

Java Opdrachten

Opdracht 1 - Basic Syntax

Voordat je Java kan gebruiken moet je de syntax kunnen begrijpen. Doe onderzoek naar de volgende termen en leg uit wat ze (voor Java) betekenen:

- Object
- Class
- Package
- Constructor
- Instance
- Method

Opdracht 2 - Variables

Ook *variables* werken in Java net iets anders dan je hiervoor misschien gewend bent. Java maakt onderscheid tussen 3 soorten variables:

- **Local** variables
- **Instance** variables
- **Class** variables (ook wel **Static** variables genoemd)

Wat voor soort variabele je gebruikt hangt onder andere af van de plek waar je hem declareert .

- a) Waar wordt een local variable gedeclareerd?

Class/Static variables en Instance variables worden op dezelfde plek gedeclareerd, en houden ook allebei de gegevens van een Object bij.

- b) Wat voor soort informatie sla je op in een instance variable?
c) Wat voor soort informatie sla je op in een class variable?

De waarde van een class variable kan je in principe ook opslaan in een instance variable.

- d) Wat is het voordeel van het gebruiken van een class variable?

Opdracht 3 - Access Control Modifiers

Java gebruikt zogenaamde **modifiers** om bepaalde eigenschappen van stukjes code te veranderen. Deze zijn te verdelen in **Access Control Modifiers** en **Non-Access Modifiers**

De Access Control Modifiers zijn heel belangrijk om te begrijpen - die geven namelijk aan welke onderdelen van je applicatie van het betreffende stukje code gebruik mogen maken.

Noem de vier verschillende Access Control Modifiers en geef bij elke een korte beschrijving.

Opdracht 4 - Non-Access Modifiers

Zoals bij de vorige opdracht vermeld stond, zijn er ook enkele **Non-Access Modifiers**. Het grootste deel hiervan hoef je nog niet te kennen, maar de volgende twee wel:

- Static
- Final

Static ben je eerder al tegengekomen. Kan je ook uitleggen waar je Final voor zou gebruiken?

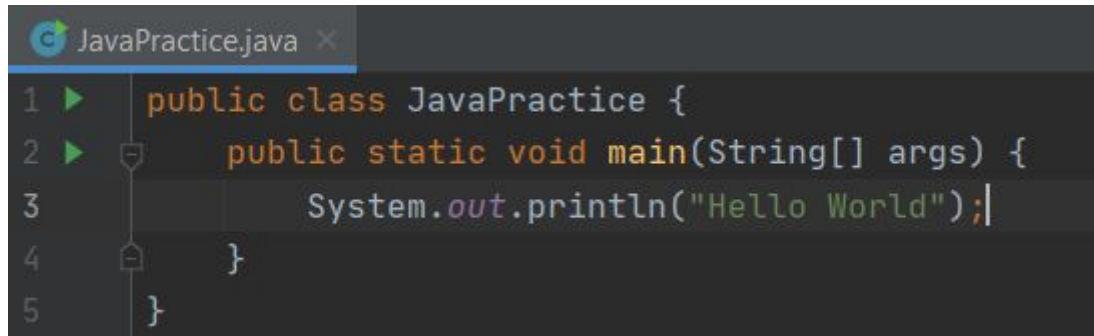
Opdracht 5 - Strictly Typed

Er wordt in IT onderscheid gemaakt tussen programmeertalen die **Strictly/Strongly Typed** zijn, en talen die **Loosely/Weakly Typed** zijn. Java is Strictly Typed - dit betekent dat je bij elke variabele in Java een datatype moet declareren. Bij Loosely Typed programmeertalen zoals JavaScript hoeft dit niet.

- a) Wat is een voordeel van Strict Typing?
- b) Wat is een voordeel van Weak Typing?

Opdracht 6 - main & Arrays

De main method van een Java class is de functie die wordt uitgevoerd als je een programma draait. Dit zal er altijd zo uit zien:



```
1 ▶ public class JavaPractice {
2 ▶     public static void main(String[] args) {
3       System.out.println("Hello World");
4     }
5 }
```

Public & *static* heb je hiervoor gehad; die geven aan dat je main method toegankelijk is voor iedereen, en dat deze voor elke Object van je class precies hetzelfde is.

void geeft aan dat er geen return value wordt verwacht. Bij de main method is dit verplicht!

Tenslotte nog `main(String[] args)` Dit slaat op de naam en parameters.

Met `String[] args` geef je dus aan dat de parameters van de *main* method in een array zitten. Dit array heet *args* (kort voor *arguments*) en bestaat enkel uit strings.

- a) Kan je een reden bedenken waarom deze parameter verplicht is?

`System.out.println()` wordt gebruikt om tekst weer te geven in de console. Voor debugging doeleinden kan je dit net zo gebruiken als de `console.log()` command van JavaScript.

