

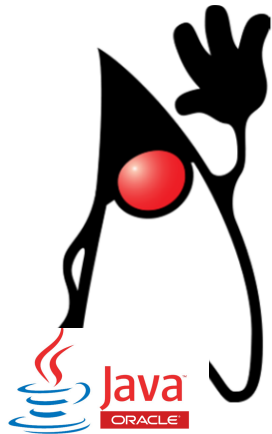
Java Fundamentals : Java API



Klasgroep	1EO-ICT
Opleiding	Bachelor Elektronica-ICT
Lokaal	groot auditorium
Tijdstip	maandag lestijd 3
Docent	Katja Verbeeck
Contact	katja.verbeeck@odisee.be
Handboek	hfst 2

Inhoud

- 1 Introductie
- 2 De klasse Math
 - keyword final
 - keyword static
 - nuttige methoden in Math
- 3 De klasse String
- 4 De Wrapperklassen
- 5 Netbeans : demo



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 LC#DB ... 📎 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)

Constanten Pi en E

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

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

Voorbeeld Math.PI

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

Zelf constanten definiëren

Via het keyword **final** kan je in java een variabele constant maken, d.i. de waarde ervan kan niet veranderd worden tijdens het programma. Je kan een constante alleen bij initialisatie een waarde geven.

```
final int MAX;  
MAX = 100; //OK  
MAX = 105; //compilatiefout!  
  
final double G = 9.81; //OK  
G += 0.01; //compilatiefout!
```

Volgens de stijlregels schrijf je constanten steeds volledig in hoofdletters !

static

De Math klasse bevat naast data ook allerlei handige methoden. Beide zijn allen **static** gedeclareerd, net zoals de *main* methode. Om statische data en methoden te gebruiken hoef 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 reeks 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

Een methode wordt steeds aangeduid met haakjes achteraan

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

Bereken de valtijd

$$t = \sqrt{\frac{2 * h}{g}}$$

Bereken de valtijd

$$t = \sqrt{\frac{2 * h}{g}}$$

```
final double G = 9.81;  
double hoogte = 10.0;  
  
double tijd = Math.sqrt(2 * hoogte / G);
```

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;
```

Math.random()

Genereer een random geheel getal in [0 10 [

```
int itoeval = (int)(Math.random() *  
    10);  
int itoeval2 =  
    (int)(Math.round(Math.random() *  
    9));
```

Merk op : getal 0 en 9 krijgen minder kans om gegenereerd te worden via de *round()* methode.

Math.random()

Genereer een random geheel getal in [1 10]

```
itoeval = (int)(Math.random() * 10 +  
    1);  
itoeval2 =  
    (int)(Math.round(Math.random() * 9  
    + 1));
```

Merk op : getal 1 en 10 krijgen minder kans om gegenereerd te worden via de *round()* methode.

Math.random()

Let op

```
double random = Math.random() * 100 +  
    100;  
System.out.println("uitvoer:" +  
    random);  
System.out.println("uitvoer: " +  
    Math.random() * 100 + 100);  
System.out.println("uitvoer: " +  
    (Math.random() * 100 + 100));
```

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 :

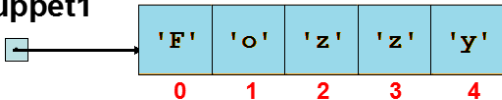
```
String muppet1 = new String("Fozzy");
```

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

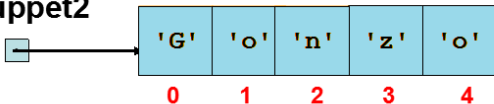
```
String muppet2 = "Gonzo";  
String muppet3 = "Kermit";
```


In het geheugen is een String een rij van karakters

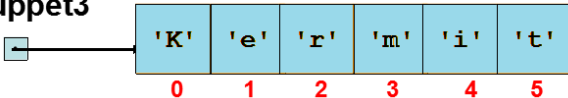
muppet1



muppet2



muppet3



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—— String index out of bounds</pre>
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)	int i
	i = muppet1.indexOf('y'); 4
	i = muppet1.indexOf('z'); 2
	i = muppet2.indexOf('*'); -1
int indexOf(String s)	i = muppet2.indexOf("zo"); 3
int lastIndexOf(char c)	i = muppet1.lastIndexOf('z'); 3

In java wordt er geteld vanaf 0 !

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

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!

