

Geneste klassen en anonieme klassen

Hoofdstuk 14

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

Get your facts straight!



- ❑ Het is mogelijk een klasse te definiëren binnen een andere klasse.
- ❑ De geneste klasse kan niet onafhankelijk van de nestende klasse bestaan.
- ❑ De geneste klasse is een member van de nestende klasse
- ❑ De geneste klasse heeft toegang tot alle members van de nestende klasse
 - Omgekeerd niet.
- ❑ Het is mogelijk om een geneste klasse in een codeblock te declareren.
 - Lokale scope
- ❑ Er zijn drie soorten geneste klassen: member, static en anonymous

Gewone geneste klasse



Afbeelding 107: Een geneste klasse

```
public class Outerclass{
    ...
    class Innerclass{
        ...
    }
}
```

- ❑ OuterClass.InnerClass
→ **Volledige** naam
- ❑ OuterClass\$InnerClass.class
→ Naam van het **klasse bestand**

Geneste klasse

- ❏ De geneste klasse kan alleen worden gebruikt in context van de nestende klasse.
- Er moet een object van de nestende klasse bestaan om een object van de nestende klasse te kunnen maken.

Geneste klasse

- ❏ We kunnen een object van de geneste klasse aanmaken binnen de nestende klasse.

```
public class Outer {  
  
    public class Inner{  
        public void aMethod(){  
            System.out.println("in Inner.aMethod()");  
        }  
    }  
  
    public void doSomething(){  
        Inner inner = new Inner();  
        inner.aMethod();  
    }  
}
```

Geneste klassen object maken (InnerClass).

- ❑ Of we kunnen een object maken van de geneste klasse buiten de nestende klasse.
- We moeten dan wel een object van de nestende klasse hebben.
- ★ OuterClass **outer** = **new OuterClass()**;
OuterClass.InnerClass **inner** = **outer.new Innerclass()**;

Geneste klassen

- ❑ Geneste klassen hebben toegang tot alle members van de nestende klasse.
 - ook de private members.
 - **this** verwijst in de geneste klasse naar de geneste klasse.
- ❑ Om een **verwijzing** te krijgen naar de nestende klasse gebruiken we **OuterClass.this**

verschillende componenten declaratie geneste klassen.

❏ Vermits **geneste klassen** als **members** van **de nestende klasse** beschouwd worden kunnen ook volgende componenten gebruikt worden in de definitie

- final
- abstract
- static
- public | private | protected

◆ Gewone klasse kent enkel public of package

But what's the purpose?



- ❑ De bruikbaarheid van geneste klassen is niet zo overdreven groot.
- ❑ Ze zijn wel heel handig als het gaat over event handling.
- De lokale geneste en anonieme geneste klassen zijn daar uitermate geschikt voor.

We komen daarop terug als we event handling zien [in de demo].

Lokale geneste klassen

- ❏ Local inner classes
- is een geneste klasse die binnen een code blok gedefinieerd wordt.
- ❏ Meestal binnen een methode.
- Een lokale geneste klasse kan enkel binnen deze methode geïntanceerd worden.
- ❏ Hebben toegang tot de final variabelen van de methode waarin de geneste klasse gedefinieerd is.

Ok, but how does it look in code.

```
public class OuterClass{

    public void aMethod(){
        class SubClass extends SomeSuperClass{
            //Override methods
        }
        SomeSuperClass object1 = new SubClass();

        class InterfaceClass implements SomeInterface {
            //implement methods
        }
        SomeInterface object2 = new InterfaceClass();
    }
}
```

Anonieme geneste klassen

- ❑ Anonymous inner classes.
 - zijn **lokale** geneste klassen **zonder naam**.
 - ze worden gedefiniëerd op het moment dat we er een object van willen maken.

Anonieme geneste klassen.

```
public class OuterClass{  
    public void aMethod(){  
        SomeSuperClass object1 = new SomeSuperClass() {  
            //override methods  
        };  
  
        SomeInterface object2 = new SomeInterface() {  
            //implement methods  
        };  
    }  
}
```

Static nested classes

- ❏ Een geneste klasse die als static wordt gedefinieerd.

→

```
public class OuterClass {  
    ...  
    public static class InnerClass{  
        ...  
    }  
}
```

Static nested classes.

- ❑ Behoort toe aan de klasse en niet aan een concrete instantie.
- ❑ Kunnen geen gebruik maken van instance-members.
- ❑ Kunnen wel static members gebruiken.
- ❑ Je hebt geen object nodig van de nestende klasse om er één te instantiëren.
- ❑ Alternatief voor packages.
- ❑ Worden zelden gebruikt.

Static nested classes.

- ❑ Enums worden soms gedefinieerd binnen een bestaande klasse.

→ Static nested class.

- ❑ **static** mag weggelaten worden.

- ❑

```
public class ColorEnums{  
    ...  
    public enum Color{  
        ...  
    }  
}
```


Static nested classes.

- ❏ De enum kan zowel binnen als buiten de klasse gebruikt worden.
 - In het laatste geval dient men het te laten voorafgaan door de naam van de nestende klasse.

ColorEnums.Color color = ColorEnums.Color.RED;