



Java Essentials

Hoofdstuk 1

Object-geörinteerd programmeren in Java

**DE HOGESCHOOL
MET HET NETWERK**

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



Inhoud

1. De programmeertaal Java
 1. Wat is Java?
 2. Java als platform
 3. Soorten Java toepassingen
2. Mijn eerste Java programma
 1. Kennismaking met IntelliJ
 2. Opbouw van het programma
3. Klassen en objecten
4. Opbouw van een klasse
 1. De declaratie van de klasse
 2. De klasse-omschrijving (body)
 1. Eigenschappen
 2. Getters
 3. Declaratie van methoden
 4. Implementatie van methoden
 5. Setters



Inhoud (vervolg)

5. Control flow statements
 1. Het boolean datatype
 2. Het if-then statement
 3. Relationale en logische operatoren
 4. Het if-then-else statement
6. Method overloading
7. Primitieve datatypes
8. Input van de gebruiker
9. Verkorte operatoren
10. De klasse String
11. Iteraties
 1. While-loop
 2. For-loop
12. Arrays
13. Gegevens doorgeven aan een methode

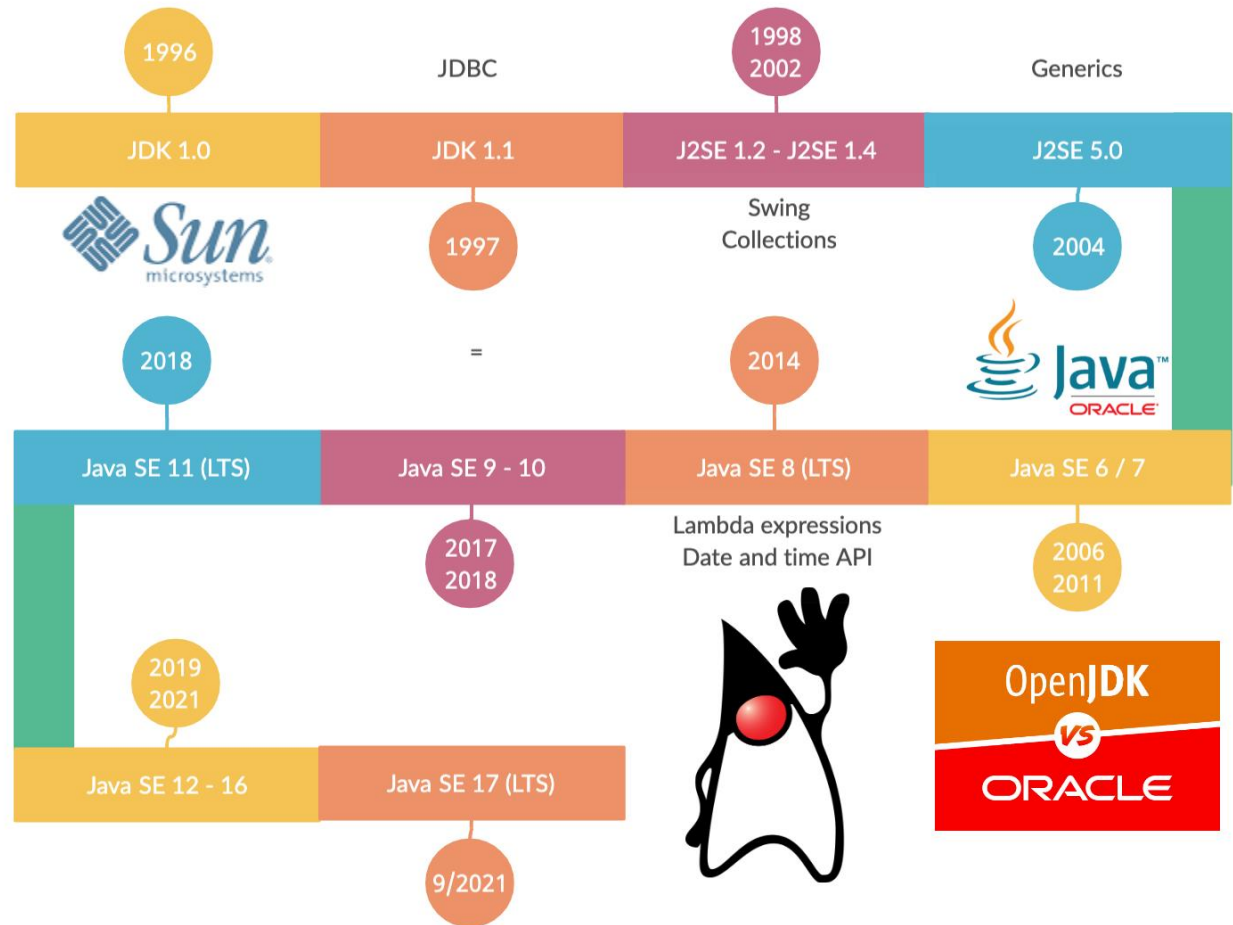




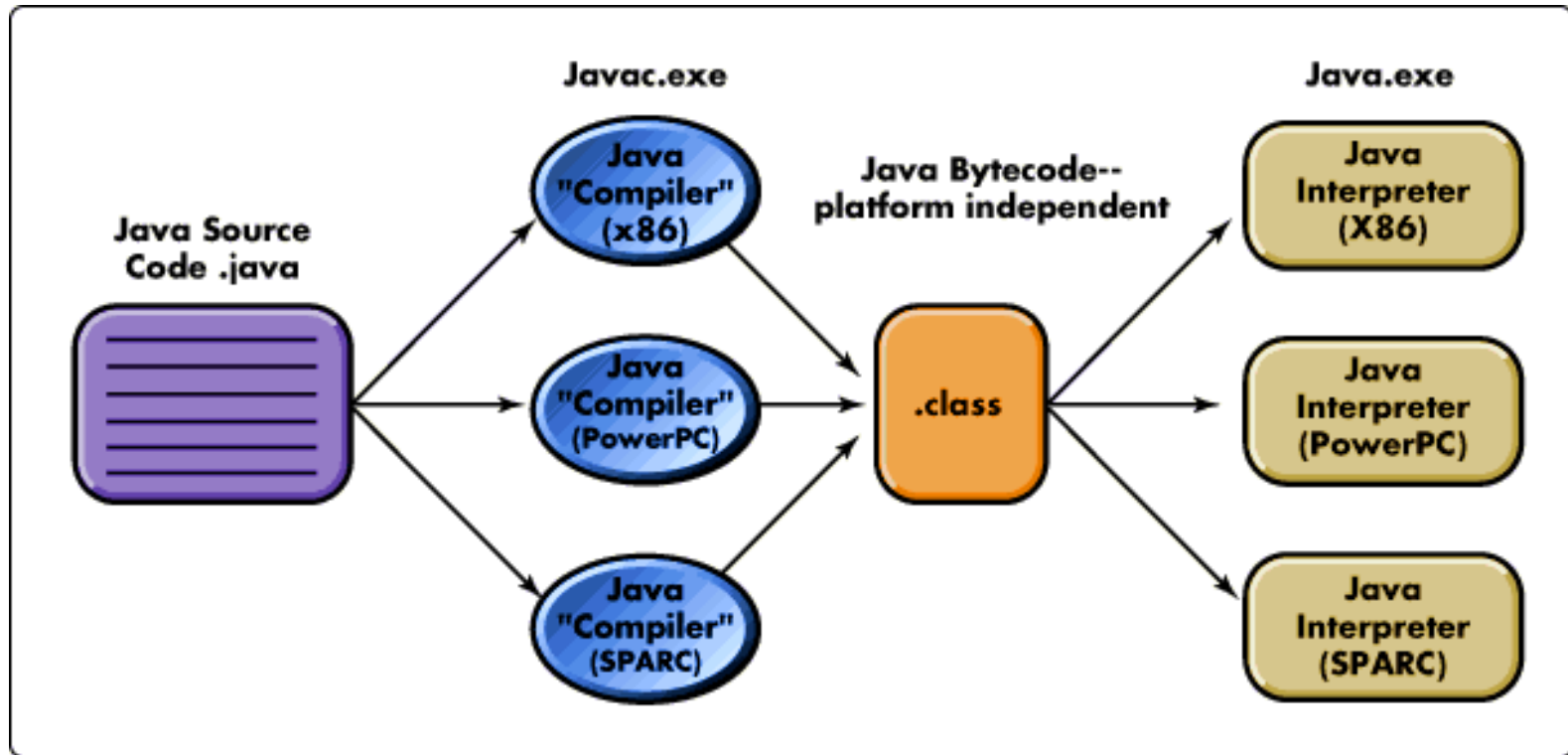
James Gosling

```
public class HelloWorld {

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

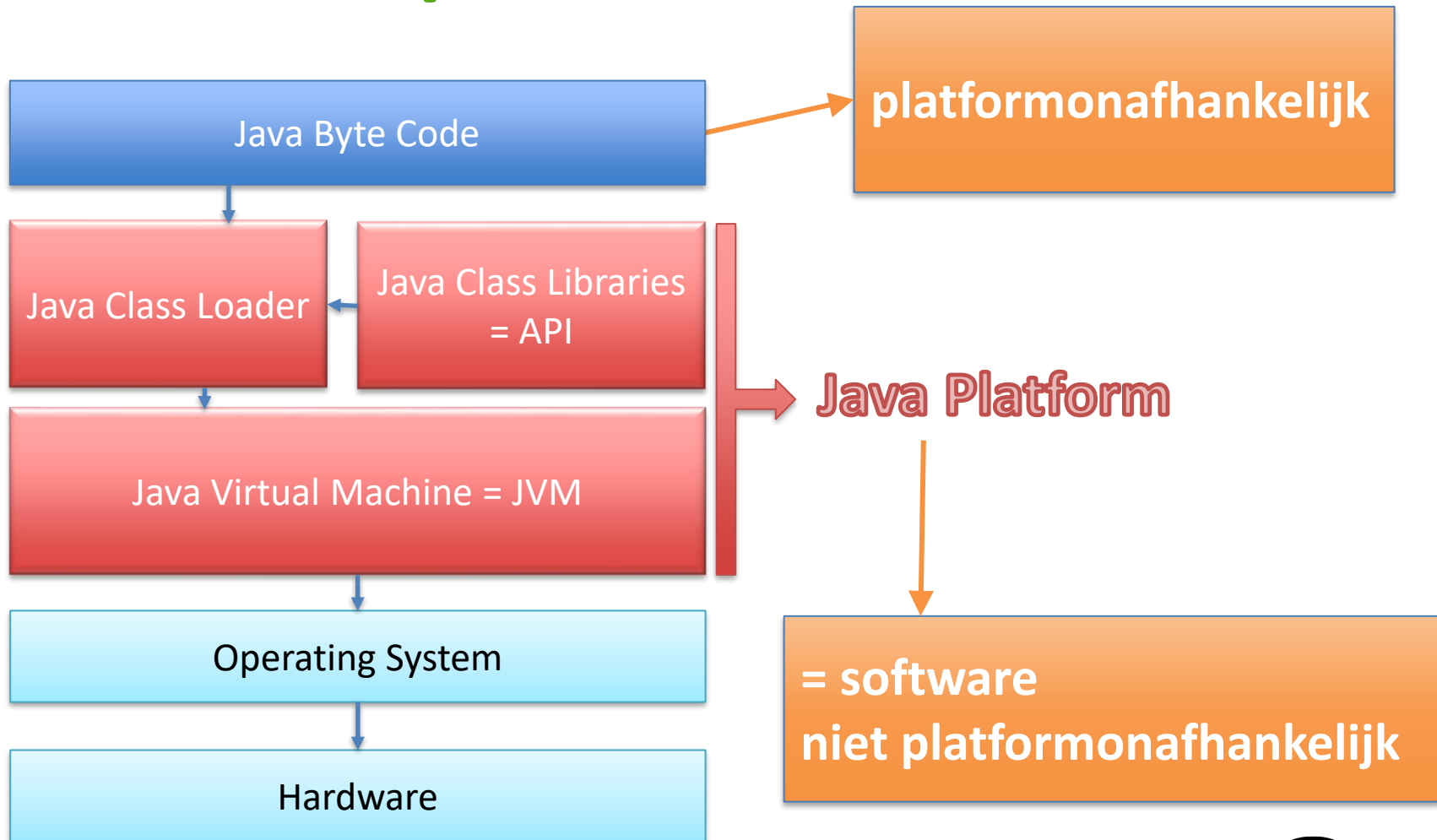


1.1 Wat is java?



Java broncode (.java) wordt eerst gecompileerd naar platform-onafhankelijke bytecode (.class). Deze bytecode wordt dan geïnterpreteerd door de Java Virtual Machine.

1.2 Java als platform



- Java Virtual Machine (**JVM**): interpreteert de bytecode en maakt gebruik van de onderliggende hardware en het onderliggende besturingssysteem om de instructies uit te voeren
- Java Application Programming Interface (**Java API**): verzameling van software-componenten die gebruikt kunnen worden door het Java-programma (gegroepeerd in packages)

1.3 Soorten Java toepassingen



Mobile Phones

If you have an Android phone you use Java every day! Android apps – and indeed the Android operating system! - are written in Java, with Google's API, which is similar to JDK.



Point of Sale Systems

Java is also used in the creation of PoS systems, helping businesses exchange goods or services for money from their customers.



Video Games

One of the most popular games of all time, Minecraft, was written in Java by Mojang. Minecraft is a sandbox construction game, where you can build anything you can imagine.



