

Programmeren 1: Java

Basisbegrippen: week 1

Programmeren 1 - Java

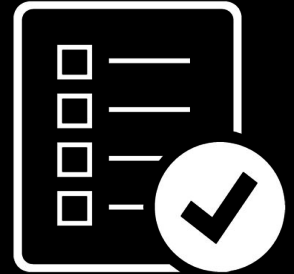
P1 - Basisbegrippen

Week 1	Inleiding - Algoritmes
Week 2	Variabelen - Operatoren
Week 3	Expressies - Constructies
Week 4	Objecten - Object-oriëntatie
Week 5	Klassen
Week 6	Arrays

Herfstvakantie

Examen

Agenda deze week




- Vooraf: afspraken, evaluatie, ...
- H1: Inleiding
- H2: Java Development Kit
- H3: Het eerste Java programma
- H4: Programmeeralgoritmen

Programmeren in het 1^e jaar

Periode	Vak	Verdeling
	Programmeren 1 – Java (sOlod – 11p)	100%
1	Basisbegrippen (dOlod – 3p)	15%
2	OO Technieken (dOlod – 3p)	35%
3	JavaFX (dOlod – 5p)	50%

ECTS-fiche

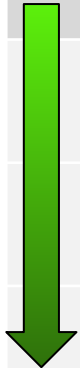
- [ECTS-fiche](#) 
- Details over het vak:
 - Inhoud
 - Doelstellingen
 - Werkvormen
 - Evaluatie
 - Lesmateriaal

Evaluatie



Pietluttig examen

Periode	Vak	Verdeling	Evaluatie
	Programmeren 1 - Java	100%	
1	Basisbegrippen	15%	Gesloten boek schriftelijk (meerkeuze / invulvragen)
2	OO technieken	35%	Open boek praktijkexamen (laptop)
3	JavaFX	50%	Project (game) + mondeling examen



Het wordt leuker en leuker

Lesmateriaal

- Blackboard

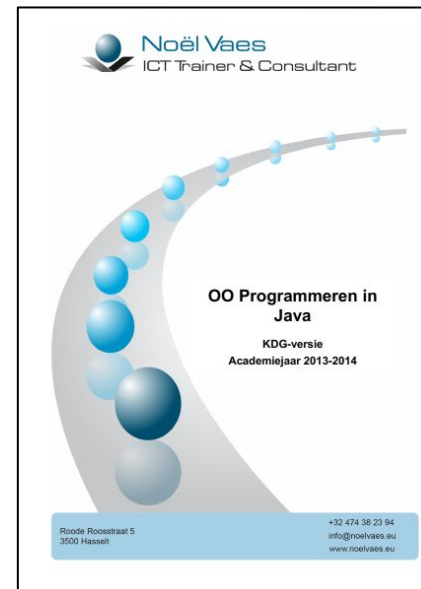
- Presentaties
- Oefeningen
- Zelftoetsen
- ...

Zie je dit icoon in de slides, dan verwijzen we naar Blackboard...



- Handboek

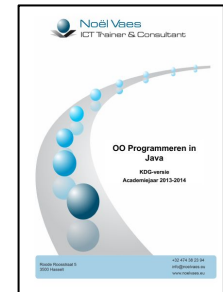
- Papieren versie
- E-Book versie
 - kan je niet printen
- Optioneel



Lesmateriaal

- Oefeningen

- Er zijn er véél beschikbaar via de slides, het cursusboek en BlackBoard
- Thuis maken, elke vrijdag vanaf 18u oplossingen beschikbaar
- Jouw oplossing kan/mag afwijken van de modeloplossing
- Modeloplossing wel begrijpen



Lesmateriaal



- Oefeningen

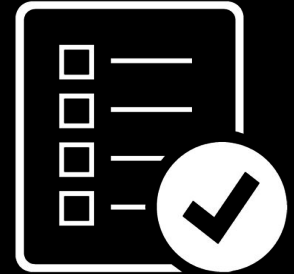
- Zijn de beste manier om de syntax en het inzicht in programmeren te leren!
- Hou ze goed bij en maak er regelmatig vanaf de eerste week!

- Zelftoetsen

- Vrijblijvend (scores tellen NIET mee)
- Tussentijds testen van je kennis

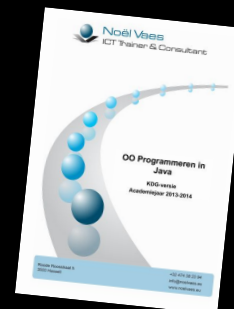


Agenda deze week



- Vooraf: afspraken, evaluatie, ...
- H1: Inleiding
- H2: Java Development Kit
- H3: Het eerste Java programma
- H4: Programmeeralgoritmen

Zie boek pagina 9 - 18



H1: Inleiding

- Programmeertaal?

- Machine Code:

```
100101011011011101100001011101110110000101101110110000101101110110  
000101101110110000101101110110000101101110110000101101110110000101  
1011101100001001...
```

1^e generatie



- Assembler:

```
MOV AX, 47104  
MOV DS, AX  
POP [3998], 36  
INT 32
```

2^e generatie



- 3th Generation Languages (3GL) → Leesbaar!

```
Scanner scanner = new Scanner(System.in);  
boolean mooiWeer = scanner.nextBoolean();  
if (mooiWeer) {  
    System.out.println("Laat je paraplu maar thuis!");  
} else {  
    System.out.println("Vergeet je paraplu niet!");  
}
```

