



Java traineeship

De basis



Leerdoelen

- Installeren prerequisites
- Het herkennen van de structuur van een .java klasse.
- Het bouwen van je eerste applicatie
- Commentaar leren schrijven in je code
- Informatie printen naar de terminal
- Variabelen leren gebruiken
- Verschillende typen leren gebruiken
- Conditionele logica gebruiken
- Loops toepassen
- In een correcte stijl programmeren
- Methodes gebruiken

Java

Java is de programmeertaal die centraal staat tijdens dit traineeship.



→ Het **doel** is het leren programmeren. Java is het **middel**.

- Objectgeoriënteerd
- Verschenen in 1995
- C-achtige syntax
- Platformonafhankelijk
- Vol nuttige programmeerconcepten

Java installeren

- Download de JDK (Java Development Kit) installer via <https://www.oracle.com/java/technologies/downloads/>
- Volg de installatie
- Nadat de JDK is geïnstalleerd moeten we de omgevingsvariabelen van windows aanpassen
- Omgevingsvariabelen worden door programma's gebruikt om te vinden waar bepaalde binaries staan.
- Zonder juiste omgevingsvariabelen kan je terminal de JDK binaries niet vinden

Update omgevingsvariabelen

- Druk op de windows-toets en zoek naar "omgevingsvariabelen" (voor engelse systemen: "environment variables")
- Ga naar "omgevingsvariabelen aanpassen" of "edit the environment variables"
- Klik op "omgevingsvariabelen" of "environment variables"
- Onder het "systeemvariabelen" kopje, klik op edit, nieuw, en voeg de bin directory van je JDK installatie toe
- Deze ziet er ongeveer zo uit: "C:\Program files\Java\JDK-21\bin"

Eerste applicatie

Creëer een bestand genaamd **HelloWorld.java** door de volgende code over te nemen in je editor.

```
/*  
 * Prints Hello, World!  
 * Everyone's first program  
 */  
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

HelloWorld.java

De volgende code bevat drie componenten:

- **Commentaar**
- Definitie van de *HelloWorld* klasse
- Definitie van de *main*-methode

```
/*  
 * Prints Hello, World!  
 * Everyone's first program  
 */  
public class HelloWorld {  
    public static void main(String[] args)  
    {  
        System.out.println("Hello,  
World!");  
    }  
}
```

Commentaar

- De compiler **negeert** commentaar
- Commentaar is belangrijk (waarom denk je?)
- Commentaar tussen `/*` en `*/` mag **meerdere** regels beslaan
- Commentaar achter `//` loopt **tot het eind** van de regel

Opdracht: Voeg eens wat verschillende stijlen commentaar toe aan je HelloWorld programma en kijk of je code nog compileert

println vs. print

Om een tekst naar het scherm weg te schrijven kun je gebruikmaken van:
System.out.println() en **System.out.print()**

```
public class HelloWorld {  
    public static void main(String[] args)  
    {  
        System.out.println("Hallo  
Wereld!");  
        System.out.print("Hallo");  
        System.out.println("Wereld!");  
    }  
}
```

```
$ javac HelloWorld.java  
$ java HelloWorld  
Hallo Wereld!  
HalloWereld!
```

println vs. print

Je kunt ook zelf een nieuwe regel invoegen door `\n` te gebruiken

```
public class HelloWorld {  
    public static void main(String[] args)  
    {  
  
        System.out.println("Hallo\n\nWereld!");  
    }  
}
```

```
$ javac HelloWorld.java  
$ java HelloWorld  
Hallo
```

```
Wereld
```

Opdracht: test zowel print, println en `\n` uit in je HelloWorld programma.

Variabelen

Om gegeven op te slaan maken we gebruik van **variabelen**.

Variabelen hebben:

- Een type
 - Primitief datatype: **int**, **float**, **char**, **byte**, **long**, **short**, **boolean**
 - Niet-primitief datatype / referentie, bijvoorbeeld: **String**
- Een identifier (unieke naam)
- Een waarde

```
public class HelloWorld {  
    public static void main(String[] args) {  
        // declareer van de variabele a,  
        met als  
        // type int  
        int a;  
        // toekenning van de waarde 3 aan de  
        // variabele a  
        a = 3;  
        // ophoging van a met de waarde 1  
        a = a + 1;  
  
