Université de Mila
Institut des sciences et de la technologie
Informatique- Semestre 5

Génie logiciel
Série de TP N°01

TP N°01 : Initiation à Java

I. Création d'un projet sur Eclipse

- Créez un nouveau projet Java. Pour cela, appuyez sur **File → New → Java Project**.
- Tapez comme indiqué le nom de votre projet. Conservez les réglages par défaut, et appuyez sur Finish. Dans l'espace de travail, sous **l'onglet Packages**, vous verrez que le projet est créé, et qu'il contient déjà un répertoire **src** (qui doit contenir tous les fichiers source que vous créez), et **JRE System Library**, qui est utilisée pour compiler votre code.
- Pour créer un package sous le répertoire **src**, cliquer sur celui-ci, puis sur l'icône  ou clic-droit sur **src**, et **choisir New -> Package**. Choisissez un nom pour le package.
- **Pour créer une classe :**
  - Cliquer sur le package qui doit contenir la classe puis sur l'icône  ou clic-droit sur le package, et choisir **New -> Class**. Dans la fenêtre qui apparaît, choisissez le nom de la classe.
  - Si la classe n'est pas définie dans un package (ce qui est déconseillé), refaites l'opération ci-dessus à partir du répertoire **src**.
  - Vous pouvez générer automatiquement la méthode main en cliquant sur la case public **static void main(String[] args)**.
- La classe générée apparaîtra sous le package que vous avez choisi. Double-cliquez dessus pour modifier son code dans la partie édition. Vous verrez qu'un squelette de la classe vous est proposé. Vous n'aurez qu'à terminer le reste du code.
- Avec Eclipse, vous n'avez pas besoin de compiler explicitement votre code : la compilation se fait en temps réel.
- De plus, les erreurs de syntaxe seront affichées pendant l'écriture du code, avec des propositions de corrections.

II.2. Nouveau projet : Exemple

II.2.1. Version simplifié

- Créer un nouveau projet **Helloworld** comme indiqué dans la partie précédente. Créer un package nommé **helloPack**, contenant une classe **Helloworld** qui contient une méthode main.
- Dans la méthode main, écrire **"System.out.println("Bonjour!");"**

- Exécuter votre programme en cliquant directement sur l'icône  . L'affichage apparaîtra dans la partie inférieure, sous l'onglet Console.

III. Rappel sur les principes de bases du langage Java

- **Déclaration de deux entiers et calcul de la division et le reste de la division**

```
int valA = 7;
int valB = 2;
int valC = valA / valB;
int valD = valA % valB;
System.out.println("la valeur de C = " + valC);
System.out.println("la valeur de D = " + valD);
```

- **Déclaration de deux nombres réels et calcul de la division**

```
double reelA = 7;
double reelB = 2;
double division = reelA / reelB;
System.out.println("la valeur de la division = " + division);
```

- **Condition if …else**

```
int temperature = 52;
if (temperature > 50) {
    System.out.println("La température est supérieure à 50°");
} else {
    System.out.println("La température est inférieure ou égale à 50°");
}
```

- **Boucle For**

```
for (int celsius2 = 0; celsius2 < 10; celsius2 = celsius2 + 1) {
    double fahrenheit2 = ((9.0 / 5.0) * celsius2) + 32.0;
    System.out.print(celsius2);
    System.out.print(" degres Celsius2 convertit en Fahrenheit2 vaut ");
    System.out.println(fahrenheit2);
}
```

Switch…case

```
int age = 70;
switch (age) {
    case 18:
        System.out.println("L'age est légal pour voter");
        break;
    case 40:
        System.out.println("L'age est légal pour être candidat à la présidence");
        break;
    case 60:
        System.out.println("L'age de la retraite");
        break;
    default:
        System.out.println("L'age n'est pas pris en considération");
}
```

- **Boucle While**

```
while (celsius3 < 10) {
    double fahrenheit3 = ((9.0 / 5.0) * celsius3) + 32.0;
    System.out.print(celsius3);
    System.out.print(" degres Celsius3 convertit en Fahrenheit3 vaut ");
    System.out.println(fahrenheit3);
    celsius3 = celsius3 + 1;
}
```

- **Méthodes propres**

```
static int addition(int x, int y) {//x et y sont les paramètres formels
    return x + y;
}

int s = 47;
int v = 56;
int somme = addition(s, v);  //s et v sont les paramètres effectifs de l'appel x
System.out.println("La somme de " + s + " et " + v + " est égale à : " + somme);
```

- **Les tableaux à une dimension**

```
char[] tabChar; // Déclaration d'un tableau de caractères
int[] tabInt; // Déclaration d'un tableau d'entiers
tabInt = new int[10]; // Instanciation d'un tableau de 10 entiers
tabInt[0] = 5;
int[] tabCinq = {12, 33, 44, 0, 50}; //Initialisation expresse
int j = 0;
int valeur1 = tabCinq[j]; //Renvoie 12, l'élément d'indice 0
int valeur2 = tabCinq[4]; //Renvoie 50, l'élément d'indice 4
System.out.println("le 1er élement du tableau tabInt est " + tabInt[0]);
System.out.println("le 1er élement du tableau est " + valeur1);
System.out.println("l'élement dont l'indice est 4 est " + valeur2);

int taille = tabCinq.length; //Renvoie 5, le nombre de cases du tableau tabCinq
System.out.println("la taille du tableau tabcinq est " + taille);
```

