

# Informatica: Java reeks 2: klassen en objecten

Ruben De Smet

2<sup>de</sup> semester 2021

Vorige les hebben we in oefeningen 4 en 5 enkele eenvoudige methodes gemaakt, alsof ze Python functies waren. Deze reeks focust op klassen, objecten en methoden, de basisbegrippen van het objectgeoriënteerd programmeren.

Maak alvast een nieuw project Les2 aan, om alle code van vandaag in op te slaan.

**Oefening 7** (Gebruiken van ingebouwde klassen). *In de vorige reeks hebben we al een ingebouwde klasse gebruikt: de Scanner. Die las van het toetsenbord en trachtte integers, doubles en andere datatypes te vormen.*

*Java heeft erg veel ingebouwde klassen. In deze eerste oefening gebruiken we de ArrayList, Random klassen. Een ArrayList is zoals een list in Python: een geordende verzameling van elementen. Daar je met Python types in een lijst mixen naar wens (bv. `l = [10, "abc"]`), kan dat niet in Java. De Random klasse kan pseudorandom data genereren.*

- (a) Maak een `class MyMath`, (zonder `public static void main(...)`) waarin je onderstaande `static` methodes maakt. Test de volgende methodes vanuit een aparte klasse met `public static void main(...)`. Als twee klassen in dezelfde package staan, kan je elkaars `public` methodes oproepen met `Klasse.methodeNaam()`.
- (b) Maak een methode `ArrayList<Integer> range(int start, int end, int step)`, equivalent aan de `range` functie van Python. Gebruik bijvoorbeeld een `for` lus en vul een (initieel lege) `ArrayList`.
- (c) Maak een methode `ArrayList<Integer> permute(ArrayList<Integer> in)`. Deze methode geeft een *nieuwe* array terug met dezelfde getallen uit `in`, in een willekeurige volgorde (een willekeurige permutatie). Gebruik de `Random` klasse.
- (d) Gebruik je `range` methode om je permutatie uit te testen. Gebruik bijvoorbeeld `permute(range(...))`, zo kan je zien wat je permutatiemethode doet.
- (e) In opgave 8.(f) hebben we straks de grootste gemeenschappelijke deler nodig. Maak een methode voor de "greatest common divisor" `int gcd(int a, int b)`, die de gcd van `a` en `b` teruggeeft.

**Hint:** In het boek staat het Euclidisch algoritme<sup>1</sup> uitgeschreven in hoofdstuk 0 op p. 10!

---

<sup>1</sup>Met het Euclidisch algoritme kan je de grootste gemeenschappelijke deler uitrekenen: [https://en.wikipedia.org/wiki/Euclidean\\_algorithm](https://en.wikipedia.org/wiki/Euclidean_algorithm)

- (f) In opgave 8.(h) hebben we het kleinste gemene veelvoud van twee gehele getallen nodig. Maak een methode voor de “lowest common multiple”<sup>2</sup> `int lcm(int a, int b)`, die het kleinste gemene veelvoud berekent van  $a$  en  $b$ .

De eenvoudigste manier is waarschijnlijk door gebruik te maken van

$$\text{lcm}(a, b) = \frac{|a \cdot b|}{\text{gcd}(a, b)}.$$

We gaan nu experimenteel testen hoezeer `class Random` en de nieuwe `MyMath.permute()` methode echt random zijn. Neem waar dat voor een willekeurige permutatie van  $0, 1, 2, \dots, 6$ , we verwachten dat 2 volgt op 1 in  $1/6$  van de gevallen. Die eigenschap gaan we testen aan de hand van een breuk.

**Oefening 8** (Breuken). Een breuk is een verhouding tussen een teller  $a$  en een noemer  $b$ . Door een breuk als combinatie van teller en noemer op te slaan — in plaats van als een enkele `float` of `double` — heb je altijd een exacte representatie van het voorgestelde getal.

- (a) Maak een `class Fraction`<sup>3</sup>. Voeg velden toe van gehele getallen voor teller en noemer<sup>4</sup>.
- (b) Geef `class Fraction` twee constructors. Eén constructor neemt teller en noemer als argument, de andere enkel een geheel getal; stel de noemer dan gelijk aan 1.
- (c) Genereer een getter en een setter<sup>5</sup> voor de *noemer*. Zorg dat de setter van de noemer een exception geeft wanneer je de noemer 0 probeert te maken. Maak het teller veld `public`, maar de noemer `private`.
- (d) Creëer een methode `String toString()`, die een string “ $a/b$ ” teruggeeft — uiteraard met de teller en de noemer ingevuld<sup>6</sup>.
- (e) Voeg een methode `double evaluate()` toe, die de (imperfecte) deling  $a/b$  maakt.
- (f) Voeg een methode `void reduce()` toe, die de breuk vereenvoudigt. Herinner je dat een breuk  $a/b$  gelijk is aan een breuk  $a/c/b/c$ , met  $c$  de grootste gemene deler. Gebruik je methode uit opgave 7.(e) om  $c$  te berekenen, en pas de velden aan.
- (g) Voeg `static Fraction multiply(Fraction lhs, Fraction rhs)` toe<sup>7</sup>. Deze methode neemt twee breuken als argument, en geeft hun product als *nieuwe* breuk terug.
- (h) Voeg `static Fraction add(Fraction lhs, Fraction rhs)` toe. Deze methode neemt twee breuken als argument, en geeft hun optelling als *nieuwe* breuk terug.

