H1 Introductie Java-applicaties

Table of Contents

1. Doelstellingen

- kan een eenvoudige Java-applicatie opbouwen en begrijpt de achterliggende werking van het Java platform
- kan in- en uitvoerstatements schrijven
- kan primitieve datatypes gebruiken
- kan gebruikmaken van rekenkundige operatoren met voorrangsregels

2. Inleiding

- Java is één van de meest gebruikte programmeertalen. Zie TIOBE.
- Draait op 15 biljoen devices
- Java ondersteunt
  - Object georiënteerd programmeren
  - Generiek programmeren
  - Functioneel programmeren
- Java heeft reeds veel releases gekend en wordt nog steeds up-to-date gehouden evolutie van JAVA

3. Ontstaansgeschiedenis Java

Java is ontworpen in **1991** door Sun Microsystems voor de programmering van elektronische apparatuur. Voor **1995** had bijna niemand ervan gehoord, vermits twee overeenkomsten met bedrijven op het laatste moment niet doorgingen. In **1993** kwam MOSAIC uit, de eerste grafische browser waarmee je afbeeldingen tussen de tekst kon plaatsen Bijgevolg herschreven de ontwerpers de compiler (C→Java) in **1994** en bouwden ze een op Java gebaseerde browser, Webrunner, om Java te kunnen positioneren als een taal voor het internet. Dit bleek samen met de **gratis verspreiding** een succesverhaal te zijn!

`Hoe heeft men gerealiseerd dat Java overzetbare (of platformonafhankelijke) programma□s kan maken?`

Java (oorspronkelijke naam OAK was reeds gebruikt) is in 1991 ontworpen voor de programmering van allerlei intelligente elektronische gebruikersapparatuur.

Elektronische apparatuur bevat **chips**.

Een chip kan

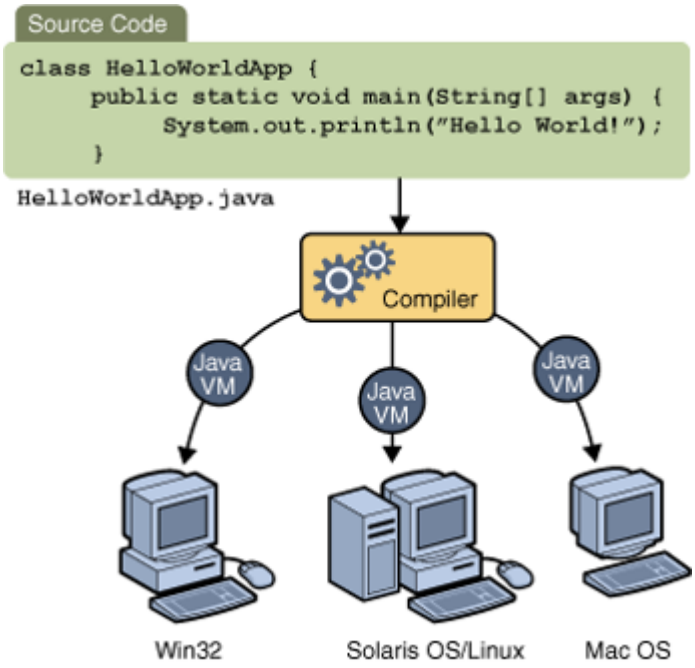
- gegevens onthouden
- een reeks van instructies automatisch uitvoeren = **programma**

Een programma wordt uitgevoerd door een speciale chip, **de processor**.

Gezien een processor enkel zijn eigen specifieke machinecode begrijpt, die onleesbaar is voor de mens (0 en 1), moeten we een programma schrijven in een **hogere programmeertaal** (Cobol, Java, Pascal, …). Een **compiler** (vertaler) zal het programma omzetten naar een specifieke machinecode. Door deze vertaalslag zal het programma **platformafhankelijk** zijn (platform = combinatie van een bepaald type processor met een bepaald besturingssysteem).

Oplossing die men in Java toepast:

1. Elk Java-programma wordt door een compiler vertaald naar een soort **TUSSENTAAL (=JAVA-bytecode)**, die betrekkelijk dicht tegen machinecode aanleunt.
2. De tussentaal wordt begrijpelijk gemaakt voor een specifieke processor, door deze van een programma te voorzien = **JAVA VIRTUAL MACHINE (JVM)**. Een JVM is een tamelijk klein programma (256 Kb), vermits het vertalen niet zo'n grote klus is want Java is een kleine taal (weinig keywords).



