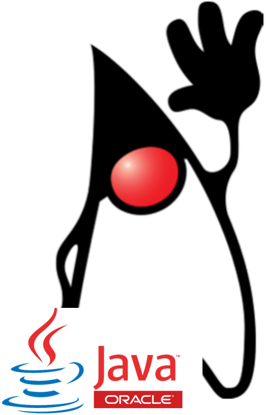
Primitieve operatoren

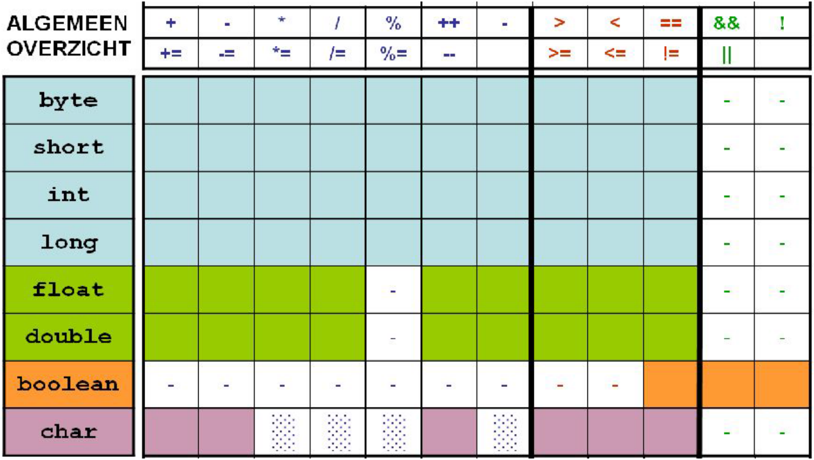
Klasgroep	1EO-ICT
Opleiding	Bachelor Elektronica-ICT
Lokaal	groot auditorium
Tijdstip	maandag lestijd 3
Docent	Katja Verbeeck
Contact	katja.verbeeck@odisee.be
Handboek	hfst 2

Inhoud

1. Introductie
2. Operatoren
3. Casting



Overzicht primitieve Operatoren



ALGEMEEN OVERZICHT	+	-	*	/	%	++	-	>	<	==	&&	!
	+=	-=	*=	/=	%=	--		>=	<=	!=	||	
byte	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	-	-
short	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	-	-
int	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	-	-
long	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	-	-
float	🟩	🟩	🟩	🟩	-	🟩	🟩	🟩	🟩	🟩	-	-
double	🟩	🟩	🟩	🟩	-	🟩	🟩	🟩	🟩	🟩	-	-
boolean	-	-	-	-	-	-	-	-	-	🟧	🟧	🟧
char	🟪	🟪	▒	▒	▒	🟪	▒	🟪	🟪	🟪	-	-

Primitieve Operatoren

Java kent volgende types operatoren :

Arithmetic Rekenkundige operatoren : $+$, $-$, $*$, $/$, %, $++$, $--$

Primitieve Operatoren

Java kent volgende types operatoren :

- Arithmetic Rekenkundige operatoren : $+, -, *, /,$ %, $++, --$
- Relational Vergelijkingsoperatoren : $==, != , >, <, \geq, \leq$

Primitieve Operatoren

Java kent volgende types operatoren :

- Arithmetic Rekenkundige operatoren : $+$, $-$, $*$, $/$, %, $++$, $--$
- Relational Vergelijkingsoperatoren : $==$, $!=$, $>$, $<$, $\geq$, $\leq$
- Logic Logische operaties : & (AND), $|$ (OR), $\wedge$ (XOR), && $(lazyAND)$, $||$ $(lazyOR)$, ! (NOT)

Primitieve Operatoren

Java kent volgende types operatoren :

- Arithmetic Rekenkundige operatoren : $+$, $-$, $*$, $/$, %, $++$, $--$
- Relational Vergelijkingsoperatoren : $==$, $!=$, $>$, $<$, $\geq$, $\leq$
- Logic Logische operaties : & (AND), $|$ (OR), $\wedge$ (XOR), && ($lazyAND$), $||$ ($lazyOR$), ! (NOT)
- Bitwise Bit operatoren : zie later