Om twee breuken op te tellen, zetten we beide eerst op gelijke noemer. De nieuwe noemer is het kleinste gemene veelvoud van de oorspronkelijke noemers. Gebruik de methode uit opgave 7.(f).

<sup>2</sup>[https://en.wikipedia.org/wiki/Least\\_common\\_multiple](https://en.wikipedia.org/wiki/Least_common_multiple)

<sup>3</sup>Fraction is Engels voor breuk.

<sup>4</sup>Programmeren gebeurt meestal in het Engels; een teller is dan een numerator, en een noemer is dan een denominator

<sup>5</sup>Een getter en een setter zijn methodes die één enkel veld van een klasse lezen en schrijven.

<sup>6</sup>De `String toString()` methode wordt gebruikt door Java zelf, wanneer je probeert het object zelf te printen (met `System.out.println(...)`).

<sup>7</sup>lhs en rhs staan voor left hand side en right hand side.

**Oefening 9** (Testen van de Random klasse). De Random klasse is een pseudorandomgenerator, meer bepaald een lineaire congruente generator<sup>8</sup>. Als de output sequentie  $(x_1, \dots, x_i, \dots)$  uniform verdeeld is, moet de permutatie uit opgave 7.(c) ook uniform zijn.

- (a) Gebruik `range(...)` om een ArrayList te maken van de 7 getallen  $0, \dots, 6$ .
- (b) Gebruik een **for** lus om  $n$  keer een willekeurige permutatie te genereren. Gebruik daartoe je methode uit opgave 7.(c). Hou in een breuk bij hoe vaak 2 volgt op 1 in de permutaties. In de noemer staat het totaal aantal trekkingen, in de teller het aantal keer dat 2 volgt op 1.
- (c) Reken uit hoe ver de resulterende breuk  $a/b$  afwijkt van  $1/6$ . Gebruik hiertoe je vermenigvuldiging en optelling  $(100 \cdot (a/b + -1/6))$ , en de evaluate methode uit opgave 8.(e). Print het resultaat.
- (d) Pas opgave 9.(c) aan, zodat je ook de afwijking van  $a/b$  met  $1/5$  en  $1/7$  ziet. Zorg dat de resultaten netjes geprint worden als percentage.

**Extra oefening 10** (Extra methodes voor breuken). Opgave 9.(c) vereiste nogal een artificiële manier van methodes aanroepen om tot het gewenste resultaat te komen. Door enkele andere methodes toe te voegen aan **class Fraction**, kan opgave 9.(c) mooier geschreven worden.

- (a) Voeg een methode **void multiply(int factor)** toe, die de breuk vermenigvuldigt met een scalair. Analoog voeg je een methode **void divide(int dividend)** toe, die de noemer vermenigvuldigt met een scalair.
- (b) Maak een methode **Fraction invert()** die het omgekeerde ( $b/a$ ) als een *nieuwe* breuk teruggeeft.
- (c) Maak een methode **Fraction negate()** die het tegengestelde ( $-a/b$ ) teruggeeft als *nieuwe* breuk.
- (d) Maak **static Fraction subtract(Fraction lhs, Fraction rhs)**, die de aftrekking van twee breuken berekent en teruggeeft als *nieuwe* breuk.
- (e) Herschrijf nu opgave 9.(c) gebruik makende van de nieuwe methodes.

**Extra oefening 11** (Complexe getallen). Een complex getal  $c$  wordt doorgaans op twee manieren voorgesteld; ofwel met reëel en imaginair deel ( $a$  en  $b$ :  $c = a + ib$ ), ofwel in poolcoördinaten ( $r$  en  $\theta$ :  $c = r \cos \theta + ir \sin \theta$ ).

- (a) Maak een **class ComplexNumber**. Voeg velden toe voor het reëel en imaginair deel.
- (b) Maak een constructor die reëel en imaginair deel neemt als argumenten.
- (c) Maak één enkele “setter” die de grootte  $r$  en het argument  $\theta$  neemt.
- (d) Maak “getters” die de representatie in poolcoördinaten uitrekenen.

**Hint:** Hierbij enkele nuttige formules voor complexe getallen.

$$\begin{aligned} r &= \sqrt{a^2 + b^2}, \\ \theta &= \arctan2(b, a)^9, \\ a &= r \cos \theta, \\ b &= r \sin \theta. \end{aligned}$$

<sup>8</sup>Een LCG heeft als relatie  $x_{n+1} = px_n + q \pmod{m}$  ([https://en.wikipedia.org/wiki/Linear\\_congruential\\_generator](https://en.wikipedia.org/wiki/Linear_congruential_generator)).

**Version management** Projecten met meerdere ontwikkelaars worden niet op bovenstaande manier gedeeld, maar met software die hier specifiek voor bedoeld is. Momenteel het meest populair is Git<sup>10</sup>. Wegens de complexiteit van Git komt dit niet expliciet aan bod in de cursus, maar raden we het wel aan voor de avontuurlijke studenten.

---

<sup>9</sup>arctan2 is een functie die rekening houdt met het kwadrant waarin een punt zich bevindt.

<sup>10</sup><https://git-scm.com/>