Inhoudsopgave

OVERZICHT THEORIE

Introductie Java

- Introduction to computers: Liang 1.1 – 1.6
- A simple JAVA program: Liang 1.7 tot aan bladzijde 36, Listing 1.2
- Compiling, executing, style documentation and errors: Liang 1.8 – 1.10
- Elementary programming, introduction: Liang 2.1 en 2.2
- Invoer van de gebruiker krijgen: Liang 2.3

Variabelen

- Variabelen en datatypes: Liang 2.4 - 2.6
- Constanten: Liang 2.7
- Naamgeving van variabelen en constanten: Liang 2.8
- Numerieke datatypes en hun operatoren: Liang 2.9.1 - 2.9.3
- Operatoren vervolg: Liang 2.14 en 2.15
- Type casting: Liang 2.16
- Strings: Liang 4.4.1 – 4.4.6

Selections

- Booleans: Liang 3.1 en 3.2
- If en if-else: Liang 3.3 en 3.4
- Geneste if-else statements: Liang 3.5
- Logische operatoren: Liang 3.10
- Strings vergelijken: Liang 4.4.7
- Switch-statements: Liang 3.13 en 3.14

Math class en Methods

- Math class: Liang 4.1, 4.2.1 – 4.2.5
- Methoden: Liang 6.1 – 6.4

While en do-while loop

- While-loop: Liang 5.1, 5.2 en 5.6

For loop

- Do-while-loop en for-loop: Liang 5.7 - 5.9
- Output formatteren: Liang 4.6

Arrays

- Arrays: Liang 7.1, 7.2.1 - 7.2.6
- Arrays en methoden: Liang 7.6 en 7.7

Methods en parameters

- Methodes en waarden doorgeven aan methodes: Liang 6.5 en 6.6

1 INTRODUCTIE JAVA

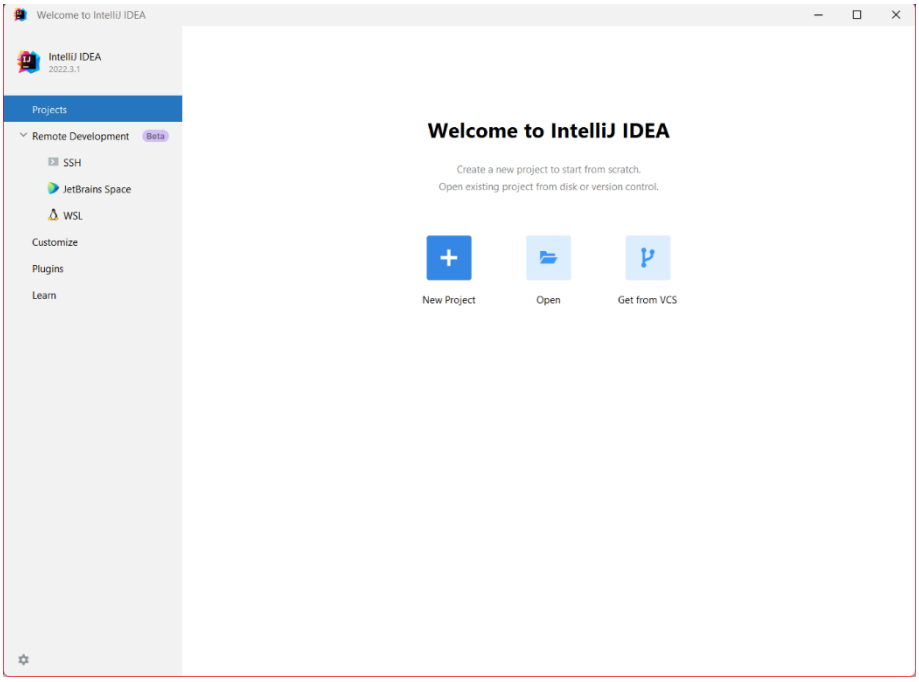
1.1 Wat is Java?

1. Log in bij LinkedIn Learning: https://www.linkedin.com/learning/. Klik op ***Aanmelden*** en log in met je HvA e-mailadres.
2. Bekijk What is java?

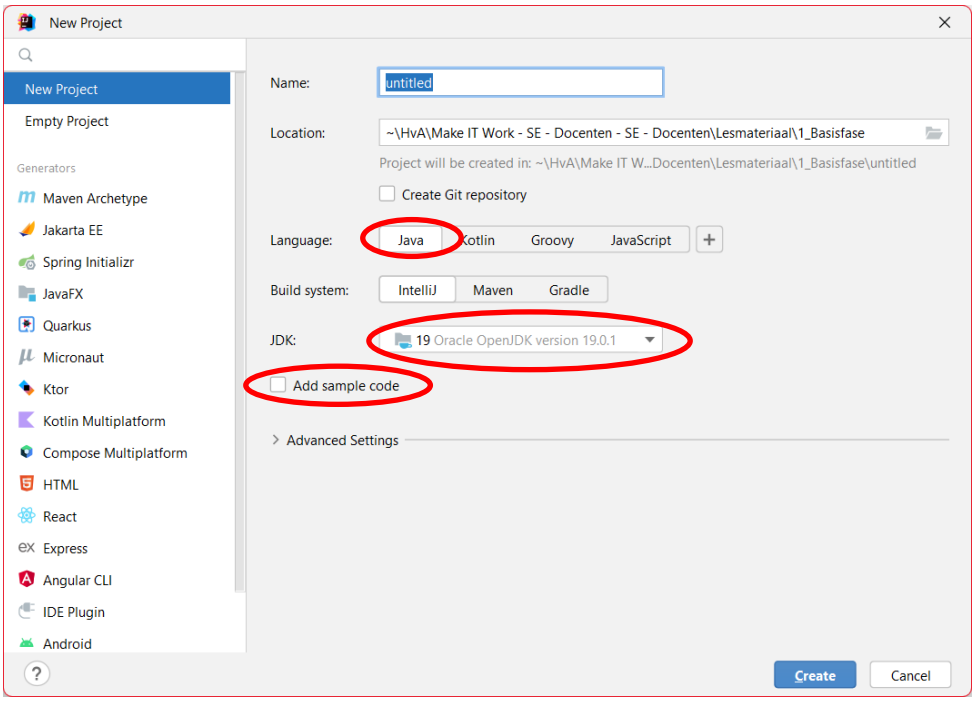
1.2 IntelliJ opstarten

1. Bekijk verder het onderdeel Exploring an integrated development environment (IDE)
2. Lees Liang 1.1 – 1.6.

   Let op: De volgende instructies moet je erg goed en erg zorgvuldig doornemen. Sla geen stap over, of doe niet iets anders als je niet zeker weet wat je doet.
   Let op: De schermafdrukken zijn gebaseerd op *IntelliJ Ultimate* versie 2022.3.1. Het kan zijn dat het scherm er bij jou anders uitziet.
3. Start *IntelliJ* op.
   Als het goed is dan krijg je het startscherm te zien, zoals hieronder:



