# Langage Java – Mémento 1

## Types de données

### Types de donnée de base

| Type de données Java | Type de données Algo / description | Exemple de déclaration de variables | Valeur possible |
|---|---|---|---|
| **boolean** | booleen | **boolean** test = true ; | **true**, **false** |
| **char** | caractere | **char** car = 'a' ; | Toute valeur Unicode[1]<br>'\u0000' à '\uffff' (0 à 65535) |
| **Nombres entiers** | | | |
| **byte** | entier /<br>Nombre entier (8 bits) | **byte** numMois = 12 ; | De -128 à +127<br>$-2^7$ à $2^7-1$ |
| **short** | entier /<br>Idem. (16 bits) | **short** annee = 2008 ; | De -32.768 à +32.767<br>$-2^{15}$ à $2^{15}-1$ |
| **int** | entier /<br>Idem.(32 bits) | **int** distance = 0 ; | De -2.147.483.648 à +2.147.483.647<br>$-2^{31}$ à $2^{31}-1$ |
| **long** | entier /<br>Idem.(64 bits) | **long** gdeDistance = 0 ; | De -9.223.372.036.854.775.808 à +9.223.372.036.854.775.807<br>$-2^{63}$ à $2^{63}-1$ |
| **Nombres réels : valeurs approchées** | | | |
| **float** | reel /<br>Nombre à virgule flottante (32 bits) | **float** f = 3.14159 ; | De $-3{,}40282347*10^{38}$ à $+3{,}40282347*10^{38}$<br>Plus petite valeur positive : $1{,}40239846*10^{-45}$ |
| **double** | reel /<br>Idem. (64 bits) | **double** d = 0 ; | de $-1{,}7976931348623157*10^{308}$ à $+1{,}7976931348623157*10^{308}$<br>Plus petite valeur positive : $4{,}9406564584124654*10^{-324}$ |

### Types évolués : classes

| | | | |
|---|---|---|---|
| **String** | chaine | **String** msg = "hello"; | |

String est une classe Java proposée dans le package java.lang (ce package est inclut automatiquement)

### Constantes

| | | |
|---|---|---|
| **final** | (const) | **final float** PI = 3.14159 ; |

[1] Unicode : jeu de caractères standardisé permettant le codage des caractères de toutes les langues

# Langage Java – Mémento 1

## Affectation(s)

| Type d'opérateur | opérateur | | Exemples |
|---|---|---|---|
| Affectation | `=` | | `i = (i+4)` |
| Affectation composée | `+=, -=, *=` | | `i+=4 ;` équivaut à `i=i+4 ;` |
| Incrémentations | `++`<br>`--` | `incrémentation`<br>`décrémentation` | `++i ;` équivaut à `i=i+1;`<br>`i++` renvoie la valeur de i avant d'ajouter 1 |

## Opérateurs

| Type d'opérateur | opérateur | | Exemples |
|---|---|---|---|
| Arithmétique | `+, -`<br>`*, /, %` | | `(4 + 3)` |
| Comparaison logique | `==`<br>`!=`<br>`<, >`<br>`<=, >=,` | `égal`<br>`différent`<br>`inférieur, supérieur`<br>`inférieur ou égal, supérieur ou égal` | `boolean b = (1 == 2) ;` |
| Connecteurs logiques | `&&`<br>`\|\|`<br>`!` | `ET`<br>`OU`<br>`NON` | `boolean b = ((i==1)&&(j==2)) ;` |
| Ternaire | `? :` | | `int max = (i > j) ? i : j ;`<br>equivalent à :<br>`if (i>j) max = i ;`<br>`else max = j ;` |

## Structure de base d'une classe « programme »

Le code source Java est stocké dans le fichier texte « NomClasse.java » et contient :

```
ici, éventuellement, les packages importés
class NomClasse {
      public static void main(String [] args) {
            ici les déclarations puis les instructions
      } // fin main
} // fin class
```