Primitieve Operatoren

Java kent volgende types operatoren :

- Arithmetic Rekenkundige operatoren : $+, -, *, /, \%, ++, --$
- Relational Vergelijkingsoperatoren : $==, != , >, <, \geq, \leq$
- Logic Logische operaties : $\& \, (AND), \mid (OR), \wedge (XOR),$
  $\&\& \, (lazyAND), \, || \, (lazyOR), \, ! \, (NOT)$
- Bitwise Bit operatoren : zie later
- Andere Assignement
  $=, += , -= , *= , /= , \%= , \&= , |= , \wedge =$
  Ternaire operator

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c = a + b;	
c = a - b;	
c = a * b ;	
c = a / b ;	
c = a % b;	
c ++ ; ++c;	
c - - ; - -c	
System.out.println(c++);	
System.out.println(++c);	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c = a + b;	13
c = a - b;	
c = a * b ;	
c = a / b ;	
c = a % b;	
c ++ ; ++c;	
c - - ; - -c	
System.out.println(c++);	
System.out.println(++c);	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c = a + b;	13
c = a - b;	7
c = a * b ;	
c = a / b ;	
c = a % b;	
c ++ ; ++c;	
c - - ; - -c	
System.out.println(c++);	
System.out.println(++c);	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c = a + b;	13
c = a - b;	7
c = a * b ;	30
c = a / b ;	
c = a % b;	
c ++ ; ++c;	
c - - ; - -c	
System.out.println(c++);	
System.out.println(++c);	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie
c = a + b;
c = a - b;
c = a * b ;
c = a / b ;
c = a % b;
c ++ ; ++c;
c - - ; - -c
System.out.println(c++);
System.out.println(++c);
- -c+2 ;

waarde van c
13
7
30
3

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c = a + b;	13
c = a - b;	7
c = a * b ;	30
c = a / b ;	3
c = a % b;	1
c ++ ; ++c;	
c - - ; - -c	
System.out.println(c++);	
System.out.println(++c);	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c = a + b;	13
c = a - b;	7
c = a * b ;	30
c = a / b ;	3
c = a % b;	1
c ++ ; ++c;	3
c - - ; - -c	
System.out.println(c++);	
System.out.println(++c);	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c = a + b;	13
c = a - b;	7
c = a * b ;	30
c = a / b ;	3
c = a % b;	1
c ++ ; ++c;	3
c - - ; - -c	1
System.out.println(c++);	
System.out.println(++c);	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c = a + b;	13
c = a - b;	7
c = a * b ;	30
c = a / b ;	3
c = a % b;	1
c ++ ; ++c;	3
c - - ; - -c	1
System.out.println(c++);	2
System.out.println(++c);	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie
c = a + b;
c = a - b;
c = a * b ;
c = a / b ;
c = a % b;
c ++ ; ++c;
c - - ; - -c
System.out.println(c++);
System.out.println(++c);
- -c+2 ;

waarde van c
13
7
30
3
1
3
1
2
3

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c = a + b;	13
c = a - b;	7
c = a * b ;	30
c = a / b ;	3
c = a % b;	1
c ++ ; ++c;	3
c - - ; - -c	1
System.out.println(c++);	2
System.out.println(++c);	3
- -c+2 ;	3

