



Programming Basics

Hoofdstuk 6

Objectgeoriënteerd programmeren

DE HOGESCHOOL MET HET NETWERK

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



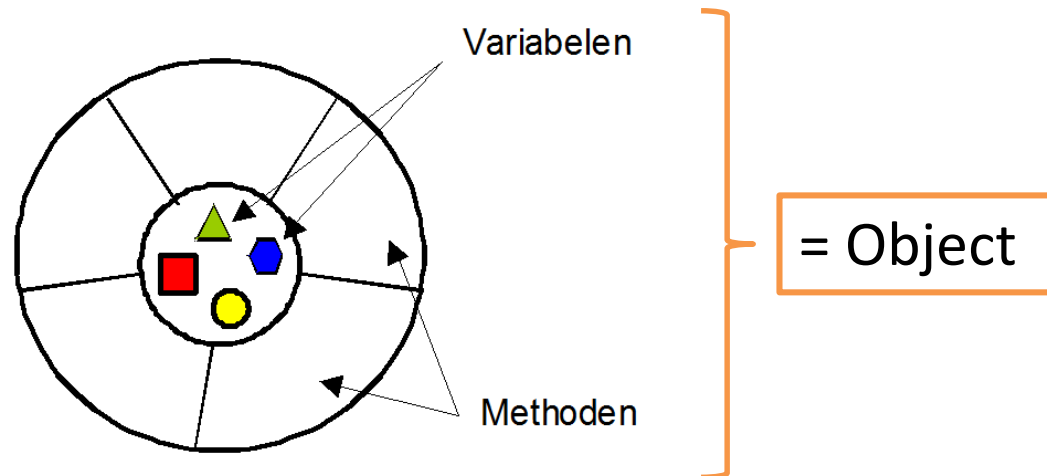
1. Inhoud

1. Inleiding in het objectgeoriënteerd programmeren
2. Werken met bestaande objecten
3. Tekenreeksen
4. Samenvatting



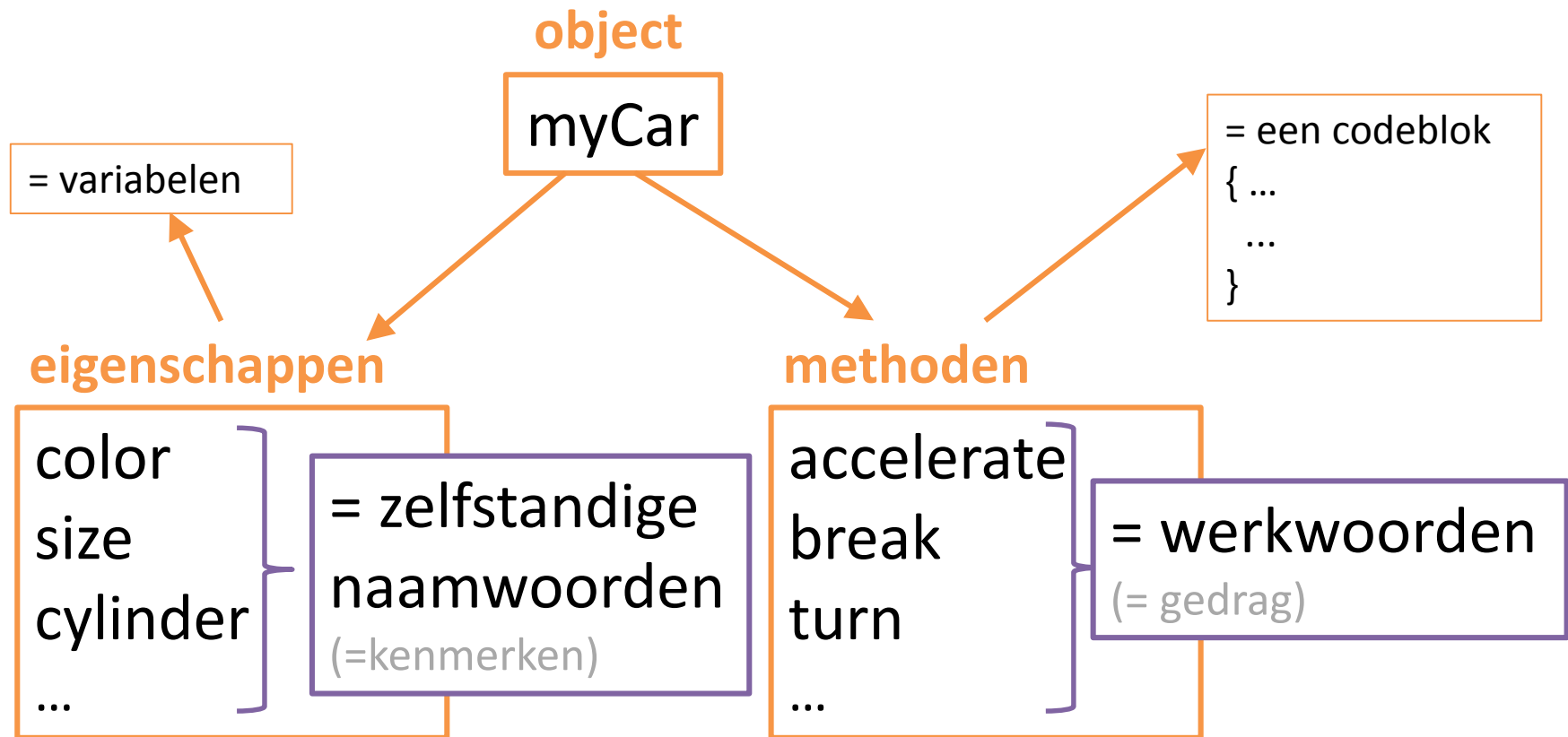
2. Inleiding in het object-georiënteerd programmeren