        String hallo = "Hallo Wereld";  
        System.out.println(hallo + a);  
    }  
}
```

Variabelen

```
public class PrintText {  
    public static void main(String[] args) {  
        System.out.println(5);  
        System.out.println(5 + 3);  
        System.out.println("Som is " + 5 + 3);  
        System.out.println("Som is " + (5 + 3));  
        System.out.println("Som van " + 5 + " en " + 3 + "  
is " + (5 + 3));  
    }  
}
```

```
$ java PrintText  
5  
8  
Som is 53  
Som is 8  
Som van 5 en 3 is 8
```

Namen van variabelen

- Hoofdlettergevoelig
- Beginnen met een letter
- Geen van de **literals**, **keywords** of **reserved words**

abstract	default	goto	package	this
assert	do	if	private	throw
boolean	double	implements	protected	throws
break	else	import	public	transient
byte	enum	instanceof	return	true
case	extends	int	short	try
catch	false	interface	static	void
char	final	long	strictfp	volatile
class	finally	native	super	while
const	float	new	switch	
continue	for	null	synchronized	

Integer-operaties in Java

```
public class IntMath {  
    public static void main(String[] args) {  
        int a = 13; b = 10;  
        int prod, quot, over;  
        prod = a * b;  
        quot = a / b;  
        over = a % b;  
        System.out.println(a + " * " + b + " = " + prod);  
        System.out.println(a + " / " + b + " = " + quot);  
        System.out.println(a + " % " + b + " = " + over);  
    }  
}
```

\$ java IntMath

13 * 10 = 130

13 / 10 = 1

13 % 10 = 3

Rekenkundige operaties met Integers

Java	Wiskundig	Integer voorbeeld	Resultaat
+	+	$3 + 4$	7
-	-	$3 - 4$	-1
*	x	$3 * 4$	12
/	÷	$3 / 4$	0
%	mod	$4 \% 3$	1

Rekenkundige operaties met Doubles

Java	Wiskundig	Double voorbeeld	Resultaat
+	+	3.1 + 4.2	7.300000000000001
-	-	3.0 - 4.0	-1.0
*	x	3.2 * 4.0	12.8
/	÷	3.0 / 4.0	0.75
%	mod	4.0 % 3.2	0.7999999999999998

Boolean datatype

- Waarde **true** of **false**
- Toekenning met **true**, **false** of een **boolean expressie**

```
boolean b = false;  
System.out.println("boolean b:" + b);  
boolean c = (2 > 0);  
System.out.println("boolean c:" + c);  
boolean d = (5 % 3 == 2);  
System.out.println("boolean d:" + d);
```

```
boolean b: false  
boolean c: true  
boolean d: true
```

Conditionele statements: if

Met behulp van een **if-statement** kun je code laten uitvoeren indien een bepaalde bewering **waar** is.

```
if (conditie) {  
    // Deze code wordt uitgevoerd als de expressie van de if-statement waar is  
}
```

Indien de bewering niet waar is, vervolgt de uitvoering van de code na het if-statement:

```
if (leeftijd > 18) {  
    System.out.println("Je mag de kroeg in");  
}
```

Conditionele statements: if

Opdracht:

- Maak een boolean `lichtIsAan`.
- Zet de waarde op *'true'* of *'false'*
- Creëer vervolgens een if-statement.
 - Als de variabele `lichtIsAan` waar is, print dan de tekst: "Het licht is aan."

Conditionele statements: if-else

- Meerdere gevallen?
- Met behulp van een **else-statement** is het mogelijk om een stuk code uit te laten voeren indien de bewering **niet** binnen de if valt.
- Als (if) bewering anders (else).

```
if (bewering) {  
    // uitgevoerd als bewering  
    true oplevert  
}  
else {  
    // uitgevoerd als bewering  
    false oplevert  
}
```

```
if (radius >= 0) {  
    double surface = Math.PI * radius * radius;  
    System.out.println("De oppervlakte van de cirkel: " +  
surface);  
}  
else {  
    System.out.println("De oppervlakte van de cirkel is  
ongeldig");  
}
```

Conditionele statements: if-else

Opdracht:

- Maak een boolean `lichtIsAan`.
- Zet de waarde op *'true'* of *'false'*.
- Creëer vervolgens een if-statement.
 - Als de variabele `lichtIsAan` waar is, print dan de tekst: "Het licht is aan."
 - Als de variabele `lichtIsAan` niet waar is, print dan de tekst: "Het licht is uit."

Conditionele statements: else if

Je kunt een **if**- en **else-statement** ook combineren tot een **if-else-statement**.

Let op: Indien je geen accolades plaatst, behoort enkel de eerstvolgende regel bij het if of else-statement.