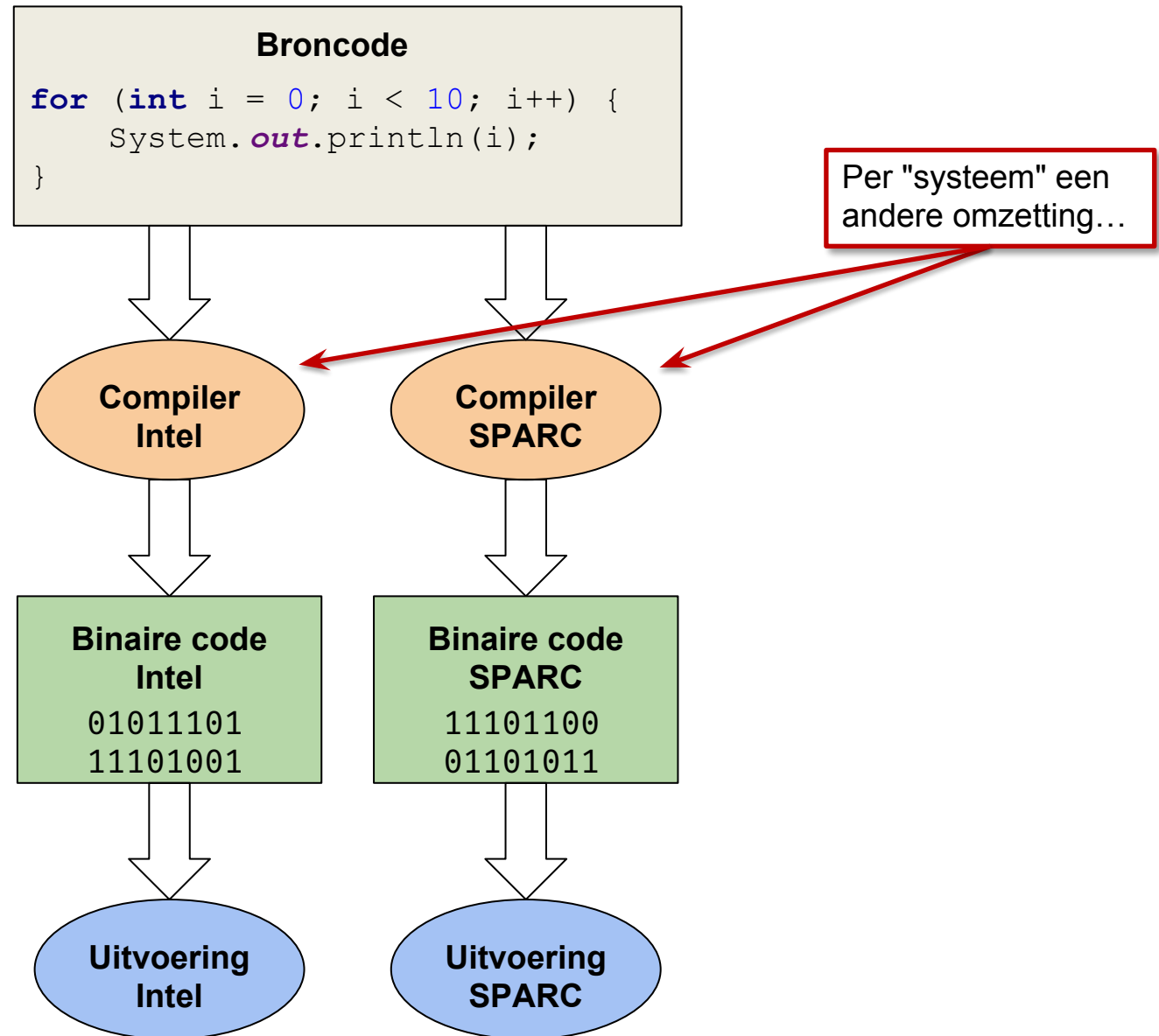
3^e generatie



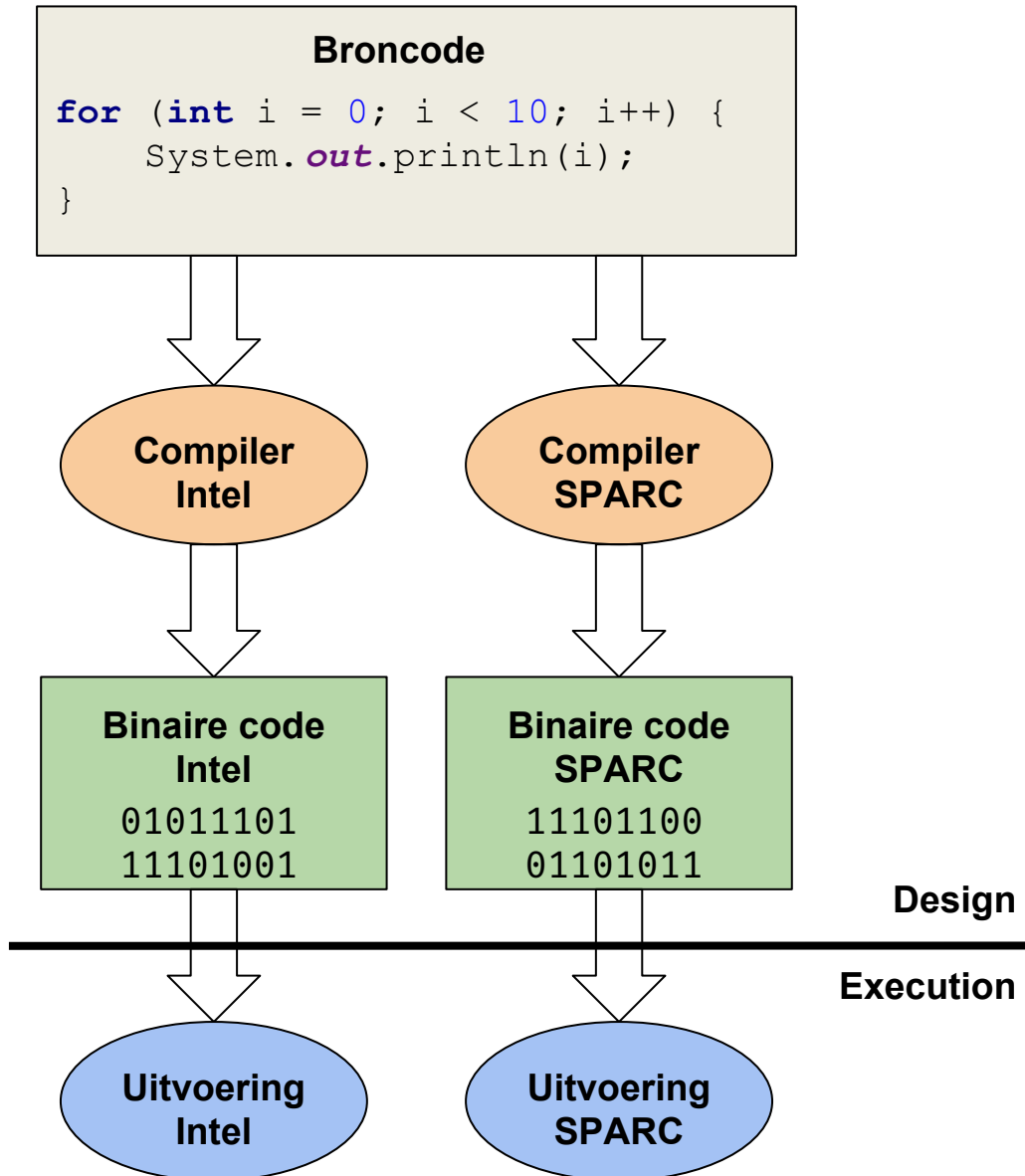
Voorbeelden 3GL

Java – C – C++ – Visual Basic – VB.Net –
C# – Scala – Kotlin – Rust – D – Go –
Erlang – Delphi – Pascal – Modula2 –
Oberon – Perl – Python – PL/SQL – SQL –
Javascript – VBScript – Cobol – Fortran –
Ruby – JRuby – ActionScript – Groovy –
JFX – Prolog – Lisp – Shellscript – TCL –
Smalltalk – Postscript – Pearl – Logo –
Basic – J++ – PHP – ...

Omzetting 3GL naar binair



Compileren



Voorbeeld:

- C

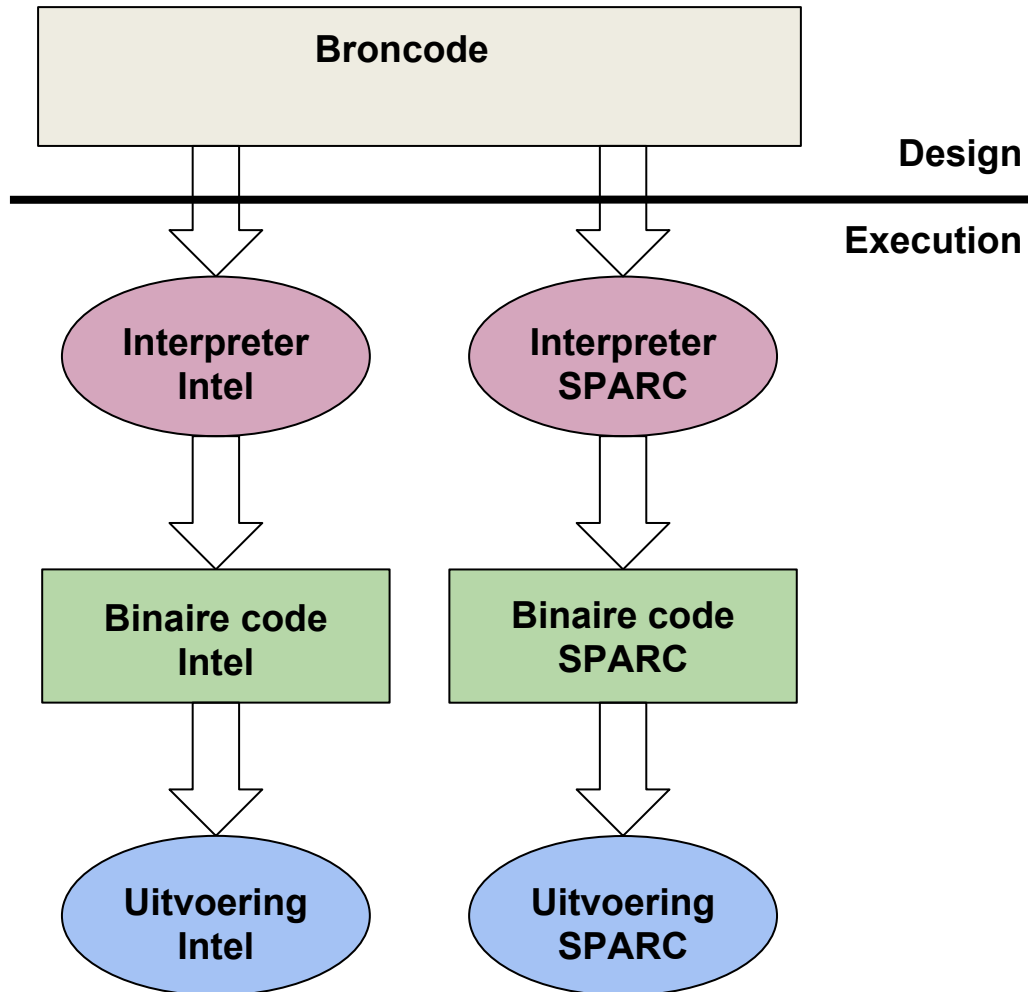
Voordelen:

- Snelle uitvoering!
- Broncode overdraagbaar
- Code beschermd

Nadelen:

- "Platform afhankelijk": hercompileren voor ander systeem
- Debugging en testing vraagt extra stap

Interpreteren



Voorbeeld:

- JavaScript

Voordelen:

- Aanpassingen aan code eenvoudig
- Onmiddellijk overdraagbaar ("platform onafhankelijk")

Nadelen:

- Tragere uitvoering
- Broncode onbeschermd

Java: best of both!