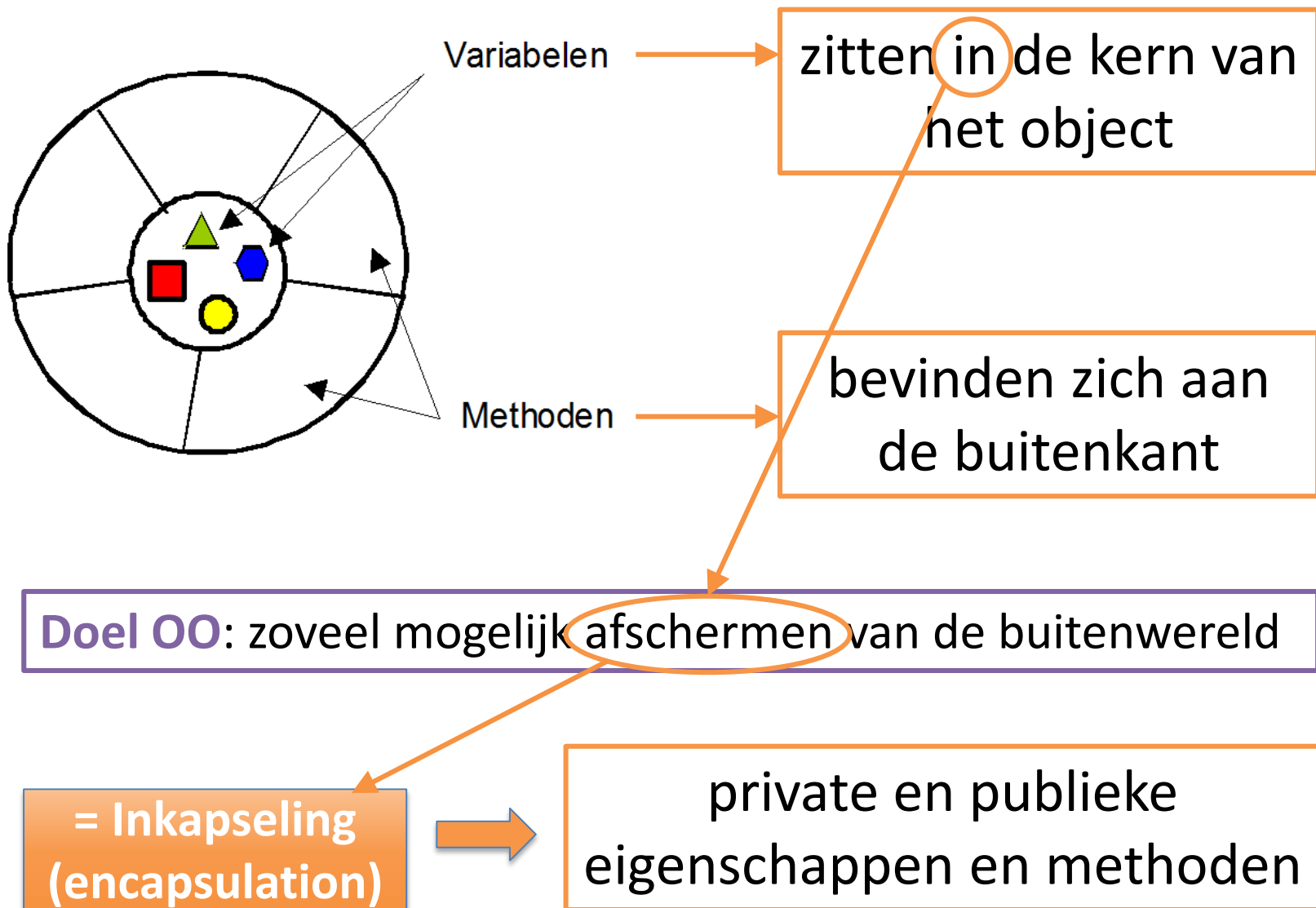
2.1 Objecten



Object = groepering van variabelen (**eigenschappen**) en gerelateerde codeblokken (**methoden**)

