

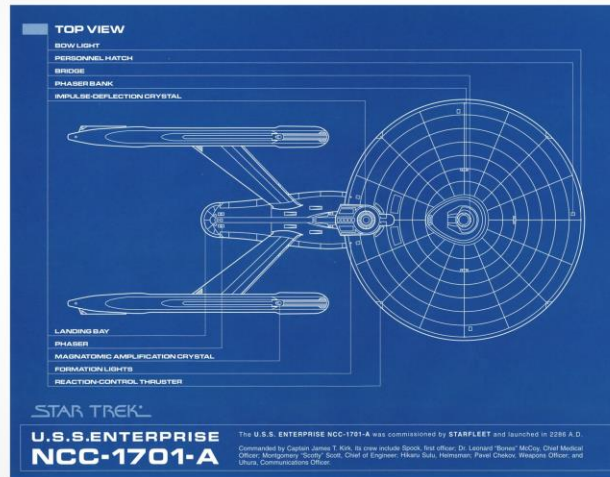
ECTTP Les 8: Classes

Valentijn Muijers

Classes

- Blauwdruk van een object
- Bevat eigenschappen van het object
- (dit zijn variabelen en functies die bij het object horen)
- Voorbeeld:
 - Class Car
 - 4 wielen
 - 1 stuur
 - 4 deuren
 - Kan rijden

Class



Een class is een blauwdruk waarin je data op kunt slaan.

Hierna kun je deze blauwdruk gebruiken om instanties te maken, deze instanties heten *objecten*.

Er kunnen meerdere instanties zijn van een class (bijvoorbeeld meerdere units in een spel allemaal met verschillende health/ posities etc., waarbij de class de eigenschappen van de unit beschrijft).

Er is altijd maar 1 definitie van een class.

Object



Een Object is een *instantie* van een class.

Objecten bestaan uit een logische groepering van functionaliteit, een object bevat:

- Eigen *attributen* (variabelen)
- Eigen *methodes* (functies)

Attributen

- Variabelen die bij een class horen heten *attributen*
- Functies die bij een class horen heten *methoden*

Trooper



Attributen

- Naam
 - "Sardauker Trooper"
- Prijs
 - Getal, (geldbedrag)
- Positie_x en Positie_y
 - Getallen (scherm-coördinaten)
- Health
 - Getal tussen 0 en 100



De attributen zijn de eigenschappen (variabelen) die de Trooper heeft

Methoden

- Attack(x, y)
- Move(x, y)
- Retreat()
- Guard(x, y)



De Methoden zijn de acties die de Trooper kan ondernemen.

De x en y die tussen de haakjes staan zijn de *parameters* of *argumenten* die mee worden gegeven bij het aanroepen van de methode, deze kunnen daarna binnen de methode worden gebruikt.

Interface

- Class Trooper

Attributen	Methoden
Naam	Attack(x,y)
Prijs	Move(x,y)
Positie_x	Retreat()
Positie_y	Guard(building)
Health	

Attributen en Methoden van de Trooper uit Dune2.

Een Interface beschrijft hoe een class gebruikt kan worden.

De Interface is tevens de documentatie van de class.

In het voorbeeld staan een aantal attributen, mogelijk zijn dit er meer (denk aan Damage, MoveSpeed etc.).

Classes in Python

```
4 class Car(object):  
5  
6     def __init__(self, x, y, speed):  
7         self.x = x  
8         self.y = y  
9         self.speed = speed  
10  
11     def move(self):  
12         self.x += self.speed
```

Hier een voorbeeld van de class genaamd Car

Om een class te instantiëren roep je de `init()` functie aan, deze zal het object initialiseren en de startwaarden aangeven.

Wanneer het object is aangemaakt kun je de `move()` methode aanroepen op het object, waarmee de auto zich zal verplaatsen

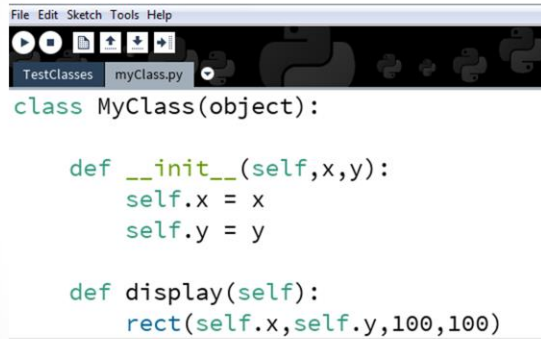
Object

```
15 | c = Car(10,10,5)
16 | c.move()
```

Om een auto aan te maken roep je de class naam aan en geef je de beginwaarden mee. Je slaat dit alles op een een variabele zodat je het object later kunt beïnvloeden. In dit voorbeeld wordt er een Car aangemaakt met x positie 10, y positie 10 en speed 5.

Daarna wordt van de auto de methode move() aangeroepen.

Processing Class



The screenshot shows the Processing IDE interface. The menu bar at the top includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu is a toolbar with icons for running, saving, and other functions. The file explorer shows two tabs: 'TestClasses' and 'myClass.py'. The main code editor displays the following Python code:

```
class MyClass(object):  
  
    def __init__(self,x,y):  
        self.x = x  
        self.y = y  
  
    def display(self):  
        rect(self.x,self.y,100,100)
```

Processing Voorbeeld!



```
from myClass import MyClass
```

```
def setup():  
    size(640, 360)  
    global r  
    r = MyClass(100,100)
```

```
def draw():  
    background(0)  
    r.display()
```

Nu jullie!

- Maak een class aan genaamd Square(object)
- Met als methode de functie display(self), die een rectangle op het scherm tekent
- Maak vervolgens een object aan wat een Square is en sla deze op in een variabele
- Roep op dit object de methode display() aan.
- Als het goed is verschijnt er een vierkant op het scherm!
- Breid dit uit naar eigen wens!

•

•