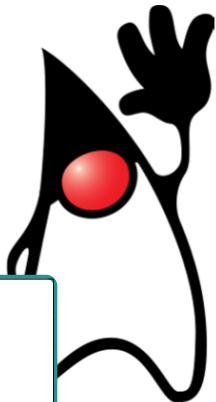


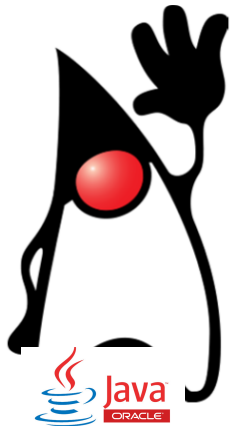
Java Fundamentals : Primitieve types



Klasgroep	1EO-ICT
Opleiding	Bachelor Elektronica-ICT
Lokaal	groot auditorium
Tijdstip	maandag lestijd 3
Docent	Katja Verbeeck
Contact	katja.verbeeck@odisee.be
Handboek	ch. 2 pagina 31 - 42 + 368 - 369

Inhoud

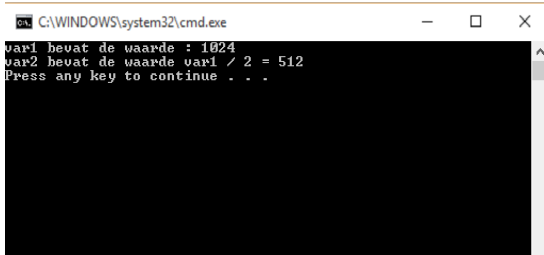
- 1 Introductie
- 2 Variabelen in Java
- 3 Types in Java
- 4 Primitieve types
 - integer types
 - floating point types
 - karakters
 - booleans
- 5 Literals
- 6 Basic I/O : lezen van en naar de console



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



```
C:\WINDOWS\system32\cmd.exe
var1 bevat de waarde : 1024
var2 bevat de waarde var1 / 2 = 512
Press any key to continue . . .
```

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 vooral 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
```

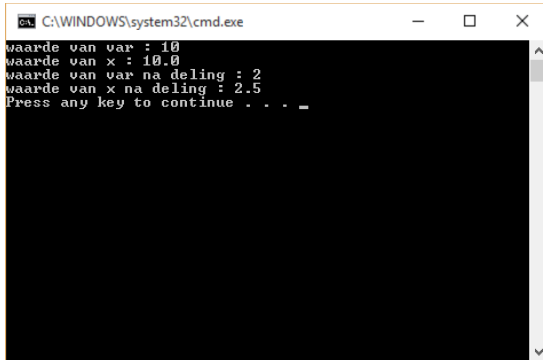
Assignment: toekenning van een waarde van een variabele

```
var1 = 1024; // assignment 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
```

int versus double

```
class IntVsDouble {  
    public static void main (String args[]){  
        int var; // gehele variabele  
        double x; // reele variabele  
  
        var = 10;  
        x = 10.0;  
  
        System.out.println("waarde van var : " + var);  
        System.out.println("waarde van x : " + x);  
  
        var = var / 4;  
        x = x / 4;  
  
        System.out.println("waarde van var na deling : " + var);  
        System.out.println("waarde van x na deling : " + x);  
    }  
}
```


int versus double



```
C:\WINDOWS\system32\cmd.exe
waarde van var : 10
waarde van x : 10.0
waarde van var na deling : 2
waarde van x na deling : 2.5
Press any key to continue . . . _
```

int versus double

Gehele Deling vs Reële deling

Een gehele deling geeft het gehele resultaat na deling terug zonder de rest!

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)
- stijlfpraak 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*

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 zijn 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
boolean	binaire waarden : true of false
byte	een geheel getal van max. 8 bits
short	een geheel getal van max. 16 bits
int	een geheel getal van max. 32 bits
long	een geheel getal van max 64 bits
char	een karakter
float	een reëel getal met enkele precisie (32 bit)
double	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 = 2.1;  
double prijs, minPrijs = 20;  
boolean ingelogd = true;  
ingelogd = false;  
char geslacht = 'm';
```

