

# Programming fundamentals

## Primitive types



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	<a href="mailto:katja.verbeeck@odisee.be">katja.verbeeck@odisee.be</a>

# Inhoud

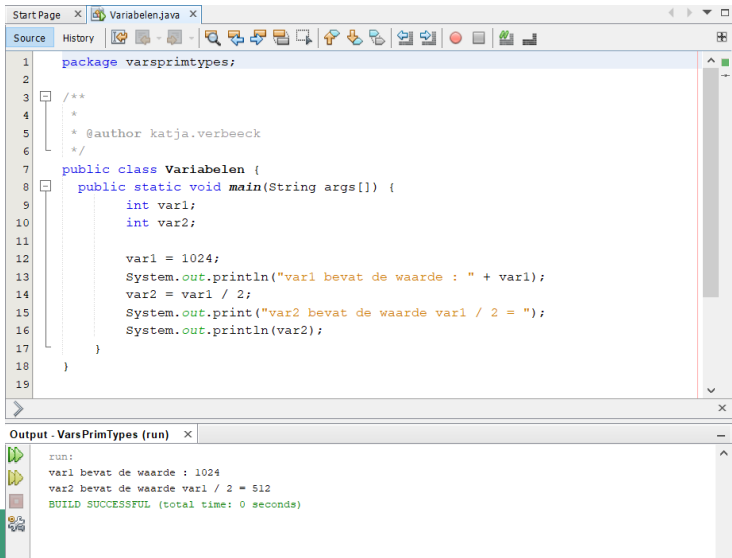
- 1 Variabelen in Java
  - Constanten in Java
- 2 Types in Java
- 3 Primitieve types
  - integer types
  - floating point types
  - karakters
  - booleans
- 4 Literals



# Variabelen en hun type

```
class Variabelen {  
    public static void main(String args[]){  
        int var1;  
        int var2;  
  
        var1 = 1024;  
        System.out.println("var1 bevat de  
            waarde : " + var1);  
        var2 = var1 / 2;  
        System.out.print("var2 bevat de waarde  
            var1 / 2 = ");  
        System.out.println(var2);  
    }  
}
```

# Variabelen en hun type



The screenshot shows an IDE window titled 'Variabelen.java'. The code defines a package 'varsprimtypes' and a class 'Variabelen' with a 'main' method. Inside 'main', two integer variables are declared: 'var1' and 'var2'. 'var1' is assigned the value 1024, and 'var2' is assigned the value of 'var1' divided by 2. The program prints the values of both variables. Below the code editor, the 'Output' window shows the execution results, confirming the values and indicating a successful build.

```
1 package varsprimtypes;
2
3 /**
4  *
5  * @author katja.verbeeck
6  */
7 public class Variabelen {
8     public static void main(String args[]) {
9         int var1;
10        int var2;
11
12        var1 = 1024;
13        System.out.println("var1 bevat de waarde : " + var1);
14        var2 = var1 / 2;
15        System.out.print("var2 bevat de waarde var1 / 2 = ");
16        System.out.println(var2);
17    }
18 }
19
```

Output - VarsPrimTypes (run)

```
run:
var1 bevat de waarde : 1024
var2 bevat de waarde var1 / 2 = 512
BUILD SUCCESSFUL (total time: 0 seconds)
```

# Declaratie van een variabele

```
int var1; // declaratie van een  
          variabele van type int  
int var2; // declaratie van een tweede  
          variabele van type int
```

Een variabele is een naam voor een locatie in het geheugen. In Java moet elke variabele gedeclareerd worden vooraleer deze kan gebruikt worden. Een declaratie specificeert van welk type de data is die op die bepaalde geheugenlocatie kan bijgehouden worden. In dit voorbeeld wordt er plaats in het geheugen gezocht om 2 gehele getallen bij te houden.