4. Kies de optie New Project. Je ziet dan het volgende scherm (Het kan zijn dat je scherm minder mogelijkheden heeft.)

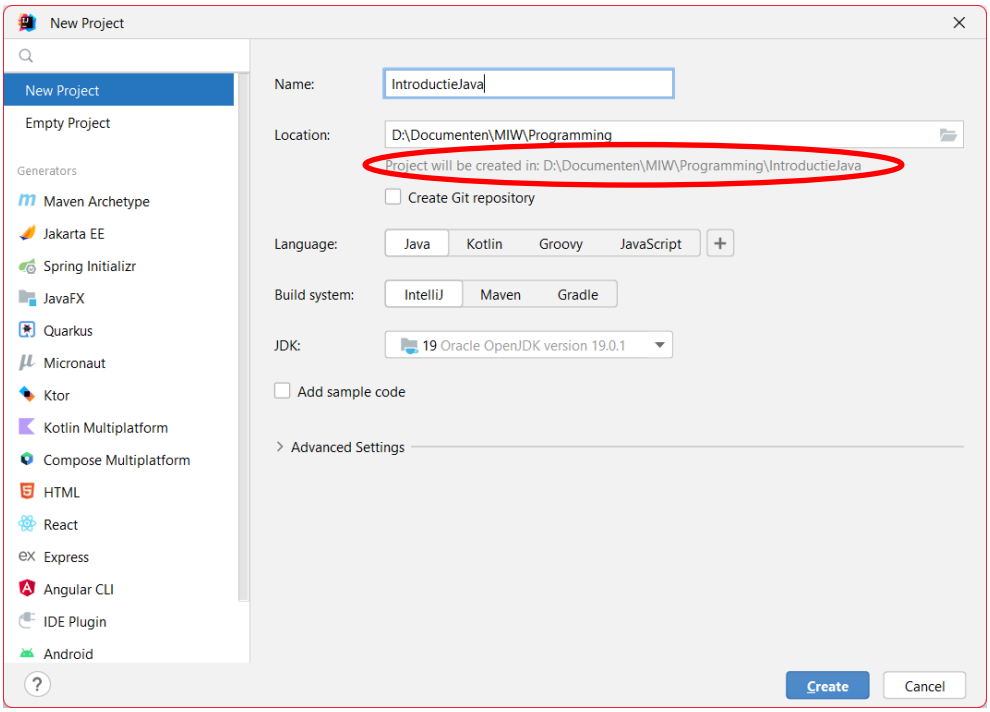


5. Zorg dat bij ***Language*** de optie Java geselecteerd is.
6. Als het goed is heb je bij ***JDK*** een versie staan (18, 19 of 20, afhankelijk van de versie die je geïnstalleerd hebt).
7. Haal het vinkje weg bij ***Add sample code***.

Let op: het is heel erg belangrijk dat nieuwe projecten in folders komen, die nog niet bestaan en ook niet in een andere folder zit waar al programma's staan. Maak bijvoorbeeld een folder **MIW** aan met daaronder een folder **Programming** en daaronder stop je ieder project in een eigen folder!
Let op: het is gebruikelijk, hoewel niet verplicht, om de folder en het project dezelfde naam te geven.

---

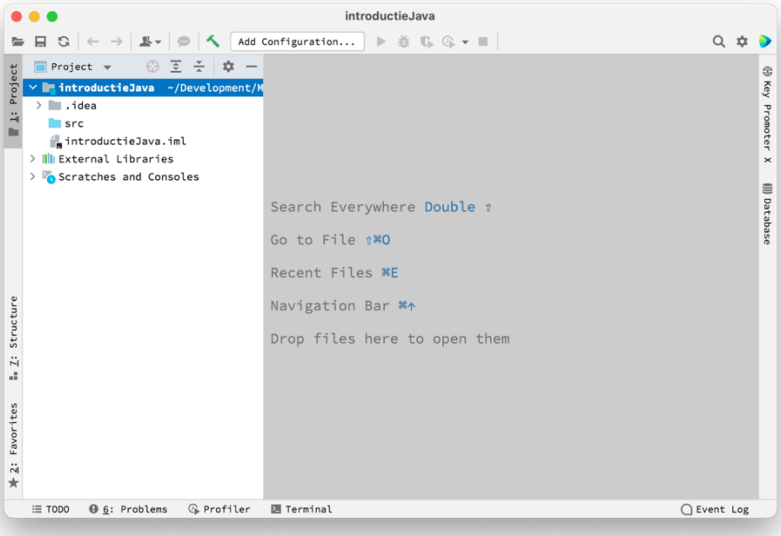
8. Geef je project de naam ***IntroductieJava*** en zorg dat er ook een folder met die naam aangemaakt wordt, zoals hieronder:



↑ Je kiest een bestaande folder bij ***Location***.
↑ Je geeft bij ***Name*** het project de naam ***IntroductieJava***.
↑ IntelliJ maakt dan automatisch een nieuwe folder aan met een naam, die hetzelfde is als de naam van je project.

9. Klik op Create.

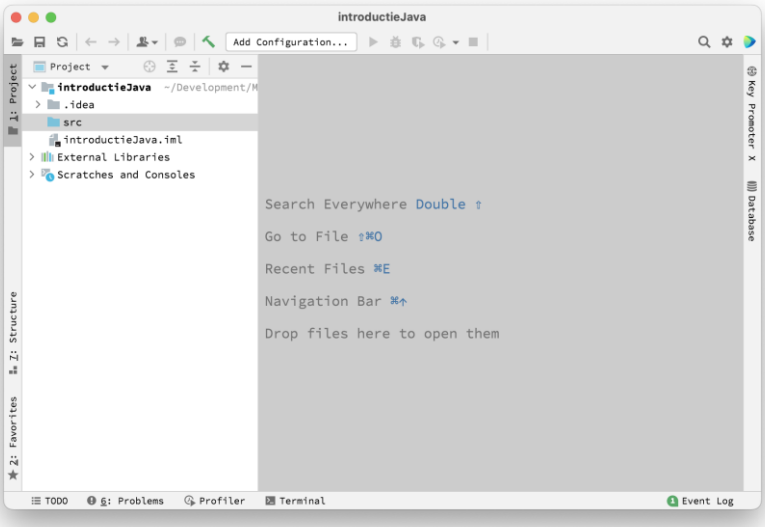
10. Je scherm ziet er nu ongeveer zo uit (op een *Windows* pc ziet het er mogelijk iets anders uit):



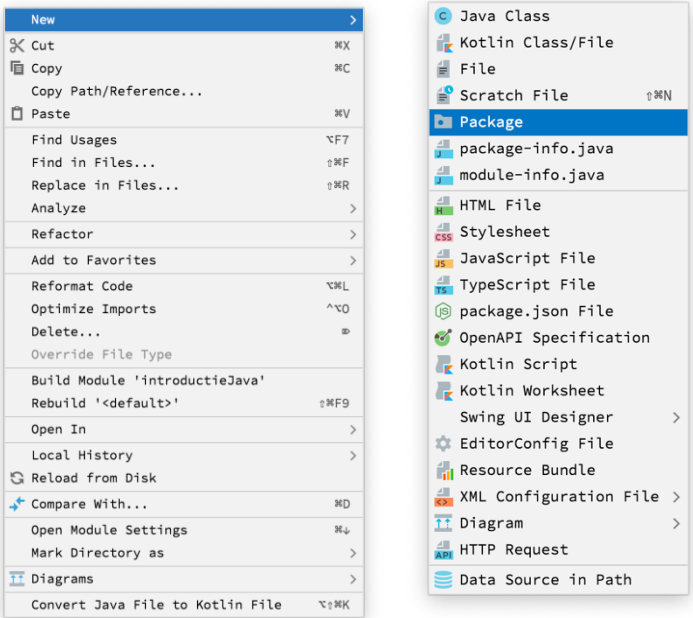
1.3 Packages en Classes

Je ***Integrated Development Environment*** staat nu klaar voor gebruik. Je gaat nu eerst een structuur neerzetten voor het bestand dat je gaat gebruiken om je eerste programma te maken.

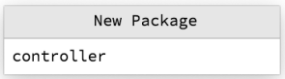
1. Klik op > voor ***IntroductieJava*** om de onderdelen uit te klappen.
2. Klik vervolgens op de map met `src`, zoals hieronder.



