

Programming fundamentals

Java API



Klasgroep	1EO-ICT
Opleiding	Bachelor Elektronica-ICT
Theorie	DI: 8:45 - 9:45 WOE: 11:45 - 12:45
(Werk)Labo	DI: 9:45 - 12:45 + 13:30 - 14:30 WOE: 13:30 - 15:30 + 15:30 - 17:30
Docent	Katja Verbeeck
Contact	katja.verbeeck@odisee.be

Inhoud

- 1 De Java API
- 2 Klassen en objecten
- 3 De String klasse
- 4 De Scanner klasse
- 5 De klasse Math
- 6 De Wrapperklassen



De Java API

De Java API (application Programming Interface) is een verzameling van stukjes code die men **klassen** noemt en die je vrij kan gebruiken in je code.

Een klasse definieert een nieuw type. Naar analogie met de primitieve types wil dat zeggen dat er data en operaties op die data gedefinieerd worden. De operaties van een klasse noemt men **methoden**.

Voorbeeld:

- **primitief type int** : de data van dit type zijn de gehele getallen; de operaties van dit type zijn de rekenkundige bewerkingen.
- **het type String** uit de Java API : de data zijn de karakters die samen een stuk tekst voorstellen; en zoals je zo dadelijk zal zien, zijn er allerlei methoden (operaties dus) gedefinieerd in de klasse String om teksten te manipuleren.

Java programma

Een Java programma schrijven komt neer op een set van nieuwe klassen / types te ontwerpen die je vervolgens gebruikt om een probleem op te lossen. Maar je hoeft uiteraard niet alles steeds opnieuw te schrijven vandaar een uitgebreide collectie van klassen die telkens (her)bruikbaar zijn in je eigen code.

Klassen

Er zijn 2 soorten types : **primitieve types** en **object of referentie types**. Deze laatste kan je zelf maken via klassen.

Een **klasse** zelf is een abstracte beschrijving van een nieuw type, en drukt uit hoe een element van dat type (wat men een **object** noemt) gebouwd moet worden. Het legt tevens vast welke **data** het object bevat en welke **operaties** op die data toegelaten zijn.

De operaties worden beschreven aan de hand van **methoden**, de data aan de hand van variabelen. Beide (methoden en variabelen) worden **members** van de klasse genoemd.

Objecten van een klasse maken

Je maakt **een object** via het **new** commando. Een object krijgt via de new operatie de nodige geheugenplaatsen om de toestand van de data bij te houden. Een object is dus niet meer dan een pointer naar een locatie in het geheugen.

```
String muppet1 = new String("Fozzy");
```

Van 1 klasse kan je heel veel objecten maken en die hebben elk een unieke plaats in het geheugen met elk hun eigen data.

Voorbeeld : objecten uit de echte wereld in je programma

Stel je wil een programma schrijven voor garagisten. Je wil dat deze de auto's in zijn gamma kan voorstellen in dat programma.

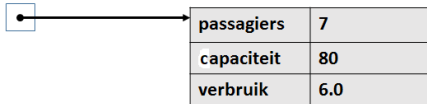
Je kan dan een **klasse Auto** aanmaken. De klasse legt de data (eigenschappen) vast : kleur, merk, type motor, ... en de methoden : bvb bereken het verbruik van de wagen.

Elke effectieve auto uit de garage kan je nu voorstellen als een element of object van de klasse Auto. Dus voor elke auto zal je via de new operator deze een uniek plekje in het geheugen geven zodat elke auto zijn eigen data kan bijhouden.

```
Auto miniBus = new Auto(7,80,6.0);  
Auto _2pk = new Auto(4,30,10.0);
```

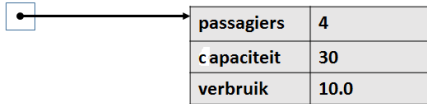
Elk object heeft zijn eigen data

miniBus



passagiers	7
capaciteit	80
verbruik	6.0

_2pk



passagiers	4
capaciteit	30
verbruik	10.0

```
Auto miniBus = new Auto(7,80,6.0);  
Auto _2pk = new Auto(4,30,10.0);
```


Uitzonderingen : klassen met statische velden

Er zijn echter ook klassen waarvan de members (zowel data als methoden) **static** zijn. Dit wil zeggen dat deze members behoren bij de klasse en niet uniek voor elk object zullen worden aangemaakt. Ze hoeven dus niet via objecten aangesproken te worden, dit kan rechtstreeks via de klassenaam. Bijvoorbeeld constante *PI* uit de *Math* klasse kan je rechtstreeks opvragen via *Math.PI*. Dit is een globale constante, niet elk object hoeft dit zelf bij te houden.