Trading Applications

Several third-party trading applications use Java. Murex, which is used by many banks for front to back connectivity, is also written in Java.

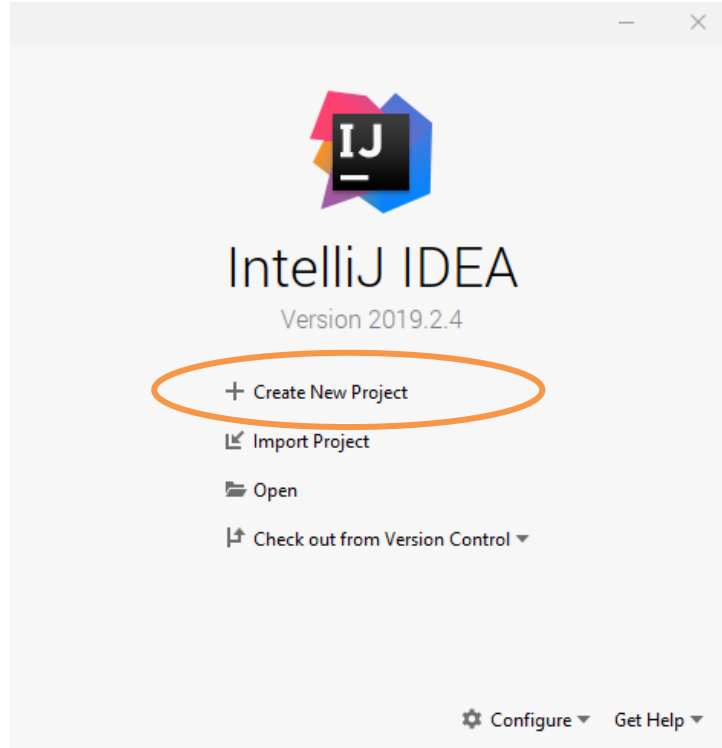


Big Data Technologies

The Java platform is very popular in writing high-performance systems. Hadoop and Elasticsearch are both written in Java and are often used in Big Data projects.

2. Mijn eerste Java programma

Opdracht 1: Kennismaking met IntelliJ.



<https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-application.html>



The screenshot shows the IntelliJ IDEA interface with the Project tool window on the left. The project structure is as follows:

- Project: JavaEssentials (C:\Users\20003575\IdeaProjects\JavaEssentials)
 - .idea
 - H1
 - src
 - be.pxl.h1.hello
 - HelloWorldApp
 - H1.iml
 - src
 - JavaEssentials.iml
- External Libraries
- Scratches and Consoles

Annotations on the right side of the image explain the components:

- = Projectnaam**: Points to the "JavaEssentials" project name in the breadcrumb and the "Project" dropdown.
- = module**: Points to the "H1" module in the project tree.
- src = source = map met broncode**: Points to the "src" directory.
- package (pakket) = submap**: Points to the "be.pxl.h1.hello" package.
- klassenaam**: Points to the "HelloWorldApp" class.

➔ C:\projectfolder\JavaEssentials\H1\src\be\pxl\h1\hello\HelloWorldApp.java



```
HelloWorldApp.java x
1  package be.pxl.h1.hello;
2
3  public class HelloWorldApp {
4      public static void main(String[] args) {
5          System.out.println("Hello World!");
6      }
7  }
8
```

Punkomma



2.2 Opbouw van het programma

1. Het pakket definiëren

```
package be.px1.h1.hello;
```

Pakketten gebruiken we om orde / structuur te brengen in de code (cfr. mappenstructuur Verkenner)

2. De klasse definiëren

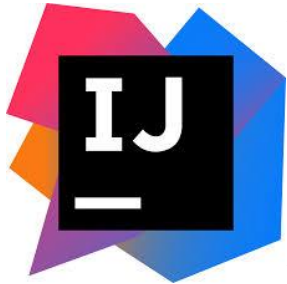
Hierin schrijf je de eigenlijke code.



3. De methode main()

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

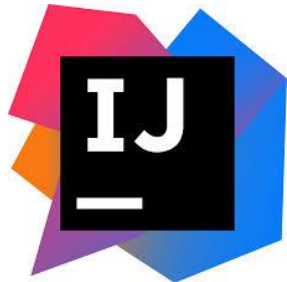
Klasse met methode main() = het hoofdprogramma.



Je kan deze main()-methode genereren door **main** of **psvm** te typen en daarna op <Enter> te drukken!

4. Het eigenlijke werk