4. Java-applicaties

4.1. Eigenschappen

1. een applicatie is een Java-programma, cfr. programma's in C++ (vb. Word, tekenprogramma,…)
2. een applicatie wordt bewaard en uitgevoerd op lokale PC
3. een applicatie is uitvoerbaar op elke PC mits JVM (geen webbrowser nodig, zoals bij een JAVA-applet)
4. een applicatie kan bestanden op schijf maken, lezen en verwijderen

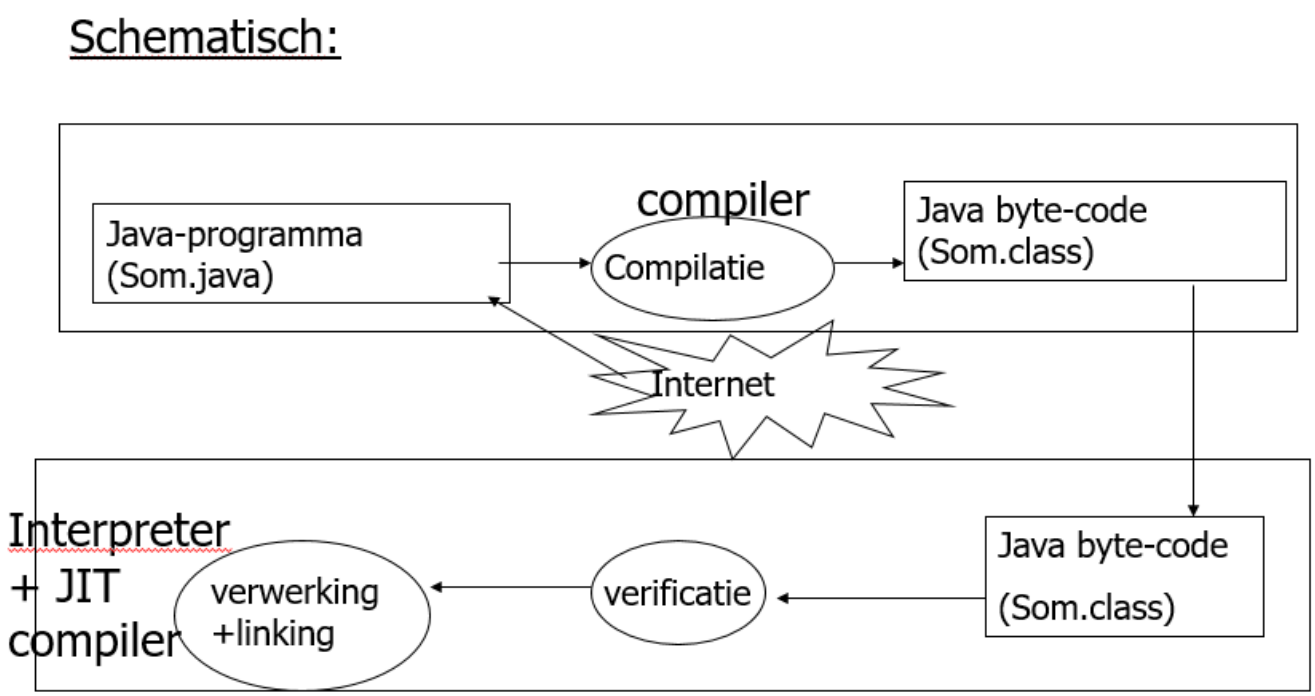
4.2. Eerste voorbeeld

```java
package cui;
//een project delen we op in packages, voorlopig enkel cui = console-user-interface

import java.lang.*;
//mag weggelaten worden, want wordt automatisch geïmporteerd

public class Som extends Object
// mag ook weggelaten worden gezien ELKE klasse in Java erft van Object
{
    public static void main(String[] args)
    {
        int x = 2, y = 3, som;
        som = x * x + y * y;
        System.out.println("Som = " + som);
    }
}
```

