



# EXERCICES

## Travail sur la programmation Objet

Créez un package **geometrie** afin d'y mettre toutes les classes de l'exercice.

### A. Une classe Point

Écrire une classe **Point** stockant un point graphique en coordonnées cartésiennes entières. On prendra soin de tester chaque exercice pour en valider le résultat.

1. *Déclarer une classe **Point** contenant les deux attributs privés **x** et **y**. Puis essayer le code suivant dans une méthode **main** de la classe **Point**.*
2. *Soit le code suivant :*

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

*Expliquer le phénomène observé.*

3. *Créer une classe **Main** (dans un fichier **Main.java**) et déplacer le **main** de **Point** dans la classe **Main**. Quel est le problème ? Comment peut-on le corriger ?*
4. *Écrire les accesseurs et mutateurs pour les 2 attributs et corriger le **main** de la classe **Main**.*
5. *Écrire le constructeur à 2 paramètres initialisant les coordonnées d'un point.*
6. *Compléter la méthode **main** afin de tester les méthodes écrites précédemment.*

On veut afficher les caractéristiques d'un point, par le code Java suivant

```
Point point = new Point(4,7);  
System.out.println(point);
```

Pour cela, il faut redéfinir dans la classe **Point** une méthode **public String toString()** (la définition de cette méthode est dans la classe **java.lang.Object**) retournant une chaîne de caractères, construite à partir des attributs de l'objet.

7. *Écrire la méthode **toString** qui permet d'afficher un point sous la forme (x,y). Cette méthode peut être générée automatiquement par le menu contextuel « **code source** ». Dans ce cas, il apparait « **@override** » avant la méthode. Chercher (sur Internet) à quoi sert cette marque et s'il est nécessaire de la laisser.*
8. *Soit le code suivant :*

```
Point p1 = new Point(1,2);  
Point p2 = p1;  
Point p3 = new Point(1,2);  
System.out.println(p1 == p2);  
System.out.println(p1 == p3);
```

a) *Qu'affiche le code ci-dessus ? Que peut-on en déduire par rapport aux références de points ?*

b) *Écrire la méthode equals qui vérifie que 2 points ont les même coordonnées.*

9. *Créez un attribut static compteur qui tiendra compte du nombre de points créés dans le programme. Pour cela, vous devez mettre à jour le constructeur, créer une méthode finalize() et un mutateur.*

a) *Pourquoi ne faut-il pas créer le mutateur de compteur ?*

b) *Affichez par son accesseur la valeur du compteur entre chaque instruction du programme ci-dessus. Que se passe-t-il ?*

## **B. Une classe LigneBrisée**

On veut créer une classe **LigneBrisée** qui contiendra une liste de points, de type la classe **Point** vu ci-dessus, stockés dans un tableau.

1. *Écrire la classe LigneBrisée, contenant la liste de points, avec ses accesseurs, mutateurs et toString, listant les points de la ligne. Une instance contiendra au moins 2 points au départ.*

2. *Écrire une méthode contientPoint qui renvoie vrai si le point passé en paramètre est dans la ligne brisée et faux sinon.*

3. *Écrire une méthode addPoint permettant d'ajouter un point dans la première case libre du tableau, à condition qu'il ne soit pas déjà dedans.*

4. *Écrire une méthode nbPoints permettant d'afficher le nombre de points de la ligne brisée.*

5. *Écrire une méthode nbMaxPoints permettant d'afficher le nombre de points possibles restant à mettre dans le tableau.*

6. *Écrire une méthode deletePoint permettant de supprimer un point et de décaler les suivants d'un cran en arrière dans le tableau pour éviter d'avoir des trous dans le tableau.*

7. *Écrire une classe Main qui permettra de tester la création de la ligne brisée, l'ajout et la suppression d'éléments, avec à chaque fois un affichage de la ligne brisée.*