```
System.out.println("Hello World!");
```



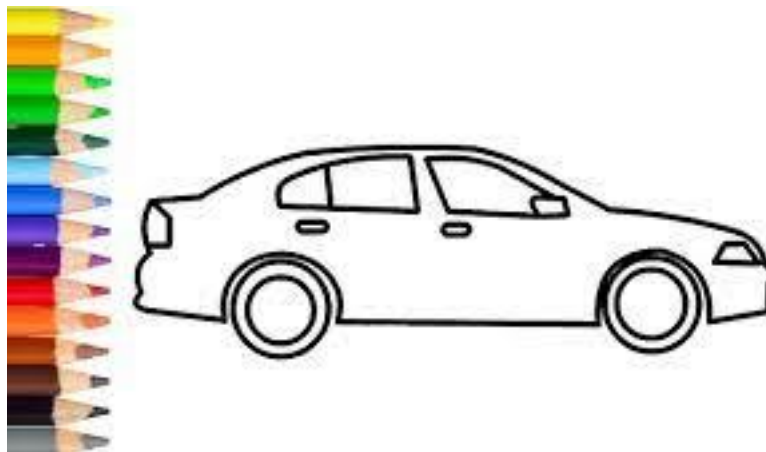
Je kan de bovenstaande methode genereren als je **sout** typt en daarna op <Enter> drukt!



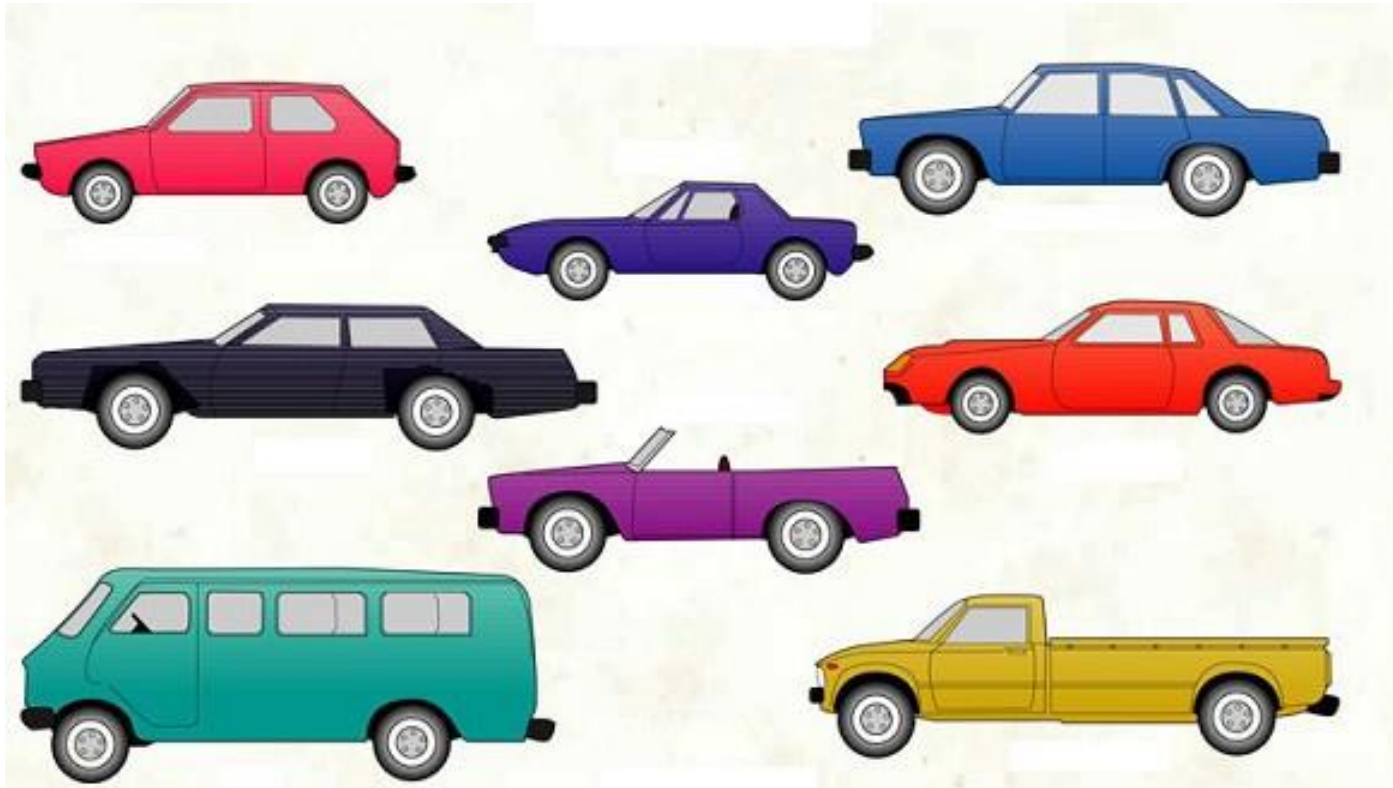
3. Klassen en objecten

Opdracht:

- Teken een auto in paint
 - Welk merk heeft je auto?
 - Welke kleur heeft je auto?
 - Hoeveel deuren heeft je auto?
 - ...



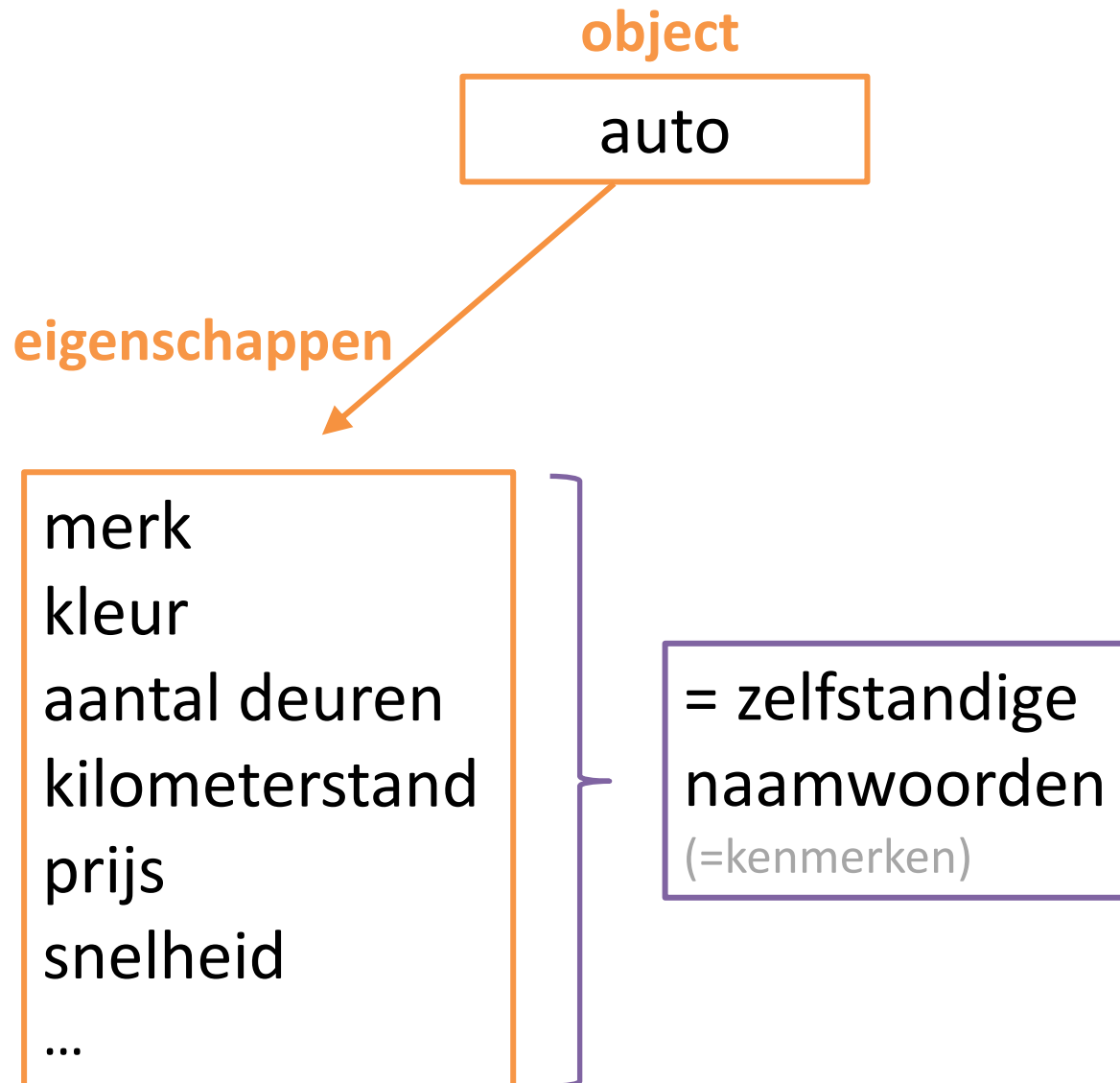
Voorbeeld:



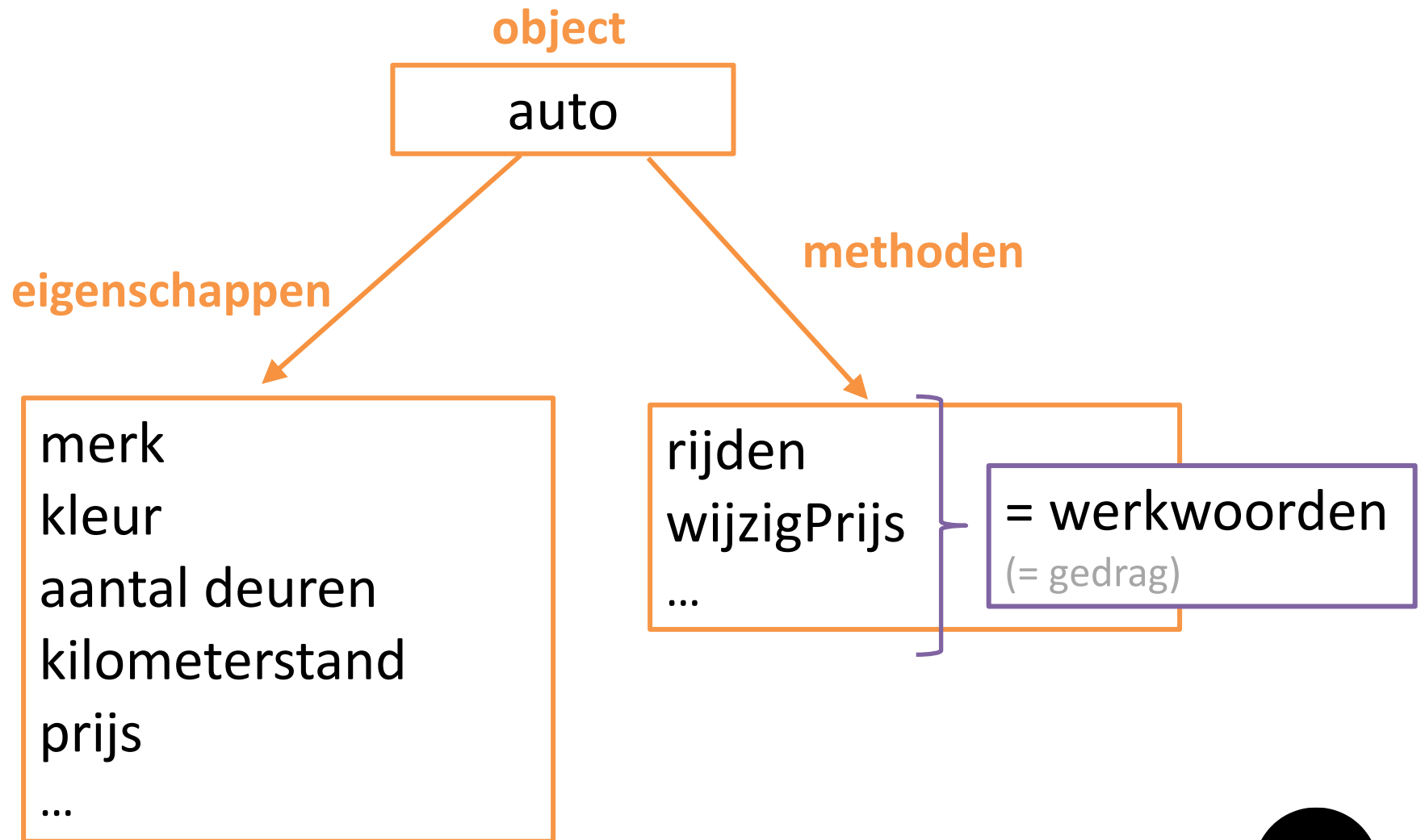
Elke auto is een **object**.

Deze auto's hebben een aantal gemeenschappelijke kenmerken.

3. Klassen en objecten

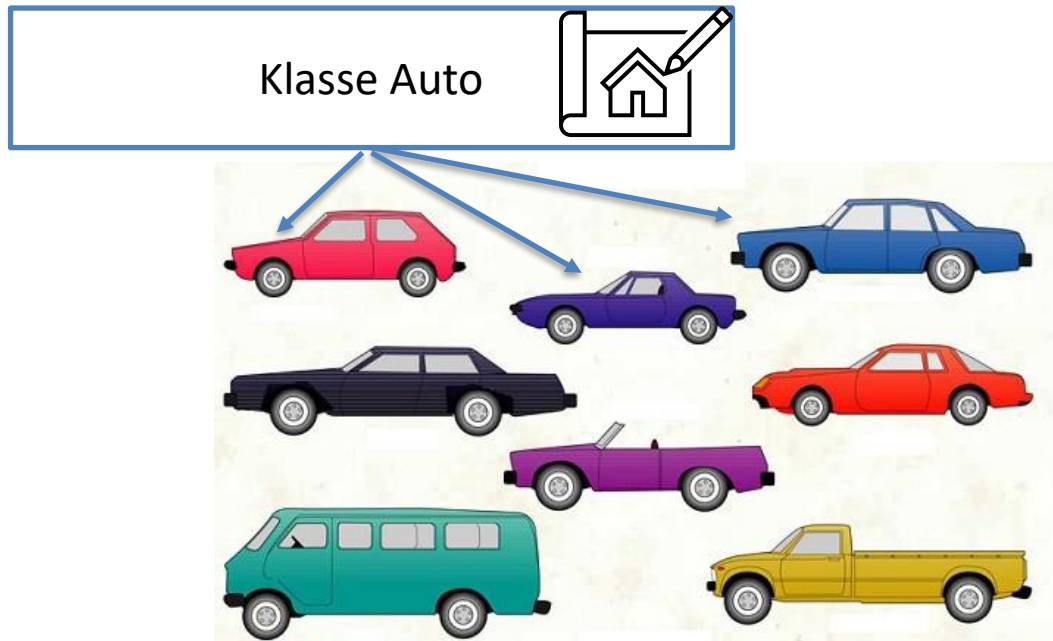


3. Klassen en objecten



3. Klassen en objecten

Al de auto's zijn gemaakt op basis van een blauwdruk (= klasse).



➔ Een **klasse** is een blauwdruk of plan voor objecten die de eigenschappen en methoden van objecten van dezelfde soort bepaalt.

3. Klassen en objecten



Je kan de klasse ook zien als een sjabloon, of koekjes uitsteker (cookie cutter), voor objecten.

4.1 De declaratie van de klasse