- **Cast (conversion) de type de variables**

```
// les conversions de type de variables
// du réel vers entiers
double x = 2;
int i = (int) (x * 42.3); // i vaut 84
System.out.println("la valeur de i = " + i);

// du caractère vers entier
char d = 'b';
int p = (int) d;
System.out.println("Le caractère " + d + " est codé en machine sur l'entier " + p);

// de l'entier vers caractère
int v = 99;
char w = (char) v;
System.out.println("L'entier " + v + " en machine correspond au caractère " + w);
```

```
// Conversion d un type primitif vers chaine de caractères
int val = 22;
String chaine = String.valueOf(val); // chaine contient "22"
System.out.println(chaine);

// Conversion d une chaine de caractères vers un entier
String chaine2 = "22";
int val2 = Integer.parseInt(chaine2); //valeur vaut 22
System.out.println(val2);
```

IV. Programmation orienté objet

IV.1. Classes et objets

- Une classe commence toujours par une **majuscule**, tous les autres identifiants commencent en **minuscule** (les objets, les méthodes, les types primitifs, les variables…)

Si un identifiant est composé de plusieurs mots, ces mots sont collés l'un à la suite de l'autre avec une majuscule pour chaque début de mot, mais en respectant la convention 1. Par exemple, pour une classe : **NomDeClasse**, et pour un objet **nomDeLObjet**. **« public class NomDeClasse { } »**

- Un constructeur est une méthode spécifique appelée automatiquement lors de l'instanciation d'un objet. Le constructeur a les particularités suivantes :
  - **C'est la seule méthode qui porte le nom de la classe**
  - **C'est la seule méthode qui ne définit pas de variable de retour**
  - **C'est la seule méthode dont le nom commence par une majuscule (puisque c'est le nom de la classe) « public NomDeClasse (x, y, …) {initialisation des attributs} »**
- Les attributs représentent l'état interne d'un objet. Il est déclaré de la façon suivante dans le corps de la classe : **[portée] [type] [identifiant];**
- Les méthodes permettent de définir le comportement des objets. Une méthode est définie pas sa **signature** qui spécifie sa portée, son type de retour, son nom et ses paramètres entre parenthèses. La signature est suivie d'un bloc de code que l'on appelle le **corps** de méthode. Il est déclaré de la façon suivante : **[portée] [type de retour] [identifiant] ([liste des paramètres]) { [code]}**

Il est possible de limiter la visibilité des attributs et méthodes définies à l'intérieure des classes.

- **public** : pas de limitation
- **protected** : visible depuis l'intérieur de la classe et depuis classes qui en héritent
- **private** : visible uniquement depuis intérieur de la classe

- Une méthode d'accesseur est utilisée pour renvoyer la valeur d'un champ privé. Il suit un schéma de dénomination préfixant le mot «get» au début du nom de la méthode. Ces méthodes renvoient toujours le même type de données que leur champ privé correspondant (par exemple, String), puis renvoient simplement la valeur de ce champ privé.
- Une méthode mutator est utilisée pour définir la valeur d'un champ privé. Il suit un schéma de dénomination préfixant le mot «set» au début du nom de la méthode. Ces méthodes n'ont pas de type de retour et acceptent un paramètre qui est du même type de données que leur champ privé correspondant. Le paramètre est ensuite utilisé pour définir la valeur de ce champ privé.

Exemple : la classe personne

```
package univ;
 public class Personne {
   private String nom;
   private String prenom;
   public Personne(final String n, final String p) {
      nom = n; prenom = p;  }
public String getNom() {
 return nom; }
public void setNom(String nom) {
this.nom = nom; }
public String toString(){
      return "Personne    " + nom + ",    "+ prenom ;
 }
```

Exercice :

Vous devez créer une classe java portant le nom "Personne"
regroupant les attributs "nom" et "age". Cette classe propose également un constructeur (qui va initialiser les attributs avec les paramètres passés) et les méthodes getNom, getAge, setNom, et setAge.
2. Vous devez implémenter une second classe PersonneTest qui va créer deux instances de la classe Personne : Mohammed qui a 23 ans et Hicham qui a 34 ans.
3. Vous devez enrichir la classe PersonneTest pour permettre :
- l'affichage du nom de la première instance de Personne (p1).
- l'affichage du nom de la seconde instance de Personne (p2).
- l'affichage des âges de p1 et p2.
- modifier l'âge de p1 à 24 ans.
- l'affichage de l'âge de p1.
- modifier le nom de p2 en "Oussama".
- l'affichage du nom de p2.
4. On veut maintenant gérer un grand nombre de personnes. Pour cela, on va définir un tableau du type de la classe Personne de taille 5. Vous devez enrichir le tableau avec les données suivantes :
- Mohammed 24 ans, Hicham 25 ans, Oussama 33 ans, Karim 28 ans et Fateh 18 ans.
Vous devez programmer l'affichage de l'ensemble des données des instances (nom et âge), modifier le prénom de "Karim" en "Meriem" ainsi que modifier son âge à 21 ans.
5. Faire en sorte que l'on puisse supprimer et ajouter une personne dans le tableau de la question.