Objecten en klassen 1

1.1 Objectgeoriënteerde taal

Als je een computerprogramma schrijft, doe je dat omdat de rekenkracht van de computer van pas komt bij het oplossen van een probleem in jouw leefwereld. Je kunt niet zomaar een e-mail schrijven naar je computer met de vraag om het probleem op te lossen. Je zal op een welbepaalde manier je probleem en je oplossingsstrategie moeten meedelen aan de computer. Anders gezegd, je zal de computer moeten programmeren.

Een computerprogramma programmeren doe je best niet in een teksteditor, zoals Kladblok of Pages. Je kiest eerst een programmeertaal en een gepaste ontwikkelomgeving (IDE of Integrated Development Environment). De keuze van beide is niet alleen afhankelijk van het probleem, maar ook van je eigen kennis en vaardigheden. Er spelen nog andere factoren waar hier niet dieper op ingegaan wordt.

Dit boek kiest voor de objectgeoriënteerde programmeertaal Java. De keuze ligt voor de hand: Java is een veel gebruikte programmeertaal en objectgeoriënteerd programmeren (OOP of object-oriented programming) is een veelgebruikte programmeertechniek.

Is het dan erg dat jouw vriend(in) leert programmeren in een andere programmeertaal? Neen, elke programmeertaal heeft zijn voor- en nadelen. Wanneer je een programmeertaal onder de knie hebt, is het leren van een tweede programmeertaal vrij eenvoudig. Bevindt de programmeertaal zich niet bij de objectgeoriënteerde programmeertalen, dan zal je wel geconfronteerd worden met andere concepten.

Je zou jouw eerste programmeerstappen kunnen zetten in een professionele ontwikkelomgeving zoals Eclipse of IntelliJ en je beperken tot eenvoudige programma's. Dit boek maakt deze keuze niet. In plaats van een bedrijfsklaar softwarepakket te programmeren, ga je de concepten van objectgeoriënteerd programmeren met behulp van *BlueJ* bestuderen.

BlueJ is speciaal ontwikkeld om objectgeoriënteerd te leren programmeren. Het is belangrijk te weten dat je eerst leert hoe de programmeercode van een programma werkt. Een professioneel programma programmeren, komt ongetwijfeld aan bod in een vervolgcursus.

1.1.1 Object

De kern van objectgeoriënteerd programmeren wordt gevormd door twee basisconcepten. Het eerste basisconcept is *object*.

Concept – Object. Objecten fungeren als model voor de elementen waarin een probleem kan worden opgesplitst.

In een objectgeoriënteerde programmeertaal probeer je op de computer een model te maken van een klein stukje van de werkelijkheid. Om dit goed te begrijpen, bestudeer je eerst een afbeelding van het spelletje EA SPORTS FIFA 23.

Figuur 1.1: EA SPORTS FIFA 23

Indien je een voetbalwedstrijd wilt vertalen naar een computerspelletje, dan moet je een model maken van de wedstrijd. Wat je ziet, splits je op in kleine entiteiten die makkelijk te beschrijven zijn. Kijk je alleen nog maar naar de gegeven afbeelding, dan merk je heel wat objecten op:

- de speler in het lichtblauwe shirt
- de speler met donkerblauwe shirt
- de bal
- het voetbalveld

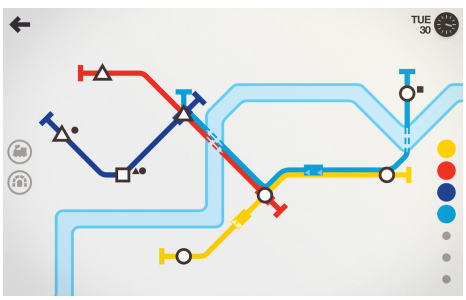
Hoe gedetailleerd de werkelijkheid gemodelleerd wordt, hangt af van het probleem.

1.1.2 Klasse

De objecten uit sectie 1.1.1 kunnen ingedeeld worden in categorieën. Spelers van het witte team of van het oranje team zijn eigenlijk, op de kleur van hun shirt na, allemaal spelers. Alle objecten van een bepaald type worden in een objectgeoriënteerde programmeertaal beschreven door een *klasse*. Een *klasse* is het tweede basisconcept van objectgeoriënteerd programmeren.

