



Informatique 1

4. Expressions et opérateurs

Objectifs du cours

Les expressions en programmation

- ▶ Les expressions
- ▶ Opérateurs
 - ▶ Type
 - ▶ Priorité

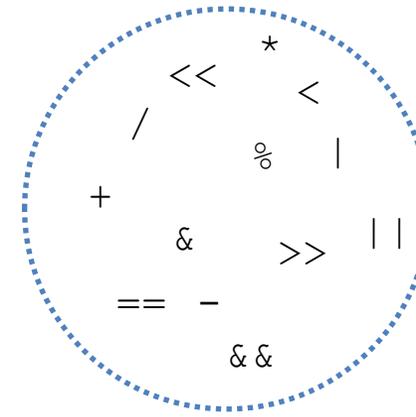
Mixing stuff

4.1 EXPRESSIONS

Expressions

Valeurs
littérales

Variables



Opérateurs



`(toto + 34)`

`y = 3 + 4 - 1`

`((a >> 1) & 0b1010) <<< 6) % 4`

`Math.sin(value) / 2.0f`

`foo == true ? 25 : 12`

Expressions

- Types **d'opérateurs**
 - Arithmétiques
 - Logiques
 - Spécifiques au langage (comme le .)
- En combinant opérateurs, variables et littéraux
→ **expressions**
- *Exemples*

$2*(toto-1)+(4/2)$
 $4.0/(2.13*2.12)$

Typed results

4.2 OPÉRATEURS ET TYPES

Opérateurs arithmétiques (1)

Opérateur	Fonction
*	Multiplication
+	Addition
-	Soustraction
/	Division
%	Modulo (reste division entière)

- *Les expressions numériques sont composées à partir d'opérateurs arithmétiques*
- Elles sont définies pour les types entiers et réels
- **L'évaluation de l'expression conduit à sa valeur**
 $(3 + (2 * 5))$ s'évalue à 13

Opérateurs arithmétiques (2)

```
int x = 4;
```

```
int y = 7;
```

```
int toto = 3;
```

```
x = (y + 3) % toto;
```

```
y = 22 + (x - toto);
```



Calcul de la moyenne

```
int note1 = 3;  
int note2 = 5;
```

Remarques sur opérateur division

- ▶ Si **a** et **b** sont des entiers, a / b est aussi entier

$$5 / 2 \rightarrow 2 \quad \leftarrow$$



- ▶ Si **a** et **b** sont des réels, a/b est réel

$$5.0 / 2.0 \rightarrow 2.5$$

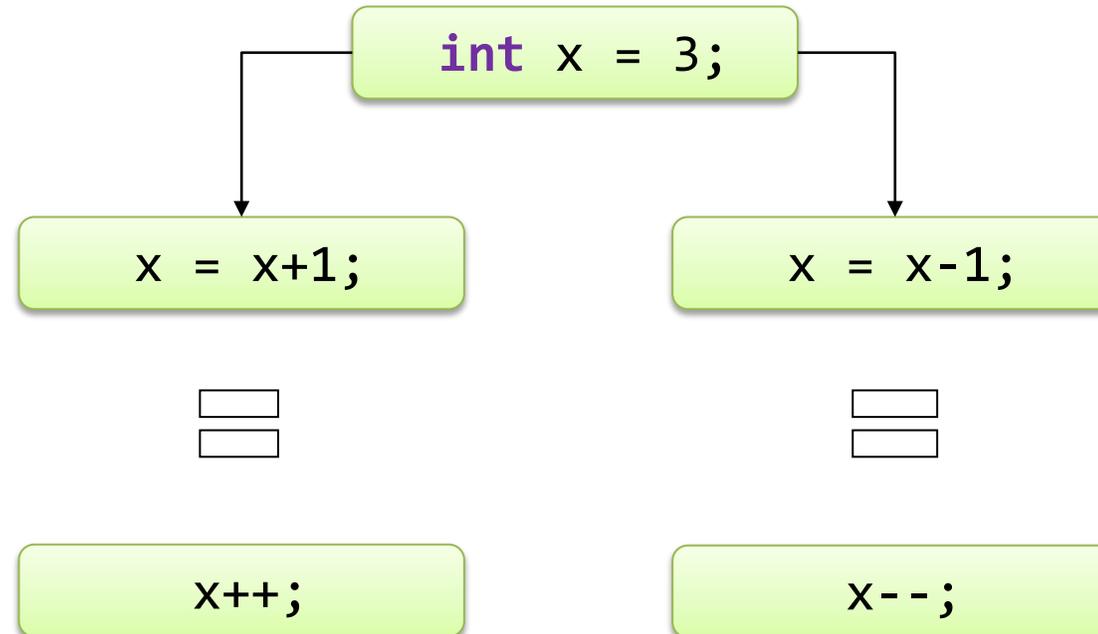
- ▶ Si **a** et **b** sont de types différents, le résultat est réel

$$5.0 / 2 \rightarrow 2.5$$

Opérateur incrémentation

- *Incrémentation* et *décrémentation*

Exemple :



Opérateur incrémentation (2)

