

Overerving

```





public class Barbaar
{
    1 reference
    public Barbaar(string naam, int levenspunten, int aanvalSterkte, double snelheid)
    {
        Naam = naam;
        Levenspunten = levenspunten;
        AanvalSterkte = aanvalSterkte;
        Snelheid = snelheid;
    }
    2 references
    public string Naam { get; set; }
    2 references
    public int Levenspunten { get; set; }
    2 references
    public int AanvalSterkte { get; set; }
    2 references
    public double Snelheid { get; set; }
    1 reference
    public void Beweeg()
    {
        Console.WriteLine($"{this.GetType()} moving");
    }
    1 reference
    public void WordWild()
    {
        Console.WriteLine("Grrrrrrrr");
    }
    19 references
    public override string ToString()
    {
        return $"[barbaar]{Naam},{Levenspunten},{AanvalSterkte},{Snelheid}";
    }
}

```





Barbaar

Class

Properties

-  AanvalSterkte
-  Levenspunten
-  Naam
-  Snelheid

Methods

-  Barbaar
-  Beweeg
-  ToString
-  WordWild

```





public class Heks
{
    1 reference
    public Heks(string naam, int levenspunten, int aanvalSterkte, double snelheid)
    {
        Naam = naam;
        Levenspunten = levenspunten;
        AanvalSterkte = aanvalSterkte;
        Snelheid = snelheid;
    }
    2 references
    public string Naam { get; set; }
    2 references
    public int Levenspunten { get; set; }
    2 references
    public int AanvalSterkte { get; set; }
    2 references
    public double Snelheid { get; set; }
    1 reference
    public void Beweeg()
    {
        Console.WriteLine($"{this.GetType()} moving");
    }
    1 reference
    public void Verschijn()
    {
        Console.WriteLine("hier ben ik");
    }
    1 reference
    public void Verberg()
    {
        Console.WriteLine("je kan me niet zien");
    }
    19 references
    public override string ToString()...
}

```

Heks

Class

Properties

-  AanvalSterkte
-  Levenspunten
-  Naam
-  Snelheid

Methods

-  Beweeg
-  Heks
-  ToString
-  Verberg
-  Verschijn

```

public class Kobol
{
    1 reference
    public Kobol(string naam, int levenspunten, int aanvalSterkte, double snelheid)
    {
        Naam = naam;
        Levenspunten = levenspunten;
        AanvalSterkte = aanvalSterkte;
        Snelheid = snelheid;
        GoudVerzameld = 0;
    }
    2 references
    public string Naam { get; set; }
    2 references
    public int Levenspunten { get; set; }
    2 references
    public int AanvalSterkte { get; set; }
    2 references
    public double Snelheid { get; set; }
    3 references
    public int GoudVerzameld { get; set; }
    1 reference
    public void Beweeg()
    {
        Console.WriteLine($"{this.GetType()} moving");
    }
    1 reference
    public void SteelGoud()
    {
        GoudVerzameld += 100;
        Console.WriteLine("ik word rijk");
    }
    19 references
    public override string ToString()...
}

```

Kobol

Class

Properties





-  AanvalSterkte
-  GoudVerzameld
-  Levenspunten
-  Naam
-  Snelheid

Methods





-  Beweeg
-  Kobol
-  SteelGoud
-  ToString

Barbaar
Class

Properties





 AanvalSterkte
 Levenspunten
 Naam
 Snelheid

Methods






 Barbaar
 Beweeg
 ToString
 WordWild

Heks
Class

Properties






 AanvalSterkte
 Levenspunten
 Naam
 Snelheid

Methods





 Beweeg
 Heks
 ToString
 Verberg
 Verschijn

Kobol
Class

Properties

 AanvalSterkte
 GoudVerzameld
 Levenspunten
 Naam
 Snelheid

Methods

 Beweeg
 Kobol
 SteelGoud
 ToString

```

static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
    Heks h = new Heks("Gwendolien", 200, 300, 2.5);
    Kobol k = new Kobol("Karel", 300, 50, 8.0);
    Barbaar b = new Barbaar("Guy", 500, 500, 1.5);
    Console.WriteLine(h);
    Console.WriteLine(k);
    Console.WriteLine(b);
    h.Verschijn();
    b.WordWild();
    h.Beweeg();
    k.Beweeg();
    b.Beweeg();
    h.Verberg();
    k.SteelGoud();
}