Concept – Klasse. Objecten worden gemaakt uit klassen. De klasse beschrijft de aard van een object; de objecten zelf zijn opzichzelfstaande exemplaren van de klasse.

Een voorbeeld om dit abstract begrip te verduidelijken. De afbeelding komt uit het computerspel *Mini Metro* van Dinosaur Polo Club. Bij dit spel is het de bedoeling dat rechthoekig gekleurde metrostellen zwarte blokjes of reizigers oppikken aan een wit metrostation. Reizigers worden afgezet aan het eerstvolgende metrostation met een zelfde vorm. Zo moeten vierkante reizigers steeds afgezet worden aan vierkante metrostations. Elk metrostel rijdt heen en weer



Figuur 1.2: Mini Metro

op een metrolijn van een bepaalde kleur en probeert zo veel mogelijk reizigers op te pikken en af te zetten aan een passend metrostation. Soms worden reizigers afgezet op een knooppunt omdat er op een metrolijn geen gepast metrostation te vinden is. Op gezette tijdstippen komen er op het metronetwerk reizigers en stations bij en kun je metrolijnen uitbreiden of nieuwe lijnen bouwen. Hoe meer reizigers een juiste bestemming bereiken, hoe meer punten je kunt verdienen. Dit duurt tot het metronetwerk de toevloed aan reizigers niet meer kan slikken.

Veronderstel dat je alle objecten op de metrokaart moet programmeren. Een van de problemen die je dan moet oplossen, is het simuleren van alle metrolijnen. Maar wat is nu een metrolijn voor dit probleem: is het een klasse of een object?

Door jezelf een aantal vragen te stellen, kun je dit probleem oplossen.

- Welke kleur heeft de metrolijn?
- Hoeveel metrostellen rijden op de metrolijn?
- Is er een vierkant station op de lijn?

Je kunt deze vragen alleen beantwoorden als je het over één bepaalde metrolijn hebt. Bijvoorbeeld, nee er is geen vierkant metrostation op de blauwe lijn. In dit geval spreekt men over objecten.

Wil je beschrijven welke eigenschappen een `Metrolijn` bezit of wat een `Metrolijn` allemaal moet kunnen dan spreek je over een klasse.

- Metrolijnen hebben een kleur.
- Metrolijnen verbinden stations.
- Op metrolijnen kunnen metrostellen rijden.

Oefening 1.1 Bij de afbeelding van het spelletje FIFA 23 werden heel wat objecten benoemd. Kun je ook verschillende klassen opsommen waartoe deze objecten behoren?

1.2 Java in BlueJ

In sectie 1.1 werd het verschil uitgelegd tussen een object en een klasse. Maar wat ga je leren programmeren: een klasse, een object of beide?

BlueJ – Super Mario. Open vanuit *BlueJ* het project *SuperMario*. Je selecteert in de map met bronprojecten van hoofdstuk 1, de map *SuperMario*.

1.2.1 Compileren

Wanneer je een computerprogramma programmeert in een IDE, doe je dat in een voor de mens leesbare programmeercode. De computer kan geen leesbare programmeercode uitvoeren. Om programmeercode te vertalen naar computertaal of machinecode gebruik je een programma: de compiler.

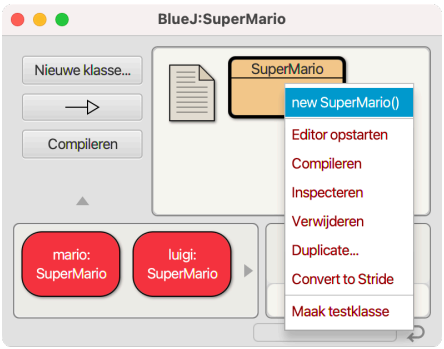
Concept – Compiler. Een compiler is een programma dat leesbare programmeercode omzet in machinecode dat kan uitgevoerd worden door een computer.

Telkens wanneer je de programmeercode of broncode van een programma wijzigt, moet je compileren. In *BlueJ* klik je dan op de knop *Compileren*. Ook wanneer je een project opent in *BlueJ*, klik je eerst op de knop *Compileren*.