Zie verdere info onder de klasse **Math**.

De klasse String

In Java, Strings zijn objecten!

De manier om in Java objecten aan te maken is door gebruik te maken van de operator *new*. Strings kunnen dus als volgt aangemaakt worden :

```
String muppet1 = new String("Fozzy");
```

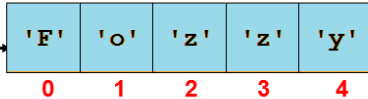
maar door gebruik te maken van string literals kan ook het volgende :

```
String muppet2 = "Gonzo";  
String muppet3 = "Kermit";
```

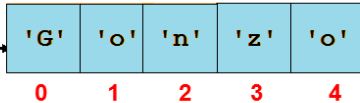
De manier waarop een String wordt aangemaakt heeft effect op waar in het geheugen deze terecht zal komen.

Een String is een rij van karakters

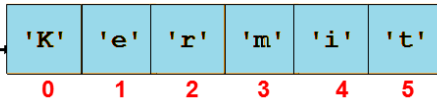
muppet1



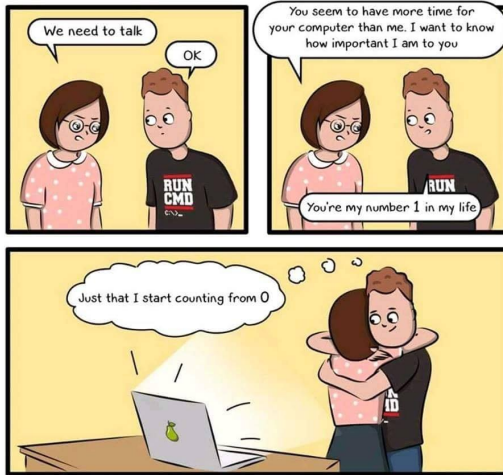
muppet2



muppet3



In java wordt er geteld vanaf 0 !



fb.com/programmingjokes

NERD4LIFE.studio

In java wordt er geteld vanaf 0 !

methode definitie	oproep
char charAt(int i)	<pre>char c ; int i c = muppet1.charAt(0); 'F' c = muppet2.charAt(2); 'n' c = muppet3.chatAt(5); 't' c = muppet2.charAt(5); —runtime error—</pre> String index out of bounds
int length()	<pre>i = muppet1.length(); 5 i = muppet2.length(); 5 i = muppet3.length(); 6</pre>

In java wordt er geteld vanaf 0 !

methode definitie	oproep
int indexOf(char c)	<pre>int i i = muppet1.indexOf('y'); 4 i = muppet1.indexOf('z'); 2 i = muppet2.indexOf('*'); -1</pre>
int indexOf(String s)	<pre>i = muppet2.indexOf("zo"); 3</pre>
int lastIndexOf(char c)	<pre>i = muppet1.lastIndexOf('z'); 3</pre>

In java wordt er geteld vanaf 0 !

methode definitie	oproep
int indexOf(char c)	<pre>int i i = muppet1.indexOf('y'); 4 i = muppet1.indexOf('z'); 2 i = muppet2.indexOf('*'); -1</pre>
int indexOf(String s)	<pre>i = muppet2.indexOf("zo"); 3</pre>
int lastIndexOf(char c)	<pre>i = muppet1.lastIndexOf('z'); 3</pre>

Method Overloading

De naam van een methode alleen definieert de methode niet. Een methode wordt gedefinieerd door zijn volledige methode definitie. **indexOf(char c)** en **indexOf(String s)** zijn dus wel degelijk verschillend!