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c += b;	
c -= b;	
c *= b ;	
c /= b ;	
c %= b;	
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c += b;	5
c -= b;	
c *= b ;	
c /= b ;	
c %= b;	
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c += b;	5
c -= b;	-1
c *= b ;	
c /= b ;	
c %= b;	
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c += b;	5
c -= b;	-1
c *= b ;	6
c /= b ;	
c %= b;	
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c += b;	5
c -= b;	-1
c *= b ;	6
c /= b ;	0
c %= b;	
$a > b$	
$a < b$	
$a \geq b$	
$a \leq b$	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c += b;	5
c -= b;	-1
c *= b ;	6
c /= b ;	0
c %= b;	2
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c += b;	5
c -= b;	-1
c *= b ;	6
c /= b ;	0
c %= b;	2
a > b	true
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c += b;	5
c -= b;	-1
c *= b ;	6
c /= b ;	0
c %= b;	2
a > b	true
a < b	false
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c += b;	5
c -= b;	-1
c *= b ;	6
c /= b ;	0
c %= b;	2
a > b	true
a < b	false
a ≥ b	true
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c += b;	5
c -= b;	-1
c *= b ;	6
c /= b ;	0
c %= b;	2
a > b	true
a < b	false
a ≥ b	true
a ≤ b	false
a == b	
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	waarde van c
c += b;	5
c -= b;	-1
c *= b ;	6
c /= b ;	0
c %= b;	2
a > b	true
a < b	false
a ≥ b	true
a ≤ b	false
a == b	false
a != b	

Ga er voor elke expressie vanuit dat **int a = 10; int b= 3; int c = 2;**

expressie	**waarde van c**
c += b;	5
c -= b;	-1
c *= b ;	6
c /= b ;	0
c %= b;	2
a > b	true
a < b	false
a ≥ b	true
a ≤ b	false
a == b	false
a != b	true

De ternaire operator

```
public class TernaireOperator {
   public static void main (String[] args){
      int a = 10, b = 3, c = 2, grootste;
      boolean res;

      res = a > b;
      System.out.println("res : " + res);

      grootste = (a > b ? a : b);
      System.out.println("grootste : " +
         grootste);
      System.out.println(a==b ? "gelijk" :
         "verschillend");
   }
}
```

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **double a = 1.2, b = 0.4, c = 2.0**

expressie	waarde van c
c = a + b;	
c = a - b;	
c = a * b ;	
c = a / b ;	
c = a % b;	
c ++ ; ++c;	
c - - ; - -c	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **double a = 1.2, b = 0.4, c = 2.0**

expressie	waarde van c
c = a + b;	1.6
c = a - b;	
c = a * b ;	
c = a / b ;	
c = a % b;	
c ++ ; ++c;	
c - - ; - -c	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **double a = 1.2, b = 0.4, c = 2.0**

expressie	waarde van c
c = a + b;	1.6
c = a - b;	0.8
c = a * b ;	
c = a / b ;	
c = a % b;	
c ++ ; ++c;	
c - - ; - -c	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **double a = 1.2, b = 0.4, c = 2.0**

expressie	waarde van c
c = a + b;	1.6
c = a - b;	0.8
c = a * b ;	0.48
c = a / b ;	
c = a % b;	
c ++ ; ++c;	
c - - ; - -c	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **double a = 1.2, b = 0.4, c = 2.0**

expressie	waarde van c
c = a + b;	1.6
c = a - b;	0.8
c = a * b ;	0.48
c = a / b ;	3.0
c = a % b;	
c ++ ; ++c;	
c - - ; - -c	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **double a = 1.2, b = 0.4, c = 2.0**

expressie	waarde van c
c = a + b;	1.6
c = a - b;	0.8
c = a * b ;	0.48
c = a / b ;	3.0
c = a % b;	–nvt–
c ++ ; ++c;	
c - - ; - -c	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **double a = 1.2, b = 0.4, c = 2.0**

expressie	waarde van c
c = a + b;	1.6
c = a - b;	0.8
c = a * b ;	0.48
c = a / b ;	3.0
c = a % b;	–nvt–
c ++ ; ++c;	3.0
c - - ; - -c	
- -c+2 ;	

Geef de waarde van variabele *c* nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **double a = 1.2, b = 0.4, c = 2.0**

expressie	waarde van c
c = a + b;	1.6
c = a - b;	0.8
c = a * b ;	0.48
c = a / b ;	3.0
c = a % b;	–nvt–
c ++ ; ++c;	3.0
c - - ; - -c	1.0
- -c+2 ;	