Uitvoer:

```
<terminated> Som [Java Application]
Som = 13
```

4.3. De 5 fasen

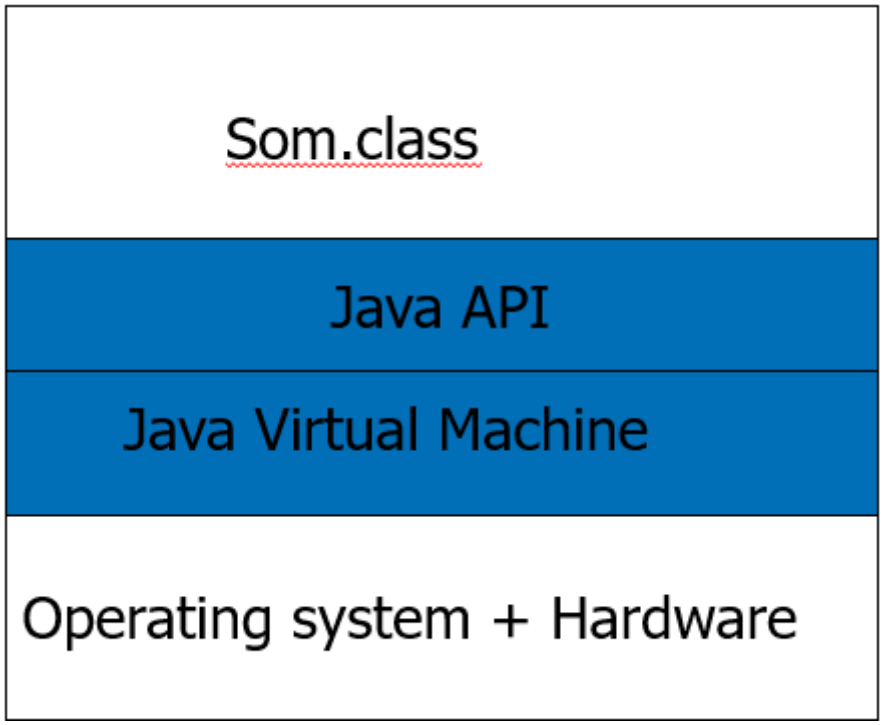


1. Met een editor het java-programma intypen en bewaren met extensie .java → **Som.java**
2. Het programma compileren met het commando javac. Het resultaat is een file met extensie .class (= bytecode). → **javac Som.java geeft Som.class**

3. Het programma (de .class-file) wordt geladen in het geheugen.
4. De bytecode wordt geverifieerd (belangrijk wanneer we klassen gebruiken van het net, die mogelijk virussen bevatten).
5. Het programma wordt uitgevoerd, door gebruik te maken van de Java-vertaler (JVM), via het commando java → **java Som**

Java-applicaties's worden door een compiler vertaald naar bytecode voor de JVM. De eerste JVM's voerden deze bytecode uit door middel van interpretatie. Omdat deze methode langzaam was ging men gebruikmaken van **JIT-compilatie (Just in Time)**, een vorm van compilatie die plaatsvindt tijdens de uitvoering van een computerprogramma. JIT-compilatie vindt plaats in de interpreter (ook wel virtuele machine genoemd). De broncode van een programma wordt meestal eerst vertaald naar bytecode door een compiler. Als de interpreter deze bytecode uitvoert en een stuk code (een functie of een methode) tegenkomt dat aan bepaalde voorwaarden voldoet, stopt de interpreter met de uitvoering. Hij compileert het stuk code naar machinetaal en deze machinetaal wordt vervolgens door de CPU uitgevoerd. Als de CPU klaar is gaat de interpreter weer verder. De machinetaal die het resultaat is van de compilatie wordt echter ook opgeslagen. De volgende keer dat dezelfde code uitgevoerd moet worden (denk hierbij aan lussen en functies die vaker worden aangeroepen) hoeft deze niet meer gecompileerd te worden en wordt de bijbehorende machinetaal direct aan de CPU gegeven om uitgevoerd te worden.

4.4. Het Java platform



Om een class-file uit te voeren, volstaat het dat op je computer de **Java Runtime Environment (= JRE)** staat. Dit is een combinatie van de JVM met de Java API, m.a.w. het programma om te vertalen naar je eigen machinecode en de bibliotheek van Java met alle klassen die kunnen gelinkt worden aan je eigen klassen.