Strings vergelijken

methode definitie	oproep
boolean equals(String s)	boolean b b = muppet1.equals(muppet3); false b = muppet1.equals(muppet1); true b = muppet1.equals("Fozzy"); true
== operator	b = muppet1 == muppet1; true b = muppet1 == "Fozzy"; false b = muppet2 == "Gonzo"; true

equals versus ==

Inhoud of object ref vergelijken?

De `==` operator vergelijkt niet de inhoud maar de object referentie (= het adres van dat object in het geheugen). String literals worden intern bijgehouden in een pool van constante string objecten. Wanneer "*Fozzy*" al in die pool zit, wordt die niet meer opnieuw aangemaakt. Wanneer je strings aanmaakt via *new* wordt wel een nieuwe plaats in het geheugen gezocht.

Strings kan je niet muteren

Dit wil zeggen dat je de rij van karakters intern niet kan wijzigen :

`muppet1.charAt(1)` ^{NOK} `= 'u'`;

Er is wel een **replace** methode voorzien, maar die maakt een nieuwe string aan.

methode definitie	oproep
<code>String replace(char c1, char c2)</code>	<code>String mup;</code> <code>mup = muppet1.replace('o','u') ;</code> <code>→ "Fuzzy"</code>
<code>String replace(String s1, String s2)</code>	<code>mup =</code> <code>muppet1.replace("Fozz","Pigg");</code> <code>→ "Piggy"</code>

Merk op

Niet meer gebruikte String objecten worden door de **Garbage Collector** verwijderd.

Een **methode-oproep** herken je aan het gebruik van de ronde haken (,). Tussen deze haken kan je waarden meegeven als input. Ook als er geen waarden worden meegegeven moeten deze ronde haken er staan!

```
String mup = muppet1.replace("Fozz","Pigg");  
int i = muppet1.length();
```

Andere nuttige methoden

methode definitie	oproep
	boolean b; String mup;
boolean startsWith(String s)	b = muppet3.startsWith(" Ker"); → true
String toUpperCase()	mup = muppet3.toUpperCase(); → "KERMIT"
String toLowerCase()	mup = muppet3.toLowerCase(); → "kermit"
String substring(int i)	mup = muppet1.substring(1); → "ozzy"
String substring(int i, int j)	mup = muppet2.substring(1,3); → "on"

volledige *String* API : [https:](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html)

[//docs.oracle.com/javase/8/docs/api/java/lang/String.html](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html)

De klasse Scanner

Scanning

De standaard manier om te lezen van de console is door gebruik te maken van de **Scanner** klasse. Met behulp van de Scanner klasse kan tekst input opgebroken worden in stukken volgens een *delimiter* zoals een spatie. De tekst input kan van verschillende bronnen komen zoals een gewone *String* maar ook *System.in*.

```
String input = "1 2 3 ";  
Scanner s = new Scanner(input);
```

```
Scanner s = new Scanner(System.in);
```

import

Merk op : de *Scanner* klasse wordt niet automatisch ingeladen zoals bvb. *System* of *Math* dus moet je dit zelf doen via het *import* keyword, helemaal bovenaan je code :

```
import java.util.Scanner;
```


Methoden van de Scanner klasse

Method	Description
<code>nextBoolean()</code>	Reads a <code>boolean</code> value from the user
<code>nextByte()</code>	Reads a <code>byte</code> value from the user
<code>nextDouble()</code>	Reads a <code>double</code> value from the user
<code>nextFloat()</code>	Reads a <code>float</code> value from the user
<code>nextInt()</code>	Reads a <code>int</code> value from the user
<code>nextLine()</code>	Reads a <code>String</code> value from the user
<code>nextLong()</code>	Reads a <code>long</code> value from the user
<code>nextShort()</code>	Reads a <code>short</code> value from the user

```
import java.util.Scanner;

public class ScanString {
    public static void main(String args[]){
        String input = "1 2 3 ";
        Scanner s = new Scanner(input);
        System.out.println(s.nextInt());
        System.out.println(s.nextInt());
        System.out.println(s.nextInt());

        input = "one two three ";
        s = new Scanner(input);
        System.out.println(s.next());
        System.out.println(s.next());
        System.out.println(s.next());
    }
}
```

```
input = "one_two_three";  
s = new  
    Scanner(input).useDelimiter("_");  
System.out.println(s.next());  
System.out.println(s.next());  
System.out.println(s.next());  
  
}  
  
}
```

Locatie gebonden instellingen

Let op bij het inlezen van een float getal zal je onder een Belgisch toetsenbord 3,14 moeten intypen ipv 3.14.