Geef de waarde van variabele c nadat volgende assignments uitgevoerd werden. Ga er voor elke expressie vanuit dat **double a = 1.2, b = 0.4, c = 2.0**

expressie	**waarde van c**
c = a + b;	1.6
c = a - b;	0.8
c = a * b ;	0.48
c = a / b ;	3.0
c = a % b;	–nvt–
c ++ ; ++c;	3.0
c - - ; - -c	1.0
- -c+2 ;	3.0

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	
c -= b;	
c *= b ;	
c /= b ;	
c %= b;	
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	**waarde van c**
c += b;	2.4
c -= b;	
c *= b ;	
c /= b ;	
c %= b;	
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	2.4
c -= b;	1.6
c *= b ;	
c /= b ;	
c %= b;	
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	2.4
c -= b;	1.6
c *= b ;	0.8
c /= b ;	
c %= b;	
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	2.4
c -= b;	1.6
c *= b ;	0.8
c /= b ;	5.0
c %= b;	
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	2.4
c -= b;	1.6
c *= b ;	0.8
c /= b ;	5.0
c %= b;	–nvt–
a > b	
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	2.4
c -= b;	1.6
c *= b ;	0.8
c /= b ;	5.0
c %= b;	–nvt–
a > b	true
a < b	
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	2.4
c -= b;	1.6
c *= b ;	0.8
c /= b ;	5.0
c %= b;	–nvt–
a > b	true
a < b	false
a ≥ b	
a ≤ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	2.4
c -= b;	1.6
c *= b ;	0.8
c /= b ;	5.0
c %= b;	–nvt–
a > b	true
a < b	false
a $\geq$ b	true
a $\leq$ b	
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	2.4
c -= b;	1.6
c *= b ;	0.8
c /= b ;	5.0
c %= b;	–nvt–
a > b	true
a < b	false
a ≥ b	true
a ≤ b	false
a == b	
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	**waarde van c**
c += b;	2.4
c -= b;	1.6
c *= b ;	0.8
c /= b ;	5.0
c %= b;	–nvt–
a > b	true
a < b	false
a ≥ b	true
a ≤ b	false
a == b	false
a != b	

Ga er voor elke expressie vanuit dat **double a = 1.2, int b= 0.4, c = 2.0;**

expressie	waarde van c
c += b;	2.4
c -= b;	1.6
c *= b ;	0.8
c /= b ;	5.0
c %= b;	–nvt–
a > b	true
a < b	false
a $\geq$ b	true
a $\leq$ b	false
a == b	false
a != b	true

Oefening

Stel : **int score = 0, totaal = 10;**
Wat is dan de uitvoer op het scherm van volgende expressie ?
System.out.println(++score*totaal++);

