med forskellige output afhængig af hvilken ”Child” man kalder fra. Ved en sådan situation vil man sandsynligvis få brug for et Keyword i c# der hedder Override. Hvis ikke man gør brug af dette vil outputtet fra ”Parent Method” altid over skrive ”Child Method”

```
6 references
class Bird  // parent class
{
    5 references
    public virtual void birdSound()
    {
        Console.WriteLine("Fuglen laver en lyd");
    }
}

1 reference
class Blackbird : Bird  // child class)
{
    5 references
    public override void birdSound()
    {
        Console.WriteLine("Solsorten fløjter en smuk lyd");
    }
}

1 reference
class Seagul : Bird  // child class
{
    5 references
    public override void birdSound()
    {
        Console.WriteLine("Havmågen siger nogle mega trælse lyde");
    }
}
```

```
class Program_override
{
    0 references
    static void Main(string[] args)
    {
        Bird myBird = new Bird();  // Create a Bird object
        Bird myBlackbird = new Blackbird();  // Create a Blackbird object
        Bird mySeagul = new Seagul();  // Create a Seagul object

        myBird.birdSound();
        myBlackbird.birdSound();
        mySeagul.birdSound();
    }
}
```

```
Microsoft Visual Studio Debug Console
Fuglen laver en lyd
Solsorten fløjter en smuk lyd
Havmågen siger nogle mega trælse lyde
```

Uden Override vil dette være output:

```
6 references
class Bird  // parent class
{
    3 references
    public virtual void birdSound()
    {
        Console.WriteLine("Fuglen laver en lyd");
    }
}

1 reference
class Blackbird : Bird  // child class)
{
    0 references
    public void birdSound()
    {
        Console.WriteLine("Solsorten fløjter en smuk lyd");
    }
}

1 reference
class Seagul : Bird  // child class
{
    0 references
    public void birdSound()
    {
        Console.WriteLine("Havmågen siger nogle mega trælse lyde");
    }
}
0 references
```

```
class Program_override
{
    0 references
    static void Main(string[] args)
    {
        Bird myBird = new Bird();  // Create a Bird object
        Bird myBlackbird = new Blackbird();  // Create a Blackbird object
        Bird mySeagul = new Seagul();  // Create a Seagul object

        myBird.birdSound();
        myBlackbird.birdSound();
        mySeagul.birdSound();
    }
}
```

```
Microsoft Visual Studio Debug Console
Fuglen laver en lyd
Fuglen laver en lyd
Fuglen laver en lyd
```