```
package be.px1.h1.opgave2;  
  
public class Auto {  
}
```

Dit is de 'meest eenvoudige' klasse-declaratie.

Geen `main()`-methode. Een programma bestaat meestal uit meerdere klassen waarvan er maar 1 de methode `main()` heeft.



Opdracht 2: *Een klasse maken*

Maak een klasse met de naam Auto in de package be.pxl.h1.opdracht2

```
package be.pxl.h1.opdracht2;  
  
public class Auto {  
}
```



Opdracht 3: *Een hoofdprogramma maken*

- Maak een hoofdprogramma in de klasse met de naam **AutoApp**
- Maak een object van de klasse *Auto*.

```
public class AutoApp {  
    public static void main(String[] args) {  
        System.out.println("Dit programma maakt een auto.");  
        Auto auto = new Auto();  
    }  
}
```



Drie soorten commentaar

`/* indien er meerdere regels commentaar
in een programma geschreven worden */`

`/** documentation JAVADOC */`

`// een regel commentaar`



4.2 De klasse-omschrijving (body)

- Body: tussen accolades
- Bevat:
 - Eigenschappen (instantievariabelen, velden, ...)
 - Methoden
 - Constructoren



4.2 De klasse-omschrijving (body)

Klasse Auto

Eigenschappen:

```
merk  
kleur  
kilometerstand
```

Methoden:



```
setMerk()  
getMerk()  
setKleur()  
getKleur()
```

4.2.1 Eigenschappen

= datatype

```
public class Auto {  
    private String merk;  
    private String kleur;  
    private int kilometerstand;  
    private int aantalDeuren;  
}
```

= access modifier

Instantievariabelen

Auto		
f	merk	String
f	kleur	String
f	kilometerstand	int
f	aantalDeuren	int

= UML-schema van de klasse Auto

5.2 Getters

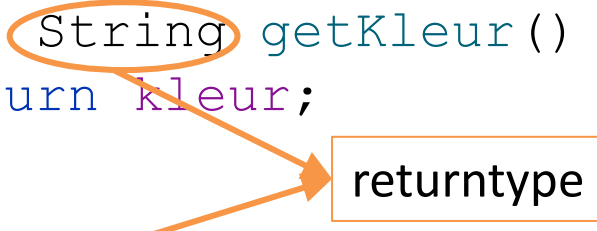
- Manier om de waarde van een eigenschap van een object op te vragen.
- Methode **bvb. `getKilometerstand()`, `getKleur()`.**
- `lowerCamelCasing`

```
objectNaam.getEigenschapNaam()
```



5.2 Getters

```
public class Auto {  
    private String merk;  
    private String kleur;  
    private int kilometerstand;  
    private int aantalDeuren;  
  
    public String getKleur() {  
        return kleur;  
    }  
  
    public int getKilometerstand() {  
        return kilometerstand;  
    }  
  
    /* overige getters worden niet getoond */  
}
```



A diagram with two orange ovals. The first oval is around the word 'String' in the 'getKleur()' method signature. The second oval is around the word 'int' in the 'getKilometerstand()' method signature. Two orange arrows originate from these ovals and point to a rectangular box labeled 'returntype'.

5.2 Getters

```
public class AutoApp {  
  
    public static void main(String[] args) {  
        Auto auto = new Auto();  
  
        System.out.println(auto.getKilometerstand());  
        System.out.println(auto.getKleur());  
    }  
}
```

0
null

objectNaam.getEigenschapNaam()





4.2.3 Declaratie van methoden

```
[public | private] returntype methodeNaam (parameters)
{
    ...
}
```

- Vet gedrukt = verplicht
- Een methode kan geen waarde of 1 waarde teruggeven
 - geen waarde => terugkeertype = void
 - 1 waarde => terugkeertype is het datatype van hetgeen teruggeven wordt
- Voor elke parameter moet worden opgegeven van welk datatype de parameter is.



Voorbeeld: methode zonder terugkeerwaarde zonder parameters

```
public void vbMethode () {  
...  
}
```

Er wordt geen waarde teruggegeven

De methode is toegankelijk vanuit andere klassen
↔ *private*: enkel toegankelijk vanuit de klasse zelf



Voorbeeld: methode met terugkeerwaarde met parameters

```
public int vbMethode(int par1, double par2) {  
    ...  
}
```

Er wordt een int teruggegeven

Bij het oproepen moeten er 2 argumenten meegegeven worden van het juiste type.



4.2.4 Implementatie van methoden

```
public void vbMethode () {  
    int nummer = 6;  
    System.out.println(nummer);  
}
```

Binnen codeblok
kunnen variabelen
gedeclareerd worden



lokale variabelen

moeten geïnitieerd worden
enkel gekend binnen de methode vbMethode



Instantievariabelen (moeten niet geïnitieerd worden)
Instantievariabelen zijn in elke methode van de klasse gekend



Een lokale variabele mag dezelfde naam hebben als een instantievariabele → instantievariabele wordt als het ware verborgen door de lokale variabele

= shadowing

```
public class MijnKlasse {  
    private int nummer = 5;  
  
    public void vbMethode() {  
        int nummer = 6;  
        System.out.println(nummer);    → 6  
        System.out.println(this.nummer); → 5  
    }  
}
```

Om de instantievariabele toch te benaderen
→ gebruik maken van het *this*-object
(= referentie naar het huidig object)

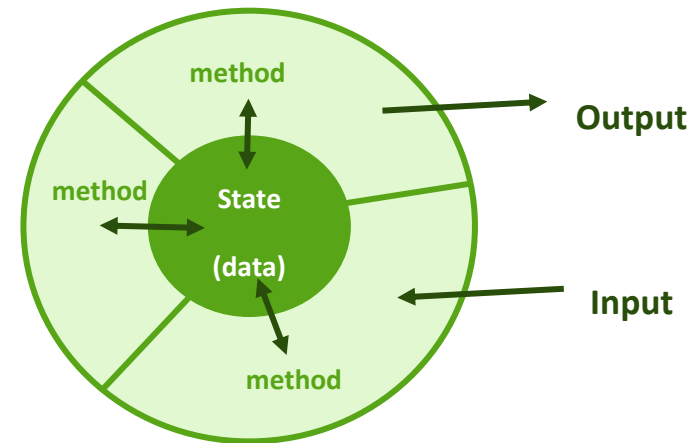


Een parameter mag ook dezelfde naam hebben als een instantievariabele → de instantievariabele wordt als het ware verborgen door de parameter
this gebruiken als je de instantie variabele wil benaderen

```
public class MijnKlasse {  
    private int nummer;  
  
    public void setNummer(int nummer) {  
        this.nummer = nummer;  
    }  
}
```



- Instantievariabelen zijn **private** (= afschermen van de buitenwereld)



= encapsulation
(inkapseling;
data hiding)

- Je maakt deze variabelen enkel toegankelijk via **public** methoden in de klasse.

4.2.5 Setters

- Manier om de waarde van een eigenschap van een object aan te passen.
- Methode `bvb. setKilometerstand(int), setKleur(String)`.



4.2.5 Setters