Voorbeeld:





Een eigenschap aanspreken:

`objectName.property`

○ = kleine letter

Vb: `myCar.color`

Een methode aanspreken:

`objectName.method()`

→ een methode heeft
altijd haakjes

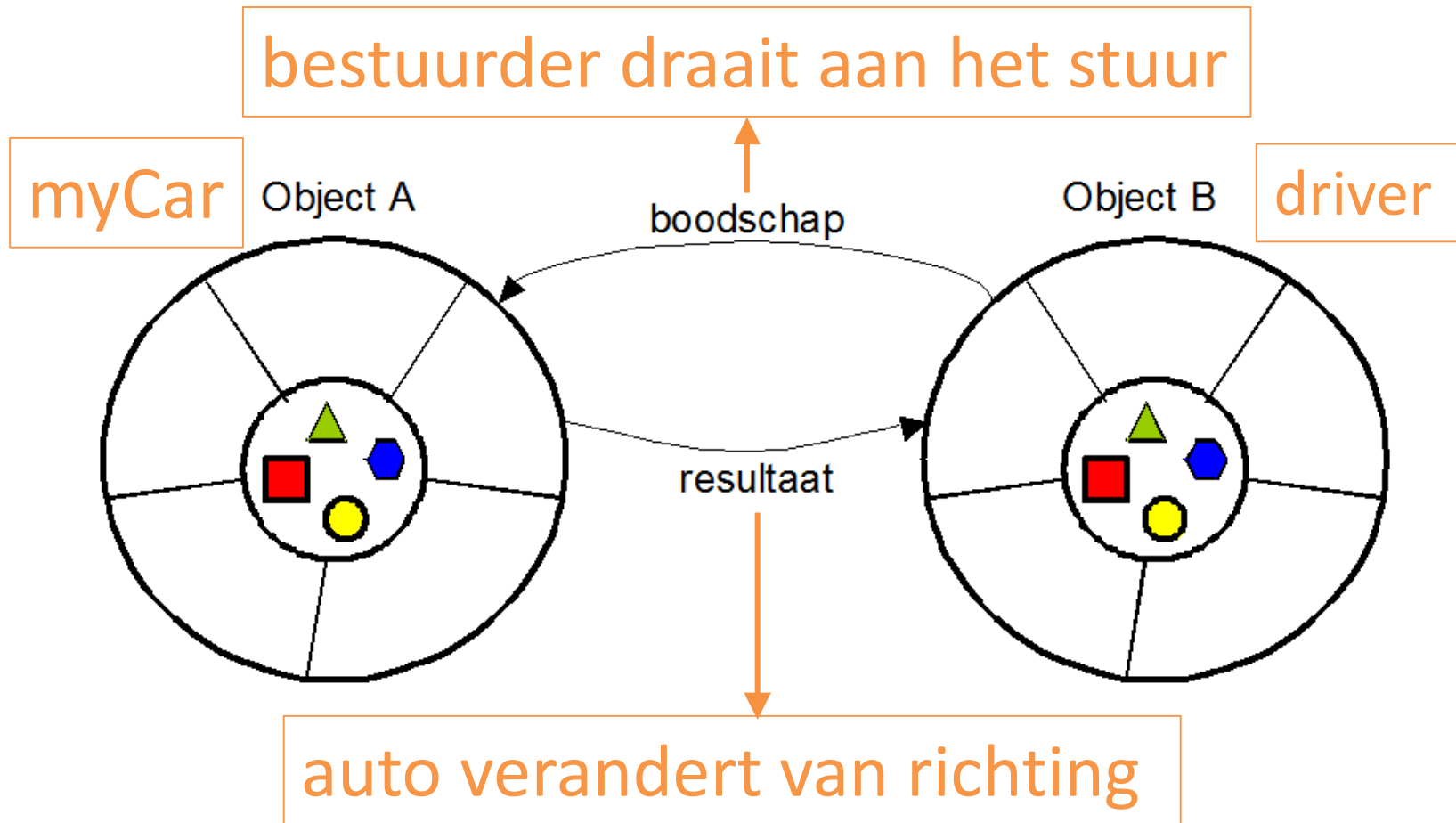
Vb: `myCar.turn()`

Voordelen OO:

- **Modulariteit:** we gaan de code opdelen in modules → zijn herbruikbaar (reusable)
(ik kan mijn auto uitlenen aan vriend en die kan daar ook mee rijden)
- **Afscherming** van informatie (de auto-constructeur kan perfect een nieuwe motor in mijn auto steken; ik ga daar nog kunnen mee rijden)



2.2 Boodschappen (message / methodcall)

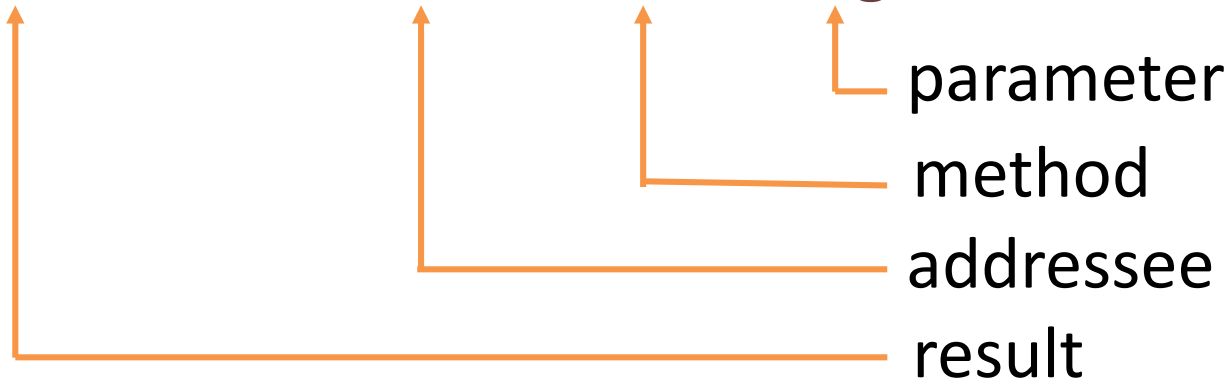


Boodschappen hebben 4 kenmerken:

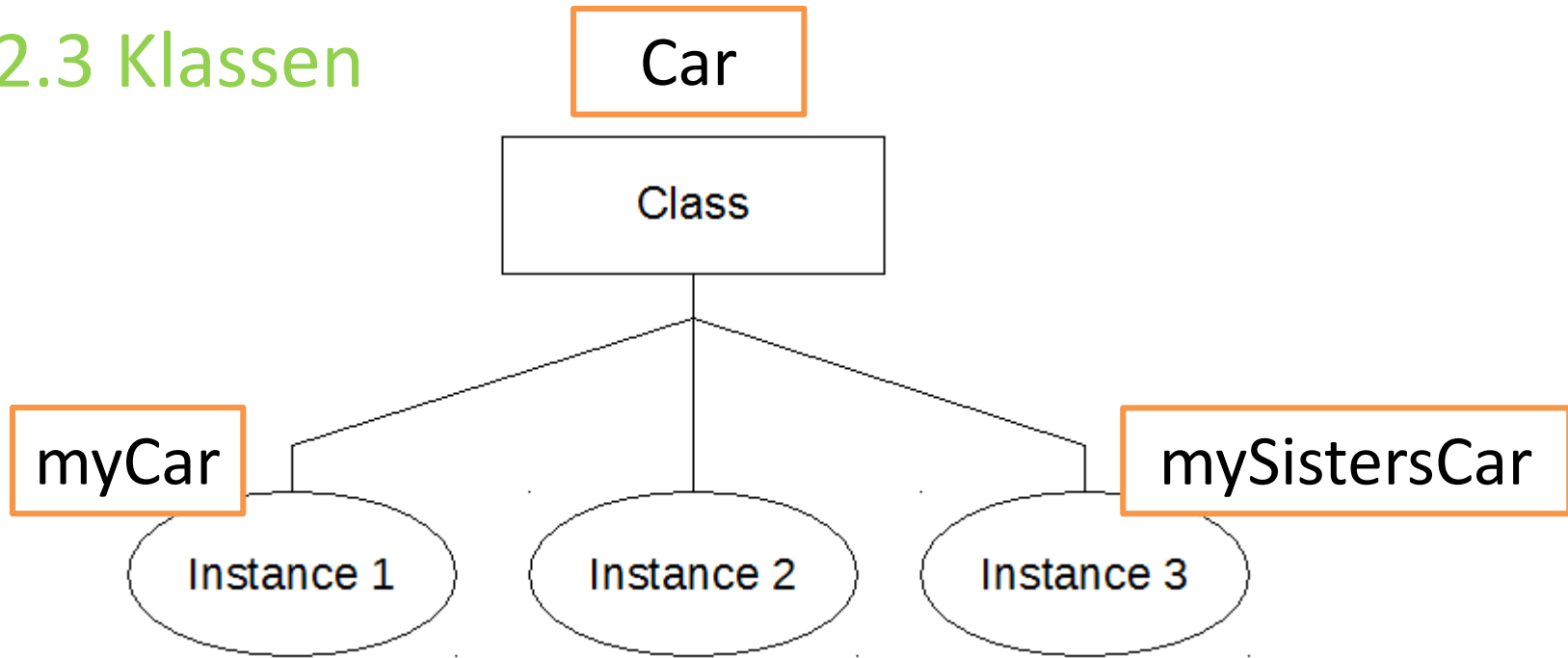
- Bestemming
- Methode-aanroep
- Parameters (optioneel)
- Resultaat van de boodschap (optioneel)

