Vereiste kennis

Dit boek richt zich op het leren programmeren door het oefenen met programmeercodes. Veel theorie komt in het begin niet aan de orde. Dat is een grote uitdaging want het is niet makkelijk om een heel klein uitvoerbaar programmaatje te schrijven over alleen een Java-onderwerp. In elk uitvoerbaar programma komen verschillende thema's voor. Alle programma's demonstreren een bepaald Java-onderwerp. Dit boek verdeelt de Java-programmeertaal in een aantal thema's. In elk hoofdstuk wordt een aantal kleine, uitvoerbare programma's aangeboden om dat specifieke onderwerp te demonstreren.

Om dit mogelijk te maken moet je als student de volgende programmeerbegrippen kennen. Het gaat er niet om dat je alle punten hieronder goed begrijpt, want details worden later behandeld in dit boek. Het enige wat je moet leren is wat de rol van deze punten is om een programma te kunnen compileren en uitvoeren.

Elk hoofdstuk van dit boek begint met een beknopte uitleg, waarna de quizzen volgen. De programmacodes zijn compleet en je kunt die testen, compileren en uitvoeren.

1 Java-editor

We gebruiken als Java-editor Eclipse inclusief JDK 7 (Java Development Kit) of hoger. Eclipse is een gratis Java-editor. Je kunt Eclipse vinden op en downloaden via internet. Bij de JDK worden tools meegeleverd, die nodig zijn bij het bewerken van Java-programma's. De broncodes van Java zijn bestanden met de extensie *.java*. De gecompileerde bytecode-bestanden hebben de extensie *.class*. Java is een platformonafhankelijke programmeertaal, daarom kun je de gecompileerde programma's op elk besturingssysteem uitvoeren.

2 Het compileren van een programma

Het compileren is het vertalen van broncode in machinetaal met behulp van een hulpprogramma (compiler). Na het compileren van een broncode is het resulterende programma direct uitvoerbaar (executable).

3 Java-klassen en interfaces

Elk Java-programma bestaat uit klassen en interfaces. Deze twee begrippen worden in het boek in detail behandeld. Het enige wat je voorlopig moet leren is dat een klasse in Java begint met een statement als `class MijnKlasse`. `MijnKlasse` is de naam van de klasse en je kunt zelf een naam verzinnen. Elke klasse in Java wordt opgeslagen in een bestand met de naam van de klasse. De klasse `MijnKlasse` moet dus opgeslagen worden in een bestand met de naam van de klasse en de extensie *.java*. De naam van het bestand in ons voorbeeld wordt dus *MijnKlasse.java*. Elke klassennaam begint met het woord `class` en elke interfacenaam begint met het woord `interface`.
Klassen en interfaces hebben ook leden zoals variabelen en methoden, die staan tussen accolades.

Voorbeeld 1

```
class MijnKlasse
{
  // code
}
```

De volgende interface moet ook opgeslagen worden in een bestand met de naam *MijnInterface.java*.

Voorbeeld 2

```
interface MijnInterface
{
  // code
}
```

4 Statements

Statements in Java zijn vergelijkbaar met zinnen in natuurlijke talen en dit zijn uitvoerbare eenheden. Een statement wordt meestal afgesloten met een puntkomma (;).

5 Blokcode in Java

Code in Java staat binnen een begin- en eindaccolade. Dit wordt blokcode genoemd. Hieronder enkele voorbeelden van blokcodes.

Bloktype	Blokcodes

Klasse	`class MijnKlasse`
	`{`
	`  // code`
	`}`

Methode	`void mijnMethode`
	`{`
	`  // code`
	`}`
	
	Het sleutelwoord `void` betekent dat de methode geen waarde oplevert.

Conditionele statements	`if(x > 3)`
	`{`
	`  // code`
	`}`

Iteratiestatements	`for( int i=0; i<5; i ++)`
	`{`
	`  // code`
	`}`

6 De main-methode

De `main`-methode is een methode die nodig is om een Java-programma uit te kunnen voeren. Elk Java-programma moet een `main`-methode hebben, die begint als volgt:

```
public static void main(String [] args)
```

Voorlopig is het belangrijk om alleen het volgende te leren over `main`-methode. Alle uitvoerbare programma's moeten voorzien zijn van een `main`-methode zoals hieronder staat. Het programma voert de statements binnen de `main`-methode uit. Het uitvoeren van de statements gebeurt van boven naar beneden. In het volgende voorbeeld wordt statement 1 als eerste uitgevoerd dan statement 2 en als laatste statement 3.