1.2.2 Klassen en objecten in BlueJ

Bestudeer in figuur 1.3 een afbeelding van het project *SuperMario* in *BlueJ*. Onmiddellijk valt de oranje rechthoek met de tekst `SuperMario` op. *BlueJ* stelt een klasse voor met een oranje rechthoek. Wanneer deze rechthoek gearceerd is, klik je op de knop *Compileren*.

Met de rechtermuisknop klikken op de oranje rechthoek of klasse geeft o.a. de optie `new SuperMario()`. Klik je hierop dan geeft een pop-upvenster je de mogelijkheid een naam in te tikken, bv. `mario`. Links onderaan zie je nu een rode rechthoek verschijnen met de tekst `mario: SuperMario`.

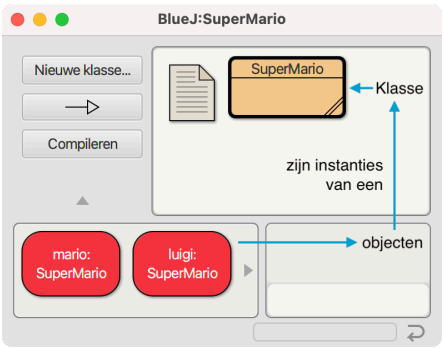


Figuur 1.3: Klassen en objecten in BlueJ

1.2.3 Instantie

Wat heb je nu eigenlijk gedaan? De opdracht `new Supermario()` laat uit de klasse `SuperMario` een object geboren worden. Via de pop-up heb je dat object de naam `mario` gegeven.

Merk op dat je van de klasse `SuperMario` veel objecten kunt maken. In het voorbeeld werden twee instanties van de klasse `SuperMario` gemaakt. Het zijn twee `SuperMario`-objecten met *BlueJ*-namen `mario` en `luigi`. Ze verschijnen allebei links onderaan in een rode rechthoek in de *Objectenbank* van *BlueJ*.



Figuur 1.4: Klasse, object en instantie

Als men het over objecten heeft, zal men vaak spreken van een instantie. Deze twee begrippen zijn bijna synoniemen.

Concept – Instantie. Objecten noemen je instanties wanneer je wilt benadrukken dat ze van een bepaalde klasse zijn.

1.2.4 Gedrag van een object

Klik je met de rechtermuisknop op een van deze objecten, dan zie je dat je een object een aantal opdrachten kunt geven. Met `void links()` en `void rechts()` kun je een object verplaatsen. Met `void spring()` kun je omhoog springen.

Concept – Gedrag. Objecten hebben een gedrag. Het gedrag wordt aangeduid door de opdrachten die je aan het object kunt geven.

1.2.5 Toestand van een object

Dubbelklik je op een object dat je aangemaakt hebt, dan krijg je alle waarden te zien die het object bijhoudt. Dit venster heet in *BlueJ* het *toestandsvenster* van een object. Wanneer je het object inspecteert, dan bekijk je de toestand van het object.

Concept – Toestand. Objecten hebben een toestand. De toestand wordt aangeduid door de waarden die het object bijhoudt.

Je moet nu ongetwijfeld het gevoel hebben dat deze drie objecten 'leven'. Jammer genoeg niet zoals op een Nintendo, maar je bent voorlopig nog geen professional.

Oefening 1.2 Maak een object aan met *BlueJ*-naam `yoshi`.

a) Bestudeer de toestand van object `yoshi`. Welke waarden houdt het object bij?
b) Bestudeer het gedrag van object `yoshi`. Laat het object tweemaal naar rechts bewegen, vervolgens éénmaal springen en ten slotte éénmaal naar links bewegen.
c) Bestudeer de nieuwe toestand van het object. Doet het object wat je verwacht?

BlueJ – Werking. Hoe zie je nu in de programmeeromgeving *BlueJ* een klasse, object, instantie en toestand?

- Een gele rechthoek stelt een klasse voor.
- Een rode rechthoek stelt een object voor. Je leest af van welke klasse dit object een instantie is.
- Objecten krijgen van *BlueJ* een naam.
- De toestand van een object inspecteren, is nagaan hoe het zit met de informatie die door het object wordt bijgehouden. Dit doe je door te dubbelklikken op een object in de *Objectenbank*.