```

Microsoft Visual Studio Debug Console

```

Hello World!
[heks]Gwendolien,200,300,2,5
[kobol]Karel,300,50,8,0
[barbaar]Guy,500,500,1,5
hier ben ik
Grrrrrrrr
Inheritance.Heks moving
Inheritance.Kobol moving
Inheritance.Barbaar moving
je kan me niet zien
ik word rijk

```


```


public void Beweeg()
{
    Console.WriteLine($"{this.GetType()} moving");
}


```


Barbaar
Class

Properties


 AanvalSterkte


 Levenspunten


 Naam


 Snelheid

Methods

 Barbaar


 Beweeg


 ToString


 WordWild


Heks
Class

Properties


 AanvalSterkte


 Levenspunten


 Naam


 Snelheid


Methods

 Beweeg

 Heks


 ToString


 Verberg


 Verschijn


Kobol
Class


Properties

 AanvalSterkte


 GoudVerzameld


 Levenspunten


 Naam


 Snelheid

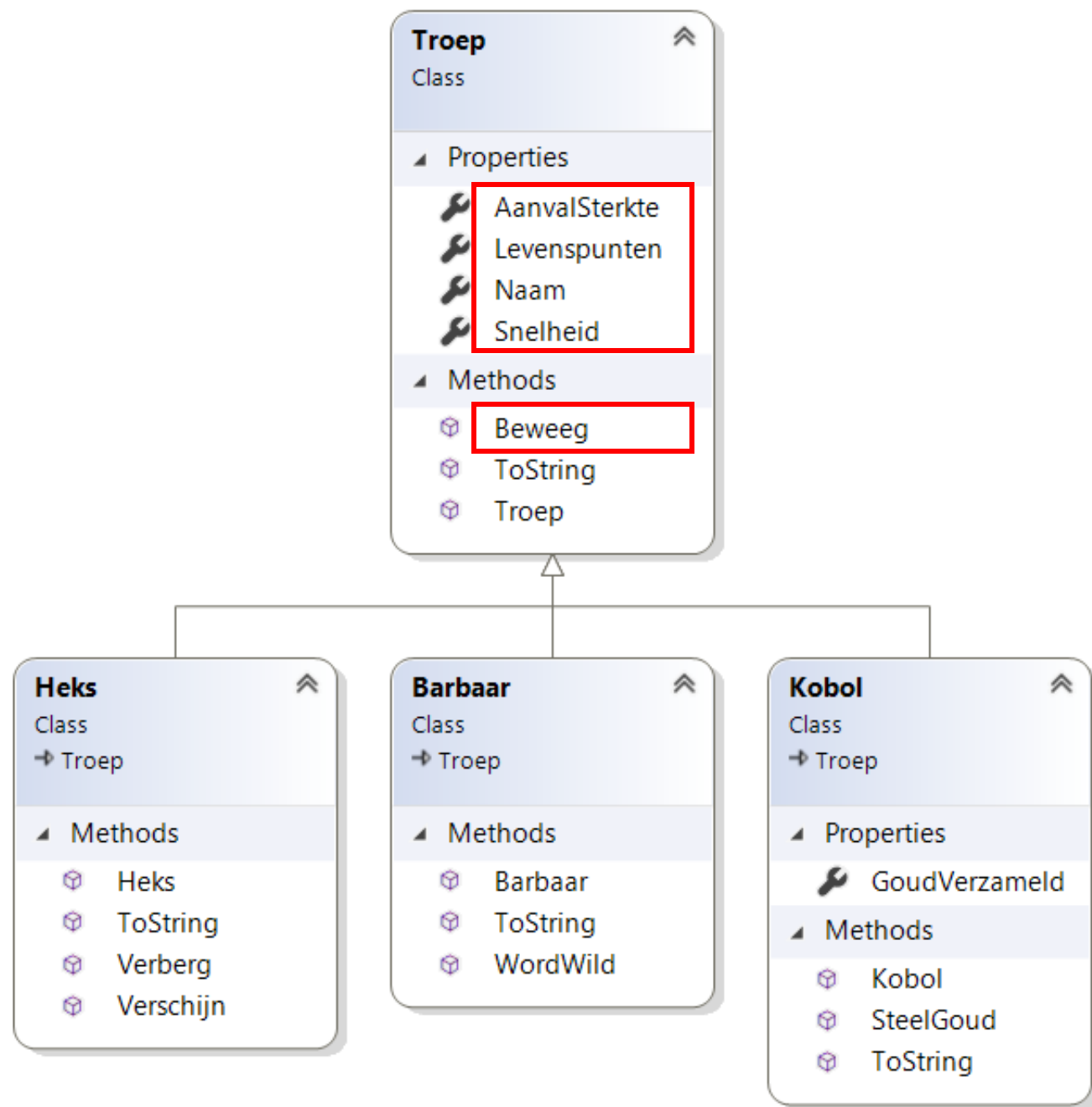
Methods

 Beweeg

 Kobol

 SteelGoud


 ToString





```





public class Troep