```
public class Auto {  
    private String merk;  
    private String kleur;  
    private int kilometerstand;  
    private int aantalDeuren;  
  
    public void setKleur(String kleur) {  
        this.kleur = kleur;  
    }  
  
    public void setAantalDeuren(int aantalDeuren) {  
        this.aantalDeuren = aantalDeuren;  
    }  
    /* overige setters en getters worden niet getoond */  
}
```



4.2.5 Setters

```
public class AutoApp {  
  
    public static void main(String[] args) {  
        Auto auto = new Auto();  
        auto.setAantalDeuren(5);  
        auto.setKleur("Rood");  
        System.out.println(auto.getAantalDeuren());  
        System.out.println(auto.getKleur());  
    }  
}
```

```
objectNaam.setEigenschapNaam(arg)
```



Opdracht 4: *De klasse Auto*

- Vervolledig de klasse Auto met de eigenschappen, getters en setters zoals beschreven in de voorgaande slides.
- Maak in het hoofdprogramma een object favorieteAuto aan. Geef alle eigenschappen een waarde zodat het object jouw favoriete auto voorstelt.
- Toon alle eigenschappen van jouw favoriete auto op het scherm.



Opdracht 5: *De klasse Stripboek*

- Maak de klasse Stripboek. Een stripboek heeft volgende eigenschappen:
 - reeks (bijv. “Suske en Wiske”, “Kuifje”, ...)
 - titel
 - album (het nummer van het album in de reeks)
 - auteur (bijv. “Willy Vandersteen”)
- Maak een hoofdprogramma in de klasse StripboekApp.
 - Maak een object van de klasse Stripboek.
 - Geef alle eigenschappen een waarde.
 - Toon alle eigenschappen in de console.



5. Control flow statements

Auto

- *auto* (object) is een *Auto* (class)
- *Kleur*, *aantalDeuren*, *kilometerstand* en *merk* zijn eigenschappen van *auto*
- *getPrijs()* is een methode (taak) om de prijs van *auto* te berekenen. De prijs van een auto wordt als volgt berekend:
Voor iedere auto is de basisprijs €20000.
Voor 3-deurs is er een korting van €1000.



5.1 Het boolean datatype

Het boolean datatype heeft 2 mogelijke waarden: `true` en `false`.

```
public class BooleanApp {  
  
    public static void main(String[] args) {  
        boolean isJavaFun = true;  
        boolean isFishTasty = false;  
        System.out.println(isJavaFun);  
        System.out.println(isFishTasty);  
    }  
}
```

true
false



5.2 Het if-then statement

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```





5.3 Relationele en logische operatoren

operator	Betekenis
>	groter dan
>=	groter of gelijk aan
<	kleiner dan
<=	kleiner of gelijk aan
==	gelijk aan
!=	verschillend van

Twee of meer relatiecondities kunnen gecombineerd worden m.b.v. logische operatoren

Logische operator	Betekenis	Resultaat
&&	and	true als beide condities true zijn
	or	true als 1 van beide condities true is
!	not	is het tegenovergestelde



```
public class Auto {  
    private String merk;  
    private String kleur;  
    private int kilometerstand;  
    private int aantalDeuren;  
    /* overige setters en getters worden niet getoond */
```

```
    public int getPrijs() {  
        int prijs = 20000;  
        if (aantalDeuren == 3) {  
            prijs = prijs - 1000;  
        }  
        return prijs;  
    }  
}
```

() zijn verplicht



46

```
public class AutoApp {
```

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

```
        Auto auto = new Auto();
```

```
        auto.setAantalDeuren(3);
```

```
        System.out.println(auto.getPrijs());
```

19000

```
    }
```

```
}
```



5.4 Het if-then-else statement

```
if (condition) {  
    // block of code to be executed if the condition is true  
} else {  
    // block of code to be executed if the condition is false  
}
```



Opdracht 6: *If-then-else statement*

Voeg in de klasse Auto een boolean eigenschap automaat toe.

Voorzie een getter en setter voor deze eigenschap.

Pas de methode getPrijs() aan.

- Voor iedere auto is de basisprijs €20000.
- Voor 5-deurs is er een toeslag van €2000, voor 3-deurs is er een korting van €500.
- Voor auto's met automaat is er een toeslag van €2000, indien er geen automaat is gaat er 2% van de prijs af.

6. Method overloading

= methoden met dezelfde naam maar met verschillende (types/aantal) parameters

```
public class Auto {  
    private String merk;  
    private String kleur;  
    private int kilometerstand;  
    private int aantalDeuren;  
  
    /* overige getters en setters worden niet getoond */  
  
    public void rijden(int afstand) {  
        kilometerstand += afstand;  
    }  
  
    public void rijden(int uren, int snelheid) {  
        kilometerstand += uren * snelheid;  
    }  
}
```

De compiler kiest de juiste methode op basis van de argumenten die worden doorgegeven.

= Method-overloading



```
public class AutoApp {  
  
    public static void main(String[] args) {  
        Auto auto = new Auto();  
        auto.rijden(320);  
        auto.rijden(2, 60);  
        System.out.println(auto.getKilometerstand());  
    }  
}
```

```
public void rijden(int afstand) {...}
```

```
public void rijden(int uren, int snelheid) {...}
```

= Method-
overloading



7. Primitieve datatypes

Type	Formaat	Bereik
Gehele getallen		
int	32 bit	-2147483648 ... 2147483647
long	64 bit	-9223372036854775808 ... 9223372036854775807
Reële getallen		
double	64 bit	
Andere types		
boolean	1 bit	
char	16 bit	



De declaratie van variabelen

```
public class Demo1 {  
    public static void main(String[] args) {  
        int number;  
        number = 5;  
        int sum = number + 2;  
        double salary = 1234.5;  
        long bigNumber = 123456789123456789L;  
        number = (int) salary;  
        System.out.println("Salary: " + number);  
        number = (int) bigNumber;  
        System.out.println("Big number: " + number);  
        char letter = 'a';  
        boolean geslaagd = true;  
        String text = "Goedenacht!";  
    }  
}
```

L of l

Casting

Concatenatie

Opdracht 7: *Primitieve datatypes*

- Maak een klasse Planet.
- Voorzie de eigenschappen name, diameter en distanceFromSun.
- Maak een hoofdprogramma waar je 2 Planet-objecten aanmaakt (zie tabel op volgende slide).
- Welke planeet staat het dichtst bij de zon? Druk de naam af.
- De Aarde staat gemiddeld op een afstand van 1 AE (Astronomische eenheid) van de Zon. Op welke afstand staat Mars (in AE)? Op welke afstand staat Neptunus?



Planet	Diameter (km)	Distance from Sun (km)
Sun	1,391,400	-
Mercury	4,879	57,900,000
Venus	12,104	108,200,000
Earth	12,756	149,600,000
Mars	6,792	227,900,000
Jupiter	142,984	778,600,000
Saturn	120,536	1,433,500,000
Uranus	51,118	2,872,500,000
Neptune	49,528	4,495,100,000





Typeconversie

= gegevens van een bepaald datatype omzetten naar een ander datatype

- **Impliciete conversie** (automatisch door de compiler)

```
int anInteger = 5;  
long aLong;  
aLong = anInteger;
```

int
(32 bit)

past in!!!

long
(64 bit)





- **Expliciete conversie (= casten)**

```
long aLong = 115L;  
int anInteger;  
anInteger = aLong;
```

long
(64 bit)

Past NIET
in!!!

int
(32 bit)

casten

= verantwoordelijkheid van
de programmeur!
Wanneer loopt het mis?



```
anInteger = (int) aLong;
```



Rangorde van de datatypes

Rangorde	Type
1	double
2	long
3	int

int + long = long

int + double = double

*int * long = long*

*double * double = double*

int / int = int

int / double = double

!!! Hier maak je snel
een fout tegen.

Vb. $10 / 4 = 2$ i.p.v. 2.5



Bij wiskundige bewerkingen is het datatype van het resultaat het type van de operand met de hoogste rang.

Opdracht 8:

Geef na iedere bewerking de waarde van de variabele

```
int i;  
int n = 3;  
double d;  
i = n + 4;  
i = n * 4;  
i = 7 + 2 * 3;  
n = n * (n + 2) * 4;  
d = 3.5 / 2;  
n = 7 / 4;
```

Datatype char

```
public class CharVoorbeeld {  
    public static void main(String[] args) {  
        char karakter = 'a';  
        System.out.println(karakter);  
        System.out.println(karakter + 1);  
        karakter = (char) (karakter + 1);  
        System.out.println(karakter);  
    }  
}
```

a
98
b

8. Input van de gebruiker

```
import java.util.Scanner;

public class Demo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Geef je naam:");
        String naam = input.next();
        System.out.println("Welkom, " + naam);
        input.close();
    }
}
```

Object van de **Scanner** class aanmaken.

De Scanner class kan je vinden in de bibliotheek (library) **java.util**. Om de Scanner class te gebruiken, moet je deze toevoegen in je code door gebruik te maken van het keyword **import**.

Scanner object moet je steeds sluiten (close()) aan het einde van je programma.