De bestuurder laat een stuk interne code van de auto uitvoeren om de auto te laten draaien, nl.

```
direction = myCar.turn(angle);
```



2.3 Klassen



Zoals mijn auto (= object / instantie) zijn er meerdere. Al deze wagens zijn gefabriceerd o.b.v. een blauwdruk (= klasse).



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



Soorten variabelen:

- Instantievariabelen: vb: color (iedere auto kan een andere kleur hebben)
- Klassevariabelen: vb: aantalWielen (elke auto van een bepaalde klasse heeft 4 wielen)

Soorten methoden:

- Instance methods
- Class methods

Dit alles zal duidelijker worden in het verdere verloop van deze cursus!



3. Werken met bestaande objecten

3.1 Inleiding

We gaan voorlopig nog geen klassen zelf schrijven, maar we gaan gebruik maken van **bestaande klassen** die reeds aanwezig zijn in het Java-platform.



3.2 Objecten maken

Wij gaan een object maken van de bestaande klasse *Random* (bevindt zich in het pakket *java.util*) → volledige naam van de klasse = *java.util.Random*. De bytecode bevindt zich in **Random.class**.

```
java.util.Random rand = new java.util.Random();
```



Genereert een reeks random getallen

= aanroep van de **constructor** (= speciale methode die het object initialiseert) van de *Random* klasse



```
java.util.Random rand = new java.util.Random();
```

of

```
import java.util.Random;  
public class oef1{
```

```
    ...
```

```
        Random rand = new Random();
```

Volledige klassenaam
importeren (om niet telkens
java.util. te moeten herhalen).



```
Random rand = new Random();
```

of

```
Random rand; //declaratie van de referentievariabele  
rand = new Random(); //creatie en initialisatie
```

= instantiatie

= constructor

met new-operator

Om object tijdens creatie te initialiseren

rand

adres

random
generator
object

Wat zou het verschil zijn tussen

```
import java.util.Random;
```

en

```
import java.util.*;
```

?



3.3 Objecten gebruiken

= methode van de
klasse Random

```
int value = rand.nextInt();
```

Neemt het eerstvolgend geheel getal van
de reeks gegenereerde random getallen.

Start de JavaDoc (snelkoppeling op je bureaublad). Ga naar de klasse Random. Vind je de uitleg over de constructor en de *nextInt()* method? Vind je ook een methode om getallen te genereren tussen 0 en een bepaalde waarde?

```
rand.nextInt(int bound);
```

Bovengrens niet inclusief!

```
vb: value = rand.nextInt(5); → 0, 1, 2, 3 of 4
```



Opdracht 1: *Objecten maken en gebruiken*

- Maak een programma dat een random geheel getal genereert. Druk het getal af op het scherm.
- Pas het programma aan zodanig dat er 20 gehele getallen gegenereerd worden tussen 0 en 10 (10 incl.). Druk ze af op het scherm.
- Pas het programma aan zodanig dat er 10 getallen gegenereerd worden tussen 20 en 30 (30 incl.)



4. Tekenreeksen

4.1 Inleiding

We behandelen 2 bestaande klassen voor het bewerken en opslaan van tekst:

- String
- StringBuilder



4.2 De klasse *String*

- Behoort tot het pakket *java.lang* (wordt standaard geïmporteerd)
- Bij de creatie van het object wordt een initiële waarde toegekend die nadien **niet meer kan veranderd worden!** (strings zijn immutable)

- `String text = new String("Hello World");`

We maken hier een object van de klasse *String* en we kennen via de constructor de **string literal** “Hello World” toe (= initialisatie).

= referentie naar het object

= argument



```
String text = new String("Hello World");
```



Vermits Java elke string literal automatisch omzet in een object van de klasse *String*, kan men een string ook als volgt declareren.