{
    4 references
    public Troep(string naam, int levenspunten, int aanvalSterkte, double snelheid)
    {
        Naam = naam;
        Levenspunten = levenspunten;
        AanvalSterkte = aanvalSterkte;
        Snelheid = snelheid;
    }
    2 references
    public string Naam { get; set; }
    2 references
    public int Levenspunten { get; set; }
    2 references
    public int AanvalSterkte { get; set; }
    2 references
    public double Snelheid { get; set; }
    1 reference
    public void Beweeg()
    {
        Console.WriteLine($"{this.GetType()} moving");
    }
    19 references
    public override string ToString()
    {
        return $"[troep]{Naam},{Levenspunten},{AanvalSterkte},{Snelheid}";
    }
}


```




Troep


Class

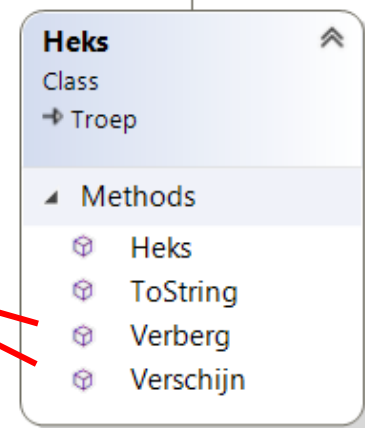
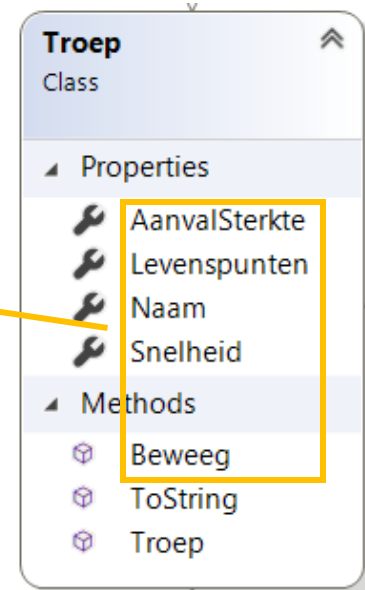
 Properties