```

import java.util.Scanner;

public class Demo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Geef nummer: ");
        int number = scanner.nextInt();
        System.out.println("Geef je favoriete film:");
        String movie = scanner.nextLine();
        System.out.println("Geef je favoriete acteur:");
        String actor = scanner.next();
        System.out.println("Geef je favoriete band:");
        String band = scanner.nextLine();
        System.out.println(number + ", " + movie + ", "
            + actor + ", " + band);
    }
}

```

Geef nummer:

15

Geef je favoriete film:

Geef je favoriete acteur/actrice:

Brad Pitt

Geef je favoriete band:

15, , Brad, Pitt

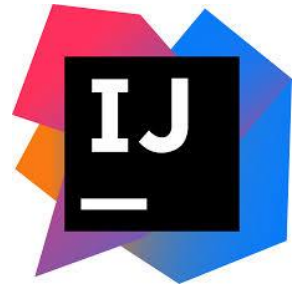


```
import java.util.Scanner;

public class Demo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Geef nummer: ");
        int number = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Geef je favoriete film:");
        String movie = scanner.nextLine();
        System.out.println("Geef je favoriete acteur:");
        String actor = scanner.nextLine();
        System.out.println("Geef je favoriete band:");
        String band = scanner.nextLine();
        System.out.println(number + ", " + movie + ", "
                           + actor + ", " + band);
    }
}
```



De plaats van de accolades is belangrijk! Zorg verder voor een nette inspruing van de accolades (= een code convention). Dit verhoogt de leesbaarheid van het programma.



Code > Reformat code
CTRL + ALT + L



Opdracht 9: Omrekening Celsius → Fahrenheit

Vraag de temperatuur in Celsius en reken deze temperatuur om naar Fahrenheit. Maak hiervoor gebruik van een object van de klasse Thermometer.



C Thermometer		
f	temperature	double
m	setTemperature(double)	void
m	getTemperature()	double
m	getFahrenheit()	double

$$F = \frac{9}{5} C + 32$$

Celsius to Fahrenheit Formula



```
public class Thermometer {  
    private double temperature;  
  
    public void setTemperature(double temperature) {  
        this.temperature = temperature;  
    }  
  
    public double getTemperature() {  
        return temperature;  
    }  
  
    public double getFahrenheit() {  
        return 9 / 5 * temperature + 32;  
    }  
}
```

Geef temperatuur: 26
Fahrenheit: 58.0

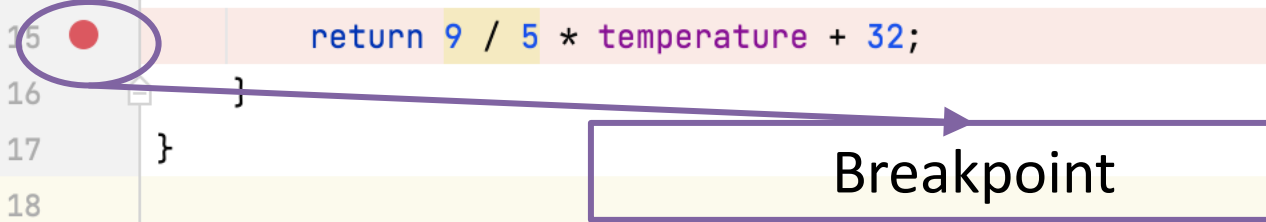
1 (omdat: $\text{int} / \text{int} \rightarrow \text{int}$)



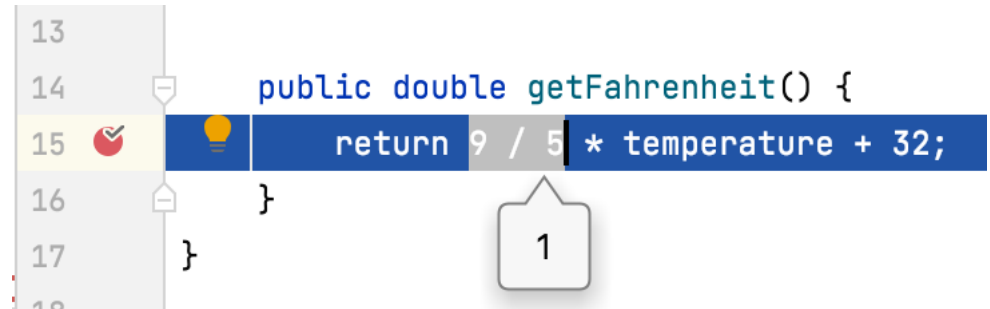
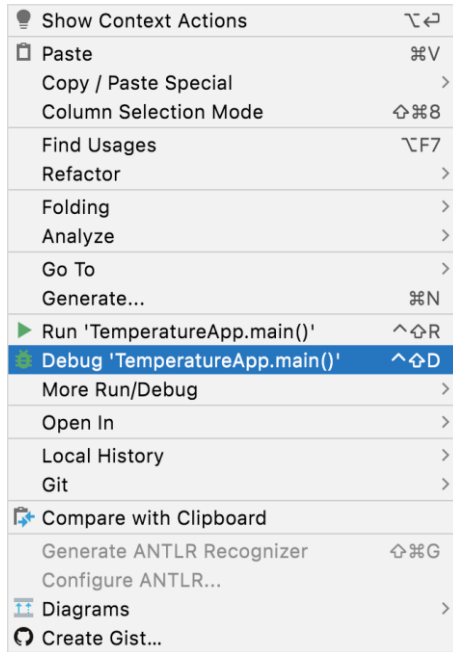
Debuggen

– Stap 1: breakpoint plaatsen

```
1  package be.px1.h1.opdracht9;
2
3  public class Thermometer {
4      private double temperature;
5
6      public void setTemperature(double temperature) {
7          this.temperature = temperature;
8      }
9
10     public double getTemperature() {
11         return temperature;
12     }
13
14     public double getFahrenheit() {
15         return 9 / 5 * temperature + 32;
16     }
17 }
18
```

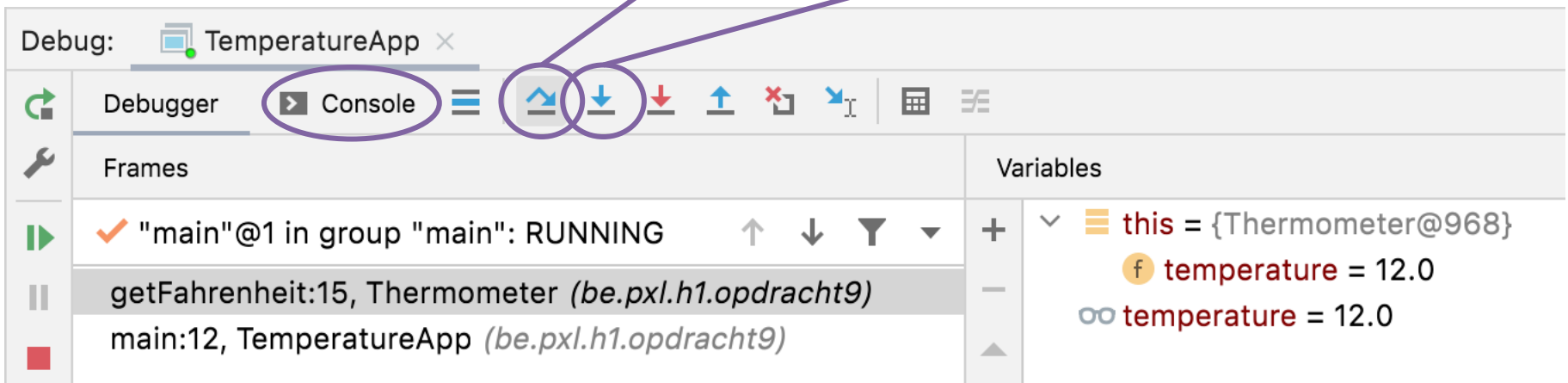


– Step 2: start programma in debug mode



Step over

Step into




```
public class Thermometer {  
    private double temperature;  
  
    public void setTemperature(double temperature) {  
        this.temperature = temperature;  
    }  
  
    public double getTemperature() {  
        return temperature;  
    }  
  
    public double getFahrenheit() {  
        return 9.0 / 5 * temperature + 32;  
    }  
}
```

Geef temperatuur: 26
Fahrenheit: 78.800000000000001

9. Verkorte operatoren

operator	Betekenis
++	Verhoog var met 1
--	Verlaag met 1
+=	Verhoog met gegeven getal
-=	Verlaag met gegeven getal
/=	Deel door het gegeven getal
*=	Vermenigvuldig variabele met het gegeven getal
%=	Rest bij deling door het gegeven getal

`i++;` // is een kortere notatie voor `i = i + 1;`

`i--;` // is een kortere notatie voor `i = i - 1;`

`i += 3;` // is een kortere notatie voor `i = i + 3;`

`i *= 2;` // is een kortere notatie voor `i = i * 2;`

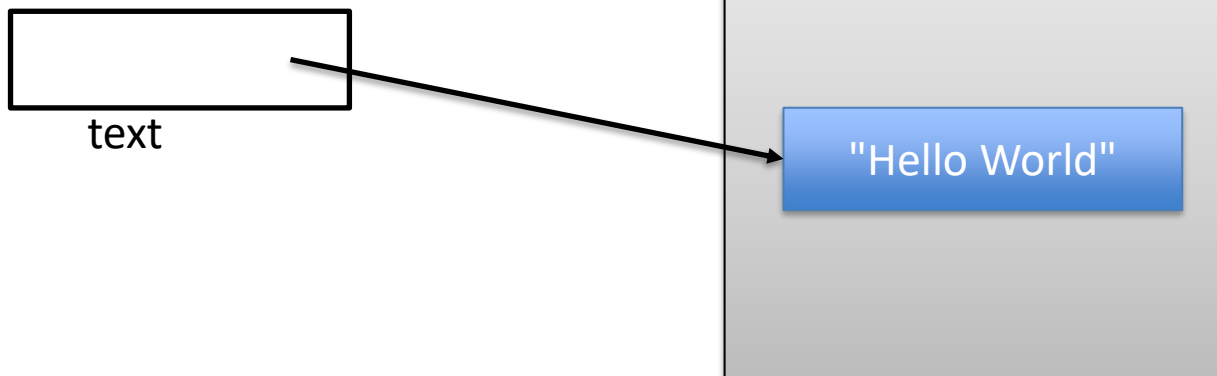


10. De klasse String

- Behoort tot het pakket *java.lang* (wordt standaard geïmporteerd)