3. Klik met je rechtermuisknop om een menu te tonen.
4. Kies New en dan Package zoals hieronder.

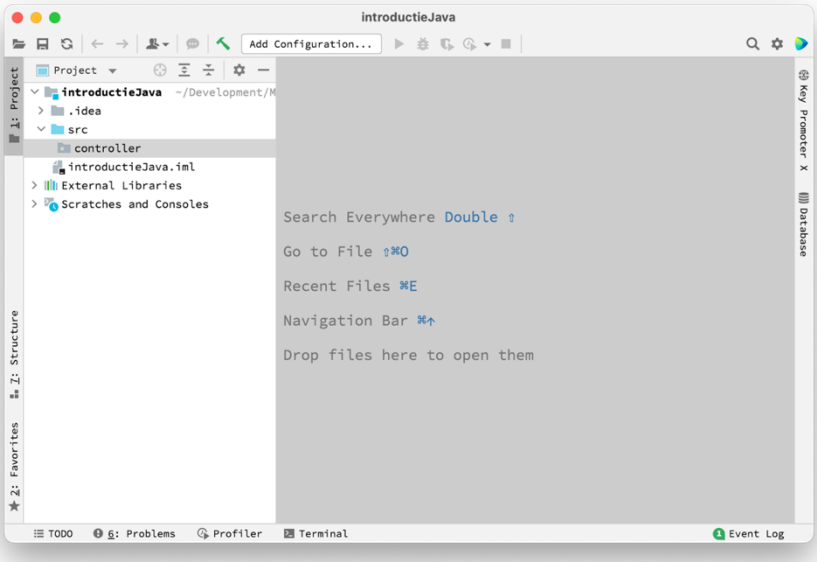


5. Type ***controller*** in de ***New Package*** dialoog, zoals hieronder.

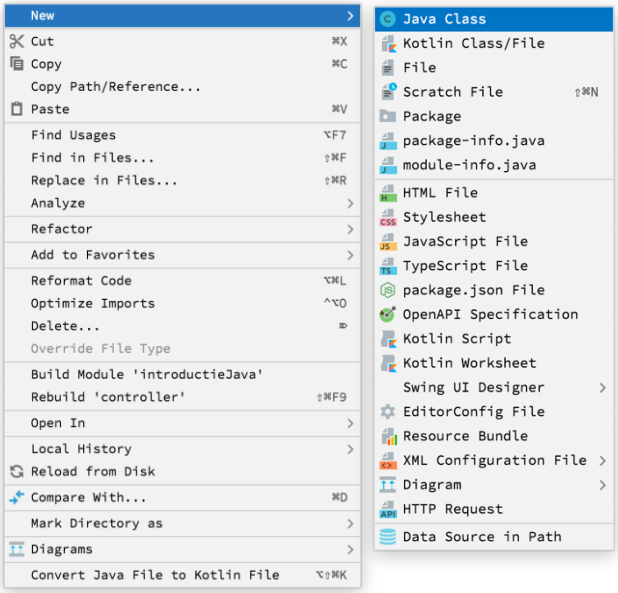


Een *package* wordt gebruikt om de verschillende klassen te structureren.
Later zullen we bij MIW gebruik maken van de ***Model-View-Controller***- indeling. In ***Programming*** zullen we alleen gebruik maken van een controller.
De conventie is dat we de naam van een package met kleine letters schrijven (pas op, want Java is hoofdlettergevoelig, dus controller, Controller en CONTROLLER zijn drie verschillende dingen).

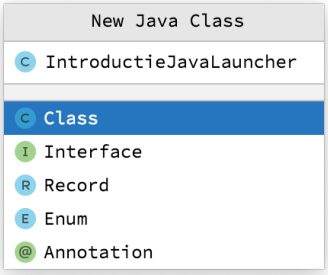
6. Druk op Enter en je krijgt het volgende scherm:



7. Zorg dat `controller` geselecteerd is en klik dan met je rechtermuisknop, selecteer dan weer New en Java Class, zoals hieronder.



8. Type `IntroductieJavaLauncher` in de dialoog zoals hieronder en druk op Enter



Toelichting: Klassen in Java beginnen altijd met een hoofdletter en eventuele volgende delen, die een zelfstandig woord bevatten ook. Dit wordt *CamelCase* genoemd (zie ook Liang 2.8).

Binnen de basiscursus zullen we de klasse waar het programma draait een *Launcher* noemen, zodat we tot de ***IntroductieJavaLauncher*** komen.
Je ziet op je scherm de volgende code:

```
package controller;

public class IntroductieJavaLauncher {
}
```

↑ Er is nu dus een klasse `IntroductieJavaLauncher` in de package `controller`.

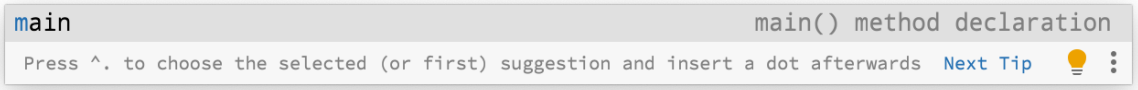
1.4 De main methode

Om een programma te kunnen draaien (*runnen*) heb je een zogeheten *main-methode* nodig.

1. Zorg dat er een witregel komt tussen de twee accolades.

```
package controller;

public class IntroductieJavaLauncher {

}
```

2. Type vervolgens een m.
   Je ziet dat het volgende verschijnt:



IntelliJ toont direct een suggestie, de enige optie die op deze plek zinvol is.

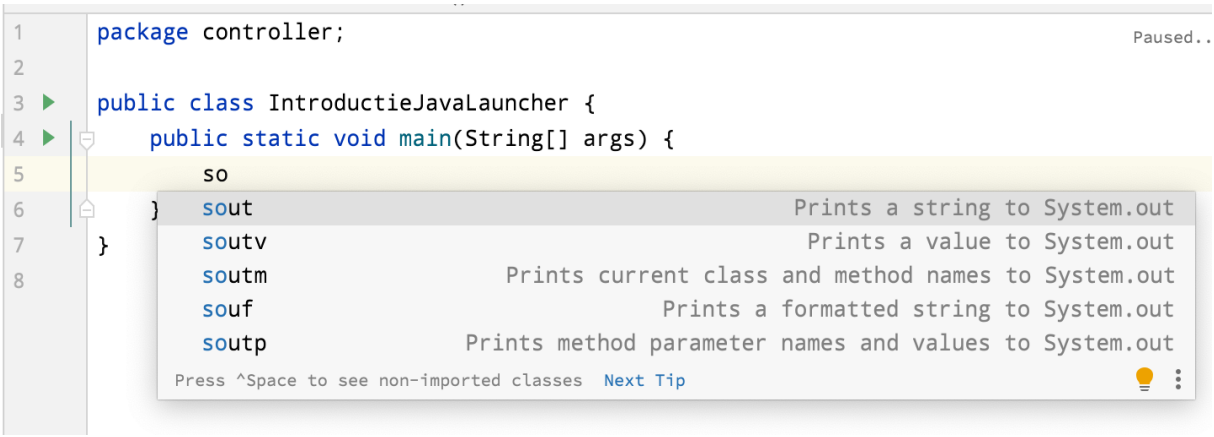
3. Druk op Enter en de volgende code verschijnt:

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {

    }
}
```

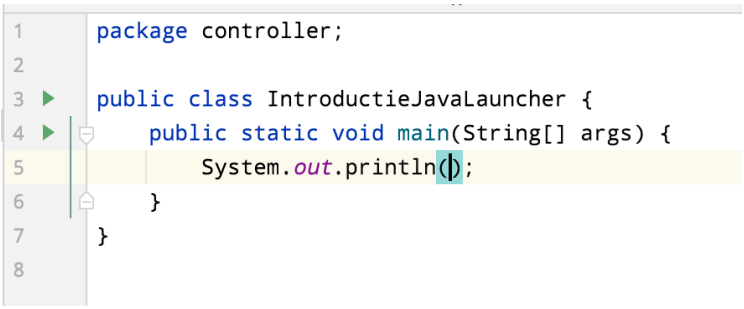
↑ Op dit moment gaan we niet in op wat er allemaal staat, dat komt later in de opleiding aan de orde.

1.5 Output genereren

1. Zorg dat je cursor aan het begin van de lege regel staat (als het goed is, dan is dat regel 5) en type so
   Je ziet dat het volgende verschijnt:



2. Druk weer op Enter (zorg dat je `sout` hebt geselecteerd) en het volgende verschijnt:

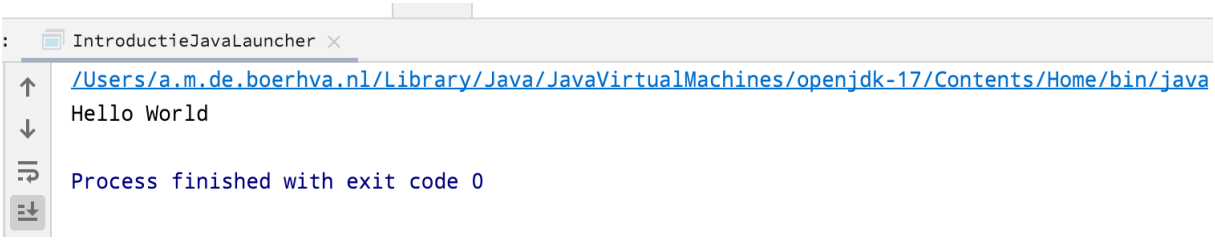


3. Type dan `"Hello World!"` (tussen de haakjes achter `println`) en je krijgt de volgende code:

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

4. Je bent nu eindelijk zover dat je je eerste programma van de opleiding kunt draaien. Klik op een groene driehoek links van de code (op regel 3 of op regel 4), zie de afbeelding hierboven. Je krijgt dan het volgende te zien:

5. Klik op de bovenste optie (`Run 'IntroductieJava.main()'`)
   Als het goed is, zie je nu onderaan je scherm het volgende verschijnen:



1.6 Oefening

1. Lees Liang 1.7 tot aan bladzijde 36, Listing 1.2.
   Liang maakt voor elk nieuw voorbeeld een eigen klasse met een eigen main-methode. In deze cursus gebruiken we dezelfde `Launcher`, omdat dat veel tijd bespaart.
2. Typ (of knip en plak) de regels 3, 4 en 5 van Listing 1.2 in Liang in je eigen code zodat het resultaat als volgt is:

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.out.println("Programming is fun!");
        System.out.println("Fundamentals First");
        System.out.println("Problem Driven");
    }
}
```

   Je kunt je oude code, die je niet meer wilt gebruiken, deactiveren door het te 'commenten' (bij gebrek aan een beter Nederlands woord) door aan het begin van de regel twee slashes (`//`) te plaatsen.
3. Zet twee slashes aan het begin van regel 5, zodat het er zo uit ziet:

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {
//        System.out.println("Hello World!");
        System.out.println("Programming is fun!");
        System.out.println("Fundamentals First");
        System.out.println("Problem Driven");
    }
}
```

4. Klik weer op een groene driehoek en 'run' het programma.
   Als je dit programma nu draait, dan krijg je een resultaat zoals dat in Listing 2 van Liang is beschreven.
5. Je kunt Listing 1.3 overslaan. Hier komen we later op terug.
6. Lees nu Liang 1.8 - 1.10.

Let op: In 1.8 heeft Liang het over een javac command. Dat gebruiken we niet, we doen het zoals hierboven beschreven is met *IntelliJ*.
Let op: In 1.9.2 heeft Liang het over inspringingen (indentations). *IntelliJ* regelt de inspringing voor je, standaard staat deze op vier spaties. Dit is ook de standaard binnen MIW. Liang gebruikt in zijn voorbeelden een inspringing van twee spaties.

1.7 Opdracht

Doe nu de opdracht **IntroJava.1-Kerstgroet** (niveau: basis) die je op de DLO/Brightspace kunt vinden.

2 VARIABELEN

2.1 Variabelen declareren en vullen

1. Zet de drie regels code in comments, zodat je code er als volgt uit ziet:

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {
//        System.out.println("Hello World!");
//        System.out.println("Programming is fun!");
//        System.out.println("Fundamentals First");
//        System.out.println("Problem Driven");
    }
}
```

2. Bekijk op ***LinkedIn Learning:*** Primitive data types
3. Ga verder met: Data types and variables
4. Ga verder met: Strings
5. Je kunt ***Using indexes with strings*** overslaan.
6. Ga verder met: Concatenating strings
7. Lees Liang 2.1 en 2.2 tot aan Listing 2.1.
8. Voer de code op regel 3 – 14 van Listing 2.1 in. Dit ziet er dan als volg uit:

```java
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {
//        System.out.println("Hello World!");
//        System.out.println("Programming is fun!");
//        System.out.println("Fundamentals First");
//        System.out.println("Problem Driven");
        double radius; // Declare radius
        double area; // Declare area

        // Assign a radius
        radius = 20; // radius is now 20

        // Compute area
        area = radius * radius * 3.14159;

        // Display results
        System.out.println("The area for the circle of radius " +
                radius + " is " + area);
    }
}
```

9. Run je programma en check je resultaat met Liang.

2.2 Oefening

1. Je gaat nu een nieuwe oefening doen en de vorige code heb je niet meer nodig (maar die wil je misschien wel bewaren). Hierboven heb je een enkele regel ‘gecomment’. Het is ook mogelijk om een heel blok van code in comments te zetten.
   Selecteer het blok met de vorige code zoals hieronder: (tussen de accolades van `main`)

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {

        double radius; // Declare radius
        double area; // Declare area

        // Assign a radius
        radius = 20; // radius is now 20

        // Compute area
        area = radius * radius * 3.14159;

        // Display results
        System.out.println("The area for the circle of radius " +
                radius + " is " + area);
    }
}
```

2. Druk nu in Windows ctrl + shift + / of in Mac op option + command + / en je krijgt het volgende te zien:

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {
/*
        double radius; // Declare radius
        double area; // Declare area

        // Assign a radius
        radius = 20; // radius is now 20

        // Compute area
        area = radius * radius * 3.14159;

        // Display results
        System.out.println("The area for the circle of radius " +
                radius + " is " + area);
*/
    }
}
```

3. Probeer het nu zelf:
4. Declareer een variabele van het type *String* om je voornaam in op te slaan.
5. Declareer een variabele van het type *int* om een getal in op te slaan.
6. Ken waarden toe aan deze variabelen.
7. Declareer een variabele van het type *int* om het kwadraat van het getal in op te slaan.

8. Ken het kwadraat van het eerste getal toe aan deze nieuwe variabele (voor de zekerheid: een kwadraat van een getal wordt berekend door het getal met zichzelf te vermenigvuldigen).
9. Toon de volgende zin op het scherm (met de juiste waarden op de stippen):

```
Hallo …, het kwadraat van … is …
```

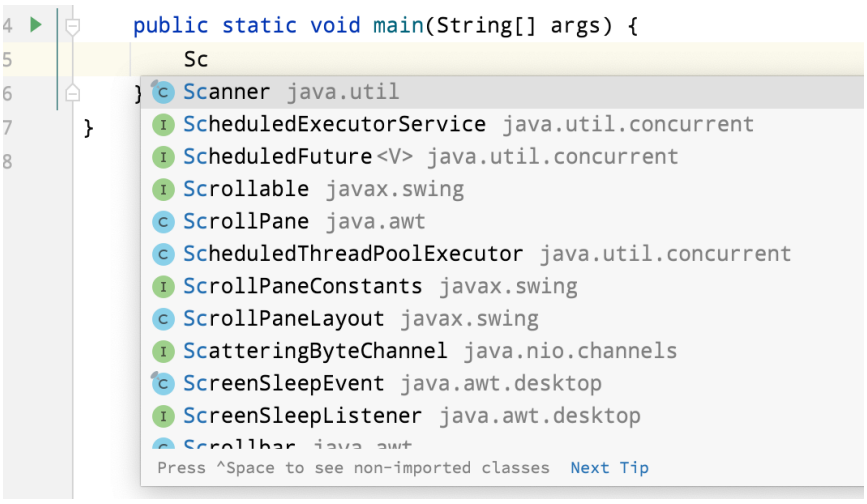
10. Onze uitwerking:

