

Informatica: Java reeks 1: van Python naar Java




Ruben De Smet




2^{de} semester 2021

Python programmeerden we met Spyder. In dit tweede semester gaan we Java programmeren, waarvoor we een completere “IDE” nodig hebben; een integrated development environment. Een IDE, volgens Wikipedia:

An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools, and a debugger. Most of the modern IDEs have intelligent code completion.¹

Wij gaan Eclipse² gebruiken. Eclipse is een extreem krachtige en flexibele IDE, waarin je overigens zowat *alle* programmeertalen kan gebruiken, inclusief Python³.

Start alvast Eclipse. Wanneer Eclipse gestart is, creëer je een nieuw Java project:   . “Execution environment” zet je op JavaSE-1.8; noem je project Reeks1. We zullen voor elke les een project aanmaken.

We maken typisch voor *elke oefening* minstens één *nieuwe klasse*. Vandaag kan je een Java-klasse zien als een Python bestand: de klassen van vandaag gaan allemaal uitvoerbaar zijn. Selecteer    en vergeet niet `public static void main(String[] args)` aan te duiden om de klasse te kunnen starten; in deze methode programmeer je de oefeningen!

Oefening 1 (String scanner). *Met de `java.util.Scanner` klasse kan je een eenvoudige maar krachtige vorm van interactiviteit inbouwen in je applicatie. In het boek staat de scanner uitgelegd in hoofdstuk 1, paragraaf 1.2.1. De Java 8 documentatie van de Scanner klasse staat op <https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html>.*

- (a) Creëer een Scanner object. Gebruik de Scanner constructor die als argument enkel een InputStream neemt; geef als InputStream de standard-input⁴ mee:

```
1 Scanner scanner = new Scanner(System.in);
```

- (b) Wat valt je op?

- (c) Laat je scanner een double lezen (gebruik de `nextDouble()` methode van Scanner), waarvan je de vierkantswortel uitrekent (gebruik `Math.sqrt`). Print je vierkantswortel.

¹https://en.wikipedia.org/wiki/Integrated_development_environment

²<https://www.eclipse.org/>. De laatste versie op moment van schrijven is 2018-12.

³Als je Python in Eclipse wil gebruiken, heet dat PyDev: <http://www.pydev.org/>

⁴Standard-input is de stroom van symbolen die je op de console ingeeft, wanneer je programma loopt.

Hint: De Scanner haalt de input van het programma uit de *console* onderin Eclipse. De console opent wanneer je iets print of via `Window >> Show View >> Console` of via sneltoets `Alt + ↑ + Q`. De Scanner leest lijn-per-lijn, dus druk op `Enter ↵` om te bevestigen.

- (d) Wat gebeurt er als je een niet-numerieke input geeft aan je programma?
- (e) Maak een programma dat de vraag “Wat is je naam?” stelt en vervolgens de gebruiker groet met de ingegeven naam: print “Hallo Ruben!”.

Oefening 2 (String operaties). *Maak twee String variabelen.*

- (a) Zet in de éne variabele `"Hallo"` en in de tweede `"wereld"`.
- (b) Print `Hallo wereld` door beide variabelen te combineren.
- (c) Wat kan je nog doen met Strings? Probeer de `*` operator te gebruiken, zoals in Python.
Hint: Als je een syntaxfout maakt, zal Eclipse je waarschuwen met een rode gekartelde lijn. Eclipse waarschuwt echter niet enkel voor syntaxfouten! Ook fouten tegen types worden al vooraleer je je programma probeert te runnen, gerapporteerd.
- (d) Gebruik een scanner zoals in oefening 1 om *twee nieuwe strings* in te lezen. Met een **if**-test kijk je of beide strings nu gelijk zijn. Als ze gelijk zijn, print dan “gelijk” naar de console, anders print je “ongelijk”.

Test eerst `a == b` (wat merk je op?) en dan `a.equals(b)`.

Hint: Strings vergelijken doe je dus met de `equals(String other)` methode!

