

# Informatica: Java reeks 2

klassen en objecten

Ruben De Smet

2<sup>de</sup> semester 2021

Java gewenning:

- ▶ Nieuwe **syntax**
- ▶ Statisch getypeerd: **types** overal.

Java gewenning:

- ▶ Nieuwe **syntax**
- ▶ Statisch getypeerd: **types** overal.

Vandaag: zelf **types maken**: klassen

# KLASSEN EN OBJECTEN

Een object is een **toestand** met **operaties**.  
Een klasse is een **sjabloon** voor een object.

Voorbeelden van klassen:

## **Klasse**

`Auto` (merk, motor)

`Breuk` (teller, noemer)

`ComplexGetal` (reëel, imaginair)

`Speler` (positie, naam)

## **Object**

Ruben's auto, jouw auto

$\frac{5}{2}$ ,  $\frac{3}{4}$

$5 + 3i$ ,  $4 - 5i$

vooraan in de klas: Ruben

# KLASSEN

Klassen bestaan uit

- ▶ attributen/velden: “reëel deel” (**double**), “teller” (**int**), “naam” (**String**); en
- ▶ methoden: “bereken modulus”, “verplaats”, “vermenigvuldig”, “tel op”.

Alle types in Java zijn klassen, behalve de primitieve types:

**boolean**, **int**, **float**, **double**, **char**, ...

Primitieve types:

- ▶ starten met **kleine letter**,
- ▶ hebben geen methodes,
- ▶ zijn **vetgedrukt** en **gekleurd**.

Klassen beginnen met hoofdletter.

## KLASSEN GEBRUIKEN

```
1  // Object aanmaken:
2  Random rng = new Random();
3
4  // Methode gebruiken
5  double a = rng.nextDouble();
6
7  // ArrayList aanmaken:
8  ArrayList<Integer> arr = new ArrayList<Integer>();
9  arr.add(3);
10 arr.add(4);
11
12 // Element opvragen per index
13 arr.get(1); // Geeft 4 terug.
```

Maak voor oefening 7 `class MyMath`, met enkele nuttige, `static` methodes:

- ▶ `permute(...)` genereert een random permutatie.
- ▶ `range(...)` emuleert het gedrag van Python's `range(...)`.
- ▶ Om breuken te vereenvoudigen hebben we de grootste gemeenschappelijke deler nodig (opgave 8.(f)).
- ▶ Om breuken op te tellen hebben we het kleinste veelvoud nodig (opgave 8.(h)).
- ▶ Herbruikbaar voor de volgende oefeningenreeksen!

Test je methoden in de `public static void main(..)` van een `aparte` klasse.

## BREUKEN

Een breuk heeft twee attributen: een teller en een noemer.

```
1 public class Fraction {
2     public int numerator;
3     private int denominator; // should never be 0
4
5     // Constructor
6     public Fraction(int numerator, int denominator) {
7         this.numerator = numerator;
8         this.denominator = denominator;
9     }
10 }
```



- ▶ “Getter” en “setter” voor noemer verhindert derden van `breuk.denominator = 0`; te schrijven.
- ▶ `String toString()` methode om te kunnen printen.
- ▶ Optellen en vermenigvuldigen van breuken.

## BREUKEN

### GETTERS EN SETTERS

Getter en setter methodes staan toe **extra code** uit te voeren tijdens het aanpassen van een attribuut, bijvoorbeeld validiteitschecks, on-the-fly uitrekenen van waarden, ...

```
1 public class Fraction {
2     private int denominator;
3     ...
4     public void setDenominator(int denominator) {
5         if (denominator == 0) { ... }
6         this.denominator = denominator;
7     }
8     public int getDenominator() {
9         return this.denominator;
10    }
11 }
```

## CODE TESTEN

Belangrijk: altijd code **testen**.

Bijvoorbeeld

```
1 Fraction f = new Fraction(2, 6);  
2 f.reduce();  
3 System.out.println(f); // moet 1/3 printen (dankzij  
   ↪ toString()).  
4 System.out.println(f.evaluate()); // moet 0.33...  
   ↪ printen.
```