Arrays declareer je zo:



```
1 ▶ public class JavaTest {
2 ▶     public static void main(String[] args) {
3       String[] cars = {"Kia", "BMW", "Renault", "Volkswagen"};
4       System.out.println(cars[0]); // Resultaat = Kia
5     }
6 }
```

- b) Kan je elementen van een Java Array veranderen?
c) Kan je elementen aan een Java Array toevoegen?

Net als bij JavaScript worden Arrays vaak gebruikt in combinatie met Loops. In de volgende opdracht gaan we hier verder op in.

Theorie - Arrays in Arrays & ArrayLists

Met Java kan je gebruik maken van arrays in arrays. Java noemt deze 2-Dimensional Arrays. Dit ziet er als volgt uit:

```
JavaTest.java x
1 public class JavaTest {
2     public static void main(String[] args) {
3         String[][] cars = {"Kia", "Honda"}, {"BMW", "Renault", "Volkswagen"};
4         System.out.println(cars[0][1]); // Resultaat = Honda
5     }
6 }
```

Bij de vorige opdracht heb je geleerd dat je de grootte van een Java Array niet aan kan passen. Als je dit wel wil doen, moet je gebruik maken van een **ArrayList**.

Belangrijk! Voordat je een ArrayList kan gebruiken, moet je dat eerst in je eigen code aangeven. Dit doe je door de ArrayList code te **importeren**.

Voorbeeld van een ArrayList:

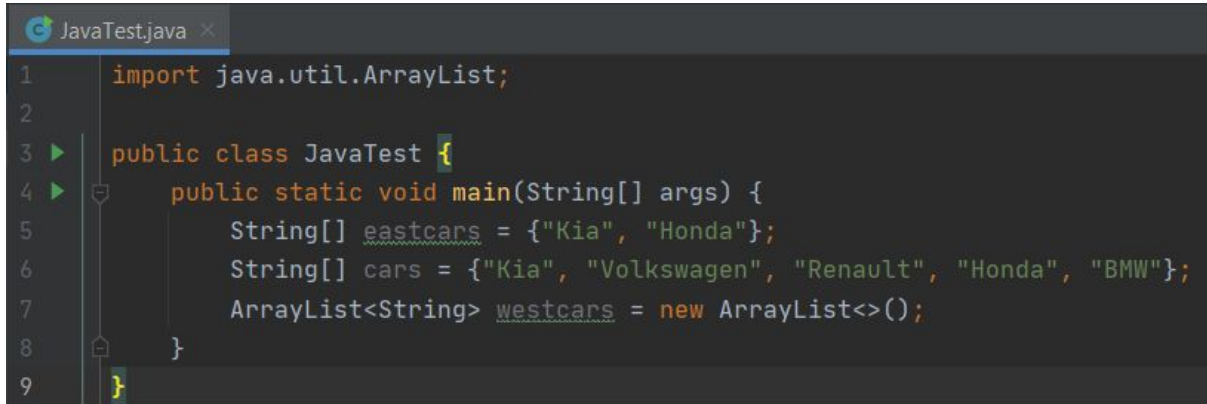
```
JavaTest.java x
1 import java.util.ArrayList;
2
3 public class JavaTest {
4     public static void main(String[] args) {
5         ArrayList<String> cars = new ArrayList<>();
6         cars.add("Kia");
7         cars.add("Honda");
8         cars.add("BMW");
9         cars.add("Renault");
10        cars.add("Volkswagen");
11        System.out.println(cars); // Resultaat = [Kia, Honda, BMW, Renault, Volkswagen]
12    }
13 }
```

Je importeert de code van een ArrayList dus uit de *java.util* package.

Opdracht 8 - Loops & Conditions

Loops werken in Java net zoals je dat in JavaScript gewend bent. Ook conditions zijn heel erg vergelijkbaar. Er zijn hier en daar enkele kleine verschillen in de syntax, maar in gebruik zijn ze vrijwel identiek.

Gegeven de volgende code:



```
1  import java.util.ArrayList;
2
3  public class JavaTest {
4      public static void main(String[] args) {
5          String[] eastcars = {"Kia", "Honda"};
6          String[] cars = {"Kia", "Volkswagen", "Renault", "Honda", "BMW"};
7          ArrayList<String> westcars = new ArrayList<>();
8      }
9  }
```

- Maak gebruik van een for loop om de eerste drie items uit het cars Array te printen.
- Maak gebruik van een for-each loop om alle items uit het cars Array te printen.

Zoals je ziet is de ArrayList westcars nog leeg - nu gaan we die vullen.

- Loop door het cars array en vergelijk deze met de items in het eastcars array. Als je item niet in het eastcars array voor komt, voeg je hem toe aan het westcars array.

Vervolgens willen we onze Arrays en ArrayList mooi alfabetisch sorteren. Net als ArrayLists is hier een class voor die je moet importeren uit de java.util package.

- Als je een Array wil sorteren, moet je de Arrays class importeren
 - Als je een ArrayList wil sorteren, moet je de Collections class importeren.
- Pas dit nu toe op de Arrays en ArrayLists die je hebt gemaakt