**Java is een sterk getypeerde taal**

# Declaratie van een variabele

```
int var3, var4; // de declaratie kan  
ook op 1 lijn
```

# Assignment: toekenning van een waarde van een variabele

```
var1 = 1024; // assignement of  
           toekennen van een waarde  
System.out.println("var1 bevat de  
           waarde : " + var1);  
var2 = var1 / 2; // het resultaat van  
           de berekening wordt toegekend aan  
           var2
```

# Variabelen en hun namen

- een *identifier* is de keuze van een naam die je geeft aan een variabele (of zie later een methode)
- een identifier kan starten met een letter, underscore of dollar teken (nooit starten met een cijfer)
- stijlspraak is om alleen kleine letters te gebruiken
- java is case-sensitive : *eenVar* is niet hetzelfde dan *eenvar*. Volgens de stijlregels moet het echter *eenVar* zijn.
- gebruik geen keywords, of reeds bestaande namen uit de library bvb. *println*



# Constanten in Java

Een constante is een bijzonder soort variabele waarvan de waarde niet meer kan veranderen nadat deze eenmaal een waarde is toegekend. Je kan deze dus slechts eenmaal in een toekenning of assignement gebruiken. In Java bekom je dit door de declaratie te laten vooraf gaan door het keyword **final**.

Volgens de stijlregels schrijf je constanten steeds volledig in hoofdletters !

Constanten worden op die manier onmiddellijk herkend door anderen die je code lezen.

Later zal je zien dat je de keyword **static** en **final** kan combineren om globale constanten te definiëren.

# 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!
```

# Primitieve versus Object types

Java kent 2 soorten types : **primitieve types** en **object types of referentietypes**.

Primitieve types zijn toegevoegd aan de taal omwille van efficiëntie redenen. Zij zijn niet afgeleid van een klasse maar gewoon opgebouwd aan de hand van binaire waarden.

Object types worden gemaakt aan de hand van Java klassen. Dit omvat

- alle Java bibliotheek klassen  
(<https://docs.oracle.com/javase/8/docs/api/>)  
System, Math, String, ...
- waaronder ook de Java Wrapper klasse Byte, Short, Integer, Long, Float, Double, Boolean, Character, BigDecimal, BigInteger, ...
- elke klasse die je zelf definieert (zie verder ... )

# Er zijn 8 primitieve types

Type	waarden van dit type
<b>boolean</b>	binaire waarden : true of false
<b>byte</b>	een geheel getal van max. 8 bits
<b>short</b>	een geheel getal van max. 16 bits
<b>int</b>	een geheel getal van max. 32 bits
<b>long</b>	een geheel getal van max 64 bits
<b>char</b>	een karakter
<b>float</b>	een reëel getal met enkele precisie (32 bit)
<b>double</b>	een reëel getal met dubbele precisie (64 bit)

Gebruik een gepast primitief type en wees bewust van de bezetting in het geheugen !

# Enkele Voorbeelden

```
byte b;  
int som = 5 + 7;  
short s = 1027;  
long l;  
float broodPrijs = 2.1f;  
double prijs = 20.50;  
boolean ingelogd = true;  
ingelogd = false;  
char geslacht = 'm';
```

# De gehele types

type	#bits	bereik
<b>byte</b>	8 bits	$-128 \text{ tot } 127$ of $-2^7 \text{ tot } (2^7 - 1)$
<b>short</b>	16 bits	$-32.768 \text{ tot } 32.767$ of $-2^{15} \text{ tot } (2^{15} - 1)$
<b>int</b>	32 bits	$-2^{31} \text{ tot } (2^{31} - 1)$
<b>long</b>	64 bits	$-2^{63} \text{ tot } (2^{63} - 1)$

vanaf java 8 : unsigned integer operaties via de wrapper klassen  
het meest gebruikte of default geheel type is *int*

**byte**  $\neq$  **Byte** , **short**  $\neq$  **Short** , **int**  $\neq$  **Integer** , **long**  $\neq$  **Long**

# 2-complements representatie

teken-bit									
0	1	1	1	1	1	1	1		= 127
0	0	0	0	0	0	1	0		= 2
0	0	0	0	0	0	0	1		= 1
0	0	0	0	0	0	0	0		= 0
1	1	1	1	1	1	1	1		= -1
1	1	1	1	1	1	1	0		= -2
1	0	0	0	0	0	0	0	1	= -127
1	0	0	0	0	0	0	0	0	= -128

# De reële types

type	#bits	bereik
<b>float</b>	32 bit	tekenbit (#1), mantisse (#23) en exponent (#8)
<b>double</b>	64 bit	tekenbit (#1), mantisse (#52) en exponent (#11)

voorbeeld :  $-8.2$  kan voorgesteld worden als :  $-0.82 * 10^1$  waarbij  
*mantisse* = 82 en *exponent* = 1

double is het meest gebruikte of default reëel type

**float**  $\neq$  **Float** , **double**  $\neq$  **Double**



# karakters

type	#bits	bereik
char	16 bit	0 tot 65.536

Java maakt gebruik van de *Unicode character set*.

De standaard 8-bit ASCII karakters ( 0 tot 128) zijn hier een deel van.

Vermits intern een karakter ook gerepresenteerd wordt via een enkel binair getal, wordt char ook als een geheel type beschouwd.

decimaal	karakter	decimaal	karakter	decimaal	karakter
32	spatie	64	@	96	,
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	DEL

Source: <http://www.asciitable.com/>

Figure: American Standard Code for Information Interchange