Broncode
(MyProgram.java)

Compiler
Java

Bytecode
(MyProgram.class)

Design

Execution

Interpreter
Intel

Uitvoering
Intel

Interpreter
SPARC

Uitvoering
SPARC

Voordelen:

- Onmiddellijk overdraagbaar ("platform onafhankelijk")
- Sneller dan geïnterpreteerd
- Broncode (min of meer) beschermd

Nadelen:

- JVM nodig om de bytecode uit te voeren
- Trager dan gecompileerd

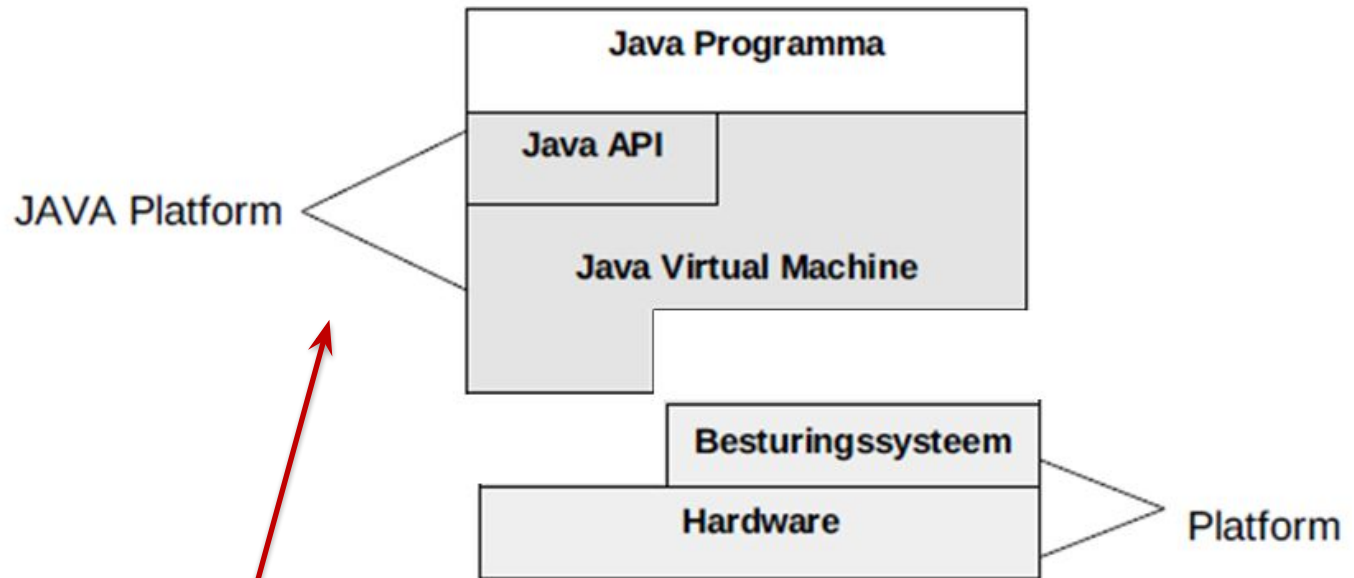
Overige kenmerken Java

- Geïnterpreteerd
- Platform onafhankelijk
- Objectgeoriënteerd
- Gedistribueerd
- Robuust
- Multithreaded
- Veilig
- Snel



Deze kenmerken worden later
nog wel duidelijk...!

Java als platform



Doordat Java een volledige laag legt op het onderliggende platform zijn Java applicaties platformonafhankelijk....

Soorten Java toepassingen

- Desktop applicaties
 - Voorbeelden: LibreOffice, Minecraft, ...
- Applets
 - In de browser, achterhaalde technologie, nu vooral HTML5, ...
- Serverapplicaties
 - Zeer populair, hoofdrolspeler in de markt! (zie μ -degree Software Architecture)

Wij focussen dit jaar op desktopapplicaties, vanaf het tweede jaar krijgen serverapplicaties de nadruk....

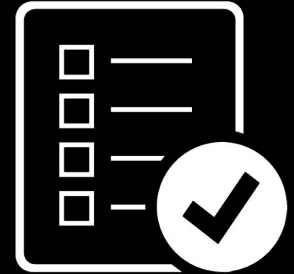
Wie is dit?



- James Gosling
- Heeft Java ontwikkeld in 1994
- Werkte bij SUN
- Ondertussen is SUN overgenomen door Oracle

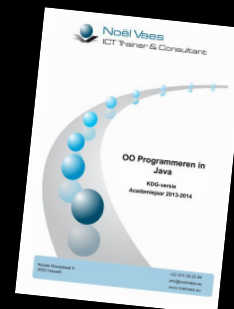
Zie ook: <https://www.youtube.com/watch?v=r19P3y1VBiw>

Agenda deze week



- Vooraf: afspraken, evaluatie, ...
- H1: Inleiding
- H2: Java Development Kit
- H3: Het eerste Java programma
- H4: Programmeeralgoritmen

Zie boek pagina 19 - 24



H2: De Java Development Kit

Om Java programma's te kunnen uitvoeren

- JRE:

- "Java Runtime Environment"
- JVM + Java API

Toolkit om programma's te kunnen schrijven

- JDK:

- "Java Development Kit" (versie 8)
- Tools: compileren, debuggen, ...
- Java API documentatie
- Bevat een JRE

Opdracht 1.1



De JDK installeren

Zie je dit icoon in de slides, dan moet je een opdracht uitvoeren...

- Download JDK8 van <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Zie de pdf's onder het menu *Software*
 - "Lees eerst dit"
 - "Installatie en controle Java en IntelliJ"
- Installeer en configureer je systeem



Je kan ook de stappen volgen in de cursus p19 en verder!



Opdracht 1.2



De JDK documentatie installeren

- Zoek "Java SE 8 Documentation" op de download pagina en pak uit in de map waar de JDK geïnstalleerd is.
- Zie de pdf's onder het menu *Software*
→ "Installatie en controle Java en IntelliJ".



Zie ook boek pagina 20 - 21



Opdracht 1.3



IntelliJ installeren

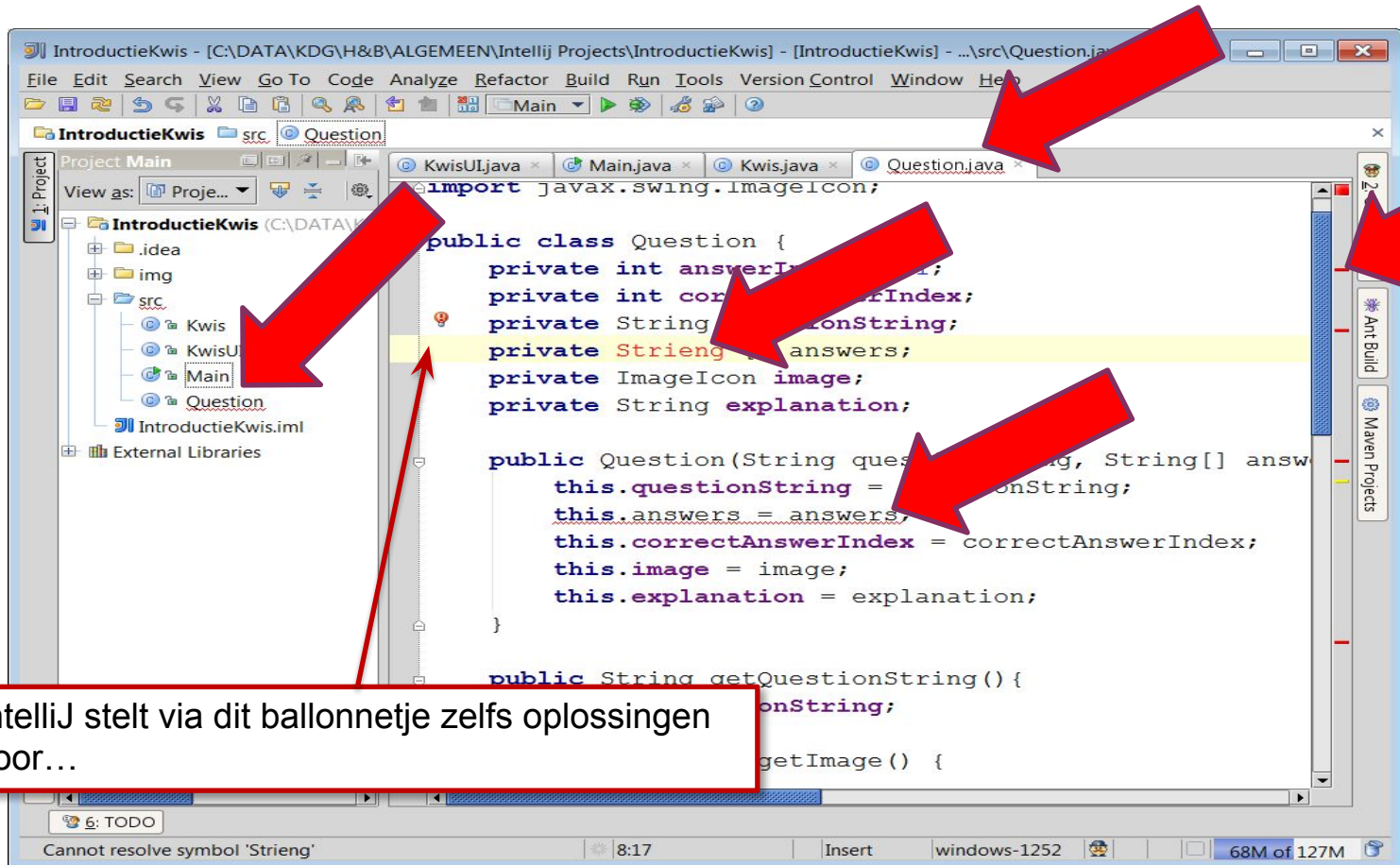
- We gebruiken IntelliJ 2017.2.4 of later als IDE
- Download de **ultimate** edition:
<http://www.jetbrains.com/idea/download/>
- Instructies en licentie:
 - zie BB → Software → Installatie en controle Java en IntelliJ