Oefening 3 (Loops). *Met lussen kan je stukken code herhalen. Net zoals in Python, heeft Java ook **for** (Boek § 0.8) en **while** lussen. De syntax is iets anders; zie slides!*

- (a) Gebruik een **for** lus om `n` keer het woord “fizz” te printen. Haal `n` uit een scanner, zoals in oefening 1, zodat je `n` kan opgeven in de console.
- (b) Gebruik een **while** lus om `n` te delen door `m`. Haal `m` uit een scanner, zoals in oefening 1.

Oefening 4 (Methodes en loops). *Methodes zijn de functies en procedures van het objectgeoriënteerd programmeren. Zoals geleerd in de theorie hangt een methode vast aan een object. Momenteel maken we nog geen eigen objecten, daarom werken we momenteel met **static** methodes. Een **static** methode behoort tot de klasse zonder object en is eigenlijk een functie zoals in Python.*

In deze oefening maken we een methode “vergelijk” die twee strings neemt als argument, en berekent welke van beide voor de andere komt volgens het alfabet.

- (a) Maak een **static** methode `vergelijk`, die twee Strings als argumenten neemt en een **boolean** returnt.
- (b) Wat neem je waar, in een lege methode die een **boolean** hoort te **returnen**?
- (c) Gebruik een **for** lus om de twee strings te vergelijken. Als de eerste String voor de tweede komt in het telefoonboek, moet de methode `true`⁵ teruggeven. Anders geeft de methode `false` terug.

⁵Niet `True`, zoals in Python, maar `true`, zonder hoofdletter.

- (d) Roep je methode op vanuit de main, gebruikmakende van twee strings uit een scanner zoals die van oefening 1.

Extra oefening 5 (Meer methodes en loops). *Methodes, types en loops moet je goed onder de knie hebben.*

- (a) Hermaak opgaves 4.(a), 4.(c) en 4.(d) met een **while** lus. Maak hiervoor een nieuwe methode (vergelijk `while`, bijvoorbeeld).
- (b) Maak een nieuwe **static** methode `schrikkeljaar`. De methode geeft **true** terug als `n` uit opgave 3.(a) deelbaar is door 4, maar niet door 100, tenzij het deelbaar is door 400.

Oefening 6 (Guessing game). *We implementeren een hoger-of-lager spel: het programma genereert een willekeurig getal tussen 1 en 100, waarna de speler moet raden. Bij een foute gok zegt het programma of het getal te laag of te hoog was.*

- (a) Maak een randomgenerator (Boek § 1.2.2). Volgende les zien we in detail hoe dit juist werkt.

```
1 Random rng = new Random();
2 int getal = rng.nextInt(100) + 1;
```

Het getal `getal` is nu een willekeurig getal zodat $1 \leq x \leq 100$.

- (b) Implementeer het spelletje: *zolang* de gebruiker fout gokt, vraag je een nieuwe gok. **Hint:** **while** is Engels voor “zolang”.
- (c) Wat is de meest efficiënte manier om het spel te spelen?

Project exporteren Je kan je Eclipse-project exporteren om mee te nemen naar huis. Klik hier-voor *rechts* op je project in de `Package Explorer`, en selecteer `Export`. Vervolgens kies je `General` `Archive File` en klik je op `Next`. Kies waar je je .zip bestand wil opslaan met de `Browse`-knop en druk vervolgens op `Finish`.

Het aangemaakte bestand kan je vervolgens uploaden op je OneDrive of ownCloud o.i.d. Deze procedure kan je gebruiken om je project naar je medestudenten te verzenden, en je zal dit moeten doen bij het indienen van je examen.

Project importeren Om een .zip file die op bovenstaande wijze gemaakt werd te importeren, gebruik je `File` `Open Projects from File System` en klik op de `Archive`-knop. Selecteer vervolgens het .zip bestand en druk op `Finish`. Het project zou nu moeten zijn aangemaakt.

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 Git⁶. Wegens de complexiteit van Git komt dit niet expliciet aan bod in de cursus, maar raden we het wel aan voor de avontuurlijke studenten.

⁶<https://git-scm.com/>