# Langage Java – Mémento 1

## Entrées clavier/ sortie écran

### Entrée : objet de la classe Scanner associé à l'objet Java System.in

Java propose

- dans un package « java.util » la classe **`Scanner`** dont le rôle est d'analyser un texte pour en extraire des valeurs
- l'objet **`System.in`** qui fait référence à l'entrée standard du système (= clavier)

Pour qu'un programme puisse récupérer les valeurs provenant du clavier :

- importer la classe `Scanner` du package `java.util`

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

- créer un objet (scanner de l'entrée standard du système)

```
Scanner sc = new Scanner(System.in);
```

soit : déclarer la variable `sc` de classe `Scanner`
affecter à cette variable un objet de la classe Scanner associé à System.in

- utiliser cet objet pour toutes les récupérations de valeurs saisies

```
numero = sc.nextInt();
```

soit : demander à l'objet `sc,` la prochaine valeur (*`next`*),
essayer de convertir cette valeur en nombre entier (*`int`*)
puis affecter cette valeur à la variable `numero` (qui a dû être déclarée précédemment)

### Sortie : objet Java System.in

Java propose

- l'objet **`System.out`** qui fait référence à la sortie standard du système (= écran, console)

Pour qu'un programme puisse afficher des valeurs à l'écran :

```
System.out.print("le numéro est "+numero) ;
```

soit :
demander à l'objet `System.out,` d'imprimer (*`print`*) la valeur entre parenthèses

## Structures de contrôle de l'exécution

### Conditionnelles, tests

| | |
|---|---|
| conditionnelle | `if (`valeur logique`) {` . . .*instructions*. . . `;}` |
| avec alternative | `if (`valeur logique`) {` . . .*instructions*. . .`;}`<br>`else {` . . .*instructions*. . .`;}` |
| avec alternatives multiples | `if (`valeur logique1`) {` . . .*instructions*. . .`;}`<br>`else if (`valeur logique2`) {` . . .*instructions*. . .`;}`<br>`else if (`valeur logique3`) {` . . .*instructions*. . .`;}`<br>. . .<br>`else {` . . .*instructions*. . .`;}` |

| | |
|---|---|
| Choix multiple | `switch (`expression_entiere ou char`) {` |

| | |
|---|---|
| | `case valeur1 : ...instructions... break ; default : ...instructions...;` `} // fin switch` |

**Itératives, boucles**

| | |
|---|---|
| Répétitions indéterminées | `while (valeur logique) { ;}` |
| Répétition « déterminée » | `for (init ;valeur logique;increment) { ;}` |
| Répétition indéterminée, au moins 1 fois | `do { ;} while (valeur logique);` |

## Structures de données composées : tableaux

| Vecteurs | Matrices |
|---|---|
| Pour définir un tableau à une dimension<br>• Déclarer une variable tableau<br>`int [] tab ;`<br>• et définir effectivement le tableau avec sa taille :<br>`tab = new int [6];` | Pour définir un tableau à 2 dimensions<br>• Déclarer une variable tableau à 2 dimensions<br>`int [][] mat ;`<br>• et définir effectivement le tableau avec sa taille :<br>`mat = new int [2][6];` |
| Pour définir un vecteur initialisé (6 élements)<br>`int [] tab = {1, 2, 3, 4, 5, 6};` | Pour définir une matrice initialisée (2 fois 6 éléments) :<br>`int [][] tab = { {1,2,3,4,5,6}, {7,8,9,10,11,12} };` |

## Sous-programmes : fonctions et procédures

**Fonction**

```
type idFonction (liste des paramètres) {
 … return valeur ;
}
```

**Procédure**

```
void idProcédure (liste des paramètres) {
 … return ;
}
```

## Annexes

**Annexe1 : mots réservés**

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| abstract | boolean | byte | char | class | const | double | enum | extends | false | final | float | |
| implements | import | int | interface | long | native | new | null | package | private | | | |
| protected | public | short | static | strictfp | super | this | transient | true | void | volatile | | |
| assert | break | case | catch | continue | default | do | else | finally | for | goto | if | instanceof |
| return | switch | synchronized | throw | try | while | | | | | | | |