Zie boek pagina 21-24



IntelliJ: onmiddellijke feedback...

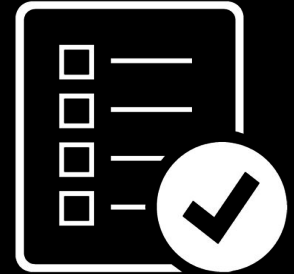


Quiz!

- Wat is een 3GL?
- Wat is compileren?
- Wat is bytecode?
- Wat is de JVM?
- Wat is de JRE?
- Wat is de JDK?
- Waarvoor staat API?
- Hoe heet het opleidingshoofd TI?

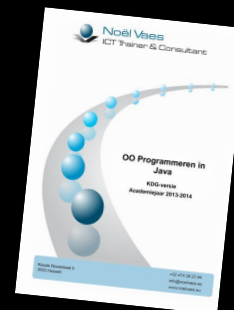


Agenda deze week



- Vooraf: afspraken, evaluatie, ...
- H1: Inleiding
- H2: Java Development Kit
- H3: Het eerste Java programma
- H4: Programmeeralgoritmen

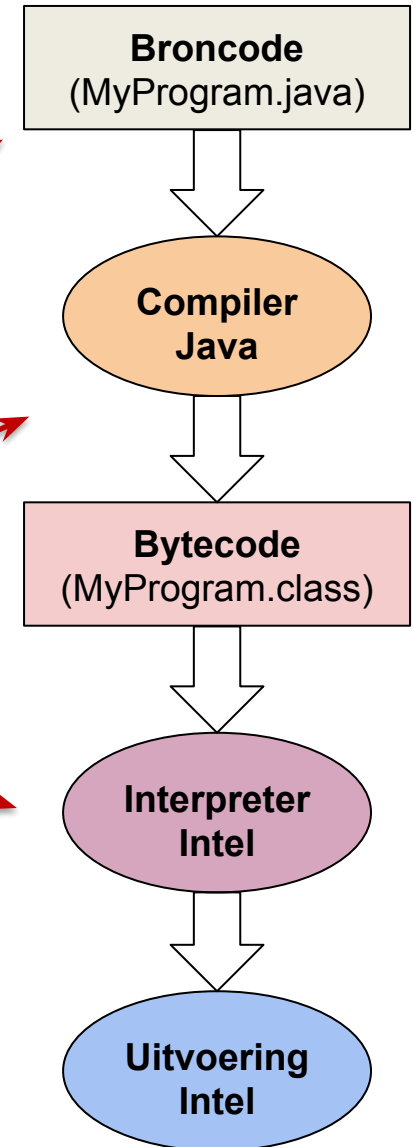
Zie boek pagina 25 - 30



Het eerste Java programma

- Drie stappen:

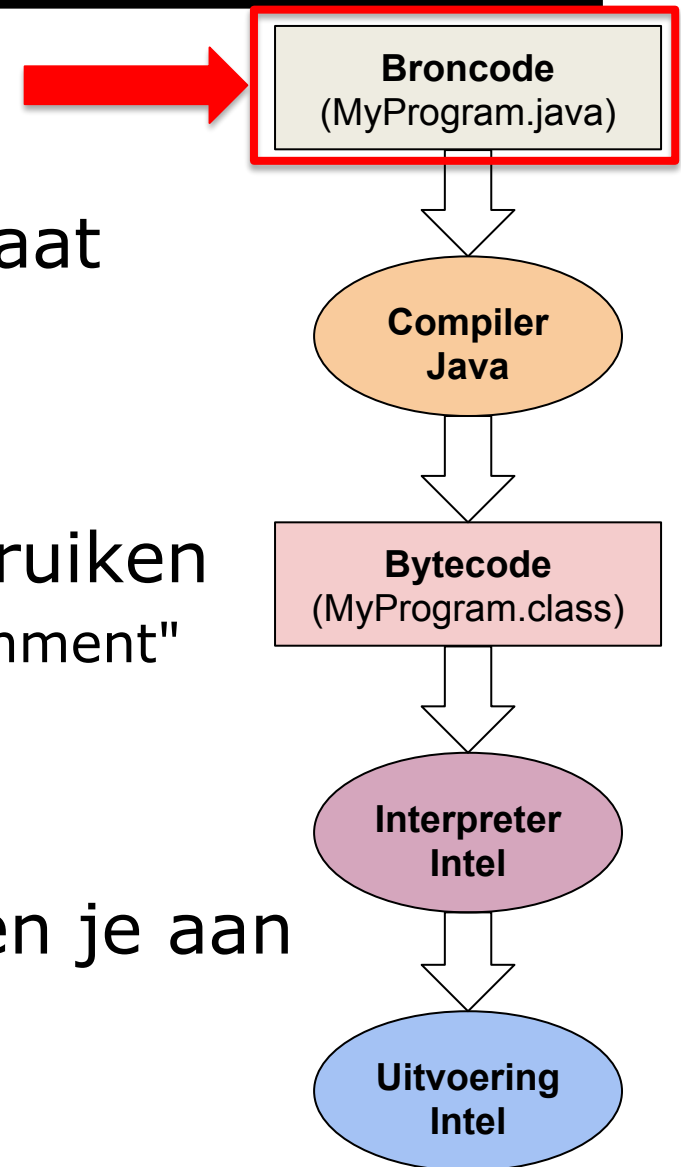
1. Broncode maken
2. Compileren naar bytecode
3. Uitvoeren



Stap 1

Broncode maken

- Een eenvoudige editor volstaat (vb: Kladblok)
- Liever: aangepaste IDE gebruiken
 - "Integrated Development Environment"
 - vbn: IntelliJ, Eclipse, Netbeans
- Het broncode bestand herken je aan extensie '**.java**'

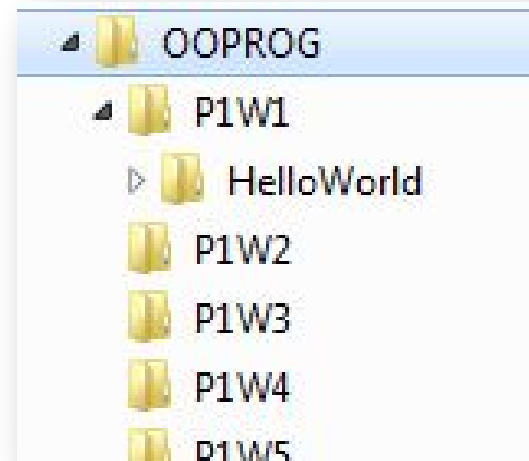


Opdracht 2.1



Hello world!

