# **Memo - Initiation à Java**

|  |
| --- |
| Version 1.0 (JBC) |

Table des matières

[**Memo - Initiation à Java** 1](#_Toc68007864)

[Les variables 2](#_Toc68007865)

[Déclaration, affectation et porté 2](#_Toc68007866)

[Les variables de type primitif 2](#_Toc68007867)

[Constante 2](#_Toc68007868)

[Conversion de type (implicite/ explicite) 2](#_Toc68007869)

[Les opétrateurs 3](#_Toc68007870)

[Les opérateurs d’affectation 3](#_Toc68007871)

[Les Opérateur arithmétiques 3](#_Toc68007872)

[Opérateur de concaténation 3](#_Toc68007873)

[Opérateurs de comparaison 3](#_Toc68007874)

[Opérateurs logiques 3](#_Toc68007875)

[Structure conditionnel 4](#_Toc68007876)

[If…else 4](#_Toc68007877)

[switch 4](#_Toc68007878)

[Structure de contrôle : les répétitives 4](#_Toc68007879)

[while 4](#_Toc68007880)

[do…while 4](#_Toc68007881)

[for… 4](#_Toc68007882)

[Methode 4](#_Toc68007883)

[Les variables complexes 4](#_Toc68007884)

[Les tableaux 4](#_Toc68007885)

[Les chaines de caractères 5](#_Toc68007886)

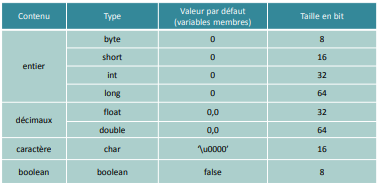
[Les enumerations 5](#_Toc68007887)

## Les variables

### Déclaration, affectation et porté

* **Porté :** une variable n’est connue que dans le bloc d’instruction dans lequel elle est déclarer
* **Déclaration :** **type nomVariable**
* **Affectation :** **nomVariable = 42**
* **Déclaration et affectation :** **type nomVariable = 42**

### Les variables de type primitif



*Note : pour utiliser un type long il faut faire suivre la valeur par l et pour un float un f*

*Ex : long nombreA = 500l ; | float taille = 51.20f ;*

*NB : au-delà d’une certaine valeur (2p64) il faut stocker les entiers dans des float ou double !*

### Constante

*Pour déclarer une constante on doit utiliser le mot clé final, ex :* **final string NOM\_ECOLE ;**

### Conversion de type (implicite/ explicite)

Certaines conversions de types sont destructrices (par exemple si on convertie un décimal en entier etc.), c’est ce qu’ont appel des conversion explicite car il faut caster (préciser) dans quel type on souhaite convertir. Exemple : **float unFLoat = (float)unDouble ;** (ici ont convertie unDouble en float).

Les autres conversions, qui ne sont pas destructrice, sont de type implicite, il n’y as pas besoin de caster. Exemple **long unLong = unEntier ;** (on convertie un int en long)

#### Classe wrapper

**Listes des classe wrapper** : Byte, Short, Integer, Long, Float, Double, Boolean, Character

**Constante des class wrapper :** .MIN\_VALUE et MAX\_VALUE (ex : Byte.MAX\_VALUE renvois la valeur maximal d’un byte)

**Méthode des class wrapper :** isDigit(), isLetter,(), isLowerCase(), isUpperCase(), etc…

#### conversion primitif => String

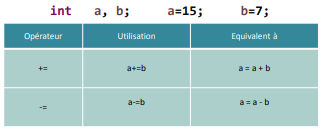
* **String chaine = String.valueOf(10.23) ;** // convertie 10.23 en chaine de caractere (String)
* **String chaine = Byte.valueOf(value).toString()**

#### conversion String => primitif

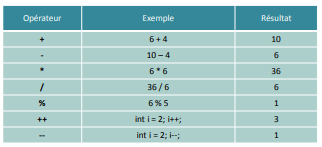
* **int unEntier = Integer.parseInt(« 10 ») ;** // convertie la chaine « 10 » en int 10

