1ème Année Master – SER- (S1) Module: **Programmation Orientée Objet**

Chargés de cours : **Pr Adil Soufi**

Devoir 01 : Notions fondamentales du langage Java

Objectif :

Le but de ce devoir est d'acquérir les notions fondamentales relatives au langage de programmation java à savoir, les types de données, les structures conditionnelles ainsi que les structures répétitives.

Notes de cours :

Dans ce qui suit, nous allons présenter brièvement les types de données utilisées pour les activités, les structures conditionnelles et répétitives.

1. Types primitifs

Le tableau suivant illustre les types primitifs du langage Java

Type	Classe éq.	Valeurs	Portée	Défaut
`boolean`	`Boolean`	`true` ou `false`	N/A	`false`
`byte`	`Byte`	entier signé	{-128..128}	o
`char`	`Character`	caractère	{\uoooo..\uFFFF}	\uoooo
`short`	`Short`	entier signé	{-32768..32767}	0
`int`	`Integer`	entier signé	{-2147483648..2147483647}	0
`long`	`Long`	entier signé	$\{-2^{31}..2^{31}-1\}$	0
`float`	`Float`	réel signé	$\{-3,4028234^{38}..3,4028234^{38}\}$ $\{-1,40239846^{-45}..1,40239846^{-45}\}$	0.0
`double`	`Double`	réel signé	$\{-1,797693134^{308}..1,797693134^{308}\}$ $\{-4,94065645^{-324}..4,94065645^{-324}\}$	0.0

Java est un langage très rigoureux sur le typage des données. Il est interdit d'affecter à une variable la valeur d'une variable d'un type différent si cette seconde variable n'est pas explicitement transformée. (Voir exemple).

`int a ;`
`double b = 5.0 ;`
`a = b ;`
	est interdit et doit être écrit de la manière suivante :	`int a ;`
		`double b = 5.0 ;`
		`a = (int)b ;`

Pour calculer le reste d'une division de a par b on utilise l'instruction : reste =a%b ; L'instruction '' = '' sert à affecter une valeur à une variable Tandis que l'instruction '' == '' sert à comparer deux valeurs.

2. Tableaux et matrices

Les deux syntaxes suivantes sont acceptées pour déclarer un tableau d'entiers :

```
int[] mon_tableau ;
int mon_tableau2[];
```

Un tableau a toujours une taille fixe [2] qui doit être précisée avant l'affectation de valeurs à ses indices, de la manière suivante : `int[] mon_tableau = new int[20];`

De plus, la taille de ce tableau est disponible dans une variable length appartenant au tableau et accessible par ***mon_tableau.length***. On peut également créer des matrices ou des tableaux à plusieurs dimensions en multipliant les crochets (ex : `int[][] ma_matrice;`). On accède aux éléments d'un tableau en précisant un indice entre crochets (`mon_tableau[3]` est le quatrième entier du tableau) et un tableau de taille n stocke ses éléments à des indices allant de *0 à n-1*.

3. Chaînes de caractères

Les chaînes de caractères ne sont pas considérées en Java comme un type primitif ou comme un tableau. On utilise une classe particulière, nommée String, fournie dans le package java.lang. On peut utiliser l'opérateur « + » pour concaténer deux chaînes de caractères :

```
String s1 = "hello" ;
String s2 = "world" ;          Après ces instructions s3 vaut "hello world"
String s3 = s1 + " " + s2 ;
```

L'initialisation d'une chaîne de caractères s'écrit :

```
String s = new String();  // pour une chaine vide
String s2 = new String("hello world");  // pour une chaîne de valeur "hello world"
```

4. Structures conditionnelles

Les mots-clés java correspondant aux structures conditionnelles sont if, else, else if et switch.

4.1. *L'instruction if* : permet d'exécuter une série d'instructions si une condition est réalisée.