4.5. Wat heb je nodig?

1. een editor, bijv. Notepad, Wordpad of Textpad (Textpad)
2. een compiler
3. een Java Virtual Machine

Compiler + JVM = Java Development Kit (JDK)

4.6. Integrated Development Environment

IDE bevat:

1. Code editor (maakt gebruikt van JDK)
2. Compiler, linker
3. Debugger
4. Help
5. Schermontwikkeling

⇒ de ontwikkelingstijd voor het schrijven van een programma wordt sterk gereduceerd.

IDE's: NetBeans, Eclipse, JBuilder, IntelliJ IDEA, JDeveloper, ...

5. Uitvoerstatements in een Java-applicatie

In deze paragraaf leren we **print, println en printf** gebruiken.

5.1. Eerste Java-applicatie: een zin afdrukken

Gewenste uitvoer:

```
<terminated> Welcome1 [Java Application] C:\Program Files\Java\jdk-11.0.6\bin\javaw.exe (28 mrt. 2020 18:54:18)
Welcome to Java Programming!
```

De applicatie:

```
1 /* Welcome1.java
2  Een eerste programma in Java. */ ①
3 ②
4 package cui; ③
5
6 public class Welcome1 ④
7 { ⑤
8     // main methode begint de uitvoering van Java applicatie ①
9     public static void main(String[] args) ⑥
10    { ⑦
11        System.out.println("Welcome to Java Programming!"); ⑧
12        ⑨
13    }// einde methode main
14 } // einde klasse Welcome1
```

Bespreking:

① Commentaar in Java

- Beschrijft stukken code
- Verhoogt de leesbaarheid van het programma
- Wordt door de Java-compiler genegeerd

> Er zijn drie soorten commentaar in Java:

1. Eén-lijn commentaar : // einde klasse Welcome 1
2. Commentaar die uit meerdere lijnen bestaat /* Dit is een traditioneel comment. Het kan over meerdere lijnen gesplitst worden */
3. Javadoc commentaar : /** … */: het javadoc-programma leest deze vorm van commentaar en maakt documentatie in HTML-formaat

② Blanco regel buiten de klasse

- Programmeurs zetten blanco regels in hun programma zodat het gemakkelijker kan gelezen worden.
- Witregels, spaties en tabs = “wit-ruimte” karakters, worden door de compiler genegeerd.

③ Package-declaratie: een project delen we op in packages, voorlopig enkel cui = console-user-interface

④ Klasse definitie: **public class Welcome1**

- Elke Java-applicatie heeft minstens één klasse door de programmeur gedefinieerd (= user-defined class)
- Keyword = sleutelwoord, heeft een speciale betekenis in Java
  - `class` gevolgd door naam van klasse (= identifier)
- Afspraak: De naam van een klasse begint altijd met een hoofdletter en ook elk woord in de

naam. *Vb.:* VoorbeeldKlasseNaam

- **Identifier**
  - Rij karakters bestaande uit letters, cijfers, underscores (_) en dollartekens ($)
  - Begint nooit met een cijfer, heeft geen spaties
  - *Voorbeelden:* Welcome1, $value, value, button7
    - 7button is ongeldig
  - Java is case sensitive (hoofdletter-gevoelig)
    - a1 is niet gelijk aan A1
- **public** sleutelwoord
  - Welcome1 is beschikbaar voor alle andere objecten, van welke packages ze ook deel uitmaken.

⑤ Linkerhaakje **{**

- Blokken vormen
- Rechterhaakje sluit het blok
- Om het begin van de klasse aan te geven

⑥ **public static void main(String[] args)**

- Onderdeel van elke Java-applicatie
  - Applicaties starten met de uitvoering van main
  - main is een methode
  - Java-applicaties bevatten één of meerdere methodes
  - Juist één methode moet main genoemd worden
- Methodes verrichten taken en kunnen informatie teruggeven
  - void betekent geen teruggeef-informatie (want de caller van main is JRE)

⑦ Linkerhaakje **{**

- Om het begin van de methode main aan te geven

⑧ In body : instructies (**System.out.println("Welcome to Java Programming!");**)

