Object référence, égalité, String, StringBuilder

Exercice 1 - Assignation, égalité, référence

Qu'affiche le code suivant :

```
String s1="toto";
String s2=s1;
String s3=new String(s1);

System.out.println(s1==s2);
System.out.println(s1==s3);
```

Expliquer

Quelle est la méthode à utilisr si l'on veux tester le contenu des chaînes de caractère ?

Qu'affiche le code suivant :

```
  private static boolean test(String u,String v) {
    return u==v;
  }
  public static void main(String[] args) {
    String s1="toto";
String s2=s1;
    System.out.println(test(s1,s2));
  }
```

Expliquer

Qu'affiche le code suivant :

```
  String s1="toto";
  String s2="toto";

  System.out.println(s1==s2);
```

Expliquer.

Exercice 2 - Point

On cherche à écrire une classe `Point` stockant un point graphique en coordonnées cartésiennes (appelons les `x` et `y`).

1 Déclarer une classe `Point` contenant les deux champs privés `x` et `y`.
Puis essayer le code suivant dans le `main` de la classe `Point`.

```
    Point p=new Point();
    System.out.println(p.x+" "+p.y);
```

Expliquer.

2 Créer une classe `Main` et déplacer le `main` de `Point` dans la classe `Main`.
Quel est le problème ? Comment peut-on le corriger ?

3 Pourquoi est-ce qu'il ne faut pas obligatoirement mettre des accesseurs lorsque l'on crée une classe ?

4 Ajouter un constructeur initialisant les coordonés du point avec deux paramètres (appelons les `px` et `py`)
Indiquer pourquoi il est possible de déclarer les champs `x` et `y` `final`.

Quel est le problème avec le code de test.

5 Écrire un autre constructeur permettant d'initialiser un point sans passer de paramètre. Le point aura pour coordonnées (0,0).
Écrire un troisième constructeur qui prend un point en paramètre et et utilise les coordonnées de celui-ci pour s'initialiser.

Exercice 3 - Test d'égalité

En utilisant la classe `Point` de l'exercice précédent. Qu'affiche le code ci-dessous :

```
Point p=new Point(1,2);
Point p2=p;
Point p3=new Point(1,2);

System.out.println(p1==p2);
System.out.println(p1==p3);
```

1 Écrire dans la classe `Point` une méthode `isSameAs()` (à vous de trouver la signature exacte de la méthode) qui renvoie `true` si deux points ont les mêmes coordonnées.

2 La classe `java.util.ArrayList` correspond à un tableau qui s'agrandi dynamiquement.
Executez le code suivant :

```
public static void main(String[] args){
  Point p=new Point(1,2);
  Point p2=p;
  Point p3=new Point(1,2);
         ArrayList list = new ArrayList();

         list.add(p);
         System.out.println(list.indexOf(p2));
         System.out.println(list.indexOf(p3));
}
```

Expliquer le résultat.
Note : ici, le compilateur génère un warning nous verrons dans les prochains TD comment l'éviter.

3 Quel méthode de `Point` est appelée par `ArrayList.indexOf` ?
Lire la doc !!!

4 Modifier la classe `Point` pour que `indexOf()` test suivant le contenu et pas les références.

5 Utiliser l'annotation `@Override` pour vérifier que vous avez bien typé votre méthode.

Exercice 4 - Conversion de String en entier

On souhaite écrire un programme affichant la somme d'entiers pris en paramètre sur la ligne de commande.
Voici un exemple d'exécution :

```
$ java sum 15 5 231
entiers: 15 5 231
sum: 251
```

Ce programme est décomposé en plusieurs fonctions :

1 Écrire une méthode qui prend un tableau de chaîne de caractère en entrée et renvoie

un tableau d'entiers de même taille contenant les entiers issus des chaînes de caractères.
La méthode statique `parseInt(String s)` de la classe `java.lang.Integer` permet de récuperer la valeur d'un entier stockée dans un chaîne de caractères.

2 Que se passe-t'il lorsqu'un mot est pris en argument n'est pas un nombre ?

3 Écrire une méthode qui prend un tableau d'entiers en entrée et renvoie la somme de ceux-ci.

4 Écrire la méthode `main` qui utilise les deux méthodes précedentes pour afficher le tableau d'entier ainsi que sa somme.
Il y a un petit piège pour afficher les tableaux. (regarder la classe `java.util.Arrays`).

Exercice 5 - En morse. Stop.

Écrire une classe `Morse` qui permet lors de son exécution d'afficher les chaînes de caractères prisent en argument séparé par des `"Stop."`.

```
$ java Morse ceci est drole
ceci Stop. est Stop. drole Stop.
```

1 Utiliser dans un premier temps, l'opérateur + qui permet la concaténation de chaînes de caractères.

2 Qu'affiche le code suivant :

```
String s="hello";
s.concat("olleh");
System.out.println(s);
```

Expliquer

3 Utiliser la classe `java.lang.StringBuilder` et sa méthode `append(String)` pour ré-écrire la classe `Morse`.
Quelle est l'avantage par rapport à la solution précédente ?

4 Recopier le code suivant dans une classe `Test` :

```
  public static void main(String[] args) {
    String first=args[0];
    String second=args[1];
    String last=args[2];
  System.out.println(first+' '+second+' '+last);
}
```

Compiler le code puis utilise la commande `javap` pour afficher l'assembleur généré

```
javap -c Test
```

Que pouvez vous en déduire ?
Dans quel cas doit-on utiliser `StringBuilder.append()` plutôt que le + ?

Exercice 6 - Tri à bulle [à la maison]

1 Écrire une méthode `swap` qui échange les valeurs de deux cases d'un tableau. `void swap(int[] array,int index1,int index2)`

2 Écrire une méthode `indexOfMin` qui renvoie l'index de la valeur minimal d'un tableau.

3 Modifier la méthode `indexOfMin` en ajoutant deux index indiquant que l'on cherche l'index du minimum non pas sur tout le tableau mais sur la partie de tableau entre les deux index (le premier compris, le deuxième non compris).

4 Écrire la méthode `sort` qui prend un tableau d'entier en paramètre et qui trie celui-ci en utilisant pour cela les méthodes `indexOfMin` et `swap`.