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

Opdracht 2: *Java API Documentation*

- Ga in JavaDoc naar de klasse *String*.
- Kijk of je de constructoren vindt.
- Zoek de methode *toUpperCase()*. Wat doet deze methode? Test onderstaande code in Eclipse.

```
String str = "abc";  
str.toUpperCase();  
System.out.println(str);
```

Waarom werkt deze code niet? Pas de code aan zodanig dat de code werkt.



Aan de hand van JavaDoc zou je alle methodes van de String klasse moeten kunnen gebruiken. Een lijst van een aantal methodes vind je in de cursus.

Enkele voorbeelden:

= positie 0!!!

```
String text = "Hello World";  
char c = text.charAt(8);  
System.out.println(c); // r
```

Positie van een bepaald karakter opvragen:

```
String text = "Hello World";  
int pos = text.indexOf('l');  
System.out.println(pos); // 2
```

➔ Geeft het eerste voorkomen van het karakter. Hoe volgende karakters opvragen?

```
System.out.println(pos); // 2  
int pos2 = text.indexOf('l', pos + 1);  
System.out.println(pos2); // 3  
int pos3 = text.indexOf('l', pos2 + 1);  
System.out.println(pos3); // 9
```

= startpositie




```
int pos = text.indexOf('1');
```

1 parameter

```
int pos2 = text.indexOf('1', pos + 1);
```

= method overloading

2 parameters

Methodes met eenzelfde naam maar verschillend aantal en/of type parameters.

Vind je nog een overloaded method van indexOf?

```
indexOf(String str)
```



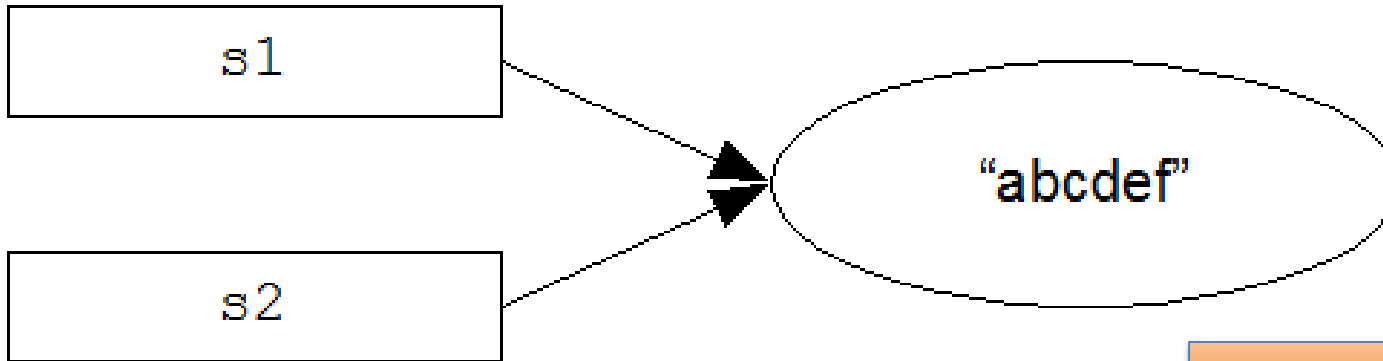
Geheugengebruik bij strings

string constant pool: zodra een nieuw string object aangemaakt wordt d.m.v. een *string literal*, gaat de VM na of deze string zich reeds in de *pool* bevindt en wordt er eventueel een referentie naar de reeds bestaande string teruggegeven.



```
String s1 = "abcdef";  
String s2 = "abcdef";  
System.out.println(s1 == s2); // true
```

s1 en s2 bevatten immers hetzelfde adres

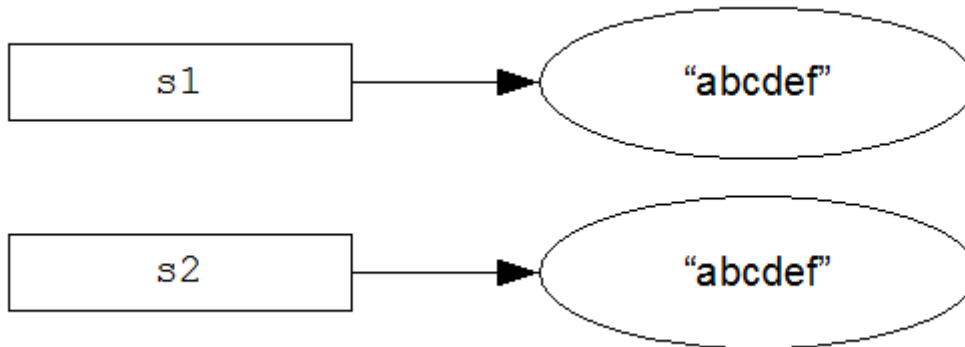


= canonieke strings

➔ Geheugen wordt geoptimaliseerd.