- System.out
  - Standaard output stream object, laat toe om bytes weer te geven op het scherm
  - Een stream is een stroom van gegevens. Een stream is een manier om allerlei vormen van gegevenstransport in een computer te beschrijven: gegevens die via het toetsenbord binnenkomen, of via een netwerkkabel of gegevens die via twee programma's worden uitgewisseld, kan je ook beschouwen als een stream.
  - Drukt af in command window (Command Prompt/shell)
- Methode System.out.println
  - Toont een zin en neemt nadien een nieuwe lijn

    - Argument tussen haakjes
    - Tekst die letterlijk moet getoond worden staat tussen dubbele quote
- Instructies = statements
    - Statements worden afgesloten met ; (maakt deel uit van het statement)

⑨ Blanco regel in een methode

Compilatie en uitvoering van de Java-applicatie

1. Bewaren met bestandsnaam = klassenaam.java (**Welcome1.java**)
2. Compileren (**javac Welcome1.java geeft Welcome1.class**)
3. Uitvoeren van de applicatie (**java Welcome1**)

5.2. Eerste Java-applicatie wijzigen

5.2.1. Eén zin afdrukken met meerdere statements

Gewenste uitvoer:

<terminated> Welcome2 [Java Application] C:\Program Files\Java\jdk-11.0.6\bin\javaw.exe (13 apr. 2020 14:15:01)

```
Welcome to Java Programming!
```

De applicatie:

```
1 package cui;
2
3 public class Welcome2
4 {
5     public static void main(String[] args)
6     {
7         System.out.print("Welcome to "); ①
8         System.out.println("Java Programming!"); ②
9
10    }
11 }
```

Bespreking:

① Methode System.out.print

- Toont een zin en neemt nadien **geen** nieuwe lijn
- Na 'to' staat er een spatie!

② Op dezelfde lijn op het scherm gaan we verder met de println

5.2.2. Meerdere zinnen afdrukken met één enkel statement

Gewenste uitvoer:

<terminated> Welcome3 [Java Application]

```
Welcome
to
Java
Programming!
```

De applicatie:

```
1 package cui;
2
3 public class Welcome3
4 {
5     public static void main(String[] args)
6     {
7         System.out.println("Welcome\nto\nJava\nProgramming!"); ①
8
9     }
10 }
```

Bespreking:

① Methode System.out.println maakt gebruik van escape character **\n** (newline karakter)

- Escape characters zijn speciale karakters bij methodes print en println en beginnen telkens met een backslash (\ )
- Het zijn karakters die output vormgeven
  - \n : newline = harde return
  - \t : horizontale tab
  - \r : carriage return
  - \\ : levert 1 backslash op
  - \" : levert de dubbele quote op

Voorbeelden:

System.out.println("\\ is een backslash");

uitvoer: \ **is een backslash**

System.out.println("\" is een double quote");

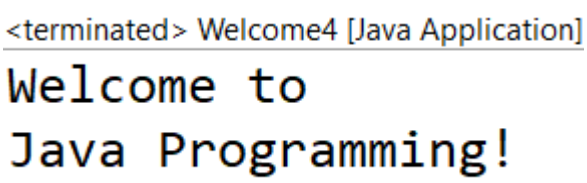
uitvoer: **" is een double quote**

5.2.3. Alternatieve methode printf

Nieuw vanaf J2SE 5.0 is de methode printf

- De f in printf betekent **“formatted”**.
- We gebruiken de methode printf voor geformatteerde uitvoer naar het beeldscherm.

Gewenste uitvoer:



De applicatie:

```
1 // Meerdere lijnen afdrukken met printf
2
3 package cui;
4
5 public class Welcome4
6 {
7     public static void main(String[] args)
8     {
9         System.out.printf("%s%n%s%n", "Welcome to", "Java Programming!"); ①
10
11    }
12 }
```

Bespreking:

① Aanroep methode **printf** naar het standaard output stream object System.out

Een aanroep van printf is van de vorm

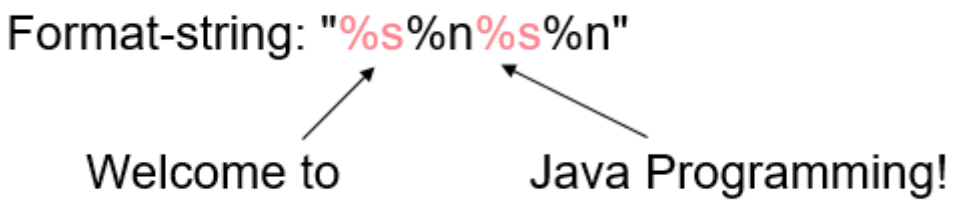
```
printf(format-string, arg1, arg2, ⋯)
```

waarin al of niet op de format-string nog een aantal argumenten kan volgen.