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {
        String voornaam;
        int getal;
        voornaam = "Olivier";
        getal = 4;
        int kwadraat;
        kwadraat = getal * getal;
        System.out.println("Hallo " + voornaam + ", het kwadraat van " + getal + "
is " + kwadraat);
    }
}
```

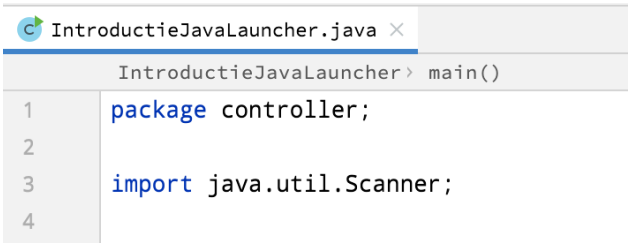
11. Ga verder met Liang 2.2.

2.3 Invoer van de gebruiker krijgen

1. Bekijk op ***LinkedIn Learning***: Input and output in Java
2. Lees Liang 2.3 tot aan Listing 2.2.
3. Type Sc op de regel na `main(String[] args) {`
   en je ziet dat *IntelliJ* weer met enkele suggesties komt. De eerste is `Scanner`.



4. Druk op Enter
Zoals je kunt zien wordt er meteen een `import` gedaan (regel 1 van Listing 2.2, dit is bij jou regel 3). *IntelliJ* doet dit automatisch voor je!



5. Voer de rest van de code van Listing 2.2 in en draai (run) je programma.

Let op: Liang gebruikt in regel 9 van Listing 2.2 `System.out.print` en niet `System.out.println`. Dit heeft als voordeel dat de invoer van de gebruiker achter de toelichtende tekst komt te staan. Het is een goede gewoonte om hier altijd `System.out.print` te gebruiken.
Let op: Het is aan de programmeur om duidelijk aan te geven wat er van de gebruiker wordt verlangd door altijd vóór een scanner een print-statement te gebruiken.

6. Lees de rest van Liang 2.3 en probeer de code in Listing 2.3 uit.

Let op: De Scanner heeft zo zijn eigenaardigheden. Zie onderstaand voorbeeld:

```
Scanner scanner = new Scanner(System.in);
System.out.print("Geef een kommagetal: ");
double eerste = scanner.nextDouble();
System.out.println("1: " + eerste);

System.out.print("Geef een tekst: ");
String tweede = scanner.nextLine();
System.out.println("2: " + tweede);

System.out.print("Geef een geheel getal: ");
int derde = scanner.nextInt();
System.out.println("3: " + derde);
```

Technisch is er niets mis met deze code, maar bij de invoer gaat het toch fout (zie hieronder). Als je het programma draait en je voert dan 3,14 in, dan krijg je het volgende te zien:

```
Geef een kommagetal: 3,14
1: 3.14
Geef een tekst: 2:
Geef een geheel getal:
```

Wat gaat hier mis?
De gebruiker drukt na de 3,14 op de Enter toets. Deze wordt gezien als een karakter. `scanner.nextDouble()` leest alleen ***3,14*** en de Enter blijft in de invoerbuffer staan.

De volgende scanner, `scanner.nextLine()` ziet de Enter staan en leest deze. Er staat verder geen tekst in de buffer en de variabele ***tweede*** wordt daarmee een lege *String*. Vandaar dat er in de uitvoer na ***2:*** niets meer staat. Vervolgens wordt gevraagd om een geheel getal.
De oplossing hiervoor is een dummy `scanner.nextline()` te doen ná de `scanner.nextDouble()` zodat de overbodige Enter wordt verwijderd.

Let op: Je hoeft maar één instantie van de *Scanner* te definiëren en te instantiëren. Dit object kun je dan steeds weer opnieuw gebruiken voor het krijgen van invoer.

Opmerking: Een fout die vaak optreedt komt doordat de notatie van doubles afhankelijk is van de taal-instellingen van de gebruiker. In bovenstaande invoer-plaatje valt al meteen op dat de gebruiker 3,14 heeft ingetypt ("Nederlands"), maar soms zal 3.14 moeten worden ingevoerd. Dit is hier en nu niet gemakkelijk op te lossen. Later kunnen we daar bijvoorbeeld *exception handling* voor gebruiken.
Opmerking 2: De Scanner klasse heeft ook een methode next(). Deze leest een stuk tekst in tot aan de spatie. Dus als een gebruiker Tom Poes intypt, dan leest de methode next() alleen de tekst “Tom”. De tekst “Poes” blijft dan in de buffer staan. Na nog een keer de methode next() aan te roepen wordt de tekst “Poes” gelezen. De Enter blijft echter nog steeds in de buffer staan. Let dus goed op en gebruik next() alleen als je een stuk van de tekst wilt lezen.

2.4 Namen, variabelen en datatypes

1. Lees Liang 2.4 - 2.6.

2.5 Constanten

1. Lees Liang 2.7.
2. Om de code van Listing 2.4 uit te proberen, kun je de code van Listing 2.3 commenten en de code van Listing 2.2 uncommenten. Dit doe je door het hele blok (inclusief `/*` en `*/`) te selecteren en weer ctrl + shift + / of option + command + / in te drukken. Je kunt dan de code aanpassen conform Listing 2.4.

2.6 Naamgeving van variabelen en constanten

1. Lees Liang 2.8.

2.7 Numerieke datatypes en hun operatoren

1. Lees Liang 2.9.1 - 2.9.3.
   Let op: onder aan pagina 68 staat een opmerking over het delen van twee gehele getallen (van het type *int*). Het resultaat is dan een geheel getal! Alles na de decimale

komma wordt weggelaten. Er wordt dus niet afgerond. Dit is in het begin een veel gemaakte vergissing. Probeer het volgende:

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {
        System.out.println( 5/2 );     // 2
        System.out.println( 5/2.0 );   // 2.5
    }
}
```

2. Probeer de code van Listing 2.5 uit. In regel 10 in Liang is sprake van *integer division*. Als je 500 seconden deelt door 60, dan is het resultaat na *integer division* gelijk aan 8.
3. Je kunt 2.10 **overslaan**, daar komen we later op terug.
4. Probeer de code van Listing 2.6 uit.
5. Je kunt 2.12 en 2.13 **overslaan**.
6. Lees Liang 2.14 en 2.15.

2.8 Type casting

1. Lees Liang 2.16.
   Let op: zet de typecasting op de juiste plaats en gebruik eventueel haken. Probeer het volgende:

```
package controller;

public class IntroductieJavaLauncher {
    public static void main(String[] args) {
        System.out.println( (int)2.5/0.5 );     // 4.0
        System.out.println( (int)(2.5/0.5) );   // 5
    }
}
```

2.9 Strings en hun methoden

1. Lees Liang 4.4.1 - 4.4.6.
2. Probeer het volgende:

```
System.out.println( 3 + 3 );     // 6
System.out.println( "3" + "3" );// 33
System.out.println( "3" + 3 );  // 33
System.out.println( 3 + "3" );  // 33
System.out.println( '3' + 3 );  // 54 Hoe kan dat? (optioneel  Liang 4.3.3)
```

2.10 Opdrachten

Doe de volgende opdracht:
IntroJava.2-Leeftijdberekening (niveau: basis)

3 SELECTIONS

3.1 Booleans

1. Lees Liang 3.1 en 3.2.

3.2 If en if-else statements

1. Lees Liang 3.3 en 3.4.
2. Kijk naar: if in Java.

Oefening stemmen

Schrijf nu zelf een programma dat vraagt om een leeftijd en teruggeeft of je al mag stemmen (vanaf 18 jaar) of niet.