```
String text = "Hello World";
```

= objectreferentie



```

public class Demo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Geef je naam:");
        String naam = input.next();
        System.out.println("Geef je naam nog een keer:");
        String naamNogEens = input.next();
        System.out.println("==" + (naam == naamNogEens));
        System.out.println("equals:" + naam.equals(naamNogEens));
        input.close();
    }
}

```

Geef je naam:

Bert

Geef je naam nog een keer:

Bert

==:false

equals:true

Als je de inhoud van 2 strings wil vergelijken, moet je de methode ***equals()*** gebruiken.



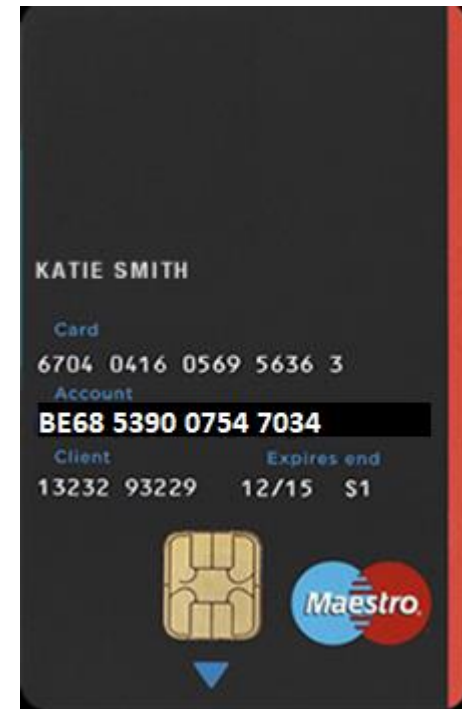
Opdracht 10: Controlegetal BBAN bepalen

Maak een klasse BankAccount.

Deze klasse heeft de volgende eigenschappen:

- name (bijv. Katie Smith)
- code (bv. "BE")
- controlDigits (de 2 cijfers na BE)
- number (de overige 12 cijfers)

Voorzie de methode *String getAccount()* die het volledige bankrekeningnummer teruggeeft (code + controlDigits + number)



Voorzie nu de methode *boolean isValid()* in de klasse BankAccount. Deze methode geeft true indien het bankrekeningnummer geldig is, en false indien de bankrekeningnummer niet geldig is.

Hoe controleer je of een bankrekeningnummer geldig is?

Vermenigvuldig de 12 laatste cijfers van een bankrekeningnummer (= eigenschap number) met 1000000.

Tel hier 111400 bij op. Vervolgens bereken je de rest van het getal bij deling door 97. Het bekomen getal trek je af van 98.

Het getal dat je nu uitkomt is het 2-cijferig controlegetal (= eigenschap controlDigits).



```
public class BankApp {  
  
    public static void main(String[] args) {  
        BankAccountNumber bankAccountNumber = new BankAccountNumber();  
        bankAccountNumber.setCode("BE");  
        bankAccountNumber.setControlDigits(68);  
        bankAccountNumber.setNumber(539007547034L);  
        System.out.println(bankAccountNumber.getAccount());  
        if (bankAccountNumber.isValid()) {  
            System.out.println("This bank account number is valid.");  
        } else {  
            System.out.println("This bank account number is not valid.");  
        }  
    }  
}
```



10.2 String formatering

```
public class Demo {  
    public static void main(String[] args) {  
        System.out.printf("My name is: %s%n", "Joe");  
        System.out.printf("My age is: %d%n", 28);  
    }  
}
```

SPECIFIER	APPLIES TO	OUTPUT
%c	character	Unicode character
%d	integer (int, long)	integer
%f	floating point	decimal number
%n	none	Platform-specific line separator
%s	any type	String value
%S	any type	Upper-case variant van %s




```

public class Demo {
    public static void main(String[] args) {
        System.out.printf("|%20d|\n", 93);
        System.out.printf("|%-20d|\n", 93);
        System.out.printf("|%020d|\n", 93);
        System.out.printf("%.2f", 2.3568888);
    }
}

```

```

|                                     93|
|93                                     |
|00000000000000000000000093|
2,36

```



String.format()

```
public class Demo {  
    public static void main(String[] args) {  
        String output = String.format("%s = %d", "Joe", 35);  
        System.out.println(output);  
    }  
}
```



11. Iteraties

11.1 Iteraties: while-loop

Lees een rij getallen in. Zolang het ingelezen getal kleiner is dan 10 druk je het getal samen met het dubbel van dit getal af.

```
Scanner keyboard = new Scanner(System.in);  
int getal = keyboard.nextInt();  
while (getal < 10) {  
    int dubbel = getal * 2;  
    System.out.println("Het dubbel van " + getal + " is " + dubbel);  
    getal = keyboard.nextInt();  
}  
keyboard.close();
```



Opdracht 11: While-loop

Voeg in de klasse BankAccount een extra eigenschap amount toe. Voorzie een setter.

Voorzie een methode withdraw(double amount) waarmee geld van de rekening opgenomen kan worden. Wanneer het gevraagde bedrag niet op de rekening staat, print je een duidelijke boodschap en wordt het geld niet van de rekening opgenomen.



In het hoofdprogramma vraag je eerst het startbedrag van de rekening.

Vervolgens stel je de vraag of de gebruiker geld wil opnemen. Indien hij “J” antwoordt, vraag je het bedrag en neem je dit op van de rekening (indien mogelijk). De gebruiker kan blijven geld opnemen van zijn rekening zolang hij “J” blijft antwoorden en de rekening niet leeg is (€0).



Voorbeeld verloop van het programma

Geef het startbedrag:

3500

Wil je geld afhalen (J/N)?

J

Hoeveel geld wil je afhalen (max. 3500.0)?

250,85

Wil je geld afhalen (J/N)?

J

Hoeveel geld wil je afhalen (max. 3249.15)?

345,67

Wil je geld afhalen (J/N)?

N

Er staat nog 2903.48 op de rekening.



11.2 Iteratives: for-loop



```
for count in range(1, 5):  
    print(count)
```

```
for (int count = 1; count < 5; count++) {  
    System.out.println(count);  
}
```

Output = ?



1
2
3
4



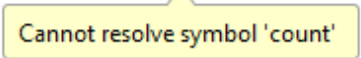
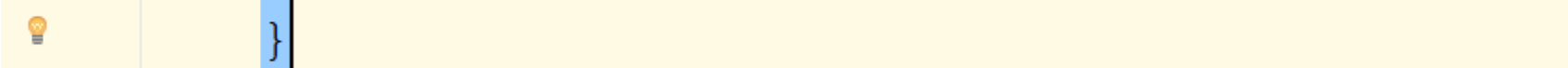

for (*initialisation*; *condition*; *increment*) {
 statements;
}



Bereik van lus-variabele

Omdat de variabele `count` geïnitialiseerd is in de `for`-loop, is deze variabele enkel zichtbaar binnen de `for`-loop.

```
public class Demo3 {  
    public static void main(String[] args) {  
        for (int count = 10; count >= 0; count--) {  
            System.out.println(count);  
        }  
        System.out.println(count);  
    }  
}
```



Opdracht 12: *For-loop*

Maak een object van de klasse Thermometer (zie opdracht 9).
Gebruik een for-loop om de volgende tabel af te drukken:

Celsius	Fahrenheit
-10,00	14,00
-5,00	23,00
0,00	32,00
5,00	41,00
10,00	50,00
15,00	59,00
20,00	68,00
25,00	77,00





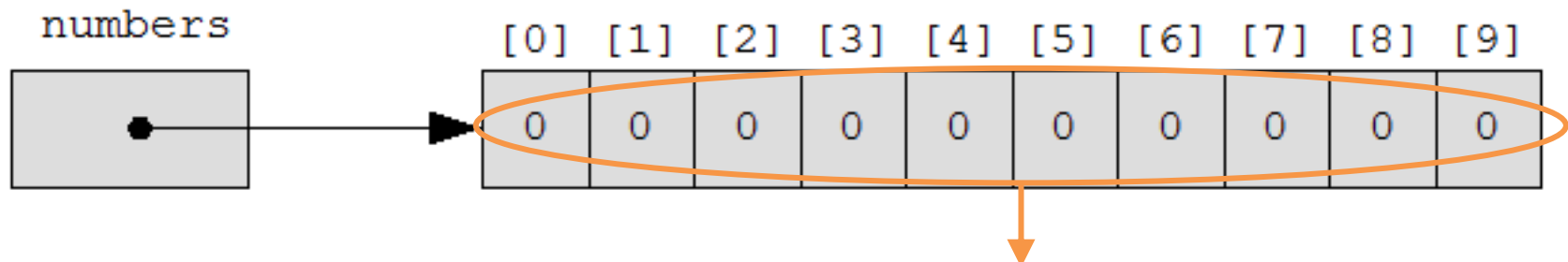
12. Arrays

array = verzameling van elementen van hetzelfde type

```
int[] numbers = new int[10];
```

Mag variabele zijn.

Arrays hebben een vaste lengte en kunnen nadien niet meer gewijzigd worden.



Wordt automatisch geïnitieerd op 0 (null voor referentietypen/false voor booleans).



Declaratie en initialisatie :

```
int[] numbers = {1, 3, 6, 8, 4};
```

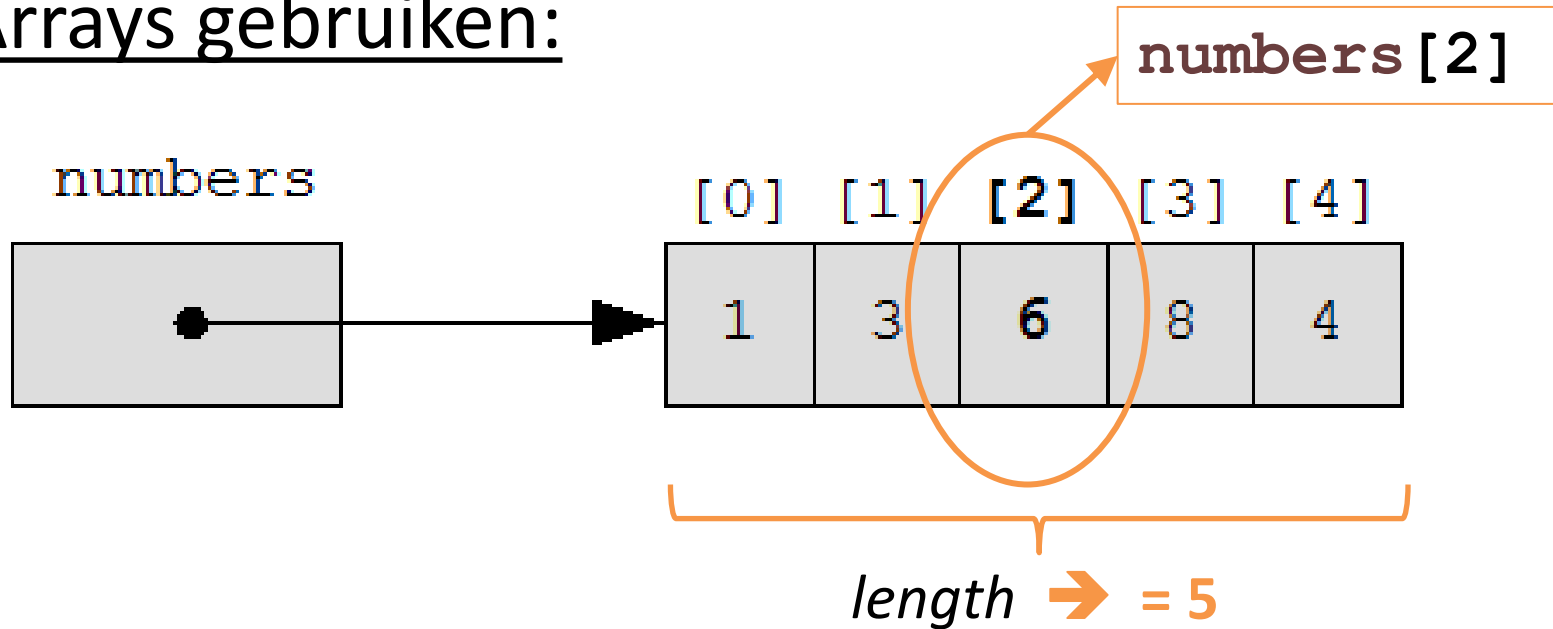