```
import java.util.Scanner;

public class ScanSter {
    public static void main(String args[]){

        Scanner scan = new Scanner(System.in);
        System.out.println("Geef een hele zin
            in : ");
        String input = scan.nextLine();
        System.out.println(input.indexOf('*')
            != -1 ? "Gevonden" : "Jammer");
        scan.close();
    }
}
```

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

```
public class TestEquals {  
    public static void main(String args[]){  
        String muppet1 = new String("Fozzy");  
        String muppet2 = "Gonzo";  
        String muppet3 = "Kermit";  
  
        System.out.println(muppet1.equals(muppet3));  
        System.out.println(muppet1.equals(muppet1));  
        System.out.println(muppet1.equals("Fozzy"));  
        System.out.println(muppet1 == muppet3);  
        System.out.println(muppet1 == muppet1);  
        System.out.println(muppet1 == "Fozzy");  
        System.out.println(muppet2 == "Gonzo");  
    }  
}
```

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 aan elkaar plakken of concateneren

"dit" + "dat"

"abc" + 5

"Dat is dan " + 10 + " euro a.u.b."

5 + "abc"

"a" + 1 + 2

1 + 2 + "abc"

"ditdat"

"abc5"

"Dat is dan 10 euro a.u.b."

"5abc"

"a12"

"3abc"

Strings kan je niet muteren

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

`muppet1.charAt(1)` ^{NOK} \neq `'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>

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

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 Wrapperklassen

Java definieert een aantal klassen die de primitieve types inpakken (*wrap around*) als een object. 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);

Omzetten naar String

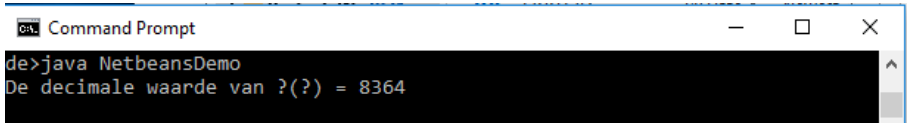
byte → String	String s = Byte.toString(b);
short → String	String s = Short.toString(sh);
int → String	String s = Integer.toString(i);
long → String	String s = Long.toString(l);
float → String	String s = Float.toString(f);
double → String	String s = Double.toString(d);

Oef : eurosymbool

De output op het scherm moet de volgende zijn : De decimale waarde van karakter '\u20AC' (€) = 8364

```
public static void main(String[] args) {  
    // oef Eurosymbol  
    char euro = '\u20AC';  
    System.out.println("De decimale waarde  
        van " + euro + "(\u20AC)" + " = " +  
        (int)euro);  
}
```

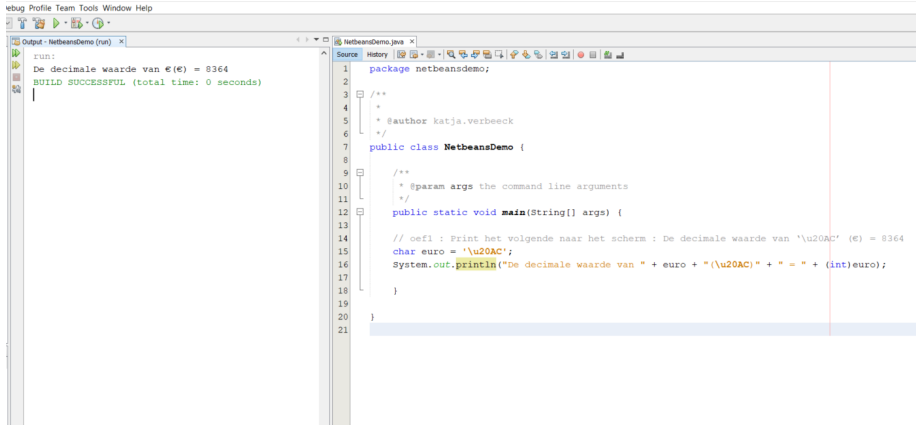
output in command prompt



A screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt" with standard minimize, maximize, and close buttons. The command prompt shows the command `de>java NetbeansDemo` being executed. The output is `De decimale waarde van ?(?) = 8364`. The text is displayed in a monospaced font on a black background.

