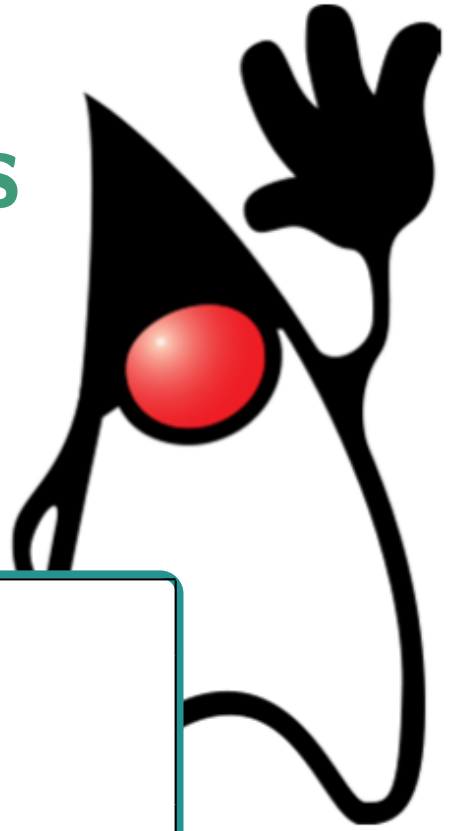
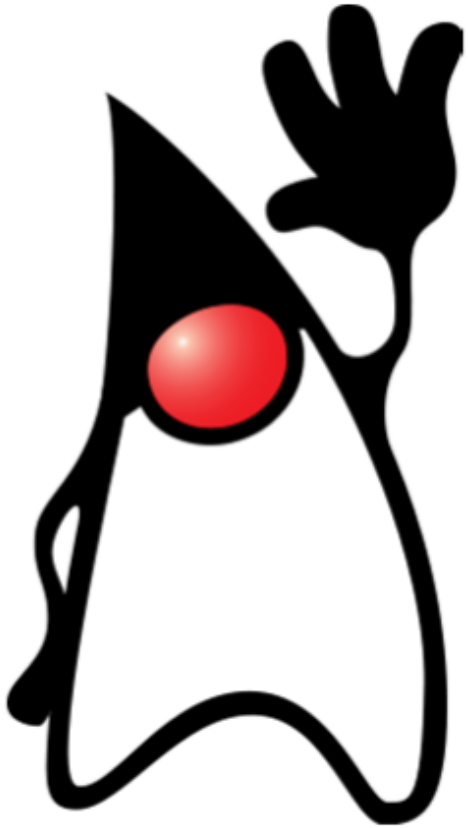
# Programming Fundamentals
# Primitieve operatoren



| | |
|---|---|
| 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





# Overzicht primitieve Operatoren

| ALGEMEEN OVERZICHT | + | - | * | / | % | ++ | - | > | < | == | && | ! |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | += | -= | *= | /= | %= | -- | | >= | <= | != | \|\| | |
| byte | | | | | | | | | | | - | - |
| short | | | | | | | | | | | - | - |
| int | | | | | | | | | | | - | - |
| long | | | | | | | | | | | - | - |
| float | | | | | - | | | | | | - | - |
| double | | | | | - | | | | | | - | - |
| boolean | - | - | - | - | - | - | - | - | - | | | |
| char | | | | | | | | | | | - | - |

# Primitieve Operatoren

Java kent volgende types operatoren :

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

Relational **Vergelijkingsoperatoren** : $==$, $!=$, $>$, $<$, $\geq$, $\leq$

Logic **Logische operatoren** : $\&$ ($AND$), $|$ ($OR$), $\wedge$ ($XOR$), $\&\&$ ($lazyAND$), $||$ ($lazyOR$), $!$ ($NOT$)

Bitwise **Bit operatoren** : zie later

Andere **Toekennings operatoren** (of Assignement)
$=$, $+=$, $-=$, $*=$, $/=$, $\%=$, $\&=$, $|=$, $\wedge=$
**De Ternaire operator** ? :
$a == b \; ? \; "gelijk" \; : \; "verschillend";$

# Gehele rekenkundige operatoren

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 |

# Toekenning en logische expressies

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 |

# 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

```java
public 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);
    }
}
```

```
Output - VarsPrimTypes (run)
run:
waarde van var : 10
waarde van x : 10.0
waarde van var na deling : 2
waarde van x na deling : 2.5
BUILD SUCCESSFUL (total time: 0 seconds)
```

# int versus double

## Gehele Deling vs Reële deling

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

# Reële rekenkundige operatoren

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 |

# Toekenning en logische expressies

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

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

| waarde van c |
|---|
| 2.4 |
| 1.6 |
| 0.8 |
| 5.0 |
| –nvt– |
| |
| true |
| false |
| true |
| false |
| false |
| true |

# Logische operatoren

Ga er voor elke expressie vanuit dat **boolean x = true, y = false**

| **expressie** | **waarde** |
|---|---|
| x == y ; | false |
| x != y ; | true |
| x && y; | false |
| x \|\| y; | true |
| ! x | false |
| x && !y \|\| x | true |
| !(x && y) | true |

# Operaties met chars

Ga er voor elke expressie vanuit dat **char c1 = 'a', c2;**

| **expressie** | **waarde** |
|---|---|
| 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 |

# Strings aan elkaar plakken of concateneren

| | |
|---|---|
| "dit" + "dat" | "ditdat" |
| "abc" + 5 | "abc5" |
| "Dat is dan " + 10 + " euro a.u.b." | "Dat is dan 10 euro a.u.b." |
| 5 + "abc" | "5abc" |
| "a" + 1 + 2 | "a12" |
| 1 + 2 + "abc" | "3abc" |

# 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” |

# 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 |

# De ternaire operator ? :

De ternaire operator **?:** is de enige operator die drie operands neemt als input :

*booleanExpression* ? *expression2* : *expression3*

- De eerste expressie is een booleaanse expressie, m.a.w. het resultaat hiervan moet true of false teruggeven.
- De tweede en derde expressie zijn expressies die een waarde teruggeven van eender welk type. Beide moeten wel hetzelfde type teruggeven.
- Het resultaat van de ternaire operator is de waarde van expressie 2 als de booleaanse expressie true oplevert. Indien deze false oplevert dan is het resultaat de waarde van expressie 3.
- Deze operator is dus een heel compacte vorm van selectie in je code, hierover later meer.

# De ternaire operator: voorbeeld

```
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");
    }
}
```

# Casting : Impliciete vs expliciet

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);
    }
}
```

# Gehele getallen genereren via 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.

# Gehele getallen genereren via 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.

# Wat gaat er fout?

```
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));
```

# Stijl

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

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

ipv.

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