TD n°1

Rappels Java et introduction à la POO (programmation orientée objet)

Exercice 1 *Rappel sur la structure d'un programme Java*
Voici le contenu du fichier `Bonjour.java` :

```
import fr.jussieu.script.Deug;  //importation de la classe Deug
                                //du package fr.jussieu.script
public class Bonjour{
  /*Fonction principale :
    point de depart de l'execution du programme*/

  public static void main(String args[]){
    String formulePolitesse = "Bonjour";
    System.out.print(formulePolitesse);
    if (args.length==0){
      System.out.println();
    }
    else{
      System.out.println(" "+args[0]);
    }
  }
}
```

1. Que fait ce programme ?
2. Étude de la structure et de la syntaxe du programme :
   - A quoi sert `import` ? Pourquoi n'est-il pas nécessaire dans le cas présent ?
   - Peut-on donner un autre nom à la classe publique `Bonjour` si le nom du fichier reste `Bonjour.java` ?
   - A quoi sert la méthode `main` ?
   - Comment écrit-on les commentaires ?
   - Où faut-il placer des point-virgules ?
   - A quoi servent les accolades ?
   - Quelles sont les conventions de style (indentation et position des accolades) ?
   - A quoi servent les parenthèses après `print` et `println`. Pourquoi faut-il les écrire même dans la ligne 11 ?
   - Quelle est la différence entre `=` et `==` ?
3. Quelles sont les deux étapes nécessaires pour exécuter ce programme Java ? Qu'est-ce que le bytecode ? Qu'est-ce que la machine virtuelle Java ?
4. Qu'est-ce que contient le tableau `args` ?
   Quels sont les résultats des commandes suivantes : `java Bonjour` et `java Bonjour Henri` ? Expliquer.

Exercice 2 *Structure d'une classe Java*
Voici le contenu du fichier `Personne.java` :

```
public class Personne{
  public String nom;
  public String titre;

  public Personne(String leNom, String leTitre){
    this.nom=leNom;
    this.titre=leTitre;
  }

  public void presenteToi(){
    System.out.println("Je m'appelle " + this.nom + ".");
  }

  public void disBonjour(Personne interlocuteur){
    System.out.println("Bonjour, " + interlocuteur.titre + " "
                                    + interlocuteur.nom + ".");
  }

  public void disAuRevoir(){
    System.out.println("Au revoir !");
  }
}
```

Et voici le contenu du fichier `Test.java` :

```
public class Test{
  public static void main(String args[]){
    Personne quidam, autreQuidam, encoreUnAutreQuidam;

    quidam = new Personne("Durand","Monsieur");
    autreQuidam = new Personne("Dupont","Madame");

    quidam.disBonjour(autreQuidam);
    quidam.presenteToi();

    autreQuidam.disBonjour(quidam);
    autreQuidam.presenteToi();

    autreQuidam.disAuRevoir();
    quidam.disAuRevoir();
  }
}
```

L'exécution du programme affiche :

```
Bonjour, Madame Dupont.
Je m'appelle Durand.
Bonjour, Monsieur Durand.
Je m'appelle Dupont.
Au revoir !
Au revoir !
```

1. Quelle est le type des variables `quidam`, `autreQuidam`, `encoreUnAutreQuidam` ?

2. Qu'est-ce qu'une instance de classe ? Combien d'instances de la classe `Personne` a-t-on créées dans la méthode `main` ? Peut-on ajouter l'instruction `encoreUnAutreQuidam.presenteToi();` ?

3. Quelles sont les propriétés des instances de la classe `Personne` ? Quel est le rôle des propriétés ? Imaginer d'autres propriétés d'instances pour cette classe ?

4. Repérer le constructeur de la classe `Personne` ? Quel est son rôle ?

5. Quelles sont les méthodes de la classe `Personne` ? Quel est le rôle des méthodes ? Imaginer d'autres méthodes d'instances pour cette classe.

6. A quoi sert le mot clé `this` ?
   Qu'est ce qui change à l'exécution si on remplace `interlocuteur.nom` par `this.nom` dans la méthode `disBonjour` ?