```
de>java NetbeansDemo
De decimale waarde van ?(?) = 8364
```

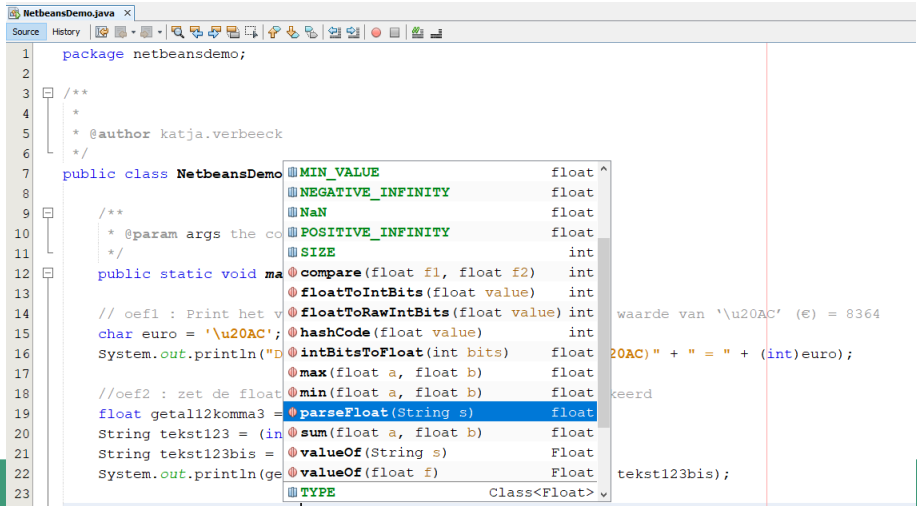

output in Netbeans



The screenshot displays the NetBeans IDE interface. On the left, the 'Output' window shows the execution results of the 'NetbeansDemo' program. The output text is: 'De decimale waarde van €(€) = 8364' followed by 'BUILD SUCCESSFUL (total time: 0 seconds)'. On the right, the 'Source' window shows the Java code for 'NetbeansDemo.java'. The code is a simple class with a main method that prints the decimal value of the Euro symbol (€) using a character escape sequence '\u20AC'.

```
1 package netbeansdemo;
2
3 /**
4  *
5  * @author katja.verbeeck
6  */
7 public class NetbeansDemo {
8
9     /**
10     * @param args the command line arguments
11     */
12     public static void main(String[] args) {
13
14         // oef1 : Print het volgende naar het scherm : De decimale waarde van '\u20AC' (€) = 8364
15         char euro = '\u20AC';
16         System.out.println("De decimale waarde van " + euro + "(\u20AC)" + " = " + (int)euro);
17     }
18 }
19
20
21
```

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



```
1 package netbeansdemo;
2
3 /**
4  *
5  * @author katja.verbeeck
6  */
7 public class NetbeansDemo {
8
9     /**
10      * @param args the command line arguments
11      */
12     public static void main(String[] args) {
13
14         // oef1 : Print het value van de euro
15         char euro = '\u20AC';
16         System.out.println("Doe euro is " + (int)euro);
17
18         //oef2 : zet de float 12.3 om naar de String "123" en omgekeerd
19         float getal12komma3 = 12.3f;
20         String tekst123 = (int)getal12komma3 + "";
21         String tekst123bis = Float.toString(getal12komma3);
22         System.out.println(getal12komma3 + " = " + tekst123 + " = " + tekst123bis);
23     }
24 }
```

MIN_VALUE float
NEGATIVE_INFINITY float
NaN float
POSITIVE_INFINITY float
SIZE int
compare(float f1, float f2) int
floatToIntBits(float value) int
floatToRawIntBits(float value) int
hashCode(float value) int
intBitsToFloat(int bits) float
max(float a, float b) float
min(float a, float b) float
parseFloat(String s) float
sum(float a, float b) float
valueOf(String s) Float
valueOf(float f) Float
TYPE Class<Float>

waarde van '\u20AC' (€) = 8364
20AC)" + " = " + (int)euro);
keerd
tekst123bis);

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

```
1 package netbeansdemo;
2
3 /**
4  *
5  * @author katja.verbeeck
6  */
7 public class NetbeansDemo {
8
9     /**
10      * @param args the command line arguments
11      */
12     public static void main(String[] args) {
13
14         // oef1 : Print het volgende naar het scherm : De decimale waarde van '\u20AC' (€) = 8364
15         char euro = '\u20AC';
16         System.out.println("De decimale waarde van " + euro + " (\u20AC)" + " = " + (int)euro);
17
18         //oef2 : zet de float 12.3 om naar de String "123" en omgekeerd
19         float getal12komma3 = 12.3f;
20         String tekst123 = (int)(getal12komma3 * 10) + "";
21         String tekst123bis = Float.toString(getal12komma3 * 10);
22         System.out.println(getal12komma3 + " " + tekst123 + " " + tekst123bis);
23
24         float getal123 = Float.parseFloat(tekst123);
25         System.out.println(getal12komma3 == getal123/10 ? "Gelukt!" : "Jammer");
26
27     }
28
29 }
```