```
int[] numbers;  
numbers = new int[] {1, 3, 6, 8, 4};
```

```
Scanner scanner = new Scanner(System.in);  
int[] numbers = new int[5];  
for (int i = 0; i < numbers.length; i++) {  
    System.out.println("Nr. " + (i + 1) + ":");  
    numbers[i] = scanner.nextInt();  
}
```



Arrays gebruiken:



```
for (int i = 0; i < numbers.length; i++) {  
    System.out.println(numbers[i]);  
}
```

Waarom '<'?





Een waarde aan een array-element toekennen:

```
numbers[2] = 12;  
numbers[4] = getal;  
numbers[5] = -4; → java.lang.ArrayIndexOutOfBoundsException
```

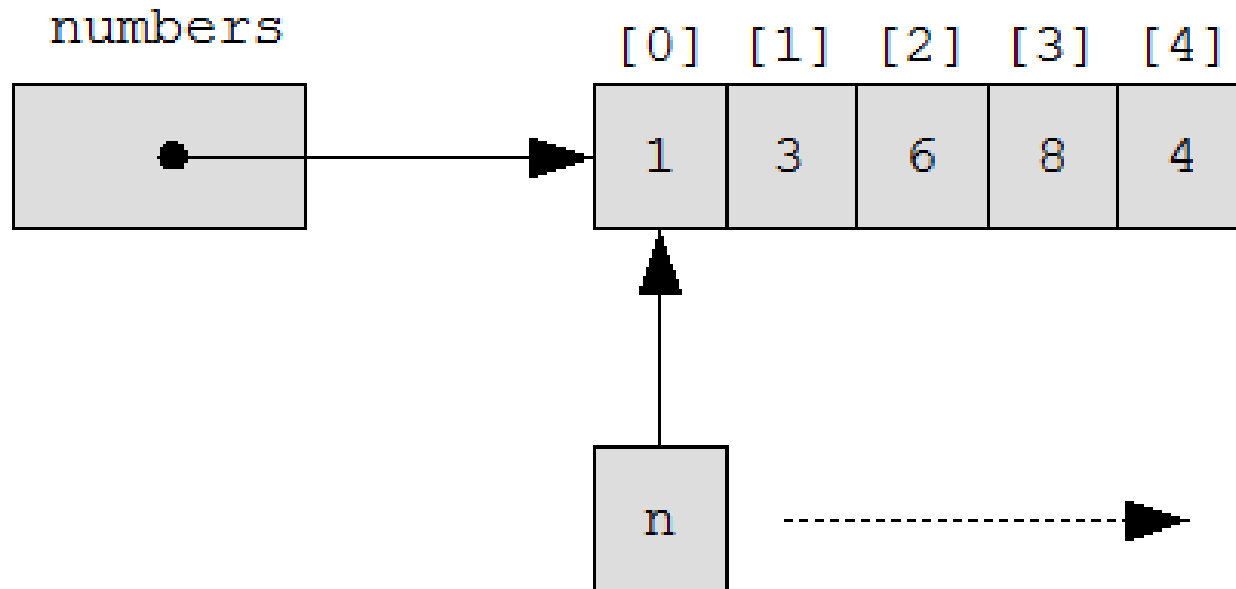




For each-loop:

```
int[] numbers = { 1, 3, 6, 8, 4 };  
  
for (int n : numbers) {  
    System.out.println(n);  
}
```

De variabele *n* neemt tijdens de iteratie één voor één de waarde aan van de elementen uit de *array*.



Voordeel: korte notatie

Nadeel:

- Je beschikt niet over de indexwaarde.
- Enkel geschikt als je alle waarden van de tabel wil doorlopen.

Opdracht 13: *Arrays gebruiken*

- Maak een klasse SalesPerson.
- Objecten van de klasse SalesPerson hebben volgende eigenschappen:
 - een naam
 - een array met de verkoops cijfers van 1 jaar
- Voorzie een methode *setMonthlySale(int, double)* om voor een gegeven maand het verkoops cijfer op te slaan.
- Voorzie de methoden *getTotalSale()* en *getAverageSale()* om respectievelijk het totale verkoops cijfer (voor 12 maanden) en het gemiddelde verkoops cijfer te berekenen.
- Maak een hoofdprogramma waarin je een object maakt van de klasse SalesPerson om alles te testen.

SalesPerson		
f	name	String
f	monthlySales	double[]
m	setMonthlySale(int, double)	void
m	getAverageSale()	double
m	getTotalSale()	double



13. Gegevens doorgeven aan een methode

```
public class Auto {  
    public int kilometerstand;  
  
    public void rijden(int uren, int snelheid) {  
        kilometerstand += uren * snelheid;  
    }  
}
```

= parameters

Bereik van
uren en
snelheid

waarde 2 wordt gekopieerd naar **uren**

= call-by-value

waarde 70 wordt gekopieerd naar **snelheid**

```
public class AutoApp {  
    public static void main(String[] args) {  
        Auto auto = new Auto();  
        auto.rijden(2, 70);  
    }  
}
```

= argumenten



```
public class Auto {  
    public int kilometerstand;  
  
    public void rijden(int uren, int snelheid) {  
        kilometerstand += uren * snelheid;  
        uren = 0;  
        snelheid = 0;  
    }  
}
```

```
public class AutoApp {  
    public static void main(String[] args) {  
        Auto auto = new Auto();  
        int uren = 2;  
        int gemiddeldeSnelheid = 70;  
        auto.rijden(uren, gemiddeldeSnelheid);  
        System.out.println(uren);  
        System.out.println(gemiddeldeSnelheid);  
    }  
}
```

→ 2
→ 70

➔ Het wijzigen van de waarde van *uren* in de methode *rijden* heeft geen invloed op de waarde van *uren* in de *main*. De waarde van de variabele *uren* wordt immers doorgegeven (gekopieerd) naar de parameter *uren*.



```
public class Auto {  
    public int kilometerstand;  
  
    public void rijden(int[] afstanden) {  
        for (int afstand: afstanden) {  
            kilometerstand += afstand;  
        }  
    }  
}
```

Bereik van
afstanden

de referentie van ritten wordt
gekopieerd naar afstanden

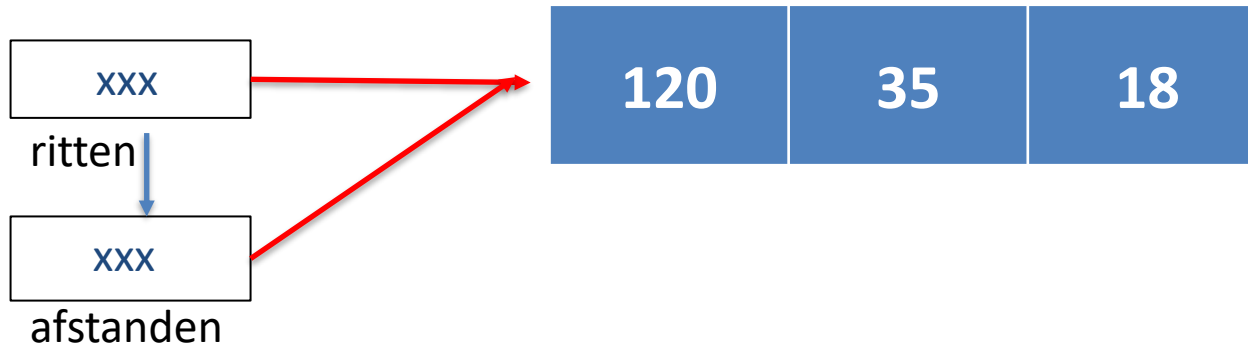
```
public class AutoApp {  
    public static void main(String[] args) {  
        Auto auto = new Auto();  
        int[] ritten = {120, 35, 18};  
        auto.rijden(ritten);  
    }  
}
```

```
public void rijden(int[] afstanden) {  
    for (int afstand: afstanden) {  
        kilometerstand += afstand;  
    }  
}
```

= call-by-reference

Voor referentietypes wordt de referentie doorgegeven.

```
auto.rijden(ritten);
```



```
public class Auto {  
    public int kilometerstand;  
  
    public void rijden(int[] afstanden) {  
        for (int afstand: afstanden) {  
            kilometerstand += afstand;  
        }  
        afstanden[0] = 2000;  
    }  
}
```

```
public class AutoApp {  
    public static void main(String[] args) {  
        Auto auto = new Auto();  
        int[] ritten = {120, 35, 18};  
        auto.rijden(ritten);  
        System.out.println(ritten[0]); ➡ ?  
    }  
}
```



```
public void rijden(int[] afstanden) {  
    for (int afstand: afstanden) {  
        kilometerstand += afstand;  
    }  
    afstanden[0] = 2000;  
}
```

= call-by-reference

Voor referentietypes wordt de referentie doorgegeven.

```
auto.rijden(ritten);
```