Je kan zelf echter via code je settings wijzigen :

```
s.useLocale(new Locale("ENGLISH",  
                        "US"));
```

De klasse Math

Math klasse

→ ↻ https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#field.summary	
Apps ☆ Bookmarks 📁 Multimedia 📁 TeTra 📁 student.ikdoeict.be ... 📄 1314 52 ptn LC4DB ... 📎 Beoordelingen (exce... 📁 slides	
Field Summary	
Fields	
Modifier and Type	Field and Description
static double	E The double value that is closer than any other to e , the base of the natural logarithms.
static double	PI The double value that is closer than any other to π , the ratio of the circumference of a circle to its diameter.
Method Summary	
All Methods	Static Methods Concrete Methods
Modifier and Type	Method and Description
static double	abs (double a) Returns the absolute value of a double value.
static float	abs (float a) Returns the absolute value of a float value.
static int	abs (int a) Returns the absolute value of an int value.
static long	abs (long a) Returns the absolute value of a long value.
static double	acos (double a) Returns the arc cosine of a value; the returned angle is in the range 0.0 through π .
static int	addExact (int x, int y) Returns the sum of its arguments, throwing an exception if the result overflows an int.
static long	addExact (long x, long y) Returns the sum of its arguments, throwing an exception if the result overflows a long.
static double	asin (double a) Returns the arc sine of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$.
static double	atan (double a)

keyword static

De Math klasse bevat constanten en methoden. Beide zijn **static** gedeclareerd, net zoals de *main* methode. Om statische data en methoden te gebruiken hoeft je niet eerst objecten aan te maken, je kan ze rechtstreeks acceren via de klasse zelf. Merk zelf het verschil op :

```
double result = Math.round(5.35);  
double min = Math.min(12.25,4.75);  
  
System.out.println("Geef een getal : ");  
Scanner scan = new Scanner(System.in);  
int num1 = scan.nextInt();
```

static

```
double result = Math.round(5.35);  
double min = Math.min(12.25,4.75);  
  
System.out.println("Geef een getal : ");  
Scanner scan = new Scanner(System.in);  
int num1 = scan.nextInt();
```

- *round()* en *min()* zijn statische methoden en kunnen rechte lijn naar de *Math* klasse gestuurd worden;
- *println()* wordt gestuurd naar het *out* object van het type *PrintStream* dat standaard binnen de klasse *System* aangemaakt wordt;
- *nextInt()* wordt gestuurd naar het *scan* object van het type *Scanner* dat je eerst zelf moet aanmaken

Constanten Pi en E

De Math klasse bevat 2 constanten die je rechtstreeks kan gebruiken.

```
static final double Math.PI  3.141592653589793  
static final double Math.E   2.718281828459045
```

Voorbeeld Math.PI

```
int  straal = 5;  
double cirkelOmtrek = 2 * Math.PI * straal;
```

Nuttige methoden in Math

methode definitie	oproep
double Math.abs (double d)	double res; int i; long l; res = Math.abs(-7.25); 7.25
long Math.round (double d)	l = Math.round(25.1); 25 l = Math.round(25.8); 26
int Math.round (float f)	i = Math.round(-7.25f); -7
double Math.ceil (double d)	res = Math.ceil(25.1); 26.0 res = Math.ceil(25.8); 26.0
double Math.floor (double d)	res = Math.floor(25.1); 25.0 res = Math.floor(25.8); 25.0

Nuttige methoden in Math

methode definitie	oproep
double Math.cos (double x) idem : sin , tan , acos , asin , atan	double res ; res = Math.cos(1); 0.54...
double Math.pow (double x, double y)	res = Math.pow(5, 2); 25.0 res = Math.pow(2, 8); 256.0
double Math.sqrt (double x)	res = Math.sqrt(49); 7.0
double Math.min (double x, double y)	res = Math.min(3, 5); 3.0
double Math.max (double x, double y)	res = Math.max(3, 5); 5.0