Uitwerking: stemmen

```
import java.util.Scanner;

public class OefeningLauncher{
    public static void main(String[] args) {
        Scanner invoer = new Scanner(System.in);
        System.out.println("Geef je leeftijd:");
        int leeftijd = invoer.nextInt();
        if(leeftijd < 18) {
            System.out.println("Je mag nog niet stemmen!");
        } else {
            System.out.println("Je mag stemmen!");
        }
    }
}
```

3.3 Geneste if-else statements

1. Lees Liang 3.5 en optioneel 3.6.

Oefening: raad getal

Schrijf een programma dat vraagt om een getal te raden (b.v. onder de 10) dat de computer "in gedachte heeft". Mogelijke antwoorden zijn "Goed!", "Jammer: te laag" of "Jammer: te hoog", gevolgd door wat het te raden getal is.

Uitwerking: raad getal

```
import java.util.Scanner;

public class OefeningLauncher{
    public static void main(String[] args) {
        final int TE_RADEN_GETAL = 7;
        Scanner invoer = new Scanner(System.in);
        System.out.println("Geef het te raden getal:");
        int geraden = invoer.nextInt();

        if (geraden == TE_RADEN_GETAL) {
            System.out.println("Dat is goed!!");
        } else if (geraden > TE_RADEN_GETAL) {
            System.out.println("Jammer: Te hoog!");
        } else {
            System.out.println("Jammer: Te laag!");
        }

        System.out.println("Het getal was " + TE_RADEN_GETAL);
    }
}
```

3.4 Logische operatoren

1. Lees Liang 3.10.

Let op: De wiskundige expressie *a < uitkomst < b* moet je in Java schrijven als:
`a < uitkomst && uitkomst < b`
Let op: Een veel gemaakte fout is dat de operator *is-gelijk-aan* niet wordt geschreven als `==` , maar als `=` . Deze laatste wordt alleen gebruikt voor toekenning van een waarde aan een variable. (`aantalAppels = 25`)
Let op: *Groter gelijk* (≥) is `>=` , *kleiner gelijk* (≤) is `<=` en *ongelijk* (≠) is `!=` .

Oefening: leerplichtig
Schrijf nu een programma dat vraagt om een leeftijd en teruggeeft of je verplicht naar school moet (vanaf 4 jaar tot je 16e) of niet. Doe dit met 1 if statement.

Uitwerking: leerplichtig

```
import java.util.Scanner;

public class OefeningLauncher{
    public static void main(String[] args) {
        Scanner invoer = new Scanner(System.in);
        System.out.println("Geef je leeftijd:");
        int leeftijd = invoer.nextInt();
        if (leeftijd < 4 || leeftijd > 16) {
            System.out.println("Je hoeft niet naar school!");
        } else {
            System.out.println("Je moet naar school gaan!");
        }
    }
}
```

3.5 Strings vergelijken

1. Lees Liang 4.4.7.
   Let op: als je in een if-statement twee strings met elkaar wil vergelijken, bijvoorbeeld een ingevoerde cursusnaam met een gewenste cursusnaam, dan moet het **niet** zo:

```
    if (ingevoerdeCursus == "Programming") { //FOUT
        System.out.println("Dit is het juiste antwoord!");
    } else {
        System.out.println("Dit is niet het juiste antwoord!");
    }
```

   maar **wel** met behulp van de `equals`() methode:

```
    if (ingevoerdeCursus.equals("Programming") ) { //GOED
        System.out.println("Dit is het juiste antwoord!");
    } else {
        System.out.println("Dit is niet het juiste antwoord!");
    }
```

3.6 Switch-statements

1. Lees Liang 3.13 en 3.14.

Oefening: weekdag

Schrijf een programma dat vraagt om een weekdag (1 – 7) en de weekdag teruggeeft (maandag – zondag), of anders (als de gebruiker geen 1 – 7 invoert) een "onbekende dag".

Uitwerking: weekdag

```
import java.util.Scanner;

public class OefeningLauncher
{
    public static void main(String[] args) {
        Scanner invoer = new Scanner(System.in);
        System.out.println("Geef een dagnummer (1 - 7)");
        int dagNummer = invoer.nextInt();

        System.out.print("Dat is een ");
        switch (dagNummer) {
            case 1:
                System.out.println("maandag");
                break;
            case 2:
                System.out.println("dinsdag");
                break;
            case 3:
                System.out.println("woensdag");
                break;
            case 4:
                System.out.println("donderdag");
                break;
            case 5:
                System.out.println("vrijdag");
                break;
            case 6:
                System.out.println("zaterdag");
                break;
            case 7:
                System.out.println("zondag");
                break;
            default:
                System.out.println("onbekende dag");
        }
    }
}
```

2. **Optioneel**: Er zijn twee "moderne" varianten van `switch` (*switch statement* en *switch expression*) waarbij bijvoorbeeld `break` niet meer nodig is. Vind dit op internet en herschrijf ***oefening weekdag*** hiermee.

3.7 Opdrachten

Doe de volgende opdrachten:
Selections.1-BMI berekening (niveau: basis).
Selections.2-Verkoopprijs (niveau: basis).

4 MATH CLASS EN METHODS

4.1 De Math klasse

1. Lees Liang 4.1, 4.2.1 - 4.2.5.
   Tip: de formule onderaan pagina 146 is erg nuttig!

Oefening: afronden

De Math klasse heeft een methode Math.round(), die kommagetallen afrondt naar een geheel getal. Er is echter geen methode, die kommagetallen afrondt naar een gegeven aantal cijfers achter de komma. Dus er is geen methode, die een kommagetal tot drie cijfers achter de komma afrondt. Daar heb je een truc voor nodig.
Stel, je wilt het getal 3,14159265 afronden tot drie cijfers achter de komma. Dat is dus 3,142. De truc is om het getal te vermenigvuldigen met 1000,0. Hierdoor verschuif je de komma drie plaatsen. Het resultaat is dan 3141,59265. Als je dit getal afrondt met de Math.round() methode, dan krijg je 3142. Deel dit vervolgens weer door 1000,0 en je krijgt 3,142. Door een getal met een macht van 10 te vermenigvuldigen kun je de komma 'opschuiven', vervolgens afronden en weer 'terugschuiven'.
Schrijf een programma dat vraagt om een kommagetal. Rond het gegeven getal af op twee cijfers achter de komma en druk dit af.

Uitwerking: afronden

```
import java.util.Scanner;

public class OefeningLauncher{
    public static void main(String[] args) {
        Scanner invoer = new Scanner(System.in);
        System.out.println("Geef een getal (bv 12,345))");
        double ingevoerdGetal = invoer.nextDouble();
        double afgerondGetal = Math.round(100.0*ingevoerdGetal)/100.0;
        System.out.println(afgerondGetal);
    }
}
```

Oefening: random

Schrijf een programma dat een willekeurig geheel getal geeft; minimaal 2, maximaal 10. Gebruik `Math.random()`. Bedenk dat deze methode een *double* geeft van 0 tot 1; bekijk de uitleg en de formule van pagina 146 nog een keer.

Uitwerking: random

```
public class OefeningLauncher{
    public static void main(String[] args) {
       int willekeurigGetal = (int) (Math.random() * 9) + 2;
        System.out.println("Het getal is: "+ willekeurigGetal);
    }
}
```

4.2 Methoden

1. Lees Liang 6.1 - 6.3.

2. Probeer de code van Listing 6.1.
   Let op: In de code staan twee methoden: de `main` methode (regel 3 – 9) en de `max` methode (regel 12 – 21). Zorg ervoor dat de methoden volledig 'los' van elkaar gedefinieerd worden. De ene methode mag dus niet binnen de accolades van een andere methode staan. Dit zal in het begin een veel voorkomende fout zijn! Hieronder nog een ander voorbeeld:

```
1  package controller;
2
3  public class IntroductieJavaLauncher
4  {
5      public static void main(String[] args) {
6          //In de main methode: programma start altijd hier
7
8          //roep max methode aan; de returnwaarde komt in grootste:
9          int grootste = max( a: 33,  b: 44);
10         System.out.println("Grootste is: " + grootste);
11     }//end of main
12
13     public static int max(int a, int b) {
14         //In de max method die door main wordt aangeroepen
15         int theMax;
16         if (a > b) {
17             theMax = a;
18         } else {
19             theMax = b;
20         }
21         return theMax; //springt terug naar main en returnt waarde
22     }//end of max()
23
24 }//end of IntroductieJavaLauncher
25
```

3. Lees Liang 6.4.
   Kijk goed naar het verschil tussen de code in Listing 6.2 en Listing 6.3. (void versus char)

4. Ook leuk en als je tijd over hebt: bekijk heel chapter 5 van ***LinkedIn Learning***, te beginnen met functions conceptually.