- Overzichtelijke mappenstructuur!
 - maak een map: OOPROG
 - maak een submap voor elke week
 - begin vanaf P1W1 (want daar zijn we nu)
 - daaronder: elk IntelliJ-project in aparte submap



Opdracht 2.1



(vervolg)

- Maak een nieuw project in IntelliJ met de naam 'HelloWorld'.
- Volg verder de instructies van het boek
- Zoek het broncode bestand:

Zie boek pagina 25-30

`\OOPROG\P1W1\HelloWorld\src\hello\HelloWorldApp.java`

- Open het eens met kladblok:

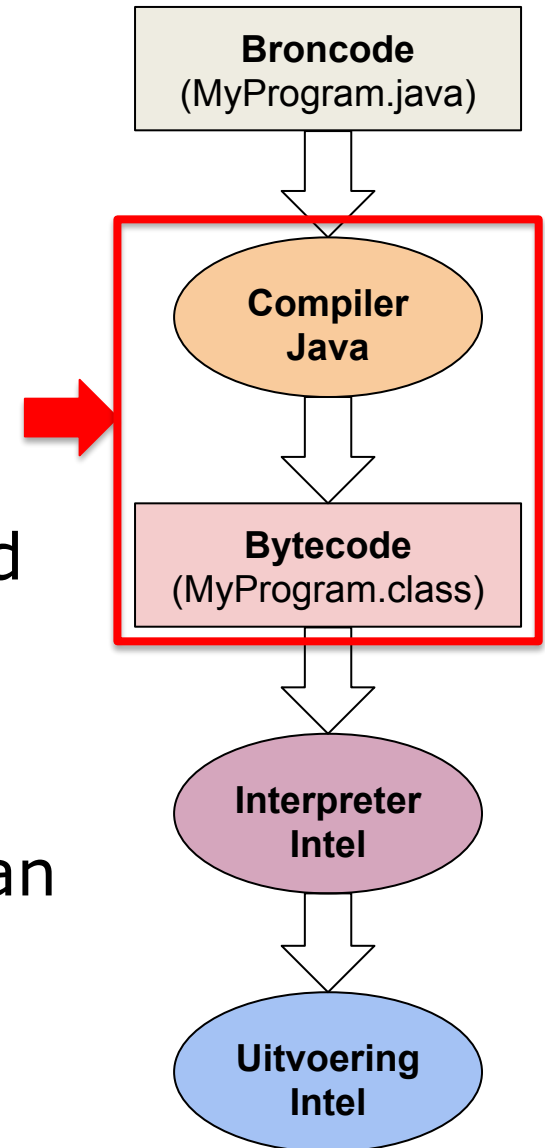
A screenshot of a Notepad window titled 'HelloWorldApp.java - Kladblok'. The window has a menu bar with 'Bestand', 'Bewerken', 'Opmaak', 'Beeld', and 'Help'. The text area contains the following Java code:

```
/* This application shows the text 'Hello world!' on the
screen. */
package hello;
public class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello world!"); // Show the text
    }
}
```


Stap 2

Compileren van de broncode

- **javac** is de compilertool die bij de JDK zit
- In IntelliJ is de compiler ingebouwd ('Make' of 'Compile')
- Het bytecode-bestand herken je aan extensie '**.class**'



Opdracht 2.2



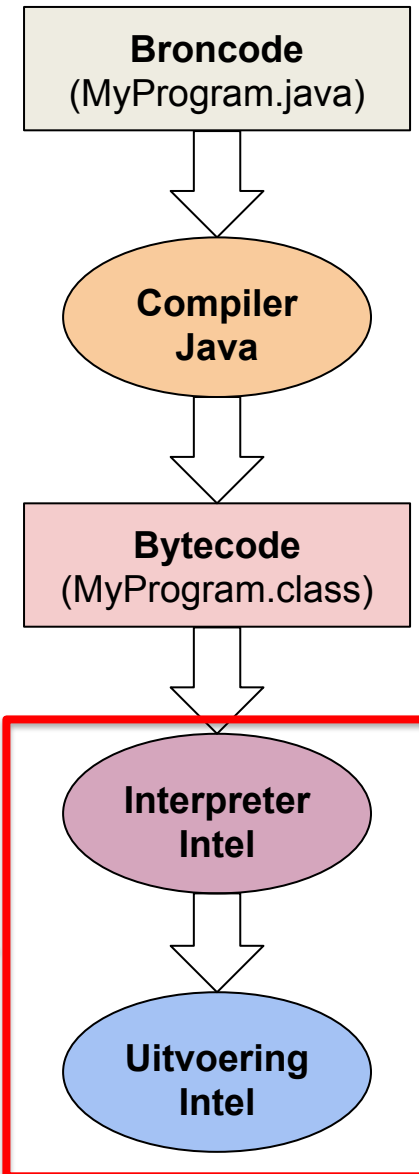
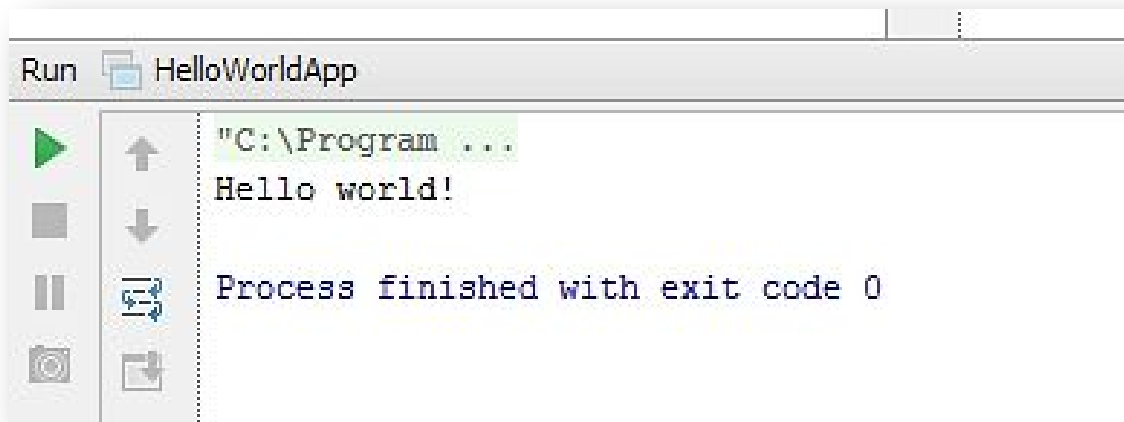
Compileren

- Compileer `HelloWorldApp`
- Zoek het bestand `HelloWorldApp.class` op in je directory-structuur

Stap 3

Uitvoeren van de bytecode

- Uitvoering van de code door de JVM die bij de JDK (of JRE) zit
- Ingebouwd in IntelliJ ('Run')
- Resultaat in commandvenster onderaan:

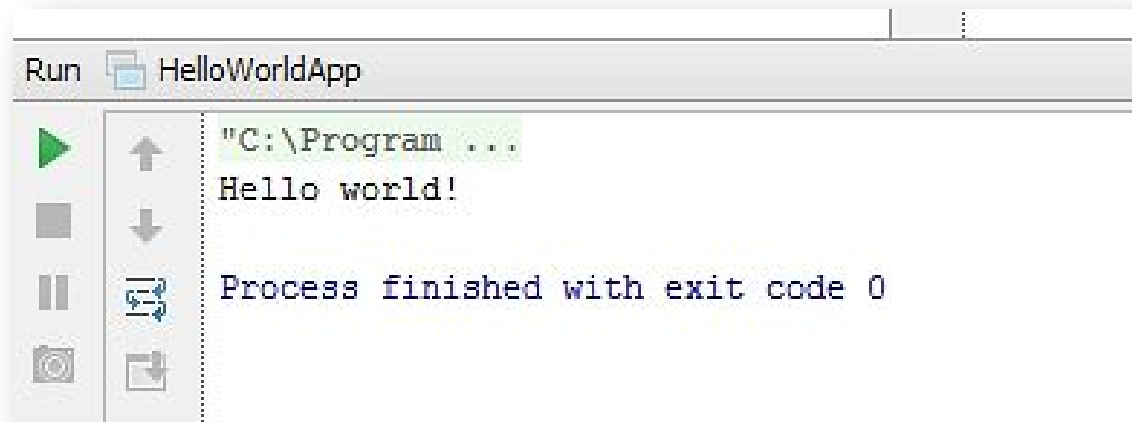


Opdracht 2.3



Runnen

- Run HelloWorldApp
- Controleer de output onderaan:



- Applaus voor jezelf!

Inspectie van de code...