Indien men echter een string maakt d.m.v. de constructor wordt er een nieuw object aangemaakt.

```
String s1 = "abcdef";  
String s2 = new String("abcdef");  
System.out.println(s1 == s2); // false
```



s1 en s2 bevatten niet hetzelfde adres

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

```
String s1 = "abcdef";  
String s2 = new String("abcdef");  
boolean eq = s1.equals(s2);  
System.out.println(eq); // true
```

Strings vergelijken met == is uit den boze tenzij men zeer bewust de referentie wil vergelijken.



Volgende code is handig om strings in te lezen

cfr. keyboard.nextInt();

```
Scanner keyboard = new Scanner(System.in);  
String text = keyboard.next();
```

```
Scanner keyboard = new Scanner(System.in);  
String text = keyboard.nextLine();
```

Wat is het verschil tussen de methode next() en nextLine()? (Zoek op in de JavaDoc)



Opdracht 3: *Werken met String-objecten*

- Maak een programma met een regel tekst. Druk de tekst en de lengte van de tekst af.
- Druk de tekst af met allemaal hoofdletters.
- Druk de tekst af met allemaal kleine letters.
- Vervang de letters 'a' door de letters 'o' en druk het resultaat af.
- Druk het aantal letters 'e' af? Hint: gebruik een *for*-lus.
- Maak 2 strings met verschillende inhoud en ga na of de inhoud gelijk is. Druk false of true af.

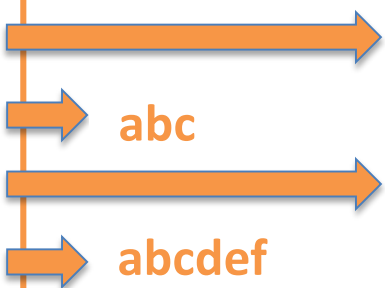


- Vergelijk de 2 strings alfabetisch en druk de kleinste eerst af.
- Maak een string met extra spaties van voor en van achter. Druk de string zonder deze extra spaties af.
- Schrijf een programma om 2 tekstvariabelen naam en voornaam samen te voegen tot 1 variabele. De nieuwe variabele bestaat uit de eerste letter van de voornaam in hoofdletter gevolgd een punt en een spatie, gevolgd door de achternaam waarbij de 1ste letter in hoofdletters is en de andere letters in kleine letters zijn.
- Schrijf een programma om de middelste of de 2 middelste letters van een woord in hoofdletters te zetten.



4.3 De klasse *StringBuilder*

```
String s1 = "abc";  
s1.concat("def");  
System.out.println(s1);  
s1 = s1.concat("def");  
System.out.println(s1);
```



s1 blijft ongewijzigd
(strings zijn immutable)



hier wordt een nieuw
object aangemaakt en
het oude object wordt
weggegooid (door
garbage collector)!

Stel je voor dat `s1 = s1.concat("def");` in een for-lus zou staan die 100x doorlopen wordt, dan wordt er 100x een nieuw object aangemaakt en 100x een object weggegooid → = CPU-tijd die verloren gaat → oplossing = `StringBuilder`

```
StringBuilder s1 = new StringBuilder("abc");  
s1.append("def");  
System.out.println(s1);
```



→ abcdef

hier “bouwt” (StringBuilder) men een string!!! De bestaande string “abc” wordt behouden en “def” wordt toegevoegd.

Indien je `s1.append("def");` in een lus zet die 100x doorlopen wordt, dan wordt er telkens gebruik gemaakt van dat ene object (er worden geen 100 objecten aangemaakt en vernietigd).

Merk op dat verschillende methoden (zie tabel in cursus) niets (**void**) teruggeven. Dat is vaak ook niet nodig want de methode zelf verandert intern de inhoud van het StringBuilder-object en er hoeft niet onmiddellijk iets terug te komen.

```
StringBuilder text = new StringBuilder("Hello World");  
text.setCharAt(4, 'a');  
System.out.println(text);
```

→ text wordt Hella World

```
StringBuilder s = new StringBuilder("dit is een tekst");
System.out.println(s);
s.reverse();
System.out.println(s);
StringBuilder s2 = s;
if (s.equals(s2))
    System.out.println("StringBuilder: equals: zelfde object");
else
    System.out.println("StringBuilder: equals: niet zelfde object");
if (s == s2)
    System.out.println("StringBuilder: ==: zelfde object");
else
    System.out.println("StringBuilder: ==: niet zelfde object");
if (s.toString().equals(s2.toString()))
    System.out.println("String: equals: zelfde inhoud");
else
    System.out.println("String: equals: niet zelfde inhoud");
if (s.toString() == s2.toString())
    System.out.println("String: ==:zelfde object");
else
    System.out.println("String: ==:niet zelfde object");
```

Tracht de output van dit programma te achterhalen.