- Hier: drie argumenten, de format-string en 2 extra argumenten
- Meerdere argumenten scheiden met komma’s.
- Het eerste argument is altijd de format-string.
- In de format-string kunnen “gewoon af te drukken karakters”, “speciale karakters (bv. %n)” en “conversiespecificaties” voorkomen.
- Elke conversiespecificatie begint met een procentteken (%) en eindigt op een conversiekarakter. Daartussen kan ook nog iets staan, bijvoorbeeld iets over de precisie.

we gebruiken %n ipv \n in printf! %n is in elk operating systeem bruikbaar en draagt dus bij tot de portabiliteit van het programma

- Hier: tweede en derde argument zijn teksten en vervangen in de format-string %s (=string).
- Als we met printf tekst willen laten weergeven dan moeten we als conversiespecificatie **%s** gebruiken.



Strings weergeven

Conversiekarakter	Beschrijving
s	Het argument is een string. s geeft de string weer.
S	Het argument is een string. S geeft de string in hoofdletters weer.

Voorbeeld:

```
1 package cui;
2
3 public class StringConversion
4 {
5    public static void main( String args[] )
6    {
7       String string = "This is also a string";
8       System.out.printf( "%s%n", "This is a string" );
9       System.out.printf( "%S%n", "This is a string" );
10      System.out.printf( "%s%n", string );
11      System.out.printf( "%S%n", string );
12   } // end main
13 } // end class StringConversion
```

```
<terminated> StringConversion [Java Application]
This is a string
THIS IS A STRING
This is also a string
THIS IS ALSO A STRING
```

Gehele getallen weergeven

Conversiekarakter	Beschrijving
d	Het (int) argument wordt geconverteerd naar decimale notatie.
o	Het (int) argument wordt geconverteerd naar octale notatie.
x of X	Het (int) argument wordt geconverteerd naar hexadecimale notatie. X geeft de getallen 0 t.e.m. 9 en de letters A t.e.m. F weer. x geeft de getallen 0 t.e.m. 9 en de letters a t.e.m. f weer.

Voorbeeld:

```
1 package cui;
2
3 public class IntegerConversion
4 {
5     public static void main(String args[])
6     {
7         System.out.printf("%d%n", 26);
8         System.out.printf("%d%n", +26);
9         System.out.printf("%d%n", -26);
10        System.out.printf("%o%n", 26);
11        System.out.printf("%x%n", 26);
12        System.out.printf("%X%n", 26);
13    } // end main
14 } // end class IntegerConversion
```

```
<terminated> IntegerConversion
26
26
-26
32
1a
1A
```

Veldbreedte

Tussen het procentteken (%) en een conversiekarakter (behalve n) kan een getal staan. Dit getal geeft de veldbreedte aan. Het geconverteerde argument zal afgedrukt worden in een veld van de opgegeven breedte, of meer als dat nodig is. Eventuele ruimte die over is wordt aan de linkerkant toegevoegd.

Voorbeeld:

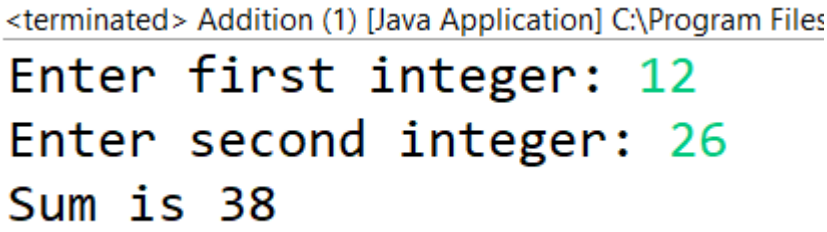
```java
package cui;

public class FieldWidth
{
    public static void main(String args[])
    {
        System.out.printf("%4d%n", 1);
        System.out.printf("%4d%n", 12);
        System.out.printf("%4d%n", 123);
        System.out.printf("%4d%n", 1234);
        System.out.printf("%4d%n%n", 12345); // getal is te groot
        System.out.printf("%4d%n", -1);
        System.out.printf("%4d%n", -12);
        System.out.printf("%4d%n", -123);
        System.out.printf("%4d%n", -1234); // getal is te groot
        System.out.printf("%4d%n", -12345); // getal is te groot
    }
}
```

```
<terminated> FieldWidth
   1
  12
 123
1234
12345

  -1
 -12
-123
-1234
-12345
```