Commentaar: code tussen `/*` en `*/` of achter `//` wordt genegeerd door de compiler en is dus enkel ter info.

```
/* This Java application shows  
the text 'Hello World!' on the screen. */
```

```
package hello;
```

Pakketnaam om sourcefiles te groeperen (creëert aparte submap)

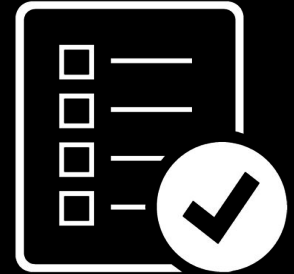
```
public class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!"); // Show the text  
    }  
}
```

De "**main** methode": dit is het startpunt van elke java desktop applicatie

De **class** wordt gedefinieerd. Een klasse is een essentieel begrip in objectgeoriënteerd programmeren en komt vanaf week 5 aan bod!

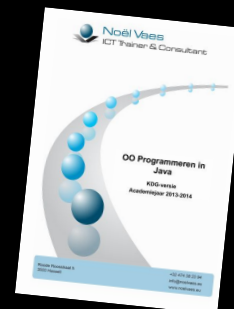
Het eigenlijke werk: de computer krijgt de opdracht de zin "Hello World" op het scherm te tonen...

Agenda deze week



- Vooraf: afspraken, evaluatie, ...
- H1: Inleiding
- H2: Java Development Kit
- H3: Het eerste Java programma
- H4: Programmeeralgoritmen

Zie boek pagina 31 - 40

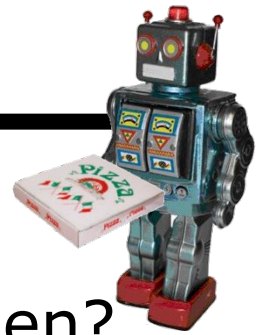


H4: Programmeeralgoritme

- Algoritme: reeks instructies die uitgevoerd moeten worden om een bepaald doel te bereiken.

→ Als je programmeert, doe je in feite niets anders dan algoritmes bedenken voor bepaalde problemen en ze dan in een programmeertaal aanbieden aan je computer...

Pizza Bestelrobot



- Wat is het algoritme om pizza te bestellen?

Volgorde
van stappen
essentieel!

1. Neem menu
2. Lees menu
3. Bel pizzaphone
4. Geef keuze
5. Geef adres
6. Wacht tot bel gaat
7. Open deur
8. Geef geld
9. Neem pizza aan

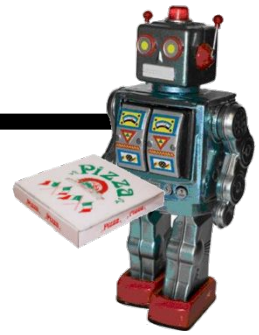
Elke stap moet je nog verder uitwerken tot op het niveau dat je robot begrijpt. Zie volgende slide..

De robot krijgt hier **input**

De robot geeft hier **output**

De robot heeft hiervoor moeten **rekenen**

Neem menu

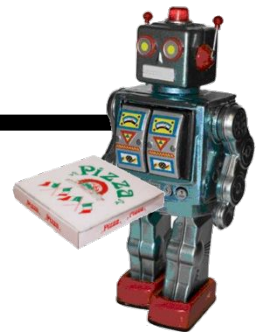


1. Wandel naar telefoonkastje
2. Open schuif
3. Haal stapel foldertjes eruit
4. Neem bovenste
5. Is deze folder van pizzaphone?
6. Indien neen: is dit laatste folder?
 1. Indien Ja: honger lijden...!
 2. Indien Neen: ga terug naar stap 4
7. Indien ja: deelalgoritme "neem menu" is klaar

Een **test**: afhankelijk van de uitkomst wordt iets anders gedaan

Een **lus**: we herhalen een aantal stappen een aantal keer

Conclusie pizzarobot:



- Algoritme:
 - Opeenvolging stappen
 - Volgorde belangrijk
 - Typische stappen zijn:
 - Input vragen
 - Output geven
 - Iets testen
 - Een aantal stappen herhalen
 - Iets berekenen
 - Je moet de stappen verder uitsplitsen tot basisinstructies die de computer begrijpt

Eenvoudiger voorbeeld

Bereken de som van 2 getallen

- Wat is het algoritme voor dit programma:

Tik een getal in: 15

Tik nog een getal in: 35

Dit is de som: 50

→ Probeer nu zelf het algoritme uit te schrijven in eenvoudige stappen (Nederlands)



Algoritme Som

input en output (IO)

1. Toon "Tik een getal in: " op output (scherm, bord, ...).
2. Lees input van keyboard en schrijf weg, noem dit **eerste**.
3. Toon "Tik nog een getal in: " op output.
4. Lees input van keyboard en schrijf weg, noem dit **tweede**.
5. Bereken de som en schrijf weg, noem dit **som**.
6. Toon het resultaat op output

Berekening met de
variabelen

Tussenresultaten die we
even wegschrijven
noemen we **variabelen**

Algoritme omzetten naar Java!

- Hoe?

- een stap uit het algoritme wordt een java instructie afgesloten met een ;

- output naar scherm:

- `System.out.print("tekst");`

- input van keyboard:

- `keyboard.nextInt();`

Algoritme omzetten naar Java

Toon "Tik een getal in: " op het scherm

Lees input van keyboard en schrijf weg, noem dit **eerste**.

Toon "Tik nog een getal in: " op het scherm

Lees input van keyboard en schrijf weg, noem dit **tweede**.

Bereken de som en schrijf weg, noem dit **som**

Toon het resultaat op het scherm

Algoritme omzetten naar Java

```
System.out.print("Tik een getal in: ");
```

Lees input van keyboard en schrijf weg, noem dit **eerste**.

Toon "Tik nog een getal in: " op het scherm

Lees input van keyboard en schrijf weg, noem dit **tweede**.

Bereken de som en schrijf weg, noem dit **som**

Toon het resultaat op het scherm

Algoritme omzetten naar Java

```
System.out.print("Tik een getal in: ");
```

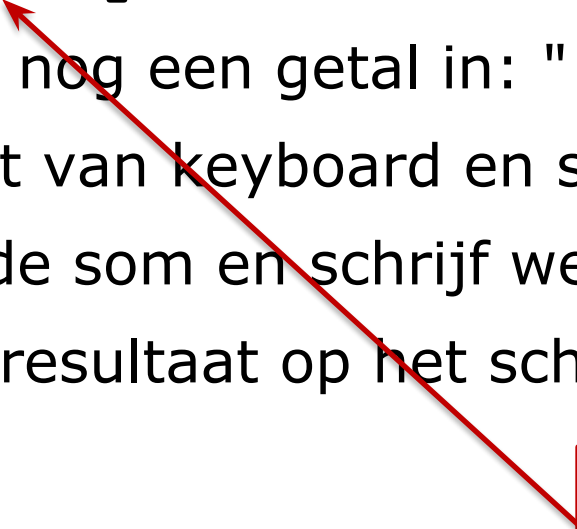
```
eerste = keyboard.nextInt();
```

Toon "Tik nog een getal in: " op het scherm

Lees input van keyboard en schrijf weg, noem dit **tweede**.

Bereken de som en schrijf weg, noem dit **som**

Toon het resultaat op het scherm



Dit is een toekenning. We schrijven het getal dat op het keyboard wordt ingetypt weg naar de variabele **getal1**.

Algoritme omzetten naar Java

```
System.out.print("Tik een getal in: ");
```

```
eerste = keyboard.nextInt();
```

```
System.out.print("Tik nog een getal in: ");
```

Lees input van keyboard en schrijf weg, noem dit **tweede**.

Bereken de som en schrijf weg, noem dit **som**

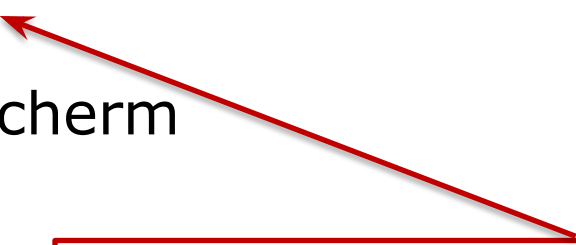
Toon het resultaat op het scherm

Algoritme omzetten naar Java

```
System.out.print("Tik een getal in: ");  
eerste = keyboard.nextInt();  
System.out.print("Tik nog een getal in: ");  
tweede = keyboard.nextInt();  
Bereken de som en schrijf weg, noem dit som  
Toon het resultaat op het scherm
```

Algoritme omzetten naar Java

```
System.out.print("Tik een getal in: ");  
eerste = keyboard.nextInt();  
System.out.print("Tik nog een getal in: ");  
tweede = keyboard.nextInt();  
som = eerste + tweede;  
Toon het resultaat op het scherm
```



Een waarde in een variabele wegschrijven doen we dus met een = teken. De doelvariabele moet aan de linkerkant staan. We noemen dit een **toekenning**.
Lees: "som **wordt** eerste + tweede"

Algoritme omzetten naar Java


```
System.out.print("Tik een getal in: ");  
eerste = keyboard.nextInt();  
System.out.print("Tik nog een getal in: ");  
tweede = keyboard.nextInt();  
som = eerste + tweede;  
System.out.print("Dit is de som: " + som);
```

Klaar? Bijna, we hebben nog een beetje voorbereidend werk nodig voor onze variabelen...

