Egalité suite, String, StringBuilder

Exercice 1 - 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 Qu'est ce qu'un accesseur ?
Quels sont les acceseurs que l'on doit mettre ici ?

4 Ajouter un constructeur initialisant les coordonnés du point avec deux paramètres (appelons les `px` et `py`)
Indiquer pourquoi il faut déclarer les champs `x` et `y` `final`.
Quel est le problème avec le code de test.

5 Écrire un autre constructeur qui prend un point en paramètre et et utilise les coordonnées de celui-ci pour initialiser le point courant.
Comment le compilateur fait-il pour savoir quel constructeur appelé ?

Exercice 2 - Test d'égalité

En ré-utilisant la classe `Point`. Qu'affiche le code ci-dessous :

```
Point p1=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 p1=new Point(1,2);
  Point p2=p;
  Point p3=new Point(1,2);

  ArrayList list = new ArrayList();
  list.add(p1);
  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()` teste suivant le contenu et pas suivant les références.

5 Utiliser l'annotation `@Override` pour vérifier que vous avez bien typé la méthode ajoutée à `Point`.

Exercice 3 - 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 A quoi sert l'objet `java.lang.StringBuilder`
Pouquoi sa méthode `append(String)` renvoie un obejt de type `StringBuilder` ?

3 Ré-écrire la classe `Morse` en utilisant un `StringBuilder`.
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 + ?