 AanvalSterkte
 Levenspunten
 Naam
 Snelheid

 Methods

 Beweeg
 ToString
 Troep

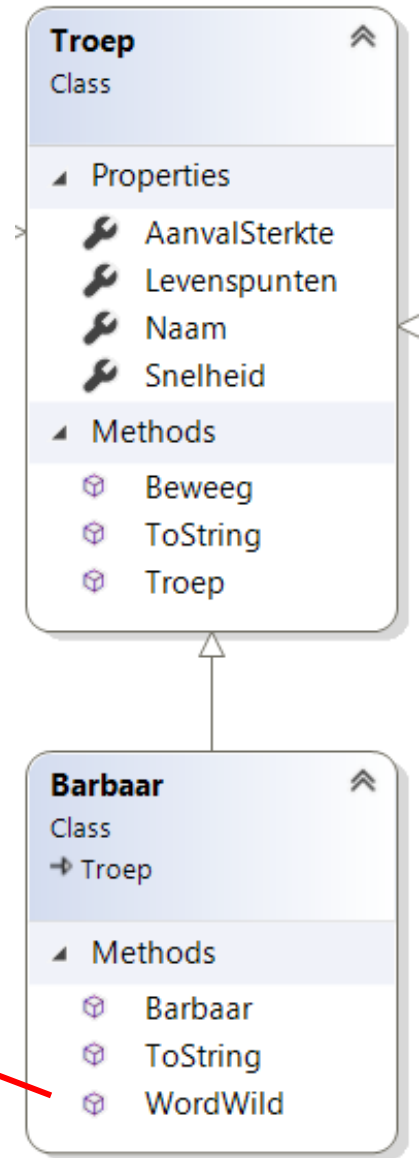
```
public class Heks Troep
{
    1 reference
    public Heks(string naam, int levenspunten, int aanvalSterkte, double snelheid)
        : base(naam, levenspunten, aanvalSterkte, snelheid)
    {
    }
    1 reference
    public void Verschijn()
    {
        Console.WriteLine("hier ben ik");
    }
    1 reference
    public void Verberg()
    {
        Console.WriteLine("je kan me niet zien");
    }
    19 references
    public override string ToString()
    {
        return base.ToString()+"[heks]";
    }
}
```



```

public class Barbaar: Troep
{
    1 reference
    public Barbaar(string naam, int levenspunten, int aanvalSterkte, double snelheid)
        : base(naam, levenspunten, aanvalSterkte, snelheid)
    {
    }
    1 reference
    public void WordWild()
    {
        Console.WriteLine("Grrrrrrrr");
    }
    - references
    public override string ToString()
    {
        return base.ToString() + $"[barbaar]";
    }
}

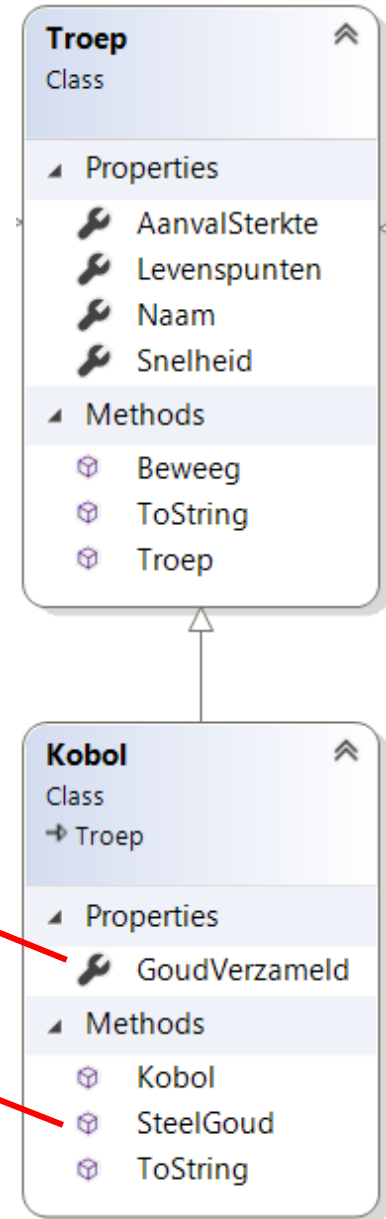
```



```

public class Kobol : Troep
{
    1 reference
    public Kobol(string naam, int levenspunten, int aanvalSterkte, double snelheid)
        : base(naam, levenspunten, aanvalSterkte, snelheid)
    {
        GoudVerzameld = 0;
    }
    3 references
    public int GoudVerzameld { get; set; }
    1 reference
    public void SteelGoud()
    {
        GoudVerzameld += 100;
        Console.WriteLine("ik word rijk");
    }
    19 references
    public override string ToString()
    {
        return base.ToString() + $"[kobol]{GoudVerzameld}";
    }
}