## Les opétrateurs

### Les opérateurs d’affectation



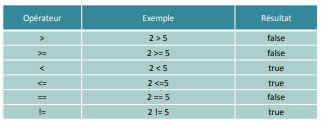
### Les Opérateur arithmétiques



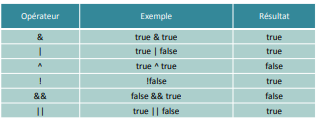
### Opérateur de concaténation



### Opérateurs de comparaison



### Opérateurs logiques

*Précision :*

*& : ET vérifie les deux conditions*

*| : OU au moins un*

*^ : OU un et un seul*

*! : DIFFERENT*

*&& : ET si la première est fausse ne vérifie pas la 2e*

*|| : OU si la première est vrai ne vérifie pas la 2e*

## Structure conditionnel

|  |  |
| --- | --- |
| If…else **If (condition) {**  Instruction  **} else if (**condition**) {**  Instruction  **} else {**  Instruction  **}** | switch **switch(variable) {**  **case 1 :** Instruction  break ;  **…**  **default :** Instruction  **}** |

## Structure de contrôle : les répétitives

|  |  |  |
| --- | --- | --- |
| while **while(condition){**  **instruction**  **}** | do…while **do{**  instruction  **} while (condition)** | for… **for(int i = 0 ; i<5 ; i++{**  instruction  **}** |

## Methode

Une méthode (= fonction) est un bloc d’instruction renvoyant ou non une valeur. On peut définir autant de paramètres que l’on souhaite.

|  |  |
| --- | --- |
| définition | appel |
| **public static void nomFonction(int val1, int val2 […]) {…}**  **public static int nomFonction(int val1, int val2 […]) {…}** | **nomFonction(20,50)**  **int value = nomFonction(60,40)** |

## Les variables complexes

### Les tableaux

**Déclaration** : **int[] nomTableau = new int [TAILLE]** ou encore **int[] tab = {2,5,78,98}**

**Déclaration tableau multidimensionnel : int[][] matrice = new int[5][]** // pas obliger de spécifier la taille du deuxième

**Propriété du tableau** : **nomTableau.lenght** pour afficher sa taille

#### Quelques méthodes de la class Arrays

**Afficher le contenu du tableau sous forme de texte** : **Arrays.toString(monTableau)** ;

**Remplir le tableau :** **Arrays.fill(monTableau, 6) ;** ici remplis monTableau de 6

**Trier le tableau : Arrays.sort(monTab) ;**

**Rechercher l’index ou se situe une valeur : Arrays.binarySearch(monTab, 8) ;** retournera l’index ou se situe la valeur 8

#### itération : for each

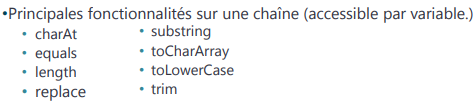
**Parcourir le tableau** :

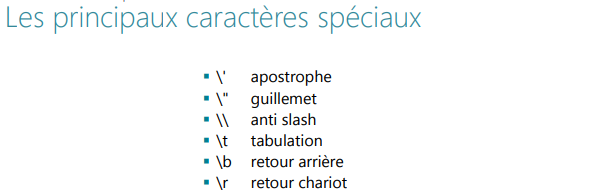
**for(int element : tableauDeInt) {**

**System.out.println(element) ;}**

## Les chaines de caractères

**Déclaration : String maPhrase = « salut c’est une phrase » ;**

****

****

**Attention**! Pour comparer 2 chaines de caractères on utilise la méthode **equals** et non **==** car String est stocké par référence et non par valeur

## Les enumerations

Une énumération est une sorte de type de variable personnalisé qui as plusieurs valeur possible (comprise dans un tableau de constante lié entre elles).

Par exemple on peut définir une énumération de Couleur, et définir une variable tel que

**Couleur maCouleurPreferer = Couleur.ORANGE ;**

|  |  |  |
| --- | --- | --- |
| Définition | Declaration | Exemple |

Dans l’exemple ci-dessus on parcours l’énumération et on affiche son nom (.name() ) ainsi que son index ( .ordinal() )