```
if (aantalMensen < 50)
    System.out.println("Er zijn minder dan 50 mensen");
else if (aantalMensen < 100)
    System.out.println("Er zijn minstens 50 en minder dan 100 mensen");
else if (aantalMensen < 200)
    System.out.println("Er zijn minstens 100 en minder dan 200 mensen");
else
    System.out.println("Er zijn meer dan 100 mensen");
```

Nested If

Een **else-statement** bindt zich aan de dichtstbijzijnde **if-statement**

Deze else wordt dus gekoppeld aan de tweede if-statement, niet aan de eerste.

```
public class NestedIf {  
    public static void main(String[] args) {  
        int number = 7;  
        if (number >= 0)  
            if (number == 0)  
                System.out.println("eerste string");  
        else  
            System.out.println("tweede string");  
        System.out.println("derde string");  
    }  
}
```



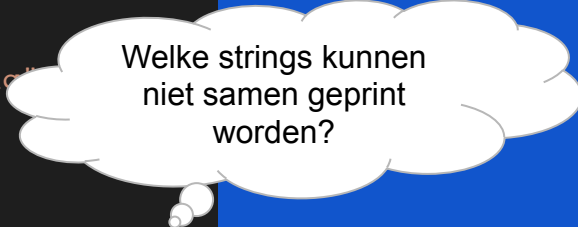
tweede string
derde string

Indentatie is belangrijk!

- Geeft structuur
- Correcte indentatie -> meer leesbaarheid!

```
public class NestedIf {  
    public static void main(String[] args) {  
        int number = 7;  
        if (number >= 0)  
            if (number == 0)  
                System.out.println("eerste string");  
        else  
            System.out.println("tweede string");  
        System.out.println("derde string");  
    }  
}
```

tweede string
derde string



Welke strings kunnen
niet samen geprint
worden?

Conditional statements: if-else

Opdracht:

- Schrijf een programma dat je vertelt of je auto mag rijden:
 - Ben je 18+ en heb je een rijbewijs? Dan mag je rijden.
 - Ben <18 en heb je een rijbewijs? Dan mag je rijden onder begeleiding.
 - Heb je geen rijbewijs? Dan mag je nooit rijden.

Schrijf hiervoor de juiste if/else-statements. Je mag 1 of meer variabelen gebruiken om bijvoorbeeld leeftijd in bij te houden.

De for-loop

Met behulp van loops kunnen we een stuk code meerdere keren laten uitvoeren.

De **for-loop** gebruik je als je van tevoren exact weet hoeveel iteraties je wilt gebruiken.

```
for (initialisatie; conditie; stap) {  
    // inhoud van de for-loop  
}
```

```
/*  
 * Bereken som van getallen 1 t/m 10  
 */  
  
int sum = 0;  
for (int i = 1; i <= 10; i++) {  
    sum += i;  
}  
  
System.out.println(sum);
```

Herhaling met de for-loop

```
/*  
 * Bereken som van getallen 1 t/m 10  
 */  
int sum = 0;  
for (int i = 1; i <= 10; i++) {  
    sum += i;  
}  
System.out.println(sum);
```

55

Herhaling met de for-loop vanaf 0

```
/*  
 * Vijf keer  
 */  
int k = 1;  
for (int i = 0; i < 5; i++) {  
    System.out.println(i + " " + k);  
    k = k * 2;  
}
```

```
0 1  
1 2  
2 4  
3 8  
4 16
```

De for-loop

Opdracht: print de onderstaande uitvoer met behulp van één for-loop

[moeilijk] Opdracht: Creëer het coördinatenstelsel van een schaakbord:

```
1 2 3
2 3 4
3 4 5
4 5 6
5 6 7
6 7 8
7 8 9
```

```
(8, 1)(8, 2)(8, 3)(8, 4)(8, 5)(8, 6)(8, 7)(8, 8)
(7, 1)(7, 2)(7, 3)(7, 4)(7, 5)(7, 6)(7, 7)(7, 8)
(6, 1)(6, 2)(6, 3)(6, 4)(6, 5)(6, 6)(6, 7)(6, 8)
(5, 1)(5, 2)(5, 3)(5, 4)(5, 5)(5, 6)(5, 7)(5, 8)
(4, 1)(4, 2)(4, 3)(4, 4)(4, 5)(4, 6)(4, 7)(4, 8)
(3, 1)(3, 2)(3, 3)(3, 4)(3, 5)(3, 6)(3, 7)(3, 8)
(2, 1)(2, 2)(2, 3)(2, 4)(2, 5)(2, 6)(2, 7)(2, 8)
(1, 1)(1, 2)(1, 3)(1, 4)(1, 5)(1, 6)(1, 7)(1, 8)
```

Klassedefinitie

```
/*  
 * Prints Hello, World!  
 * Everyone's first program  
 */  
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

```
class Name {  
    // ...  
}
```

Voorbeeld van een simpel programma met één klasse

Basis van een klasse-definitie

Public klasse

- Bestanden mogen meerdere klassen bevatten
 - Maar één klasse mag public zijn: hier wordt de file naar vernoemd.
 - Bijvoorbeeld: HelloWorld in HelloWorld.java
- Een bestand mag meer dan één klasse bevatten

```
/*  
 * Prints Hello, World!  
 * Everyone's first program  
 */  
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Methodes

- Zelfstandige, herbruikbare stukken code (met eigen variabelen)
- Methodes krijgen variabelen mee, dit noemen we de parameters van de functie
- Een functie eindigt (vaak) na een **return**-commando, waarmee een waarde teruggegeven wordt: de **return value**.

Methode definiëren

Structuur:

- Modifiers (bespreken we later)
- Return type: wat voor waarde krijgen we terug?
- Naam van de methode
- Parameters

```
modifiers returnType naam(parameters){  
    // .. code  
    // a = variabele van het type returnType  
    return a;  
}
```

Methode definiëren

Niet elke methode heeft een return-type: we gebruiken dan **void**. Soms willen we simpelweg commando's uitvoeren en hebben we geen terugkoppeling nodig.

```
// Functie die een getal teruggeeft
static int square(int getal) {
    return Math.pow(getal, 2);
}
```

```
// Functie zonder return type: void
public static void main(String[] args) {
    System.out.println("Hallo Wereld!");
}
```

Methode definiieren

```
public class FunctionProgramming {  
    public static void main(String[] args) {  
        int sum = Add(3, 4);  
        System.out.println(sum);  
    }  
  
    static int Add(int num1, int num2) {  
        return num1 + num2;  
    }  
}
```

Methodes: wanneer?

- Als je code bezig is met verschillende taken tegelijk
- Als je code aan het herhalen bent
- Als een stuk code onafhankelijk is van de boven- en onderstaande code.

Methodes: voordelen

- Je code wordt overzichtelijker,
- Je kunt stukken code makkelijker herbruiken
 - Binnen hetzelfde project
 - Of daarbuiten!
- En dus...
- Minder aanpassingen -> minder kans op fouten

Methodes

Opdracht

Maak twee methodes:

1. Een functie die een lamp aan zet
2. Een functie die een lamp uit zet

Bouw ook logica in je programma: een lamp die al aanstaat, kun je niet aanzetten.

[optioneel] Voeg de twee methodes samen om er een lichtsakelaar van te maken.

Scanner

- **Scanner** klasse kan lezen
- Bijvoorbeeld uit een bestand
- Of uit de terminal -> input van gebruiker!

Eerst importeren:

```
import java.util.Scanner;
```

Daarna kun je een scanner aanmaken:

```
Scanner scanner = new Scanner(System.in); // Scanner voor de terminal
```

Invoer integer met nextInt()

```
import java.util.Scanner;

public class ScannerInput {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Voer een getal in: ");
        int number = input.nextInt();
        System.out.println("U heeft getal " + number + " ingevoerd.");
    }
}
```


Stijl van programma's

Programma's moeten goed te lezen en onderhouden zijn.

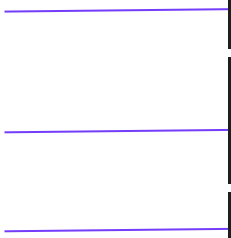
Richtlijnen:

1. Gebruik duidelijke **namen** (voor variabelen en meer)
2. Gebruik de juiste **haakjes en whitespace**
3. Schrijf altijd **commentaar** bij je programma
4. Let op de **indentatie**

1. Gebruik duidelijke namen

Constanten, types, variabelen, methoden, objecten & het programma zelf krijgen een **naam**.

- Namen van variabelen in **camelCase**
- Namen van constanten alleen in hoofdletters
- Namen van klassen in **PascalCase**