Voorbeeld 3

```
public class MijnKlasse
{
  // statements;
  // mainmethode
  public static void main(String[] args)
```

```
  {
    // statement 1;
    // statement 2;
    // statement 3;
  }
}
```

7 Schrijven van waarden van variabelen en teksten naar de standaarduitvoer

Het statement `System.out.println();` is belangrijk om je programma te kunnen testen. We behandelen geen details, maar het is wel belangrijk om te weten wat je hiermee kunt doen. Met dit statement kun je een waarde van een variabele of een tekst schrijven naar de standaarduitvoer. We maken van het vorige voorbeeld uitvoerbare code.

Voorbeeld 4

```
public class MijnKlasse
{
  public static void main(String[] args)
  {
    int x1 = 25; // een geheel getal variabele x1
    int x2 = 99; // een geheel getal variabele x2
    //statement 1 schrijft de waarde van x1 naar de standaarduitvoer
    System.out.println(x1);
    //statement 2 schrijft de waarde van x2 naar de standaarduitvoer
    System.out.println(x2);
    //statement 3 schrijft de volgende tekst naar de standaarduitvoer
    System.out.println("Mijn naam is Emma");
    //statement 4 schrijft een combinatie van een tekst en
    //een variabele naar de standaarduitvoer
    System.out.println("Leeftijd: " + x1 + " jaar");
  }
}
```

Als je het vorige programma compileert en uitvoert, wordt het volgende geschreven naar de standaarduitvoer.

```
25
99
Mijn naam is Emma
Leeftijd: 25 jaar
```

Om tekst te schrijven naar de standaarduitvoer moet je die tussen dubbele aanhalingstekens zetten, maar voor de waarde van de variabelen is dat niet nodig, zie statement 1, 2 en 3 in het vorige voorbeeld.
Om een combinatie van tekst en een variabele naar de standaarduitvoer te schrijven, moet je een plusteken (+) gebruiken, zie statement 4.
Om variabelen en tekst op één regel te schrijven naar de standaarduitvoer moet je `print` gebruiken in plaats van `println`.

8 Commentaar

Commentaar wordt genegeerd door de compiler. Hieronder worden twee manieren beschreven hoe je commentaar kunt toevoegen.

1 Commentaar van een enkele regel begint met twee schuine strepen //.
Alles wat rechts van deze tekens staat, ziet Java als commentaar:

```
// commentaar van een enkele regel
```

2 Commentaar van meerdere regels begint met /* en eindigt met */. Alles wat tussen deze twee tekens staat, ziet Java als commentaar:

```
/* hier begint een commentaar van meerdere regels
   en dat eindigt hierna */
```

9 Sleutelwoorden

Er zijn belangrijke sleutelwoorden in Java die later in dit boek worden behandeld. Om de quizcodes zo eenvoudig mogelijk te houden, moeten we de sleutelwoorden `public` en `static` gebruiken. Deze twee sleutelwoorden worden later in detail uitgelegd.

Het sleutelwoord static

Het begrip `static` is heel belangrijk in Java en het wordt behandeld in een apart hoofdstuk. Het enige wat je nu moet leren is dat dit sleutelwoord helpt om kleine uitvoerbare programma's te schrijven. Daarom komt het in sommige quizcodes vóór de naam van de variabelen voor. Om het idee te demonstreren gebruiken we het volgende voorbeeld.

Voorbeeld 5

Wat gebeurt er als het volgende programma wordt gecompileerd en uitgevoerd?

```
public class MijnKlasse
{
  //static sleutelwoord voor de variabelenaam
  static int x = 4;
  // de mainmethode om het programma uit te voeren
  public static void main(String[] args)
  {
    // statement 1: schrijft de waarde van de variabele x
    System.out.print(x);
    // statement 2: schrijft de tekst "Mijn Java code"
    System.out.print(" Mijn Java code");
  }
}
```

Het vorige programma schrijft `4 Mijn Java-code` naar de standaarduitvoer.