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	waarde
x == y ;	
x != y ;	
x && y;	
x || y;	
! x	
x && !y || x	
!(x && y)	
c2 = c1 + 2;	
c2 = ‘A’ + 32;	
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	**waarde**
x == y ;	false
x != y ;	
x && y;	
x || y;	
! x	
x && !y || x	
!(x && y)	
c2 = c1 + 2;	
c2 = ‘A’ + 32;	
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	waarde
x == y ;	false
x != y ;	true
x && y;	
x || y;	
! x	
x && !y || x	
!(x && y)	
c2 = c1 + 2;	
c2 = ‘A’ + 32;	
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	waarde
x == y ;	false
x != y ;	true
x && y;	false
x || y;	
! x	
x && !y || x	
!(x && y)	
c2 = c1 + 2;	
c2 = ‘A’ + 32;	
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	waarde
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	
x && !y || x	
!(x && y)	
c2 = c1 + 2;	
c2 = ‘A’ + 32;	
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	**waarde**
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	
!(x && y)	
c2 = c1 + 2;	
c2 = ‘A’ + 32;	
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	**waarde**
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	true
!(x && y)	
c2 = c1 + 2;	
c2 = ‘A’ + 32;	
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	waarde
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	true
!(x && y)	true
c2 = c1 + 2;	
c2 = ‘A’ + 32;	
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	**waarde**
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	true
!(x && y)	true
c2 = c1 + 2;	‘c’
c2 = ‘A’ + 32;	
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	waarde
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	true
!(x && y)	true
c2 = c1 + 2;	‘c’
c2 = ‘A’ + 32;	‘a’
c2 = ‘0’ + 5;	
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	waarde
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	true
!(x && y)	true
c2 = c1 + 2;	‘c’
c2 = ‘A’ + 32;	‘a’
c2 = ‘0’ + 5;	‘5’
boolean b = c1 > ‘A’ ;	
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	**waarde**
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	true
!(x && y)	true
c2 = c1 + 2;	‘c’
c2 = ‘A’ + 32;	‘a’
c2 = ‘0’ + 5;	‘5’
boolean b = c1 > ‘A’ ;	true
System.out.println(‘5’);	
System.out.println((int)’5’);	
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	**waarde**
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	true
!(x && y)	true
c2 = c1 + 2;	‘c’
c2 = ‘A’ + 32;	‘a’
c2 = ‘0’ + 5;	‘5’
boolean b = c1 > ‘A’ ;	true
System.out.println(‘5’);	5
System.out.println((int)’5’);	
System.out.println((char)88);	

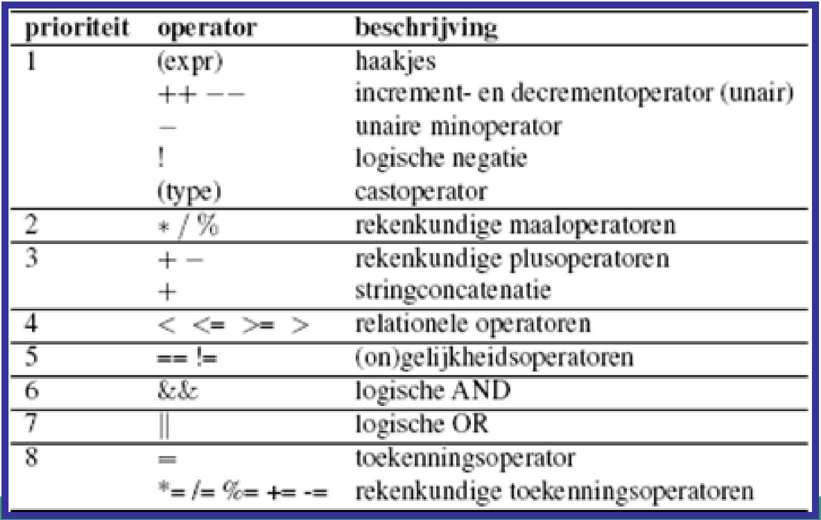
Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	waarde
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	true
!(x && y)	true
c2 = c1 + 2;	‘c’
c2 = ‘A’ + 32;	‘a’
c2 = ‘0’ + 5;	‘5’
boolean b = c1 > ‘A’ ;	true
System.out.println(‘5’);	5
System.out.println((int)’5’);	53
System.out.println((char)88);	

Ga er voor elke expressie vanuit dat **boolean x = true, y = false** en **char c1 = ’a’, c2;**

expressie	**waarde**
x == y ;	false
x != y ;	true
x && y;	false
x || y;	true
! x	false
x && !y || x	true
!(x && y)	true
c2 = c1 + 2;	‘c’
c2 = ‘A’ + 32;	‘a’
c2 = ‘0’ + 5;	‘5’
boolean b = c1 > ‘A’ ;	true
System.out.println(‘5’);	5
System.out.println((int)’5’);	53
System.out.println((char)88);	X

Volgorde van de bewerkingen



prioriteit	operator	beschrijving
1	(expr)	haakjes
	++ −−	increment- en decrementoperator (unair)
	−	unaire minoperator
	!	logische negatie
	(type)	castoperator
2	* / %	rekenkundige maaloperatoren
3	+ −	rekenkundige plusoperatoren
	+	stringconcatenatie
4	< <= >= >	relationele operatoren
5	== !=	(on)gelijkheidsoperatoren
6	&&	logische AND
7	||	logische OR
8	=	toekenningsoperator
	*= /= %= += -=	rekenkundige toekenningsoperatoren