1.3 Klassendefinitie

Waar zit nu de programmeercode die voor al dit 'leven' zorgt? Dubbelklik op de gele rechthoek of de klasse `SuperMario`. Je ziet nu de *Java*-code verschijnen van al deze objecten. Als je gaat programmeren is het dus erg belangrijk dat je inziet dat je een klasse zal programmeren. Je programmeert nooit een object.

```
1  public class SuperMario
2  {
3      private int x; // x-coordinaat van het object op het scherm
4      private int y; // y-coordinaat van het object op het scherm
5
6      /**
7       * Maakt een object aan van de Klasse SuperMario.
8       */
9      public SuperMario()
10     {
11         x = 0;
12         y = 0;
13     }
14
15     /**
16      * Geeft de x-coordinaat op het scherm terug.
17      * @return de x-coordinaat
18      */
19     public int getX()
```

```
20      {
21          return x;
22      }
23
24      /**
25       * Geeft het object een nieuwe y-coordinaat.
26       * @param   y   de nieuwe y-coordinaat.
27       */
28      public void setY(int y)
29      {
30          this.y = y;
31      }
32
33      /**
34       * Verplaatst het object een plaats naar links.
35       */
36      public void links()
37      {
38          x--;
39      }
40
41      /**
42       * Verplaatst het object een plaats naar rechts.
43       */
44      public void rechts()
45      {
46          x++;
47      }
48
49      /**
50       * Verplaatst het object twee plaatsen omhoog.
51       */
52      public void spring()
53      {
54          y += 2;
55      }
56  }
```

Codefragment 1.1: Klassendefinitie van de klasse `SuperMario`

Wanneer je via het bestandssysteem van je computer naar de projectmap van het project *SuperMario* navigeert, vind je twee belangrijke bestanden.

- *.java*-bestand: bevat de *Java*-code uit het codefragment hierboven.
- *.class*-bestand: hierin staat de voor ons onleesbare code die de computer kan uitvoeren. Dit bestand wordt aangemaakt telkens je op knop *Compileren* klikt. Wanneer je jouw programmeercode wijzigt, moet je steeds compileren zodat de computer de meest recente versie van je programma kan uitvoeren.

BlueJ – Geldautomaat. Open het project *Geldautomaat* en bestudeer wat je allemaal kunt met een instantie van de klasse `Geldautomaat`. Hiervoor hoef je de code helemaal niet te bestuderen!

De klasse `Geldautomaat` is een heel eenvoudige klasse. Je merkt het als je in *BlueJ* een instantie van deze klasse test. Toch bevat deze klasse alle elementen die je in het volgende hoofdstuk zal bestuderen om dan zelf een klasse te kunnen programmeren.

- Er is een stukje code dat ervoor zorgt dat men van een klasse een instantie kan maken.
- Het toestandsvenster laat zien dat een object een waarde bijhoudt. Hier is de waarde het geldbedrag dat reeds in de geldautomaat geworpen werd.
- Rechtsklikken op een object toont een menu met verschillende methoden om iets te veranderen aan de toestand van het object of om iets te vragen aan het object:
  - `void werpMunstukInGeldautomaat(double muntstuk)`
    Een methode om het bedrag in de geldautomaat met een gegeven bedrag te vermeerderen.
  - `double resetGeldautomaat()`
    Een methode om de ingeworpen muntstukken uit de geldautomaat te halen.
  - `double getIngeworpenBedrag()`
    Een methode om te vragen welk geldbedrag al ingeworpen werd in de geldautomaat.

Concept – Klassendefinitie. In een klassendefinitie zal je haarfijn definiëren hoe alle instanties van een klasse zich moeten gedragen: individueel of ten opzichte van instanties van dezelfde klasse of een andere klasse.

Sluit nu het project *Geldautomaat*. Wanneer je de editor sluit, worden alle wijzigingen aan de *Java*-code automatisch bewaard. Wanneer je de volgende keer het project opent, zal de rechthoek waarin de naam van de klasse `Geldautomaat` staat, gearceerd zijn. Dit betekent dat je eerst nog moet compileren alvorens je een object kunt aanmaken.