6. Invoer- en uitvoerstatements in een Java-applicatie: optellen van gehele getallen

Programma met input en output: Er worden twee waarden aan de gebruiker gevraagd. De som van de twee waarden wordt weergegeven op het scherm.

Gewenste in- en uitvoer:



De applicatie:

```
1 package cui;
2
3 import java.util.Scanner; ①
4
5 public class Addition
6 {
7     // main methode start de uitvoering van Java applicatie
8     public static void main(String[] args)
9     {
10        // creëer een object van Scanner; voor invoer vanaf het toetsenbord
11        Scanner input = new Scanner(System.in); ②
12
13        int number1; // eerste getal om op te tellen ③
14        int number2; // tweede getal om op te tellen
15        int sum; // som van number1 en number2
16
17        System.out.print("Enter first integer: "); // prompt ④
18        number1 = input.nextInt(); // leest eerste getal van de gebruiker ⑤
19
20        System.out.print("Enter second integer: "); // prompt
21        number2 = input.nextInt(); // leest tweede getal van de gebruiker
22
23        sum = number1 + number2; // de getallen optellen ⑥
24
25        System.out.printf("Sum is %d%n", sum); // de som weergeven ⑦
26
27    } // einde methode main
28
29 } // einde klasse Addition
```

Bespreking:

① **import java.util.Scanner;**

- import: importeert de klasse Scanner van de package java.util
  - Het import-statement geeft de compiler de opdracht de applicatie toegang te geven tot een klasse van een package.
- Packages
  - Verzameling van voorgedefinieerde klassen.

    - Een groep van gerelateerde klassen is een package.
    - Verzameling van alle packages = Java class library of Java Applications Programming Interface (Java API)
- Twee groepen packages in Java API
    - Core (=kern) packages en beginnen met java (vb: ***java.util.Scanner***: enkel de klasse Scanner)
    - Extension (=uitbreidings) packages en beginnen met javax (vb: ***javax.net.****: alle klassen in package 'net' onder javax)
    - De Java API documentatie kan gevonden worden op https://docs.oracle.com/en/java/javase/14/docs/api/index.html of je kan ze downloaden van http://www.oracle.com/technetwork/java/javase/downloads/index.htm

② **Scanner input = new Scanner(System.in);**

- Een **klasse** beschrijft gelijksoortige objecten:
    - *Auto*: de klasse Auto beschrijft auto's met gelijksoortige eigenschappen zoals kleur, merk, type en hetzelfde gedrag zoals rijden, remmen,…
    - *Pennenzak*: de klasse Pennenzak beschrijft pennenzakken met gelijksoortige eigenschappen zoals kleur, lengte, materiaal en hetzelfde gedrag zoals openen, sluiten, vullen,…
- **Scanner**: de klasse Scanner uit java.util beschrijft scanners die data kunnen scannen/lezen van een welbepaalde bron.
    - *System.in* = Standaard input stream object, laat toe om bytes via het toetsenbord in te lezen
    - *new Scanner(System.in)* creëert een object van de klasse Scanner door de constructor aan te roepen via de operator 'new'. Door dit object zullen we data vanaf het toetsenbord kunnen lezen.
    - *Scanner input = new Scanner(System.in);* : het object van de klasse Scanner toekennen aan de variabele input in het linkerlid
- Toekenningsstatement
    - = binaire operator - heeft twee operanden
    - Rechterlid wordt geëvalueerd en toegekend aan variabele in het linkerlid
- Variabelen
    - Geheugenlocatie waarin een waarde wordt gestockeerd
    - Heeft een naam en is van een bepaald type: *input* is van type Scanner
    - Naam van variabele = willekeurige identifier