Als je `static` verwijdert in het vorige voorbeeld krijg je de volgende foutmelding:
`Cannot make a static reference to the non-static field x.`

Om dit probleem op te lossen, moeten we een object creëren. Er wordt pas in hoofdstuk 5 uitgelegd hoe je een object maakt, daarom wordt de variabele `x` in het programma als `static` gedeclareerd. Dit trucje helpt om het maken van objecten in de eerste fase te vermijden. Hiermee kunnen we het programma uitvoerbaar en klein houden en ons meer focusen op het onderwerp dat ter sprake komt.

Het eerste statement in de `main`-methode schrijft de waarde van de variabele `x` naar de standaarduitvoer.
Het tweede statement schrijft de tekst `Mijn Java-code` naar de standaarduitvoer.

Vergeet niet dat we hier `print` hebben gebruikt in plaats van `println`, daarom worden de waarden van de variabele en de tekst op één regel geschreven naar de standaarduitvoer.

Het sleutelwoord public

Java ondersteunt toegankelijkheid van de klassen en de klassenleden met behulp van speciale sleutelwoorden. Het sleutelwoord `public` voor de naam van klassen of klassenleden (variabelen en methoden) geeft aan dat de klassen of de leden toegankelijk zijn vanaf elke andere klasse. Dit wordt later in een apart hoofdstuk behandeld.

10 Standaard-API (Application Programming Interface)

Java biedt veel code die gebruikt kan worden door programmeurs. Je kunt altijd van deze rijke bron gebruikmaken. Sommige opdrachten in dit boek behandelen het gebruik van deze standaard-API.
Je vindt Java-standaard-API hier: https://docs.oracle.com/javase/8/docs/api/

11 Escape-sequenties (escape sequences)

Een escape-sequentie is een teken voorafgegaan door een backslash. Deze tekens hebben een speciale betekenis voor de compiler.
Als je bijvoorbeeld een dubbel aanhalingsteken binnen een dubbel aanhalingsteken zet, moet je gebruikmaken van escape-sequenties. Zie het volgende voorbeeld.
Als je de zin “Hij zegt: “Ik ga naar Amsterdam”.” wilt schrijven naar de standaarduitvoer, moet je dat als volgt doen.

```
System.out.print("Hij zegt: \"Ik ga naar Amsterdam\".");
```

Voorbeeld 6

```
public class MijnKlasse
{
  public static void main(String[] args)
  {
    {
      System.out.println("Apostrof              : " + "abcde\'fghij");
      System.out.println("Dubbel aanhalingsteken: " + "abcde\"fghij");
      System.out.println("Backslash             : " + "abcde\\fghij");
      System.out.println("Nieuwe regel          : " + "abcde\nfghij");
      System.out.println("Tab                   : " + "abcde\tfghij");
      System.out.print("Er stond duidelijk \"verboden te parkeren\".");
    }
  }
}
```

Als deze code wordt uitgevoerd, wordt het volgende geschreven naar de standaard-uitvoer.

```
Apostrof              : abcde'fghij
Dubbel aanhalingsteken: abcde"fghij
Backslash             : abcde\fghij
Nieuwe regel          : abcde
fghij
Tab                   : abcde      fghij
Er stond duidelijk "verboden te parkeren".
```

De tabel van de escape-sequenties

Escape-sequentie	Omschrijving
\'	Apostrof
\"	Dubbel aanhalingsteken
\\	Backslash
\b	Laatste teken wissen (backspace)
\n	Nieuwe regel (new line)
\r	Nieuwe regel (carriage return)
\t	Tabulator
\f	Plaats een formfeed in de tekst op dit punt

Een decimaal getal gescheiden met een komma

In de quizzen en andere codes in dit boek wordt standaard tussen hele getallen en decimalen een punt gebruikt, dus € 7.31 is 7 euro en 31 cent. Dit is het gebruik in Engelstalige landen, normaal wordt in Nederland hier een komma gebruikt.

Java ondersteunt verschillende manieren om de punt te veranderen in een komma en het aantal getallen achter de komma te bepalen, maar dat wordt niet gebruikt in dit boek. Dit is gedaan om de code overzichtelijk en het aantal coderegels beperkt te houden.