```



```

Console.WriteLine("Hello World!");
Heks h = new Heks("Gwendolien", 200, 300, 2.5);
Kobol k = new Kobol("Karel", 300, 50, 8.0);
Barbaar b = new Barbaar("Guy", 500, 500, 1.5);
Console.WriteLine(h);
Console.WriteLine(k);
Console.WriteLine(b);
List<Troep> troepen = new List<Troep>();
troepen.Add(h);
troepen.Add(b);
troepen.Add(k);
h.Verschijn();
b.WordWild();
foreach(Troep t in troepen)
{
    t.Beweeg();
}
h.Verberg();
k.SteelGoud();

```

Microsoft Visual Studio Debug Console

```

Hello World!
[troep]Gwendolien,200,300,2,5[heks]
[troep]Karel,300,50,8[kobol]0
[troep]Guy,500,500,1,5[barbaar]
hier ben ik
Grrrrrrr
Inheritance_2.Heks moving
Inheritance_2.Barbaar moving
Inheritance_2.Kobol moving
je kan me niet zien
ik word rijk
[troep]xxx,100,50,5

```

```
public class Troep
```

```
Troep tr = new Troep("xxx", 100, 50, 5.0);  
Console.WriteLine(tr);
```

```
public abstract class Troep
```

```
Troep tr = new Troep("xxx", 100, 50, 5.0);  
Console.WriteLine(tr);
```

■ readonly struct System.Int32
Represents a 32-bit signed integer.

CS0144: Cannot create an instance of the abstract type or interface 'Troep'

Polymorphism

```

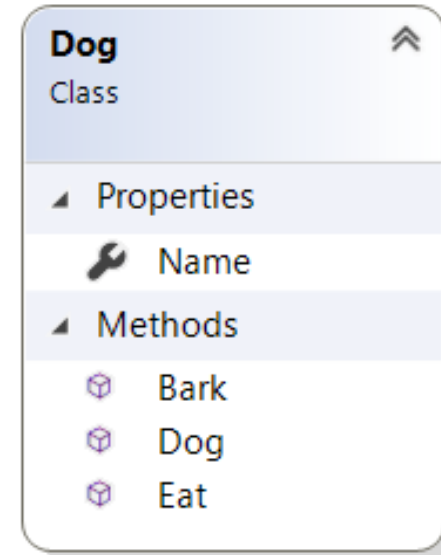
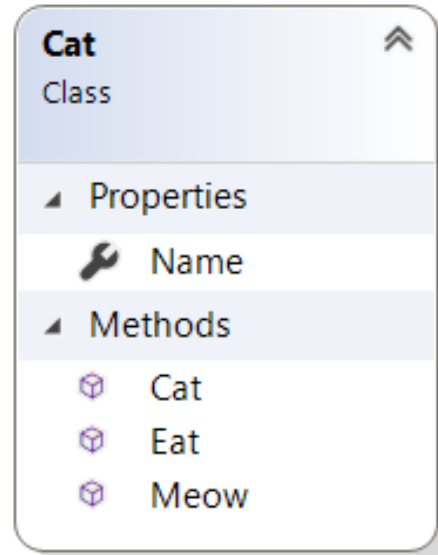
public class Dog
{
    0 references
    public Dog(string name)
    {
        Name = name;
    }
    1 reference
    public string Name { get; set; }
    0 references
    public void Bark()
    {
        Console.WriteLine("bark");
    }
    0 references
    public void Eat()
    {
        Console.WriteLine("eating dog food");
    }
}