Algoritme omzetten naar Java

```
int som;  
int eerste;  
int tweede;
```

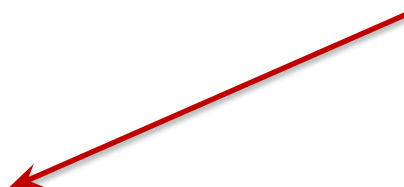
Voordat we variabelen mogen gebruiken moeten we ze eerst even vermelden en zeggen wat voor soort data ze zullen bevatten. We noemen dit **declareren** van de variabelen



```
System.out.print("Tik een getal in: ");  
eerste = keyboard.nextInt();  
System.out.print("Tik nog een getal in: ");  
tweede = keyboard.nextInt();  
som = eerste + tweede;  
System.out.print("Dit is de som: " + som);
```

Algoritme omzetten naar Java

```
int som;  
int eerste;  
int tweede;  
Scanner keyboard = new Scanner(System.in);  
System.out.print("Tik een getal in: ");  
eerste = keyboard.nextInt();  
System.out.print("Tik nog een getal in: ");  
tweede = keyboard.nextInt();  
som = eerste + tweede;  
System.out.print("Dit is de som: " + som);
```



Voordat we input van het keyboard kunnen vragen moeten we het keyboard aan het programma koppelen. Dat doe je op deze manier...

De volledige broncode

```
package berekeningen;
```

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

Dit import statement zorgt ervoor dat je het keyboard kunt gebruiken voor invoer.

```
/* We maken een programma dat 2 getallen inleest  
   en daarna de som ervan berekent */
```

```
public class Som {  
    public static void main(String[] args) {  
        int som;  
        int eerste;  
        int tweede;  
        Scanner keyboard = new Scanner(System.in);  
        System.out.print("Tik een getal in: ");  
        eerste = keyboard.nextInt();  
        System.out.print("Tik nog een getal in: ");  
        tweede = keyboard.nextInt();  
        som = eerste + tweede;  
        System.out.print("Dit is de som: " + som);  
    }  
}
```



Zoals in ons HelloWorld voorbeeld moeten we onze code in een **main methode** van een **class** plaatsen om ze te kunnen uitvoeren

Opdracht 3



Run het programma

- Voer nu de nodige stappen uit om de code te kunnen uitvoeren:
 - Maak een nieuw project **somOefening**
 - Maak een package **berekeningen**
 - Maak een klasse **Som**
 - Zet daarin de broncode
 - Compileer
 - Run →

A screenshot of a Java IDE's console window. The window has a title bar with 'Run' and 'Som'. On the left is a vertical toolbar with icons for running (green play button), stepping through (square), pausing (two vertical bars), and other debugging actions. The main area shows the output of the program: the path 'C:\Program ...' is highlighted in green. The text 'Tik een getal in: 15' and 'Tik nog een getal in: 35' are in green, indicating user input. The output 'Dit is de som: 50' is in black. The final line 'Process finished with exit code 0' is in blue. A cursor is visible at the end of the output line.

```
Run Som
"C:\Program ...
Tik een getal in: 15
Tik nog een getal in: 35
Dit is de som: 50
Process finished with exit code 0
```




• Blackboard > Basisbegrippen > Opdrachten > W1

– Basis:

- Product
- Leeftijd
- Bewerkingen
- BMI

Moeilijker voorbeeld: Hoger Lager



- Wat is het algoritme voor dit spel:

Geef een getal:100

Te groot!Geef een getal:50

Te klein!Geef een getal:60

Te groot!Geef een getal:55

Te groot!Geef een getal:54

Proficiat, u hebt het geraden na 5 gokken.

→ Probeer nu zelf het algoritme uit te schrijven!



Algoritme Hoger Lager



input en output (IO)

Drie **testen**: we vergelijken de variabelen **teZoekenGetal** en **gok**

1. Kies getal en schrijf weg, noem dit **teZoekenGetal**
2. Toon "Geef een getal" op output (scherm, bord, ...).
3. Lees input van keyboard en schrijf weg, noem dit **gok**.
4. Is **gok** gelijk aan **teZoekenGetal**?
 - Ja: Schrijf "proficiat, u hebt het geraden" op het scherm en stop.
5. Is **gok** kleiner dan **teZoekenGetal**?
 - Ja: Toon "Te klein" op output en keer terug naar stap 2
6. Is **gok** groter dan **teZoekenGetal**?
 - Ja: Toon "Te groot" op output en keer terug naar stap 2

Tussenresultaten die we even wegschrijven noemen we **variabelen**

Een **lus**

Algoritme Hoger Lager met teller!



1. Kies getal en schrijf weg, noem dit **teZoekenGetal**
2. *Schrijf 0 weg, noem dit **teller*** ←
3. Toon "Geef een getal" op output (scherm, bord, ...).
4. Lees input van keyboard en schrijf weg, noem dit **gok**.
5. *Verhoog **teller** met 1* ←
6. Is **gok** gelijk aan **teZoekenGetal**?
 - Ja: schrijf "proficiat, u hebt het geraden in " + **teller** + " keer " op het scherm en stop.
7. Is **gok** kleiner dan **teZoekenGetal**?
 - Ja: Toon "Te klein" op output en keer terug naar *stap 3*
8. Is **gok** groter dan **teZoekenGetal**?
 - Ja: Toon "Te groot" op output en keer terug naar *stap 3*

We hebben voor de teller een extra variabele nodig...

Een stap waarin we **rekenen** met deze variabele **teller**

Algoritme omzetten naar Java!

- Hoe?

- een stap uit het algoritme wordt een java instructie afgesloten met een ;
- output naar scherm: `System.out.print("tekst");`
- input van keyboard: `keyboard.nextInt();`
- test: `if (test) {`
 `// hier de "indien ja" instructies`
 `}`
- lus: `while (true) {`
 `// hier de herhaalde instructies`
 `}`



Opmerking: er zijn nog andere (en dikwijls betere) manieren in Java om lussen en testen te doen, dat zien we in de komende weken...

Algoritme omzetten naar Java!

1. Kies getal en schrijf weg, noem **teZoekenGetal**
2. Toon "Geef een getal" op output (scherm, bord, ...).
3. Lees input van keyboard en schrijf weg, noem dit **gok**.
4. Is **gok** gelijk aan **teZoekenGetal**?
 - Ja: schrijf "proficiat, u hebt het geraden" op het scherm en stop.
5. Is **gok** kleiner dan **teZoekenGetal**?
 - Ja: Toon "Te klein" op output en keer terug naar stap 3
6. Is **gok** groter dan **teZoekenGetal**?
 - Ja: Toon "Te groot" op output en keer terug naar stap 3

Algoritme omzetten naar Java!

Een waarde in een variabele wegschrijven doen we dus met een = teken. De variabele moet steeds aan de linkerkant staan. We noemen dit een **toekenning**.
Lees: "teZoekenGetal wordt 56"

teZoekenGetal = 56;

2. Toon "Geef een getal" op output (scherm, bord, ...).
3. Lees input van keyboard en schrijf weg, noem dit **gok**.
4. Is **gok** gelijk aan **teZoekenGetal**?
 - Ja: schrijf "proficiat, u hebt het geraden" op het scherm en stop.
5. Is **gok** kleiner dan **teZoekenGetal**?
 - Ja: Toon "Te klein" op output en keer terug naar stap 3
6. Is **gok** groter dan **teZoekenGetal**?
 - Ja: Toon "Te groot" op output en keer terug naar stap 3

Algoritme omzetten naar Java!

```
teZoekenGetal = 56;
```

```
System.out.print("Geef een getal: ");
```

3. Lees input van keyboard en schrijf weg, noem dit **gok**.
4. Is **gok** gelijk aan **teZoekenGetal**?
 - Ja: schrijf "proficiat, u hebt het geraden" op het scherm en stop.
5. Is **gok** kleiner dan **teZoekenGetal**?
 - Ja: Toon "Te klein" op output en keer terug naar stap 3
6. Is **gok** groter dan **teZoekenGetal**?
 - Ja: Toon "Te groot" op output en keer terug naar stap 3

Algoritme omzetten naar Java!

```
teZoekenGetal = 56;
```

```
System.out.print("Geef een getal: ");
```

```
gok = keyboard.nextInt();
```

Ook dit is een toekenning. We schrijven het getal dat op het keyboard wordt ingetypt weg naar de variabele **gok**.