Sa syntaxe est : *if (condition réalisée) {liste d'instructions}*

- La condition doit être entre des parenthèses
- Il est possible de définir plusieurs conditions à remplir avec les opérateurs ET et OU (&& et ||) par exemple l'instruction suivante teste si les deux conditions sont vraies :

if ((condition1)&&(condition2))

L'instruction suivante exécutera les instructions si l'une ou l'autre des deux conditions est vraie : *if ((condition1)||(condition2))*

- S'il n'y a qu'une instruction, les accolades ne sont pas nécessaires...
- Les instructions situées dans le bloc qui suit ***if*** sont les instructions qui seront exécutées si la ou les conditions ne sont pas remplies

4.2. *L'instruction if ... else*

L'expression *if ... else* permet d'exécuter une autre série d'instruction en cas de non-réalisation de la condition. Sa syntaxe est : *if (condition réalisée) {liste d'instructions }*
else {autre série d'instructions }

```
public class Conditionnelle1
 {
 public static void main (String [] args)
   {   int a = 21, b = 22;
```

```
    if(a < b)  {System.out.println("a est moins grand que b");}
    else  {System.out.println("a est plus grand que b");}
 } }
```

4.3. *L'instruction switch*

Elle permet de faire plusieurs tests de valeurs sur le contenu d'une même variable. Ce branchement conditionnel simplifie beaucoup le test de plusieurs valeurs d'une variable, car cette opération aurait été compliquée (mais possible) avec des *if* imbriqués. Sa syntaxe est la suivante :

```
switch (Variable) {
case Valeur1 :
        Liste d'instructions
        break;
case Valeur2 :
        Liste d'instructions
        break;
case Valeurs... :
        Liste d'instructions
        break;
default:
        Liste d'instructions
        break;
}
```

5. Structures répétitives :

Les boucles sont des structures qui permettent d'exécuter plusieurs fois la même série d'instructions jusqu'à ce qu'une condition ne soit plus réalisée.

5.1. *La boucle for*

Dans la syntaxe de for, il suffit de préciser le nom de la variable qui sert de compteur (et éventuellement sa valeur de départ, la condition sur la variable pour laquelle la boucle s'arrête et enfin une instruction qui incrémente (ou décrémente) le compteur.

La syntaxe de cette expression est la suivante :

```
for (compteur; condition; modification du compteur)
{ Liste d'instructions }
```

Exemple: *for (int i=1; i<6; i++) {System.out.println((char)i);}.* Cette boucle affiche 5 fois la valeur de *i*, c'est-à-dire 1, 2, 3, 4,5. Elle commence à i=1, vérifie que i est bien inférieur à 6, etc... jusqu'à atteindre la valeur i=6, pour laquelle la condition ne sera plus réalisée, la boucle s'interrompra et le programme continuera son cours. D'autre part, Java autorise la déclaration de la variable de boucle dans l'instruction *for* elle-même.

5.2. *L'instruction while*

L'exécution de cette instruction suit les étapes suivantes :

1. La condition (qui doit renvoyer une valeur booléenne) est évaluée. Si celle-ci est vraie on passe à l'étape 2, sinon on passe à l'étape 4 ;
2. Le bloc est exécuté ;
3. Retour à l'étape 1 ;
4. La boucle est terminée et le programme continue son exécution en interprétant les instructionnsuivant le bloc.

Syntaxe : while (<condition>) <bloc>

Exemple : *while (a != b) a++;*

5.3. *La boucle do..while*

L'exécution de cette instruction suit les étapes suivantes :

1. Le bloc est exécuté ;
2. La condition (qui doit renvoyer une valeur booléenne) est évaluée. Si celle-ci est vraie on retourne à l'étape 1, sinon on passe à l'étape 3 ;
3. La boucle est terminée et le programme continue son exécution en interprétant les instruction suivant le bloc.

Syntaxe : do <bloc> while (<condition>);

Exemple : **do** a++ **while** (a != b);

Activité 01 :

1. Prédire ce que fait le morceau de programme suivant.
2. Placez ce code dans un programme java et exécutez-le sur machine.
3. Réécrire ce code en remplaçant la boucle while avec un for.

```
int i,v;
i=0;
v=0;
while(i<10){
v=v+i;
i=i+1
}
System.out.print(v) ;
```

Activité 02 :

Ecrire un programme en langage java qui affiche les informations fournies par les méthodes suivantes (N est saisi au clavier) :

- Une méthode qui calcule la somme des N premiers entiers pairs (exemple : pour N= 5, on aura 2+4+6+8+10=30)
- Une méthode qui calcule qui affiche les dix nombres suivants le nombre N. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

Activité 03 :

Ecrire un programme en langage java qui calcule la somme des éléments impairs d'un tableau de 20 entiers.