```

```

public class Cat
{
    0 references
    public Cat(string name)
    {
        Name = name;
    }
    1 reference
    public string Name { get; set; }
    0 references
    public void Meow()
    {
        Console.WriteLine("miauw");
    }
    0 references
    public void Eat()
    {
        Console.WriteLine("eating mice");
    }
}

```




```

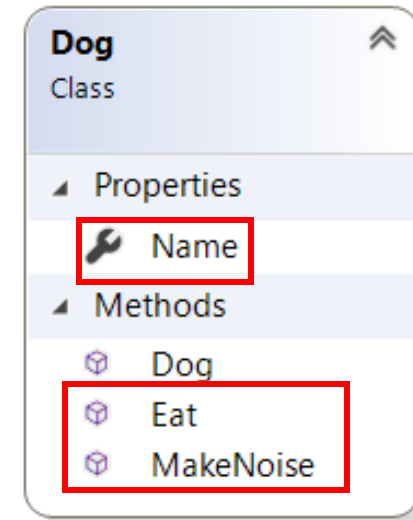
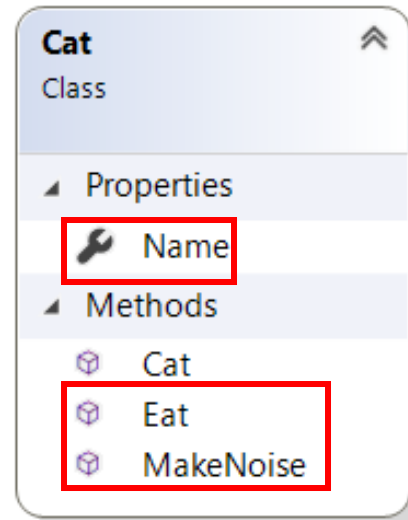
public class Cat
{
    0 references
    public Cat(string name)
    {
        Name = name;
    }
    1 reference
    public string Name { get; set; }
    0 references
    public void MakeNoise()
    {
        Console.WriteLine("miauw");
    }
    0 references
    public void Eat()
    {
        Console.WriteLine("eating mice");
    }
}

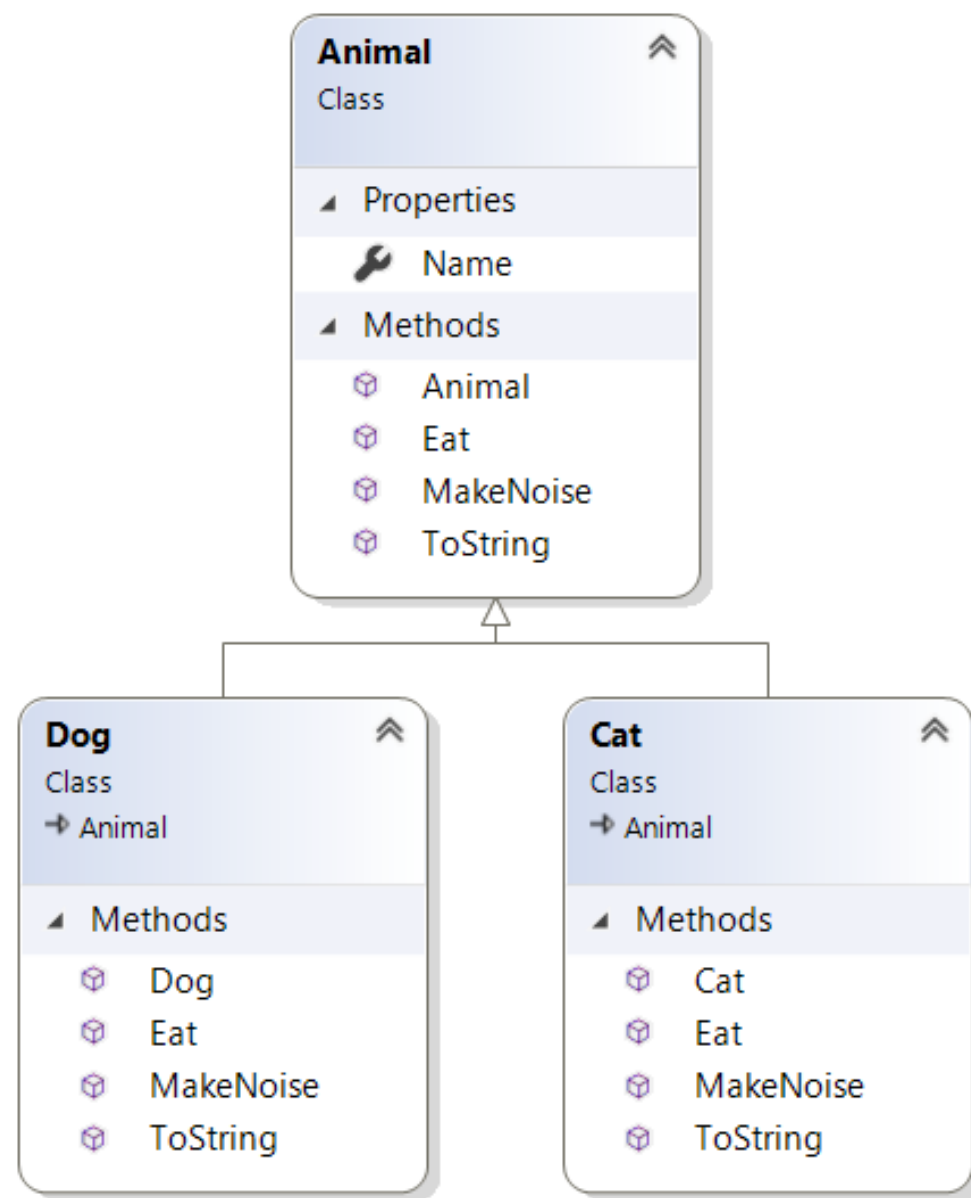
```

```

public class Dog
{
    0 references
    public Dog(string name)
    {
        Name = name;
    }
    1 reference
    public string Name { get; set; }
    0 references
    public void MakeNoise()
    {
        Console.WriteLine("bark");
    }
    0 references
    public void Eat()
    {
        Console.WriteLine("eating dog food");
    }
}