Remarque à propos de la casse *: noter les règles de syntaxe pour les identificateurs de classes. Noter les conventions pour les identificateurs de propriétés et de méthodes.*

Qu'est-ce qu'un objet ?
Les instances de classe correspondent aux objets du monde réels, on dit d'ailleurs souvent objet à la place d'instance. Une classe est donc un ensemble d'objets de même type. Les objets d'une même classe ont les mêmes types de caractéristiques (les mêmes propriétés d'instances) et les mêmes types de comportements (les mêmes méthodes d'instances). Seules les valeurs des propriétés changent d'un objet à l'autre.

Exercice 3 *Ensembles d'entiers*

Il s'agit d'écrire une classe `EnsembleDEntiers` qui implémente des ensembles d'entiers en utilisant une propriété de type tableau.

1. Écrire une classe `EnsembleDEntiers` avec une propriété privée `elements` de type tableau d'entiers.

2. Écrire un constructeur avec pour paramètre un tableau d'entiers. Pour simplifier, on se limite dans un premier temps à des ensembles de cardinaux toujours égaux à 10.
   Le constructeur doit d'abord instancier le tableau `elements` avec une taille 10. Puis il doit recopier les valeurs contenues dans le tableau transmis en paramètre dans le tableau `elements`.
   Remarque *: on suppose que le tableau transmis en paramètre a une taille égale à 10.*

3. Écrire une méthode `aPourElement` qui reçoit comme paramètre un entier et retourne un booléen qui vaut vrai si le paramètre appartient à l'instance qui exécute la méthode, faux sinon.

4. Écrire la méthode `calculerMoyenne` qui retourne la moyenne des éléments de l'ensemble. On spécifiera le type `double` pour la valeur de retour.

5. Écrire les méthodes `calculerMaximum` et `calculerMinimum` qui retournent respectivement le maximum et le minimum ?

6. Ecrire une méthode `toString` qui retourne une chaîne de caractères contenant la liste des éléments de l'ensemble, sa moyenne, son maximum et son minimum.

7. Écrire une classe `Test` contenant la méthode `main`. Dans cette méthode, instancier un ensemble et afficher-le ainsi que sa moyenne, son maximum et son minimum puis tester si l'entier `10` appartient à cette instance. Quels appels de méthodes sont nécessaires ?

8. On veut maintenant utiliser des ensembles de cardinaux différents d'une instance à une autre. Pour cela, on ajoute une propriété d'instance `cardinal`. Peut-on autoriser l'écriture sur cette propriété ? Écrire l'accesseur `getCardinal`.

9. La taille du tableau `elements` est fixée par une constante de classe publique `CARDINAL_MAXIMUM` de valeur `100`. Noter qu'il s'agit bien du cardinal maximum d'une instance de la classe `EnsembleDEntiers`.

   Pour spécifier qu'une propriété est une constante de classe publique, on utilise la suite de mots clés `public static final`.

   La déclaration de `CARDINAL_MAXIMUM` est donc :
   `public static final CARDINAL_MAXIMUM = 100`

   Propriété de classe : *Une propriété de classe possède la même valeur pour toutes les instances de cette classe contrairement aux propriétés d'instance dont la valeur est différente d'une instance à une autre. On utilise le mot clé* `static` *pour spécifier qu'une propriété est une propriété de classe.*

   Constante : *Le mot clé* `final` *spécifie qu'une propriété est constante, c'est-à-dire qu'elle n'est pas modifiable.*

   Convention : *les identificateurs des constantes de classe sont toujours en MAJUSCULES.*

10. Réécrire le constructeur pour tenir compte de ces modifications. Penser à initialiser la propriété `cardinal`.
    Remarque *: Si le tableau transmis en paramètre a une taille supérieure à* `CARDINAL_MAXIMUM`, *on tronque les éléments restants et on initialise* `cardinal` *à* `CARDINAL_MAXIMUM`.

11. Quelles modifications faut-il apporter au code des méthodes `calculerMoyenne`, `calculerMaximum`, `calculerMinimum`, `aPourElement` et `toString` ?

12. Écrire la méthode `ajouterElement` qui ajoute à l'ensemble un entier passé en paramètre.