

Mémo Java

1. Sorties et variables

Contrairement à python, en java la fin d'une ligne se termine toujours par un point-virgule (;)

1. Sorties

Pour permettre au programme en cours d'exécution d'afficher un texte ou un nombre on utilise les fonctions `System.out.println()` ou `System.out.print()` qui prennent 1 seul argument

Exemples :

```
System.out.println("toto");
```

```
Int a=2
```

```
System.out.print("le carré de "+a+ "est "+a*a );
```

```
System.out.println (1);
```

```
System.out.println (2);
```

```
System.out.println (3);
```

Affichage : 1

2

3

```
System.out.print (1);
```

```
System.out.print (2);
```

```
System.out.print (3);
```

Affichage : 1 2 3

2. Les variables

Un programme manipule des données enregistrées dans des variables de différents types.

Une variable est définie par un nom, alors que pour l'ordinateur, il s'agit en fait d'une adresse, c'est-à-dire d'une zone particulière de la mémoire

En Java, il faut **déclarer** une variable avant de pouvoir l'utiliser (note : on peut l'initialiser à la déclaration. Pour déclarer une variable, il faut préciser son **type** et son **nom**.

```
Int I = 2 // déclare et initialise une variable de type integer  
String S = "Bonjour" // déclare et initialise variable de type string.
```

Nous retrouvons les types que nous connaissons à quelques petites différences prêt :

Entier : **int** qui connait 2 déclinaisons qui permettent d'utiliser des valeurs plus grandes ou au contraire plus petites **long** et **short**

Réels : **float** qui comme les entiers peut être utilisé dans sa version plus longue **double**

Chaine de caractères : **String** qui comme en python est à la fois un type primitif et un type complexe, géré comme un objet (d'où la **Majuscule** à ne pas oublier)

Caractère : **char**

Toutes les classes existantes ainsi que celles que nous allons créer seront des types utilisables pour les objets que nous allons utiliser.

Il est possible de convertir une variable d'un type à un autre.

```
I = 2 ; S=(String) i ; // L'entier I est converti en chaine de caractère  
S = "105.3" ; F. = (float) S; // La chaine est convertie en réel
```

3. Les commentaires

L'utilisation des commentaires se fait comme dans tous les langages:

// permet de définir une ligne de commentaire

/* */ permet de définir un bloc de commentaires

```
// ligne commentée  
  
Int a = 5 ;  
  
/*  
  
Tout un bloc commenté  
  
Toujours commenté  
  
*/
```

2. Les conditions

1. Le if

L'instruction if associée à un opérateur de comparaison permet de faire des choix.

Contrairement à python les blocs liés au **if** sont entre accolades


```
if (a%2 == 0) {  
    System.out.println("nombre pair"); }  
  
Else{  
    System.out.println("nombre impair"); }
```

L'instruction **else** est facultative. Il est également possible de combiner des **if** entre eux, pour cela on rajoute **if** apres le **else**, a la suite de qui donne **else if** puis une nouvelle condition.

```
If ( x>0 ) {
    System.out.println (" positif "); }
Else if (x ==0) {           //sinon si
    System.out.println (" égal à zéro "); }
Else{
    System.out.println (" negatif ");
}
```

L'indentation et la mise à la ligne ne servent plus qu'a la lisibilité du code

2. Opérateurs de comparaison

Test en français	Écrit en langage  Python 3
Si <i>n</i> est égal à zéro	if (n==0):
Si <i>n</i> est positif	if (n>0):
Si <i>n</i> est différent de 34	if (n!=34):
Si <i>n</i> est compris strictement entre 0 et 10	if (n>0) and (n<10):
Si <i>n</i> est divisible par 5	if (n%5==0):

Combiner des conditions :

condition1 **&&** condition2 les conditions 1 **et** 2 doivent être vrais pour que le tout soit vrais

condition1 **||** condition2 1 des conditions 1 **ou** 2 doit être vrais pour que le tout soit vrais

! condition condition doit être faux pour que la combinaison soit vrais (insertion)

3. Les boucles

1. La boucle while

```
ind = 0

while (ind <= 10){

    System.out.println (I) ;

    i = i + 1;

} // affiche tous le nombre de 0 à 10
```

2. La boucle do while

```
ind = 0

do{

    System.out.println (I) ;

    i = i + 1;

} while (ind <= 10)

// affiche tous le nombre de 0 à 10
```

3. La boucle for

Nous allons devoir réapprendre cette boucle comparée à python.

Cette boucle nous permet de gérer le compteur mais n'est plus lié à une liste

Un for comprend la déclaration du compteur en 3 parties séparées pas un point-virgule

(déclaration et initialisation ; condition d'arrêt ; pas d'avancement)

```
For (int i =0 ; i<10;i++){

    System.out.println(i) ;

} // Affichera toutes valeurs de de 0 à 9
```

4. Les fonctions ou plutôt les méthodes

Les droit d'accès (encapsulation) (**public, protected ou private**)

Si elles sont des méthodes d'instance ou de classe (mot clef **static** pour des méthodes de classe) par défaut de sont des méthodes d'instance.

Les fonctions comme les variables doivent déclarer leur type de retour. Si elles n'ont pas de retour nous utiliserons le type **void**

Un certain nombre de paramètres comprenant le type et ne nom du paramètre. Chaque paramètre est séparé par une virgule

Une méthode est caractérisée par sa signature : type de retour, nom, nombre et type des paramètres.

```
Public void add(a,b){  
    System.out.println(a+b);} // methode qui affiche un résultat mais ne retourne rien  
Public int add(a,b){  
    Return (a+b);} // la même méthode qui retourne le résultat
```

5. Les classes

Nous aurons dans le cours la définition de ce que son les classes, contentons ici de la syntaxe de leur implémentation :

La déclaration d'une classe comprend ses droits d'accès le mot clef **class** ainsi que son nom. Comme pour tous les autres blocs, son contenu est entre accolades

```
Public class Personne{  
    //attributs  
    Private String nom;  
    Private int age;  
    // methodes  
    Public String getNom(){  
    }  
}
```

Le mot clef **this** permet de faire référence à l'objet courant ex:

```
Public class Personne{  
  
    //attributs  
  
    Private String nom;  
  
    Private int age;  
  
    // methodes  
  
    Public String setNom(String nom){  
  
        This.nom = nom  
  
        // this permet ici de ne pas avoir d'ambigüité entre l'attribut de la classe et le paramètre de la  
        méthode qui ont le même nom  
  
    }  
}
```

6. Les modules

Les modules ou librairies sont des programmes java, (comme en python) qui contiennent des fonctions que l'on est amené à réutiliser souvent.

Pour être utilisable un module doit être importé ;

```
import random                                // on à maintenant accès au module random  
  
random.randint (0, 100)    // utilisation d'une fonction de ce module
```

Les modules en java ne sont ni plus ni moins que des classes. Ils définissent des objets que nous pouvons créer et manipuler.