```





```
public class Animal
{
    3 references
    public Animal(string name)
    {
        Name = name;
    }
    4 references
    public string Name { get; set; }
    3 references
    public virtual void MakeNoise()
    {
        Console.WriteLine("grrrr");
    }
    7 references
    public virtual void Eat()
    {
        Console.WriteLine("eating");
    }
    9 references
    public override string ToString()
    {
        return $"[Animal]{Name}";
    }
}
```

Virtual gebruiken we als een methode mag (maar niet moet) worden overschreven

```

public class Animal
{
    3 references
    public Animal(string name)
    {
        Name = name;
    }
    4 references
    public string Name { get; set; }
    5 references
    public void MakeNoise()
    {
        Console.WriteLine("grrrr");
    }
    9 references
    public virtual void Eat()
    {
        Console.WriteLine("eating");
    }
    9 references
    public override string ToString()
    {
        return $"[Animal]{Name}";
    }
}

```

```

public class Cat : Animal
{
    1 reference
    public Cat(string name) : base(name)
    {
    }
    1 reference
    public new void MakeNoise()
    {
        base.MakeNoise();
        Console.WriteLine("miauw");
    }
    9 references
    public override void Eat()
    {
        base.Eat();
        Console.WriteLine("eating mice");
    }
    9 references
    public override string ToString()
    {
        return $"[Cat]{Name}";
    }
}