```
public class CharsASCII {  
    public static void main (String[] args){  
        char c = 'z';  
        char d = 90;  
        char dollar = 36;  
  
        System.out.println("char c bevat : " +  
            c);  
        System.out.println("char d bevat : " +  
            d);  
        System.out.println("char dollar bevat  
            : " + dollar);  
    }  
}
```

```
11  L  */
12  public class CharsASCII {
13      public static void main (String[] args){
14          char c = 'z';
15          char d = 90;
16          char dollar = 36;
17
18          System.out.println("char c bevat : " + c);
19          System.out.println("char d bevat : " + d);
20          System.out.println("char dollar bevat : " + dollar);
21      }
22  }
23
```

Output - VarsPrimTypes (run) X

run:  
char c bevat : z  
char d bevat : Z  
char dollar bevat : \$  
BUILD SUCCESSFUL (total time: 1 second)

# Unicode UTF16

## TABLE OF SPECIAL CHARACTERS

The decimal digits xxx used to create special characters, as well as accented characters in West European languages.  
For the following characters, the digits for decimal Unicode and ISO 8859-1 are identical.

Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code
	160	¡	161	¢	162	£	163	¤	164	¥	165	¦	166	§	167
¨	168	©	169	ª	170	«	171	¬	172		173	®	174	¯	175
°	176	±	177	²	178	³	179	´	180	µ	181	¶	182	·	183
,	184	¸	185	º	186	»	187	¼	188	½	189	¾	190	¿	191
À	192	Á	193	Â	194	Ã	195	Ä	196	Å	197	Æ	198	Ç	199
È	200	É	201	Ê	202	Ë	203	Ì	204	Í	205	Î	206	Ï	207
Ð	208	Ñ	209	Ò	210	Ó	211	Ô	212	Õ	213	Ö	214	×	215
Ø	216	Ù	217	Ú	218	Û	219	Ü	220	Ý	221	Þ	222	ß	223
à	224	á	225	â	226	ã	227	ä	228	å	229	æ	230	ç	231
è	232	é	233	ê	234	ë	235	ì	236	í	237	î	238	ï	239
ð	240	ñ	241	ò	242	ó	243	ô	244	õ	245	ö	246	÷	247
ø	248	ù	249	ú	250	û	251	ü	252	ý	253	þ	254	ÿ	255

Figure: Unicode is een wereldwijde standaard en beperkt zich niet tot de symbolen uit de Westerse talen.

# Unicode UTF16

## Unicode Character 'DEGREE SIGN' (U+00B0)



[Browser Test Page](#)  
[Outline \(as SVG file\)](#)  
[Fonts that support U+00B0](#)

Figure: Verschillende encodings voor het graden symbool

Encodings	
HTML Entity (decimal)	&#176;
HTML Entity (hex)	&#xb0;
HTML Entity (named)	&deg;
How to type in Microsoft Windows	Alt +00B0 Alt 0176 Alt 248
UTF-8 (hex)	0xC2 0xB0 (c2b0)
UTF-8 (binary)	11000010:10110000
UTF-16 (hex)	0x00B0 (00b0)
UTF-16 (decimal)	176
UTF-32 (hex)	0x000000B0 (00b0)
UTF-32 (decimal)	176
C/C++/Java source code	"\u00B0"
Python source code	u"\u00B0"
<a href="#">More...</a>	