4.3 Opdrachten

Doe de volgende opdrachten:
MathEnMethodes.1-BMI-vervolg (niveau: basis).
MathEnMethodes.2-Money exchange (niveau: basis).
MathEnMethodes.3-Methods (niveau: gemengd).

5 WHILE EN DO-WHILE LOOP

5.1 De while loop

1. Lees Liang 5.1 en 5.2.
   Je kunt de code in Listing 5.1 zelf uitproberen. Maak hiervoor een nieuw project aan. Noem dit **WhileLoop** (in de folder **WhileLoop**) en maak een `package controller` aan en klasse **WhileLoopLauncher** met een `main` methode. Je kunt dan alle code van Listing 5.1 regel 5 – 21 kopiëren.
2. Lees Liang 5.4 – 5.6.
   Je kunt de code in Listing 5.5 en Listing 5.6 zelf uitproberen.
3. ***LinkedIn Learning*** : while loops conceptually en while loops in java

Oefening: while random

Pas ***oefening random*** zó aan dat er net zolang willekeurige getallen worden gegeven totdat het maximale getal is "getrokken". Maak gebruik van de constanten (finals) `MINIMUM` en `MAXIMUM`. Dus geen *magic numbers* meer. De getallen 2 en 10 noemen we *magic numbers*.
Het grote voordeel van het gebruik van constanten is dat je hierdoor maar op één plek in je code de waarde hoeft aan te passen als je het wilt veranderen.
Maak twee versies met de twee verschillende while loops. Wat valt je op?
Run het programma een aantal keer. Wat is je langste reeks? ;-)

Uitwerking1: while random

```
public class OefeningLauncher{
    public static void main(String[] args) {
        final int MINIMUM = 2;
        final int MAXIMUM = 10;
        int willekeurigGetal = 0;
        while (willekeurigGetal != MAXIMUM) {
            willekeurigGetal = (int) (Math.random() * (MAXIMUM - MINIMUM + 1)) +
MINIMUM;
                // vanaf 2 t/m 10
            System.out.println("Het getal is: " + willekeurigGetal);
        }
    }
}
```

Uitwerking2: while random

```
public class OefeningLauncher{
    public static void main(String[] args) {
        final int MINIMUM = 2;
        final int MAXIMUM = 10;
        int willekeurigGetal;
        do {
            willekeurigGetal = (int) (Math.random() * (MAXIMUM - MINIMUM + 1)) +
MINIMUM;
            System.out.println("Het getal is: " + willekeurigGetal);
        } while (willekeurigGetal != MAX);
    }
}
```

5.2 Opdrachten

Doe de volgende opdrachten:
WhileLoop.1-Oppervlakteberekening (niveau: basis).
WhileLoop.2-Gemiddelde (niveau: gemengd).

6 FOR LOOP

6.1 For loop

1. Lees Liang 5.7 - 5.9.
2. ***LinkedIn Learning***: use for loops
3. De *for loop* gebruik je als je weet dat je een gegeven aantal keer een stuk code moet uitvoeren. Strafregels: schrijf 300 keer op “Ik mag niet spieken”.

```
public class OefeningLauncher{
    public static void main(String[] args) {
        final int AANTAL = 300;
        for (int getal = 0; getal < AANTAL; getal++) {
            System.out.println("Ik mag niet spieken");
        }
    }
}
```

4. In een *for loop* kun je ook aftellen. Print de getallen 1 tot en met 10 in omgekeerde volgorde.

```
public class OefeningLauncher{
    public static void main(String[] args) {
        for (int getal = 10; getal > 0; getal--) {
            System.out.println(getal);
        }
    }
}
```

5. In een *for loop* kun je de ***loop control variabele*** ook ‘grotere sprongen’ laten maken. Geef alle oneven getallen onder 20.

```
public class OefeningLauncher{
    public static void main(String[] args) {
        for (int getal = 1; getal < 20; getal += 2) {
            System.out.println(getal);
        }
    }
}
```

6.2 Output formatteren

1. Lees Liang 4.6.
2. In print-statements gebruik je soms meerdere variabelen en dan moet je meerdere malen quotes en het plusteken gebruiken. Bekijk de code hieronder.

```
public class OefeningLauncher{
    public static void main(String[] args) {
        String voornaam = “Kees”;
        String achternaam = “Smit”;
        int leeftijd = 51;
        double saldo = 12345.6789;
        System.out.println("Goedemorgen " + voornaam + achternaam + ", je leeftijd
            is " + leeftijd + "en je saldo is " + saldo);
    }
}
```

3. De code van het print-statement wordt eenvoudiger door gebruik te maken van de `printf()` methode met een *format string* in combinatie met *format specifiers*. Voor elke *format specifier* moet je dan ook een invulwaarde meegegeven en dat kan in de vorm van een variabele. Merk op dat je een nieuwe regel (*enter*) moet maken met `\n` in de *format string*.

```
public class OefeningLauncher{
    public static void main(String[] args) {
        String voornaam = "Kees";
        String achternaam = "Smit";
        int leeftijd = 51;
        double saldo = 12345.6789;
        System.out.printf("Goedemorgen %s %s, je leeftijd is %d en je saldo is
             %f\n", voornaam, achternaam, leeftijd, saldo);
    }
}
```

4. Je kunt ook uitvoer in tabelvorm maken door een getal in de *format specifier* te zetten. Dit getal geeft het aantal karakters aan dat ingenomen wordt door de invulwaarde.

5. Probeer de volgende code uit en bekijk goed wat de code doet.

```
public class OefeningLauncher{
    public static void main(String[] args) {
         String voornaam = "Kees";
         String achternaam = "Smit";
         int leeftijd = 51;
         double saldo = 12345.6789;
         System.out.printf("%-9s %-11s %9s %10s\n", "voornaam", "achternaam",
             "leeftijd", "saldo");
        System.out.printf("%-9s %-11s %9d %10.2f\n", voornaam, achternaam,
             leeftijd, saldo);
    }
}
```

6.3 Opdrachten

Doe de volgende opdrachten:
Loops.1-Tafels (niveau: basis).
Loops.2-Loops (niveau: basis).
Loops.3-Dobbelsteen (niveau: gevorderd).