```

we gebruiken **override** om aan te duiden dat we de implementatie van de methode wensen te veranderen

```
Animal a = new Animal("Leo");
Console.WriteLine(a);
a.Eat();
Console.WriteLine("-----");
Dog d = new Dog("Santa");
Console.WriteLine(d);
d.Eat();
Console.WriteLine("-----");
Cat c = new Cat("Snowball");
Console.WriteLine(c);
c.Eat();
Console.WriteLine("-----");
Console.WriteLine("-----");
```

Microsoft Visual Studio Debug Console

```
[Animal]Leo
eating
-----
[Dog]Santa
eating
eating dog food
-----
[Cat]Snowball
eating
eating mice
-----
-----
```

```
public override void Eat()
{
    base.Eat();
    Console.WriteLine("eating dog food");
}
```

```
public virtual void Eat()
{
    Console.WriteLine("eating");
}
```

```

public class Animal
{
    3 references
    public Animal(string name)
    {
        Name = name;
    }
    4 references
    public string Name { get; set; }
    5 references
    public void MakeNoise()
    {
        Console.WriteLine("grrrr");
    }
    9 references
    public virtual void Eat()
    {
        Console.WriteLine("eating");
    }
    9 references
    public override string ToString()
    {
        return $"[Animal]{Name}";
    }
}

```

```

public class Cat : Animal
{
    1 reference
    public Cat(string name) : base(name)
    {
    }
    1 reference
    public new void MakeNoise()
    {
        base.MakeNoise();
        Console.WriteLine("miauw");
    }
    9 references
    public override void Eat()
    {
        base.Eat();
        Console.WriteLine("eating mice");
    }
    9 references
    public override string ToString()
    {
        return $"[Cat]{Name}";
    }
}

```

We gebruiken **new** als we de methode uit de **base class** wensen af te schermen

```

Animal a = new Animal("Leo");
Console.WriteLine(a);
a.MakeNoise();
a.Eat();
Console.WriteLine("-----");
Dog d = new Dog("Santa");
Console.WriteLine(d);
d.MakeNoise();
d.Eat();
Console.WriteLine("-----");
Cat c = new Cat("Snowball");
Console.WriteLine(c);
c.MakeNoise();
c.Eat();
Console.WriteLine("-----");
Console.WriteLine("-----");

```

Microsoft Visual Studio Debug Console

```

[Animal]Leo
grrrr
eating
-----
[Dog]Santa
grrrr
bark
eating
eating dog food
-----
[Cat]Snowball
grrrr
miauw
eating
eating mice
-----
-----

```

New of override ? blijktbaar geen verschil ?

```

Animal a = new Animal("Leo");
Console.WriteLine(a);
Console.WriteLine("-----");
Dog d = new Dog("Santa");
Console.WriteLine(d);
Console.WriteLine("-----");
Cat c = new Cat("Snowball");
Console.WriteLine(c);
Console.WriteLine("-----");
List<Animal> animals = new List<Animal>();
animals.Add(a);
animals.Add(d);
animals.Add(c);
foreach (Animal animal in animals)
{
    Console.WriteLine(animal);
    animal.MakeNoise();
    animal.Eat();
    Console.WriteLine("-----");
}
Console.WriteLine("-----");
Console.WriteLine("-----");

```

Microsoft Visual Studio Debug Console

```

[Animal]Leo
-----
[Dog]Santa
-----
[Cat]Snowball
-----
[Animal]Leo
grrrr
eating
-----
[Dog]Santa
grrrr
eating dog food
-----
[Cat]Snowball
grrrr
eating
eating mice

```

Iedereen is een dier !

new schermt af -> enkel base beschikbaar

override overschrijft -> derived/child beschikbaar