Output:

```
dit is een tekst  
tsket nee si tid  
StringBuilder: equals: zelfde object  
StringBuilder: ==: zelfde object  
String: equals: zelfde inhoud  
String: ==: niet zelfde object
```

Bij *StringBuilder* doet "*equals()*" hetzelfde als "==" (vergelijkt of het gaat om hetzelfde object in het geheugen). Als je de inhoud van 2 *StringBuilder-objecten* wil vergelijken moet je eerst een *toString()* doen.



Opdracht 4: *Werken met StringBuilder-objecten*

- Zoek in de Java API-documentatie de beschrijving van de klasse *StringBuilder* (in package *java.lang*).
- Maak een programma met 2 regels tekst (met *StringBuilder*). Druk de regels af. (zie output op het einde van Opdracht 4)
- Voeg aan de 1^e *StringBuilder* een stukje tekst toe.



- Keer de tekens van de 2^e *StringBuilder* om. Dit kan perfect met volgende code (een *StringBuilder*-object is wijzigbaar).

```
sb.reverse();  
System.out.println(sb);
```



```
String str = "abc";  
str.toUpperCase();  
System.out.println(str);
```

String is niet wijzigbaar!

- Verwijder alle spaties uit de 1^e stringbuilder.
- Verdubbel iedere letter 't' in de 2^e stringbuilder.

Output:

This is my first line of text

This is my second line of text

This is my first line of text and this is what I added.

txet fo enil dnoceS ym si sihT

ThisismyfirstlineoftextandthisiswhatIadded.

ttxett fo enil dnoceS ym si sihT



4.4 Strings samenvoegen met de +-operator

StringBuilders worden samengevoegd met de *append*-methode. *Strings* worden samengevoegd met de *concat*-methode. Het is echter ook mogelijk om Strings samen te voegen met de +-operator (= makkelijker!).

```
text3 = text1 + text2;
```

Eigenlijk doet de compiler onderstaande.

```
text3 = new StringBuilder().append(text1).  
                                append(text2).toString();
```

Tracht bovenstaande uit te leggen!



Nog een voorbeeld:

```
System.out.println("Een getal: " + number);
```

Eigenlijk doet de compiler onderstaande.

```
System.out.println(new StringBuilder().  
    append("Een getal: ").append(number).toString());
```

Tracht bovenstaande weer uit te leggen!

Welk resultaat geeft onderstaande code als output?

```
System.out.println(3 + 7 + "Hello World!" + 3 + 7);
```

➔ 10Hello World!37



4.5 Gegevens formatteren met methode printf()

```
int number1 = 5;  
int number2 = 10;  
double number3 = 0.3388888;  
String text = "Hello";
```

```
System.out.printf("Een integer: %d", number1);
```

= 'placeholder' of 'format specifier' (= 'gat in string' dat we opvullen met *number1*)



Een integer: 5

```
System.out.printf("Een integer %d en een tweede integer %d", number1, number2);
```

wordt opgevuld met *number1*

wordt opgevuld met *number2*

➡ Een integer 5 en een tweede integer 10

```
System.out.printf("Een floating point getal: %f", number3); ➡ Een floating point getal: 0,338889
```

```
System.out.printf("Afgerond op 2 cijfers na de komma: %.2f", number3); ➡ Afgerond op 2 cijfers na de komma: 0,34
```

```
System.out.printf("Afgerond op 2 cijfers na de komma: %6.2f", number3); ➡ Afgerond op 2 cijfers na de komma: 0,34
```

```
System.out.printf("Een string: %s", text); ➡ Een string: Hello
```

6 posities waarvan er 2 na de komma (handig om uit te lijnen)



Opdracht 5: *Getallen formatteren*

Maak een programma dat de eenheid 'meter' omzet naar de eenheid 'voet'. Toon de waarden van 1 meter t.e.m. 20 meter (toename van 0,5 meter) en de overeenkomstige waarden in 'voet'. Zorg dat slechts 2 cijfers na de komma getoond worden en dat alle getallen mooi uitgelijnd zijn.

1 meter = 3.2808399 voet



5. Samenvatting

- **Object**

- bestaat uit:

- eigenschappen (**properties**)
 - gedragingen (**methods**)

- wordt gemaakt o.b.v. een blauwdruk (**klasse**)

- Random
 - constructor
 - publieke methoden
 - String en StringBuilder
 - constructor
 - publieke methoden

- Tekst hebben we geformatteerd met de methode *printf()*.