```

public class CharsUnicode {
    public static void main(String args[]) {
        System.out.println("Temperatuur
            vandaag: 25" + (char)248 + "C" );
        System.out.println();
        System.out.println("Temperatuur
            vandaag: 25" + (char)176 + "C" );
        System.out.println();
        System.out.println("Temperatuur
            vandaag: 25" + '\u00b0' + "C" );
        System.out.println();
        System.out.println("S\u00EDD
            Se\u00F1or");
        System.out.println();
    }
}

```

# booleans

type	#bits	bereik
<b>boolean</b>	8 bit	slechts 2 waarden : <b>true</b> en <b>false</b>

```
public class PrimBoolean {  
    public static void main(String args[]) {  
        boolean b1, b2;  
        b1 = false;  
        b2 = true;  
  
        System.out.println("De waarde van b1 : " + b1);  
        System.out.println("De waarde van b2 : " + b2);  
    }  
}
```



# default waarden

<b>type</b>	<b>default veldwaarde</b>
<b>boolean</b>	false
<b>byte</b>	0
<b>short</b>	0
<b>int</b>	0
<b>long</b>	0L
<b>char</b>	'\u0000'
<b>float</b>	0.0f
<b>double</b>	0.0d

# literals

literals zijn de vaste waarden of *constanten* die toegekend kunnen worden aan een type

# literals

literals zijn de vaste waarden of *constanten* die toegekend kunnen worden aan een type

character constants worden weergegeven tussen enkele quotes : 'a' of '%'  
voor sommige karakters levert dit echter een probleem, zie verder *escape sequences*

# literals

literals zijn de vaste waarden of *constanten* die toegekend kunnen worden aan een type

**character constants** worden weergegeven tussen enkele quotes : `'a'` of `'%'`  
voor sommige karakters levert dit echter een probleem, zie verder *escape sequences*

- integer literals**
- ① alle gehele getallen : `12` (default **int**), `12L` of `12L` (**long**)
  - ② hexadecimale getallen met basis 16 :  $0 \rightarrow 9 + A \rightarrow F$   
Deze moeten voorafgegaan worden door `0x` of `0X` :  
`int hexVal = 0x1a;` (nummer 26)
  - ③ binaire getallen met basis 2. Deze moeten voorafgegaan worden door `0b` of `0B` :  
`int binVal = 0b11010;` (nummer 26)

float literals alle reële getallen :

- 11.23 (default **double**)
- 11.23*f* of 11.23*F* (**float**)
- 1.234e2 (wetenschappelijke notatie =  $1.234 * 10^2 = 123.4$ )

float literals alle reële getallen :

- 11.23 (default **double**)
- 11.23f of 11.23F (**float**)
- 1.234e2 (wetenschappelijke notatie =  $1.234 * 10^2 = 123.4$ )

String literals alhoewel er **geen primitief type is voorzien in JAVA voor strings**, zijn er wel string literals, namelijk alle tekst tussen dubbele quotes : " dit is een string "

String is geen primitief type maar een object type. Het is een klasse in de java API. Om een string te maken moet je dus een object van de klasse String maken m.b.v. het keyword **new**. Doordat er wel primitieve string literals voorzien zijn kan het echter ook rechtstreeks **String hello = "Hello World !"**

# gebruik van underscore

```
long creditCardNumber = 1234_5678_9012_3456L;  
long socialSecurityNumber = 999_99_9999L;  
float pi = 3.14_15F;  
long hexBytes = 0xFF_EC_DE_5E;  
long hexWords = 0xCAFE_BABE;  
long maxLong = 0x7fff_ffff_ffff_ffffL;  
byte nybbles = 0b0010_0101;  
long bytes =  
    0b11010010_01101001_10010100_10010010;
```

# Escape Sequences

Escape Sequentie	beschrijving
<code>\ ' </code>	enkele quote
<code>\ " </code>	dubbele quotes
<code>\\ </code>	backslash
<code>\r </code>	carriage return : begin van de lijn
<code>\n </code>	nieuwe lijn
<code>\f </code>	form feed : volgende pagina
<code>\t </code>	horizontale tab
<code>\b </code>	backspace
<code>\uxxxx </code>	hexadecimale waarde met xxxx de hex constante



```
public class EscapeSeq {  
    public static void main(String args[]) {  
        System.out.println("abc\bx");  
        System.out.println("\"hallo\"");  
        System.out.println("1\t2\t3");  
        System.out.println("1\n2\n3");  
        System.out.println("auto\'s");  
        System.out.println("\\*wrong comment  
        *\\");  
        System.out.println("/*good comment  
        */");  
    }  
}
```