TP n°6 : retour sur les bases, vecteurs, templates et fichiers

1. Retour sur les bases du JAVA

Cet exercice va vous permettre de réviser vos acquis JAVA et revient sur ce que vous devez avoir compris pour poursuivre votre apprentissage du langage. Vous devez savoir tout écrire, hormis le dernier point.

- Créer une **classe** Principale dans laquelle vous implémentez une **méthode** main ;
- créer une classe Personne comportant 3 **attributs** : un attribut nom (de type String), un attribut prenom (de type String), et un attribut naissance (de type int) ;
- créer un **constructeur par défaut** à la classe Personne permettant d'initialiser les attributs de la classe à des valeurs par défaut;
- créer un second **constructeur** à la classe Personne qui prend 3 **paramètres** correspondant aux attributs de votre classe. Faire en sorte que les attributs prennent les valeurs des paramètres ;
- créer des méthodes « **getter** » et « **setter** » permettant de modifier et d'accéder aux attributs de la classe ;
- créer une méthode **toString** dans votre classe vous permettant d'afficher les 3 attributs ;
- tester la construction (ou **instancier**) de deux objets de type Personne dans votre méthode main. Un utilise le constructeur par défaut et l'autre le constructeur avec 3 paramètres de votre choix. Faites afficher la valeur des attributs de vos 2 objets ;
- créer une classe Etudiant **héritant** de la classe Personne. Ajouter un attribut formation (de type String), des méthodes getter et setter pour cet attribut, puis un constructeur prenant 4 paramètres : les 3 paramètres de nom, prenom, et naissance et un String permettant d'initialiser l'attribut formation ;
- instancier un objet e1 de type Etudiant dans votre méthode main et faites afficher la valeur des attributs de cet Etudiant ;
- instancier et initialiser un premier objet p1 de type Personne puis un nouvel objet Personne p2 de la façon suivante : `Personne p2 = p1;` Faire afficher p1 et p2. Changer la valeur d'un attribut de p1 et faire de nouveau afficher p1 et p2. Que remarquez-vous ?
- créer un **constructeur par copie** pour la classe Personne pour résoudre le problème précédent. Un constructeur par copie est un constructeur bien spécifique : il prend en paramètre un objet dont le type est celui de la classe où il est défini. Il s'appelle de la façon suivante : `Personne p2 = new Personne(p1);`

2. Utilisation de vecteurs

Supposons que l'on souhaite créer plusieurs objets de type Personne et qu'on veuille les stocker en mémoire. La première solution la plus naïve consiste à créer un tableau comportant n lignes (le nombre de personnes) et n colonnes (le nombre d'attributs à stocker pour chaque personne, dans ce TP ce seront donc 3 attributs.

Par exemple :

```
String tableau_personnes1[][] = new String[100][3];
```

Ne connaissant pas le nombre de personnes à l'avance, on est obligé d'instancier en mémoire un tableau avec une taille maximale (100 dans l'exemple). De plus, notre tableau est un tableau de String. Pour récupérer la valeur de l'âge (de type int) d'une personne, on doit effectuer une conversion de String en int. Cette première solution est très contraignante et pas du tout pratique à l'utilisation !

Une seconde solution consiste à créer un tableau de Personne.

```
Personne tableau_personnes2[] = new Personne[100];
```

On a plus le problème de conversion évoqué précédemment puisqu'on stocke cette fois des objets de type Personne. Cette solution, bien que plus élégante que la précédente, garde cependant l'inconvénient qu'une taille d'initialisation du tableau doit être spécifiée. Si par malheur on arrive a 101 personnes, le tableau n'est plus suffisant comme solution de stockage.

Heureusement, une troisième solution existe ! La **classe Vector du package java.util** est une classe permettant de gérer ce cas de situation très souvent rencontré. Elle permet de stocker des objets (quelconques) dans un tableau dont la taille évolue avec les besoins.

Créez un vecteur qui va contenir des objets de type Personne dans votre méthode main :

```
Vector<Personne> tableau_personnes3 = new Vector<Personne>();
```

Puis ajoutez à ce vecteur un objet de type personne (appelée p dans l'exemple suivant) précédemment instancié grâce à la méthode add

```
tableau_personnes3.add(p);
```

Pour « récupérer » un objet du tableau suivant son index (0 dans l'exemple suivant), vous ferez simplement :

```
Personne p = tableau_personnes3.elementAt(0);
```

Testez l'utilisation d'un vecteur dans lequel vous stockez au moins 3 personnes. Une fois le vecteur rempli, parcourez-le et faites afficher les attributs des étudiants stockés dans le vecteur (indication : la méthode size() appliquée à un vecteur retourne la taille de ce vecteur).

Question subsidiaire : pouvez-vous stocker dans votre vecteur de Personne des objets de type Etudiant ? Pourquoi ?

3. Une classe générique, indrduction à la notion de *Template*

Les vecteurs font partie des classes en JAVA qui implémentent l'interface ***Collection***. Cette interface du package java.util propose de gérer de façon uniforme des vecteurs dynamiques, des ensembles, des listes chaînées, des tables associatives… Cette classe utilise aussi le concept de **classe générique** : on peut stocker n'importe quel type d'objet dans un vecteur (des int, des String, des objets définis par vos propres classes…)

Prenons l'exemple d'une nouvelle classe Binome :

```
public class Binome<T> {
   private T b1, b2;
   public Binome(T b1, T b2) {
      this.b1 = b1;
      this.b2 = b2;
  }
   public T getPremier() {
      return b1;
   }
   public T getSecond() {
      return b2;
   }
   public String toString() {
      return b1 + " -- " + b2;
   }
}
```

On peut créer un nouvel objet de type Binome dans notre méthode main qui contiendra 2 objets de type Personne.

```
Binome<Personne> bin = new Binome<Personne>(p1 , p2);
```

Ou, puisque nous avons une classe Etudiant héritant de Personne, on peut créer un binome composé d'une personne et d'un étudiant. Mais on peut aussi créer un objet Binome qui contiendra 2 String ou 2 int ! Ce mécanisme peut s'avérer très pratique pour l'extensibilité future de votre programme : vous pourrez directement utiliser cette classe générique pour stocker de nouveaux objets dans un objet de type Binome, par exemple des « Stagiaires » ou des « Familles ».

4. Lecture de données dans un fichier

Créer une méthode permettant de lire des données de Personne formatées dans un fichier texte et de les stocker dans un vecteur. Vous pourrez utiliser les classes InputStream, InputStreamReader, BufferedReader pour lire les données du fichier. La classe StringTokenizer est aussi intéressante pour séparer les champs de données à lire.