Concatenatie

expressie
1 + ‘a’
‘a’ + 1
”” + ’a’
1 + ””
”! ” + ‘a’ + 1
”!! ” + (‘a’ + 1)
”” + (char)(‘a’ + 1)

type	waarde

Concatenatie

expressie
1 + ‘a’
‘a’ + 1
”” + ’a’
1 + ””
”! ” + ‘a’ + 1
”!! ” + (‘a’ + 1)
”” + (char)(‘a’ + 1)

type	waarde
int	98

Concatenatie

expressie	type	waarde
1 + ‘a’	int	98
‘a’ + 1	int	98
”” + ’a’		
1 + ””		
”! ” + ‘a’ + 1		
”!! ” + (‘a’ + 1)		
”” + (char)(‘a’ + 1)		

Concatenatie

expressie	**type**	**waarde**
1 + ‘a’	int	98
‘a’ + 1	int	98
”” + ’a’	String	”a”
1 + ””		
”! ” + ‘a’ + 1		
”!! ” + (‘a’ + 1)		
”” + (char)(‘a’ + 1)		

Concatenatie

expressie
1 + ‘a’
‘a’ + 1
”” + ’a’
1 + ””
”! ” + ‘a’ + 1
”!! ” + (‘a’ + 1)
”” + (char)(‘a’ + 1)

type	waarde
int	98
int	98
String	”a”
String	”1”

Concatenatie

expressie
1 + ‘a’
‘a’ + 1
”” + ’a’
1 + ””
”! ” + ‘a’ + 1
”!! ” + (‘a’ + 1)
”” + (char)(‘a’ + 1)

type	waarde
int	98
int	98
String	”a”
String	”1”
String	”! a1”

Concatenatie

expressie
1 + ‘a’
‘a’ + 1
”” + ’a’
1 + ””
”! ” + ‘a’ + 1
”!! ” + (‘a’ + 1)
”” + (char)(‘a’ + 1)

type	waarde
int	98
int	98
String	”a”
String	”1”
String	”! a1”
String	”!! 98”

Concatenatie

expressie
1 + ‘a’
‘a’ + 1
”” + ’a’
1 + ””
”! ” + ‘a’ + 1
”!! ” + (‘a’ + 1)
”” + (char)(‘a’ + 1)

type	waarde
int	98
int	98
String	”a”
String	”1”
String	”! a1”
String	”!! 98”
String	”b”

Impliciete versus expliciete cast

Soms kan je een waarde van het ene type toekennen aan het andere :

```
 byte b = 20;
int kopie = b;  // impliciet wordt b omgezet naar een type int
```

Dit kan natuurlijk alleen maar als :

- de twee types compatibel zijn : bvb. van **int** naar **boolean** kan niet, maar **int** naar **float** kan wel
- het type waaraan je toekent moet meer geheugencapaciteit hebben : wanneer je een **int** in een **byte** stockeert verlies je gegevens.

  toch kan dit maar dan met een expliciete cast

  ```
  kopie = kopie + 256 ;
  byte b2 = (byte) kopie;  // expliciete omzetting
  ```

Expliciete cast

```
public class CastExpl {
   public static void main (String[] args){
      double x,y;
      byte b;
      int i;
      char c;

      x = 10.0;
      y = 3.0;
      i = (int) (x / y); // double to int
      System.out.println("(int) ( x / y) = " +
         i);
```

Expliciete cast - vervolg

```
i = 100;
b = (byte) i;  // byte kan tot +127
System.out.println("waarde b = " + b);

i = 257;
b = (byte) i;  // informatieverlies
System.out.println("waarde b = " + b);

b = 10;
i = b * b; // bytes worden in een bew.
   altijd omgezet naar int
b = (byte)(b * b);
System.out.println("waarde i = " + i + "
   waarde b = " + b);
```