Math.random()

methode definitie	oproep
double Math.random()	double res; res = Math.random(); $res \in [0, 1[$

Genereer een random getal in $[0, 10[$

```
double dtoeval = Math.random() * 10;
```

De klasse Random

```
public class RandomIntGetallen {  
    public static void main(String[] args) {  
        //random reëel tussen 0.0 en 1.0(exclusief)  
        double randomWithMathRandom = Math.random();  
        System.out.println("Random with Math.random : " + randomWithMathRandom);  
  
        // genereer een geheel random getal tussen min en max  
        int max = 100;  
        int min = 30;  
        int randomWithMathRandomInARange = (int) ((Math.random() * (max - min)) + min);  
        System.out.println("Random with Math.random : " + randomWithMathRandomInARange);  
  
        // java.util.Random  
        Random random = new Random();  
        int randomWithNextInt = random.nextInt();  
        int randomWithNextIntInARange = random.nextInt( bound: max - min) + min;  
        double randomWithNextDouble = random.nextDouble();  
        System.out.println("Random with nextInt : " + randomWithNextInt);  
        System.out.println("Random with nextInt within a range : " + randomWithNextIntInARange);  
        System.out.println("Random with nextDouble : " + randomWithNextDouble);  
    }  
}
```

Afronden op 2 decimalen na de komma

```
public class Afronden {  
    public static void main(String[] args) {  
  
        double randomWithMathRandom = Math.random();  
        System.out.println("Random with Math.random : " + randomWithMathRandom);  
  
        Random random = new Random();  
        double randomWithNextDouble = random.nextDouble();  
        System.out.println("Random with nextDouble : " + randomWithNextDouble);  
        double randomWithNextDouble2 = random.nextDouble();  
        System.out.println("Random with nextDouble : " + randomWithNextDouble2);  
  
        System.out.printf("Value with 3 digits after decimal point %.3f \n", randomWithMathRandom);  
        double rounded = Math.round(randomWithNextDouble * 1000) / 1000.0;  
        double rounded2 = (int)(randomWithNextDouble2 * 1000) / 1000.0;  
        // merk op de eigenlijke waarde van de laatste 2 variabelen is nu wel gewijzigd  
        System.out.println("Value with 3 digits after decimal point : " + rounded);  
        System.out.println("Value with 3 digits after decimal point : " + rounded2);  
    }  
}
```

De klassen : *Double*, *Float*, *Long*, *Integer*, *Short*, *Byte* en *Character*.

De Wrapperklassen

Java definieert een aantal klassen die de primitieve types inpakken (*wrap around*) als een klasse. Zo heb je de klassen

Double, *Float*, *Long*, *Integer*, *Short*, *Byte* en *Character*.

Handig zijn de conversie methoden die ze voorzien

String → byte	byte b = Byte.parseByte(s);
String → short	short sh = Short.parseShort(s);
String → int	int i = Integer.parseInt(s);
String → long	long l = Long.parseLong(s);
String → float	float f = Float.parseFloat(s);
String → double	double d = Double.parseDouble(s);

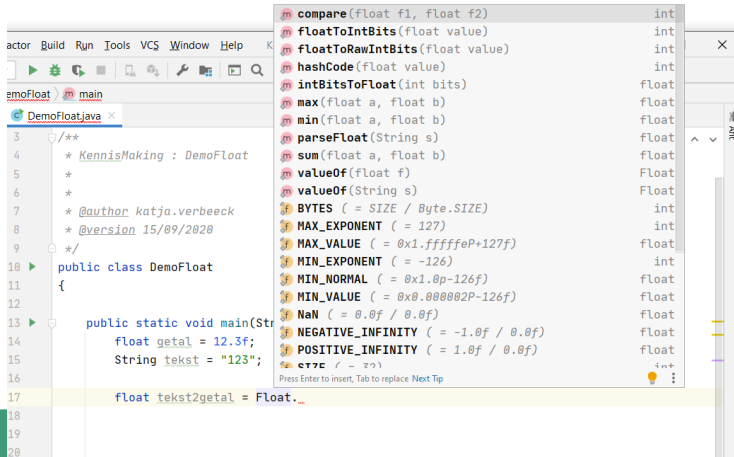
Merk op : deze zijn statisch !