4. Is **gok** gelijk aan **teZoekenGetal**?

- Ja: schrijf "proficiat, u hebt het geraden" op het scherm en stop.

5. Is **gok** kleiner dan **teZoekenGetal**?

- Ja: Toon "Te klein" op output en keer terug naar stap 3

6. Is **gok** groter dan **teZoekenGetal**?

- Ja: Toon "Te groot" op output en keer terug naar stap 3

Algoritme omzetten naar Java!

```
teZoekenGetal = 56;
```

```
System.out.print("Geef een getal: ");
```

```
gok = keyboard.nextInt();
```

```
if (gok == teZoekenGetal) {
```

- Ja: schrijf "proficiat, u hebt het geraden" op het scherm en stop.

5. Is **gok** kleiner dan **teZoekenGetal**?

- Ja: Toon "Te klein" op output en keer terug naar stap 3

6. Is **gok** groter dan **teZoekenGetal**?

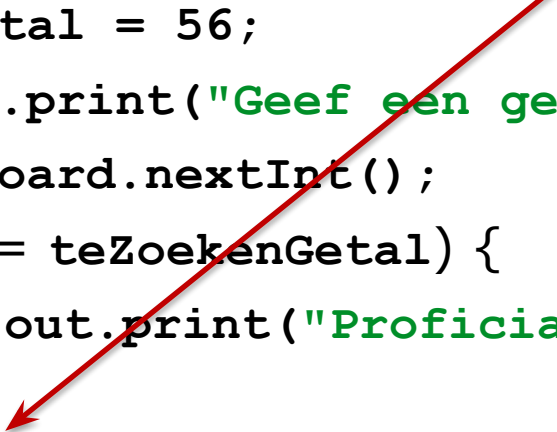
- Ja: Toon "Te groot" op output en keer terug naar stap 3

Om een gelijkheid te testen moeten we twee = tekens gebruiken.
Waarom zou één = teken niet werken?

Algoritme omzetten naar Java!

De **return** instructie zorgt ervoor dat we de applicatie hier laten stoppen...

```
teZoekenGetal = 56;
System.out.print("Geef een getal: ");
gok = keyboard.nextInt();
if (gok == teZoekenGetal) {
    System.out.print("Proficiat u hebt het geraden!");
    return;
}
```



5. Is **gok** kleiner dan **teZoekenGetal**?
 - Ja: Toon "Te klein" op output en keer terug naar stap 3
6. Is **gok** groter dan **teZoekenGetal**?
 - Ja: Toon "Te groot" op output en keer terug naar stap 3

Algoritme omzetten naar Java!

```
teZoekenGetal = 56;
System.out.print("Geef een getal: ");
gok = keyboard.nextInt();
if (gok == teZoekenGetal) {
    System.out.print("Proficiat u hebt het geraden!");
    return;
}
if (gok < teZoekenGetal) {
    System.out.print("Te klein! ");
}
if (gok > teZoekenGetal) {
    System.out.print("Te groot! ");
}
```




En die lus?

Algoritme omzetten naar Java!

Klaar? Bijna, we hebben nog een beetje voorbereidend werk nodig voor onze variabelen...

```
teZoekenGetal = 56;
while (true) {
    System.out.print("Geef een getal: ");
    gok = keyboard.nextInt();
    if (gok == teZoekenGetal) {
        System.out.print("Proficiat u hebt het geraden!");
        return;
    }
    if (gok < teZoekenGetal) {
        System.out.print("Te klein! ");
    }
    if (gok > teZoekenGetal) {
        System.out.print("Te groot! ");
    }
}
```



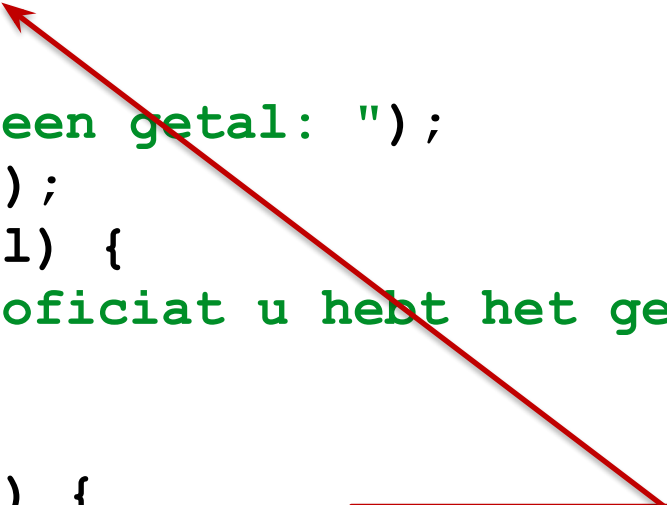
Algoritme omzetten naar Java!

```
int teZoekenGetal;  
int gok;  
teZoekenGetal = 56;  
while (true) {  
    System.out.print("Geef een getal: ");  
    gok = keyboard.nextInt();  
    if (gok == teZoekenGetal) {  
        System.out.print("Proficiat u hebt het geraden!");  
        return;  
    }  
    if (gok < teZoekenGetal) {  
        System.out.print("Te klein! ");  
    }  
    if (gok > teZoekenGetal) {  
        System.out.print("Te groot! ");  
    }  
}
```

Voordat we variabelen mogen gebruiken moeten we ze eerst even vermelden en zeggen wat voor soort data ze zullen bevatten. We noemen dit: **declareren** van de variabelen

Algoritme omzetten naar Java!

```
int teZoekenGetal;  
int gok;  
Scanner keyboard = new Scanner(System.in);  
teZoekenGetal = 56;  
while (true) {  
    System.out.print("Geef een getal: ");  
    gok = keyboard.nextInt();  
    if (gok == teZoekenGetal) {  
        System.out.print("Proficiat u hebt het geraden!");  
        return;  
    }  
    if (gok < teZoekenGetal) {  
        System.out.print("Te klein! ");  
    }  
    if (gok > teZoekenGetal) {  
        System.out.print("Te groot! ");  
    }  
}
```



Voordat we input van het keyboard kunnen vragen moeten we het keyboard koppelen. Dat doe je op deze manier...

Eindresultaat Hoger Lager

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

Dit import statement zorgt ervoor dat we het keyboard kunnen gebruiken.

```
public class HogerLager {
```

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

```
        int teZoekenGetal;
```

```
        int gok;
```

```
        Scanner keyboard = new Scanner(System.in);
```

```
        teZoekenGetal = 56;
```

```
        while (true) {
```

```
            System.out.print("Geef een getal: ");
```

```
            gok = keyboard.nextInt();
```

```
            if (gok == teZoekenGetal) {
```

```
                System.out.print("Proficiat u hebt het geraden!");
```

```
                return;
```

```
            }
```

```
            if (gok < teZoekenGetal) {
```

```
                System.out.print("Te klein! ");
```

```
            }
```

```
            if (gok > teZoekenGetal) {
```

```
                System.out.print("Te groot! ");
```

```
            }
```

```
        }
```

```
    }
```

```
}
```



Zoals in ons HelloWorld voorbeeld moeten we onze code in een **main methode** van een **class** plaatsen om ze te kunnen uitvoeren

Opdracht



- Test de werking van het programma HogerLager uit.
- Werkt alles naar behoren?
- Probeer het programma dan uit te breiden met de teller-functionaliteit
- Compileer en run



• Blackboard > Basisbegrippen > Opdrachten > W1

– Basis:

- Sommeren
- Omwisselen
- Reeksen

– Extra:

- Middelste
- Tafels

Samengevat



- Vooraf: afspraken, evaluatie, ...
- H1: Inleiding:
 - 3GL, Compileren, Interpreteren, JVM
- H2: Java Development Kit
- H3: Het eerste Java programma
 - HelloWorldApp, javac, java
- H4: het algoritme
 - Stappenplan
 - Variabelen, rekenen, IO, testen, lussen
 - Verfijnen tot Java instructies