③ **int number1; // eerste getal om op te tellen**

- Declaraties eindigen met ;
- Mogelijk meerdere variabelen van hetzelfde type tegelijkertijd te declareren: **int number1, number2, sum;**

- Voeg commentaar toe om het doel van variabelen te beschrijven
- De variabelen number1, number2 en sum zijn van het primitieve type *int*
- *int* bevat gehele getallen : 0, -4, 97
- Andere primitieve types: boolean, char, float, double, short, byte, long
- Namen van variabelen beginnen ALTIJD met een kleine letter

④ **System.out.print("Enter first integer: "); // prompt**

- De vraagstelling "Enter first integer: " wordt op het scherm weergegeven
- Methode print: de cursor blijft op dezelfde lijn staan na het dubbele punt

⑤ **number1 = input.nextInt(); // leest eerste getal van de gebruiker**

- Klassen hebben methodes, die we gebruiken om met de objecten te communiceren. Notatie: **object.methode**
- De klasse Scanner heeft een methode 'nextInt' om integers te lezen
- De methode nextInt geeft het ingegeven getal, via het toetsenbord, terug.
- Het ingegeven getal wordt toegekend aan de variabele number1
  - number1 werd als int gedeclareerd

⑥ **sum = number1 + number2; // de getallen optellen**

- Toekenningsstatement
  - Berekent de som van number1 en number2 (rechterzijde)
  - Gebruikt de toekenningsoperator = om het resultaat in de variabele sum te stoppen
  - Lees als: sum krijgt de waarde van **number1 + number2**
  - number1 en number2 zijn operanden

⑦ **System.out.printf("Sum is %d%n", sum); // de som weergeven**

- De format-string is "Sum is %d%n"; deze bestaat uit vaste tekst, conversiespecificatie %d en een newline %n
- Tweede argument (sum) is een geheel getal: in de formatstring wordt %d vervangen door de waarde die in de variabele sum zit
- Als we met printf een geheel getal willen laten weergeven dan moeten we als conversiespecificatie %d gebruiken

7. Rekenkundige bewerkingen

- Gebruik:
  - * voor vermenigvuldiging
  - / voor deling
  - +, -

    - Geen operator voor exponent (zie hoofdstuk 2)
- Integer deling kapt quotiënt af:
    - 7 / 5 = 1
    - 1 / 2 = 0
    - 123 / 10 = 12
- Rest-operator %:
    - 7 % 5 = 2 want 7 - (5 * 1) = 2
    - 1 % 2 = 1 want 1 - (2 * 0) = 1
    - 123 % 10 = 3 want 123 - (10 * 12) = 3
- Prioriteitsregels:

haakjes: ( )
* , / , %
+, -

Bij gelijke prioriteit ⇒ regels van de associativiteit (van L naar R)

Voorbeeld: 7 * 3 / 5 % 2 = 21 / 5 % 2 = 4 % 2 = 0

Voorbeelden: formules in de algebra omgezet in Java

Algebra: $m = \frac{a + b + c + d + e}{5}$

Java : `m = (a + b + c + d + e ) / 5;`

Algebra: $m = a + b + c + d + \frac{e}{5}$

Java : `m = a + b + c + d + e / 5;`

Algebra: $z = pr\%q + w/x - y$

Java : `z = p * r % q + w / x – y;`

Algebra: $y = ax^2 + bx + c$

Java : `y = a * x * x + b * x + c;`

In Java mag je ook overbodige haakjes plaatsen:

`y = (a * x * x) + (b * x) + c;`

In Java mag je geneste haakjes plaatsen:

`a = ((b + c) * d) / 2`

8. On the Web

- http://www.deitel.com/: Deitel & Associates home page
- http://www.oracle.com/technetwork/java/index.html: Oracle Java home page
- http://www.oracle.com/technetwork/java/javase/overview/index.html: de home page voor het Java Platform, Standard Edition
- http://www.oracle.com/technetwork/java/javase/downloads/index.htm: de download page voor JDK en API documentatie
- https://docs.oracle.com/en/java/javase/14/docs/api/index.html: online API documentatie JDK14
- http://www.uml.org/: informatie omtrent UML