- Autre forme et signification, `--x` et `++x`
-  ordre différent !!

```
int x = 3, y = 3, toto, titi;  
toto = 1 + x++;  
titi = 1 + --y;
```

Opérateur incrémentation (3)

Opérateur	Nom	Exemple	Explication
<code>++</code>	Preincrement	<code>++a</code>	
<code>++</code>	Postincrement	<code>a++</code>	
<code>--</code>	Predecrement	<code>--b</code>	
<code>--</code>	Postdecrement	<code>b--</code>	

Opérateurs binaires

Opérateur	Fonction
&	and
	or
^	xor
~	not
>>, <<	<i>Décalage signé</i>

Exemples

```
short a = 0b1101; short b = 0x3e; short result;  
result = a & b;  
result = a | b;  
result = a >> 1;  
result = a >> 3;  
result = a ^ 0b11;  
result = ~a;
```

Opérateurs relationnels

Opérateur	Fonction
==	Egalité
> et >=	Supérieur (ou égal)
< et <=	Inférieur (ou égal)
!=	Différent de

- Permettent d'effectuer des comparaisons
- Résultat type **boolean**

Exemples

```
int a = 3; int b = 5;  
boolean result;  
result = a < b;  
result = a >= 3;  
result = b != 2;  
result = a == 3;
```

Opérateurs booléens (logiques)

Opérateur	Fonction
&&	et
	ou
!	négation

- Travaillent sur **boolean** et produisent des types **boolean**
- Utilisés pour réaliser des conditions complexes

Exemples

- a est positif **et** $c > 5$
 $((a > 0) \ \&\& \ (c > 5))$
- a n'est **pas** plus petit ou égal à 5 (càd ?)
 $!(a \leq 5)$

Opérateurs d'assignation

- Assignation, opérateur '='
- *Notation courte* pour affectations

Forme normale

```
x = x + y;
```

```
toto = toto * 4;
```

Forme courte

```
x += y;
```

```
toto *= 4;
```

- Formes existantes:

`+=` `-=` `*=` `/=` `%=` `&=` `|=` `^=` `<<=` `>>=` `>>>=`

= et ==

- Ne pas confondre "==" et "=" !!!
 - ▶ Erreur très fréquente en *C/C++*
 - ▶ *Java* signale l'erreur si rencontrée



Opérateurs d'assignation, remarque

- L'assignation n'est pas commutative :

$x = y;$

n'est **PAS** équivalent à :

$y = x;$

- Il faut imaginer l'assignation comme une flèche de droite à gauche

$x = y;$ signifie $x \leftarrow y;$

$y = x;$ signifie $y \leftarrow x;$

Opérateurs conditionnels

- Opérateur conditionnel du type "? :"
- Syntaxe

`expr ? a : b`

- Si **expr** vaut **true**, l'expression vaut a, autrement b

```
int toto = 0;  
toto = (3 > 5)? 24 : 4;  
toto = (5 <= 23)? 1 : 0;
```

Surcharge de l'opérateur +

```
String s1 = "Garfield"; String s2 = " est un chat";  
String s3 = s1 + s2;  
System.out.println(s3);
```

- L'opérateur + peut être appliqué a des **String** → opérateur *concaténation*
- Si l'une des opérandes du + est un String, alors l'opération est la concaténation
- Attention à la priorité !

What to do first

4.3 PRIORITÉ DES OPÉRATEURS

Evaluation des expressions

- Les expressions **s'évaluent** (donnent une valeur) pour être utilisées, par exemple pour assigner une variable.
- L'évaluation **s'arrête** lorsque tous les opérateurs ont été appliqués.
- L'évaluation se fait **dans l'ordre** de priorité des opérateurs (basé sur évaluation mathématique)

Priorité opér.

- Question récurrente dans tous les langages
- Ordre appliqué :
 1. Priorité des opérateur
 2. Si deux opérateurs de même priorité, associativité à gauche
- Conseil :

Dans le doute, utilisez des parenthèses !

Priorité max



Priorité min

Opérateur	Description	Assoc.
[] . ()	access array element access object member invoke a method	left to right
++ -- ++ -- + - ! ~	post-in(dec)crement pre- inc(dec)rement unary plus minus logical and bitwise NOT	right to left
() new	cast and object creation	right to left
* / %	multiplicative	left to right
+ -	additive, string concatenation	left to right
<< >> >>>	shift	left to right
< <= > >= instanceof	relational type comparison	left to right
== !=	equality	left to right
&	bitwise AND	left to right
^	bitwise XOR	left to right
	bitwise OR	left to right
&&	conditional AND	left to right
	conditional OR	left to right
? :	conditional	right to left
= += - = *= /= %= &= ^= = <<= >>= >>>=	assignment	right to left

Conclusion

- Vous avez appris dans ce cours:
 - ▶ L'existence de plusieurs types d'opérateurs
 - ▶ Comment les combiner dans des expressions
 - ▶ Comment sont évaluées les expressions arithmétiques par l'ordinateur
 - ⇒ Vous connaissez l'arithmétique à écrire dans un programme !
- Faites le quiz online (COURS 4)