Expliciete cast - vervolg

```
      b = 61; // ASCII voor '='
      c = (char)b;
      System.out.println("waarde c = " + c);

      c = (char)(c + c); //chars worden in
         bew. omgezet naar int
      System.out.println("waarde c = " + c);
   }
}
```

Stijl

Gebruik voldoende tabs en spaties in je expressies om de leesbaarheid te verhogen. dus :

$x \ = \ 10 \ / \ y \ * \ (127/x);$

ipv.

$x = 10/y * (127/x);$

expressie
5 - 2 / 4 - 2
(double) (11 / 2)
(double) 11 / 2
11 / (double) 2
1 < 2 && !(2 < 1)
(int) 1.0 / 2 + 0.5
(int) (1.0 / 2) + 0.5
(int) (1.0 / 2 + 0.5)

type	waarde

expressie
5 - 2 / 4 - 2
(double) (11 / 2)
(double) 11 / 2
11 / (double) 2
1 < 2 && !(2 < 1)
(int) 1.0 / 2 + 0.5
(int) (1.0 / 2) + 0.5
(int) (1.0 / 2 + 0.5)

type	waarde
int	3

expressie
5 - 2 / 4 - 2
(double) (11 / 2)
(double) 11 / 2
11 / (double) 2
1 < 2 && !(2 < 1)
(int) 1.0 / 2 + 0.5
(int) (1.0 / 2) + 0.5
(int) (1.0 / 2 + 0.5)

type	waarde
int	3
double	5.0

expressie
5 - 2 / 4 - 2
(double) (11 / 2)
(double) 11 / 2
11 / (double) 2
1 < 2 && !(2 < 1)
(int) 1.0 / 2 + 0.5
(int) (1.0 / 2) + 0.5
(int) (1.0 / 2 + 0.5)

type	waarde
int	3
double	5.0
double	5.5

expressie
5 - 2 / 4 - 2
(double) (11 / 2)
(double) 11 / 2
11 / (double) 2
1 < 2 && !(2 < 1)
(int) 1.0 / 2 + 0.5
(int) (1.0 / 2) + 0.5
(int) (1.0 / 2 + 0.5)

type	waarde
int	3
double	5.0
double	5.5
double	5.5

expressie	type	waarde
5 - 2 / 4 - 2	int	3
(double) (11 / 2)	double	5.0
(double) 11 / 2	double	5.5
11 / (double) 2	double	5.5
1 < 2 && !(2 < 1)	boolean	true
(int) 1.0 / 2 + 0.5		
(int) (1.0 / 2) + 0.5		
(int) (1.0 / 2 + 0.5)		

expressie
5 - 2 / 4 - 2
(double) (11 / 2)
(double) 11 / 2
11 / (double) 2
1 < 2 && !(2 < 1)
(int) 1.0 / 2 + 0.5
(int) (1.0 / 2) + 0.5
(int) (1.0 / 2 + 0.5)

type	waarde
int	3
double	5.0
double	5.5
double	5.5
boolean	true
double	0.5

expressie
5 - 2 / 4 - 2
(double) (11 / 2)
(double) 11 / 2
11 / (double) 2
1 < 2 && !(2 < 1)
(int) 1.0 / 2 + 0.5
(int) (1.0 / 2) + 0.5
(int) (1.0 / 2 + 0.5)

type	waarde
int	3
double	5.0
double	5.5
double	5.5
boolean	true
double	0.5
double	0.5

expressie	type	waarde
5 - 2 / 4 - 2	int	3
(double) (11 / 2)	double	5.0
(double) 11 / 2	double	5.5
11 / (double) 2	double	5.5
1 < 2 && !(2 < 1)	boolean	true
(int) 1.0 / 2 + 0.5	double	0.5
(int) (1.0 / 2) + 0.5	double	0.5
(int) (1.0 / 2 + 0.5)	int	1