De gehele types

type	#bits	bereik
byte	8 bits	$-128 \text{ tot } 127$ of $-2^7 \text{ tot } (2^7 - 1)$
short	16 bits	$-32.768 \text{ tot } 32.767$ of $-2^{15} \text{ tot } (2^{15} - 1)$
int	32 bits	$-2^{31} \text{ tot } (2^{31} - 1)$
long	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
float	32 bit	tekenbit (#1), mantisse (#23) en exponent (#8)
double	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**

```
class Pythagoras {  
    public static void main(String args[]){  
        double x, y, z;  
  
        x = 3;  
        y = 4;  
        z = Math.sqrt(x*x + y*y); //sqrt is een  
            statische methode uit de klasse Math  
  
        System.out.println("De lengte van de  
            schuine zijde is : " + z);  
    }  
}
```

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

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);  
    }  
}
```

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)	°
HTML Entity (hex)	°
HTML Entity (named)	°
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"
More...	

```

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\u00ED Se\u00F1or");
        System.out.println();
    }
}

```


booleans

type	#bits	bereik
boolean	8 bit	slechts 2 waarden : true en false

```
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

type	default veldwaarde
boolean	false
byte	0
short	0
int	0
long	0L
char	'\u0000'
float	0.0f
double	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\b x");  
        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 */");  
    }  
}
```

Karakters lezen van de console via byte-streams

In de Java API worden standaard (stream) variabelen voorzien om input en output te verzorgen van het toetsenbord en naar het scherm. Dit zijn de streamobjecten **System.out** die naar de console verwijst en **System.in** die naar het toetsenbord verwijst. **System.in** is van het type **InputStream**, **System.out** van het type **PrintStream**, beide zijn byte-stream klassen uit de I/O library van Java. Voorziene methodes :

System.out.print() , **System.out.println()**, maar ook **System.in.read()**. De *read* methode leest 1 karakter en wacht op een ENTER om dan een karakter terug te geven onder de vorm van een *int*. Let op alles wat ingegeven werd zit in een buffer, ook de return zelf! Bij herhaaldelijk lezen moet je deze dus steeds leegmaken.

```
public class LeesChar {  
    public static void main (String[] args)  
        throws java.io.IOException {  
        int i;  
        char c;  
        System.out.println("Geef 1 karakter in,  
            gevolgd door ENTER : ");  
        i = System.in.read();  
        c = (char)i;  
        System.out.println("Je karakter was : "  
            + c );  
        // door ENTER staan er 2 karakters  
        // extra op je in-buffer  
        System.in.read(); // carriage return  
        System.in.read(); // line feed  
    }  
}
```

```
        System.out.println("Geef een tweede  
            karakter in, gevolgd door ENTER : ");  
        i = System.in.read();  
        c = (char)i;  
        System.out.println("Je 2de karakter was  
            : " + c );  
    }  
}
```

Scanning

De standaard manier om te lezen van de console is echter door gebruik te maken van de **Scanner** klasse. Met behulp van de Scanner klasse kan text input opgebroken worden in stukken volgens een *delimiter* zoals een spatie. De text 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;
```

```
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());  
  
    s.close();  
}  
  
}
```

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

public class ScanInput {
    public static void main(String args[]){
        Scanner s = new Scanner(System.in);
        System.out.println("Geef een cijfer gevolgd door
            een karakter, en een woord : ");
        System.out.println(s.nextInt()); // het cijfer
        System.out.println(s.next());    // het karakter
        System.out.println(s.next());    // het woord
    }
}
```

```
s = new Scanner(System.in);
System.out.println("Geef nog een komma getal onder
    volgende settings " + (s.locale()).toString());
System.out.println(s.nextFloat());

s = new Scanner(System.in);
s.useLocale(new Locale("ENGLISH", "US"));
System.out.println("Geef een komma getal onder
    volgende settings : " +
    (s.locale()).toString());
System.out.println(s.nextFloat());
```

```
s = new Scanner(System.in);  
System.out.println("Geef een hele zin in : ");  
System.out.println(s.nextLine()); // de zin  
s.close();  
}  
}
```

Locale

Via de klasse **Locale** kan je locatie gebonden instellingen maken, zoals bvb. 3,14 versus 3.14. Opm wij gebruiken wetenschappelijke notatie en dus 3.14 m.a.w.

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