```
int aantalStudenten;  
int lineaireCoefficient;
```

```
static final int  
MAX_AANTAL_STUDENTEN;
```

```
class ComplexeBreuk
```

2. Gebruik goede haakjes en whitespace

We gebruiken accolades bij een **methode**, **for**, **while**, of **if**

- Tussen twee methoden laten we **één** regel leeg.
- Tussen declaratie variabelen en opdrachten laten we **één** regel leeg.

```
public static void main(String[] args) {  
    double x = 0.0;  
  
    for(initialisatie; conditie; stap) {  
        if (expressie) {  
            // ...;  
        }  
    }  
}
```

3. Commentaar bij je code

In sommige gevallen is commentaar overbodig.

Ter zake doende commentaar op de juiste plekken:

```
/*  
 * Som krijgt de waarde van de som van de  
getallen          */  
int sum = 0;  
for (int i = 0; i < number; i++) {  
    sum += i;  
}
```

```
/* Vraag de gebruiker een geheel getal tussen  
min  
 * en max met gebruik van de Scanner class  
 */  
int inputNumber(int min, int max){  
    // ...  
}
```

3. Commentaar bij je code: docblocks

```
/*  
 * Naam   : K. Klaassen  
 * Datum  : DD-MM-YYYY  
 *  
 * Voorbeeld.java:  
 * - Dit programma berekent de wortels van de vergelijking  
 *    $ax^2 + bx + c = 0$   
 *   Deze oplossing wordt berekend met de a-b-c formule  
 */  
public class Example {  
}
```

4. Let op indentatie

Code zonder indentatie is slecht leesbaar!

```
public class BubbleSort{
public static void main(String a[]){
int i;
int array[] = {12,9,4,99,120,1,3,10};
System.out.println("Values before the sort:\n");
for(i=0;i<array.length;i++){
System.out.print(array[i]+" ");
System.out.println();
bubble_srt(array,array.lenght);
System.out.println("Values after the sort:\n");
for(i=0;i<array.length;i++){
System.out.print(array[i]+" ");
System.out.println();
}
}
public static void bubble_srt(int a[], int n){
int i, j ,t =0;
for(i=0;i<n;i++){
for(j=0,j<(n-1);j++){
if(a[j-1]>a[j]){
t=a[j-1];
a[j-1]=a[j];
a[j]=t;
}
}
}
}
}
```

4. Let op indentatie

En hoe is dit?

```
public class BubbleSort{
    public static void main(String a[]){
        int i;
        int array[] = {12,9,4,99,120,1,3,10};
        System.out.println("Values before the sort:\n");
        for(i=0;i<array.length;i++){
            System.out.print(array[i]+" ");
            System.out.println();
            bubble_srt(array,array.lenght);
            System.out.println("Values after the sort:\n");
            for(i=0;i<array.length;i++){
                System.out.print(array[i]+" ");
                System.out.println();
            }
        }
    }

    public static void bubble_srt(int a[], int n){
        int i, j ,t =0;
        for(i=0;i<n;i++){
            for(j=0,j<(n-1);j++){
                if(a[j-1]>a[j]){
                    t=a[j-1];
                    a[j-1]=a[j];
                    a[j]=t;
                }
            }
        }
    }
}
```

Leerdoelen

- ✓ Het herkennen van de structuur van een .java klasse.
- ✓ Het bouwen van je eerste applicatie
- ✓ Commentaar leren schrijven in je code
- ✓ Informatie printen naar de terminal
- ✓ Variabelen leren gebruiken
- ✓ Verschillende typen leren gebruiken
- ✓ Conditionele logica gebruiken
- ✓ Loops toepassen
- ✓ In een correcte stijl programmeren
- ✓ Methodes gebruiken

Vragen?

- E-mail mij op joris.vanbreugel@code-cafe.nl!
- Join de CodeCafé-community op Discord!