Oefening 1.3 Stel dat men jou vraagt een voetbalspeler uit FIFA 23 te programmeren en je op het idee komt de klasse `Voetbalspeler` te programmeren.

a) Welke waarden moet een voetbalspeler kunnen bijhouden?
b) Welke informatie moet je kunnen vragen aan een voetbalspeler?
c) Wat moet een voetbalspeler allemaal kunnen doen?

In hoofdstuk 2 ga je stap voor stap alle elementen die een klasse definiëren, bestuderen en leren programmeren zodat je zelf in staat bent een klassendefinitie te programmeren zoals de klasse `SuperMario` in codefragment 1.1.

1.4 Herhalingsoefeningen

Oefening 1.4 – Polonaise In het grote feestwoordenboek staat bij polonaise de volgende definitie:

Een sliert feestvierende personen, elk met de handen op de schouder van de volgende persoon of met de handen zwaaiend, die luidruchtig door de kamer of zaal bewegen.

Figuur 1.5: Polonaise dans

Schrap wat niet past in de uitspraken over bovenstaande afbeelding:

a) Je ziet twee ***objecten / klassen*** van ***de klasse / het object*** `Polonaise`.

b) Een instantie van ***de klasse / het object*** `Polonaise` bevat op haar beurt verschillende ***objecten / klassen*** van ***de klasse / het object*** `Feestneus`.

c) In ***de klasse / het object*** `Polonaise` programmeert men het best de mogelijkheid om meerdere ***objecten / klassen*** van ***de klasse / het object*** `Feestneus` bij te houden.

(Naar een oefening van Goderik Lefebvre, Waregem)

Oefening 1.5 – Aquarium Wanda en Rik zijn twee vissen die gelukkig rondzwemmen in een aquariaum.

Figuur 1.6: Aquarium

Zijn onderstaande uitspraken waar of vals?

a) Op de afbeelding staan drie objecten van de klasse `Vis`.
b) Wanda is een instantie van de klasse `Aquarium`.
c) Op de afbeelding staat juist één instantie van de klasse `Aquarium`.

(Naar een oefening van Goderik Lefebvre, Waregem)

Oefening 1.6 – Schaakstuk Een veel gespeeld online spelletje is schaken.

Figuur 1.7: Schaakspel

Stel dat je de klasse `Schaakstuk` moet programmeren.

a) Welke waarden moet een instantie van de klasse `Schaakstuk` bijhouden?
b) Wat moet een instantie van de klasse `Schaakstuk` kunnen?

Oefening 1.7 – E-mailclient Open je favoriete e-mailclient.

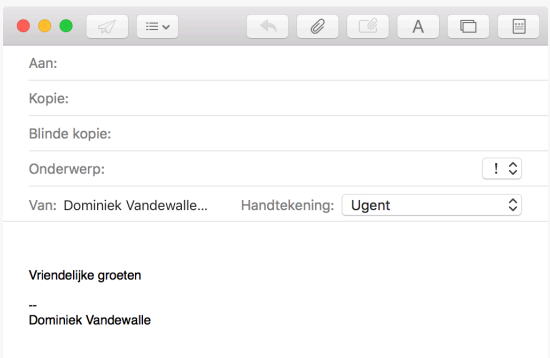
a) Welke objecten zijn een instantie van welke klasse?
b) Zijn er objecten die andere objecten verzamelen?

Oefening 1.8 – Super Mario Plus Open het project *SuperMarioPlus*. Dit project vind je in de map oefeningen bij hoofdstuk 1. Maak een object aan van deze klasse en bestudeer de werking en de toestand van het object.

a) Wanneer wordt een object 'groot'? En opnieuw 'klein'?
b) Kan een instantie van `SuperMarioPlus` door de grond zakken? Of schuin omhoog springen?

Oefening 1.9 – Mail Stel dat je de klasse `Mail` moet programmeren.

a) Welke waarden moet een e-mail kunnen bijhouden?
b) Welke informatie moet je allemaal opgeven aan een e-mail?
c) Welke informatie moet je kunnen opvragen van een e-mail?



Figuur 1.8: Mail van Apple Inc.

Index