Omzetten naar String

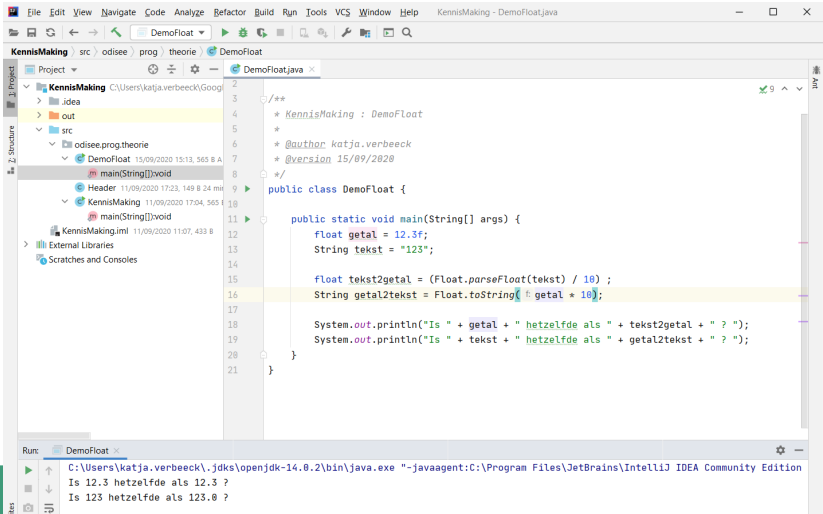
byte → String	String s = Byte.toString(b);
short → String	String s = Short.toString(sh);
int → String	String s = Integer.toString(i);
long → String	String s = Long.toString(l);
float → String	String s = Float.toString(f);
double → String	String s = Double.toString(d);

Oef : zet de float 12.3 om naar de String "123" en omgekeerd

De hele Java API kan je consulteren vanuit de IDE



Oef : zet de float 12.3 om naar de String "123" en omgekeerd



The screenshot shows the IntelliJ IDEA IDE with a project named 'Kennismaking'. The file explorer on the left shows the project structure, including a 'DemoFloat' class. The main editor window displays the code for 'DemoFloat.java'. The code defines a public class 'DemoFloat' with a static method 'main' that takes a String array 'args' as input. Inside the 'main' method, a float 'getal' is initialized to 12.3f, and a String 'tekst' is initialized to "123". The code then calculates 'float tekst2getal = (Float.parseFloat(tekst) / 10);' and 'String getal2tekst = Float.toString((int) getal * 10);'. Finally, it prints two lines: 'System.out.println("Is " + getal + " hetzelfde als " + tekst2getal + " ? ");' and 'System.out.println("Is " + tekst + " hetzelfde als " + getal2tekst + " ? ");'. The Run window at the bottom shows the output: 'Is 12.3 hetzelfde als 12.3 ?' and 'Is 123 hetzelfde als 123.0 ?'.

```
1  /**
2  * Kennismaking : DemoFloat
3  *
4  * @author katja.verbeeck
5  * @version 15/09/2020
6  */
7  public class DemoFloat {
8
9      public static void main(String[] args) {
10
11          float getal = 12.3f;
12          String tekst = "123";
13
14          float tekst2getal = (Float.parseFloat(tekst) / 10);
15          String getal2tekst = Float.toString((int) getal * 10);
16
17          System.out.println("Is " + getal + " hetzelfde als " + tekst2getal + " ? ");
18          System.out.println("Is " + tekst + " hetzelfde als " + getal2tekst + " ? ");
19      }
20  }
```

Run: DemoFloat

C:\Users\katja.verbeeck\jdk\openjdk-14.0.2\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition

Is 12.3 hetzelfde als 12.3 ?
Is 123 hetzelfde als 123.0 ?