7 ARRAYS

7.1 Arrays

1. Lees Liang 7.1, 7.2.1 - 7.2.6.
2. ***LinkedIn Learning***: arrays en for loops with arrays
3. Je moet een array declareren, instantiëren (creëren) en initialiseren (waardes geven).

```
public class OefeningLauncher{
    public static void main(String[] args) {
        // declaratie
        int[] getallen;
        // instantiatie
        getallen = new int[3];
        // initialisatie (vullen)
        getallen[0] = 3;
        getallen[1] = 5;
        getallen[2] = 8;
    }
}
```

4. Als je de waarden van te voren weet, dan kun je de drie stappen in één keer doen.

```
public class OefeningLauncher{
    public static void main(String[] args) {
        // declaratie, instantiatie en initialisatie
        int[] getallen = {3, 5, 8};
    }
}
```

5. Arrays en for loops gaan goed samen. Je kunt in een for loop alle elementen van de array aflopen, bijvoorbeeld om de waarden te geven, te printen of om te bewerken. Probeer de volgende code uit.

```
public class OefeningLauncher{
    public static void main(String[] args) {
        int AANTAL = 8;
        int[] getallen = new int[AANTAL];
        // een rij van 8 getallen vullen met random getallen tussen 1 en 6
        for (int teller = 0; teller < getallen.length; teller++) {
            getallen[teller] = (int) (Math.random() * 6) + 1;
        }
        // de getallen printen
        System.out.println("Dit zijn de getallen:");
        for (int teller = 0; teller < getallen.length; teller++) {
            System.out.println(getallen[teller]);
        }
        // tellen hoe vaak een 6 voor komt in de rij
        int aantalZessen = 0;
        for (int teller = 0; teller < getallen.length; teller++) {
            if (getallen[teller] == 6) {
                aantalZessen ++;
            }
        }
        // de som van de getallen bepalen
        int som = 0;
        for (int teller = 0; teller < getallen.length; teller++) {
            som += getallen[teller];
        }
        System.out.printf("De 6 komt %d keer voor en de som van de getallen is %d",
            aantalZessen, som);
    }
}
```

7.2 Arrays en methoden

1. Lees Liang 7.6 en 7.7.
2. Je kunt een *array* als parameter meegeven aan een methode. Schrijf een methode die de som van de waarden van een *array* met integers bepaalt en gebruik de methode door er een *array* aan mee te geven.

```
public class OefeningLauncher{
    public static void main(String[] args) {
        int AANTAL = 8;
        int[] getallen = new int[AANTAL];
        // een rij van 8 getallen vullen met random getallen tussen 1 en 6
        for (int teller = 0; teller < getallen.length; teller++) {
            getallen[teller] = (int) (Math.random() * 6) + 1;
        }
        // let op: je geeft de gehele array aan de methode, de naam van de array
        System.out.printf("De som van de getallen is %d",
            bepaalSomWaarden(getallen);
    }

    public static int bepaalSomWaarden(int[] mpRij) {
        int som = 0;
        for (int teller = 0; teller < mpRij.length; teller++) {
            som += mpRij[teller];
        }
        return som;
    }
}
```

In deze module gebruiken we het voorvoegsel “mp” enkel en alleen in de definitie van de methode parameters (zie `int[] mpRij`). Merk op dat de naam mpRij alleen binnen de methode beschikbaar en bekend is en dus alleen daar gebruikt kan worden.

3. Je kunt een methode ook een array laten returnen. Schrijf een methode die een array met willekeurige getallen kan maken. Je moet de grootte van de array en het minimum en maximum van de te genereren getallen meegeven.

```
public static int[] genereerRandomRij(int mpGrootte, int mpMin, int mpMax) {
        // declareer en instantieer de array op de gegeven grootte
        int[] rij = new int[mpGrootte];
        // vul de rij met willekeurige waarden tussen mpMin en mpMax
        for (int teller = 0; teller < mpGrootte; teller++) {
            rij[teller] = (int) (Math.random() * (mpMax - mpMin + 1)) + mpMin;
        }
        return rij;
    }
```

7.3 Opdrachten

Doe de volgende opdrachten:
Arrays.1-Tentamencijfers (niveau: basis).
Arrays.2-Arrays (niveau: gemengd).
Arrays.3-Bsa-monitor (niveau: basis).

8 METHODS EN PARAMETERS

8.1 Methodes en waarden doorgeven aan methodes

1. Lees Liang 6.5 en 6.6.
   Kijk goed naar Listing 6.4 en Listing 6.5.

2. Methodes zijn handig bij het schrijven van overzichtelijk en herbruikbare code. Schrijf een programma dat de namen en scores van een vast aantal spelers vraagt, vervolgens de namen en scores in een tabel afdrukt en de tenslotte de hoogste score bepaalt en de speller(s) met die hoogste score afdrukt. Denk eerst zelf na over stappen in het programma. Per stap schrijven we een methode die de stap uitvoert. In de main methode roepen we de methoden aan om de stappen uit te voeren.

3. Je kunt of moet soms gebruik maken van *globale variabelen*. Dat zijn variabelen die op klasse niveau gedeclareerd worden en vaak ook geinitialiseerd. Die zijn dan bruikbaar in alle methoden van de klasse, in dit geval de ***Launcher***. Bekijk de code hieronder. In de main staan de aanroepen van de methoden die de verschillende stappen uitvoeren. Denk eerst na over de code van de methode.

```
public class ScoreLauncher {
    // Declareer globale variabelen die in meerdere methodes gebruikt worden
    public static final int AANTAL_NAMEN = 4;
    public static String[] namen = new String[AANTAL_NAMEN];
    public static int[] scores = new int[AANTAL_NAMEN];

    public static void main(String[] args) {
        vraagNamenEnScores();
        drukTabelMetNamenEnScoresAf();
        int hoogsteScore = geefHoogsteScore(scores);
        drukHoogsteScoreAf(hoogsteScore);
        String spelersMetHoogsteScore = geefSpelersMetHoogsteScore(hoogsteScore);
        drukSpelersMetHoogsteScoreAf(spelersMetHoogsteScore);
    }

// methodes komen hier
}
```

4. Hier volgen alle afzonderlijk methodes met hun code.

```
public static void vraagNamenEnScores() {
    // Vraag de gebruiker om de namen van spelers
    // Vraag de gebruiker om de scores van de spelers
    Scanner invoerVrager = new Scanner(System.in);
    for (int teller = 0; teller < AANTAL_NAMEN; teller++) {
        System.out.print("Geef de naam van persoon " + (teller + 1) + ": ");
        namen[teller] = invoerVrager.nextLine();
        System.out.print("Geef de score van deze persoon: ");
        scores[teller] = invoerVrager.nextInt();
        invoerVrager.nextLine();
    }
}
```

```
public static void drukTabelMetNamenEnScoresAf() {
    // Druk een tabel af met de namen en de scores
    System.out.printf("%-15s%7s\n", "Naam", "Score");
    for (int teller = 0; teller < AANTAL_NAMEN; teller++) {
        System.out.printf("%-15s%7d\n", namen[teller], scores[teller]);
    }
}
```

```
public static void drukHoogsteScoreAf(int mpHoogsteScore) {
    System.out.println("De hoogste score is: " + mpHoogsteScore);
}
```

```
public static int geefHoogsteScore(int[] mpScores) {
    int hoogsteScore = -1;
    for (int teller = 0; teller < AANTAL_NAMEN; teller++) {
        hoogsteScore = Math.max(hoogsteScore, mpScores[teller]);
    }
    return hoogsteScore;
}
```

```
public static String geefSpelersMetHoogsteScore(int mpHoogsteScore) {
    String spelersMetHoogsteScore = "";
    // loop door de scores om te bepalen wie de hoogste score heeft
    for (int teller = 0; teller < AANTAL_NAMEN; teller++) {
        if (scores[teller] == mpHoogsteScore)  {
            // voeg de naam van speler met hoogste score toe aan de uitvoer
        if (spelersMetHoogsteScore.equals("") {
            spelersMetHoogsteScore = namen[teller];
        } else {
           spelersMetHoogsteScore = spelersMetHoogsteScore + ", " + namen[teller];
           }
        }
    }
    return spelersMetHoogsteScore;
}
```

```
public static void drukSpelersMetHoogsteScoreAf(String mpSpelersMetHoogsteScore) {
    System.out.println("Speler(s) met de hoogste score: " +
        mpSpelersMetHoogsteScore);
    }
```

8.2 Opdracht

Doe de volgende opdracht:
MethodeParameters.1-Getallen-raden (niveau: gevorderd).

9 OEFENOPDRACHTEN

9.1 Oefenopdracht

Doe de volgende opdracht:
Oefenopdrachten.1-Parkeergarage (niveau: basis)