Rappels- Java

Exercice 1 : Nombre Pair et Impair

On se propose de créer la classe **PairImpair**, qui étant donné un nombre entier saisie par l'utilisateur, détermine si cet entier est positif ou négatif et pair ou impair. Si l'entier est égal à 0, il est pair.

Exercice 2: Point

Le but est d'écrire une classe représentant des coordonnées cartésiennes.

1. Créer une classe **Point** avec deux champs privés **x** et **y**. Ecrire une méthode **main** avec le code suivant :

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

Pourquoi cela fonctionne t-il ?

2. Créer une classe **TestPoint** avec un main ayant le même code que précédemment. Que se passe-t-il ? Comment peut-on y remédier ?
3. Pourquoi il faut toujours que les champs d'une classe soient privés ?
4. Qu'est-ce qu'un accesseur ? Doit-on le faire ici ?
5. Créer un constructeur prenant les coordonnées du point en paramètre (appelés px et py). Quel est le problème au niveau du main ?
6. Modifier les paramètres du constructeur pour les appeler x et y. Que se passe-t-il ?
7. On veut pouvoir connaitre à tout moment le nombre de points qui ont été créés. Comment faire ?
8. Ecrire un autre constructeur prenant un point en argument et utilisant les coordonnées de ce dernier pour la création. Comment le compilateur sait quel constructeur appeler ?
9. Ecrire une fonction **affichePoint()** qui affiche les coordonnées du point comme ceci : (x, y).

```
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);
```

10. Qu'affiche ce code ? Pourquoi ?

11. Ecrire une méthode **isSameAs(Point)** renvoyant true si deux points ont les mêmes coordonnées.

Exercice 3 : Cercle

1. Ajouter une méthode **translate(dx, dy)** à Point.
2. Ecrire une classe Cercle, défini comme étant un point (centre) et un rayon, ainsi que son constructeur.
3. Ecrire une méthode **translate(dx, dy)** qui translate un cercle.
4. Ajouter une méthode **surface()** et l'ajouter dans l'affichage du cercle.
5. Créer une méthode **contains(Point p)** indiquant si le point p est contenu dans le cercle (indice : utiliser pythagore).
6. Créer la méthode **contains(Point p, Circle...circles)** qui renvoi vrai si le point est dans un des cercles.

Exercice 4 : Ligne brisée

On utilise toujours la classe Point. On veut maintenant écrire une classe représentant une ligne brisée, c'est-à-dire une suite de points. La ligne brisée aura un nombre maximum de points défini à la création, mais pouvant varier d'une instance à une autre.

1. On utilisera un tableau pour stocker les points d'une ligne brisée. Ecrire le constructeur d'une ligne brisée.
2. Ecrire une méthode **add** ajoutant un point à la ligne brisée.
3. Ecrire une méthode **pointCapacity()** et **nbPoints()** indiquant la capacité de la ligne brisée et le nombre de points actuellement sur la ligne.
4. Ecrire une méthode **contains** indiquant si un point passée en argument est contenu dans la ligne brisée. Vous utiliserez pour cela une boucle for each et non une boucle classique.

Exercice 5 : Recherche de nombres premiers

Pour tester si un nombre p est premier, une méthode simple consiste à tester tous ses diviseurs potentiels entre 2 et $\sqrt{p}$. Le nombre p est premier si et seulement s'il n'existe aucun $x \in \left[2 \ldots \sqrt{p}\right]$ tel que le reste de la division $\frac{p}{x}$ est nul.

1. On veut créer une classe *nbPremier* qui utilisera un tableau pour stocker les entiers. Le tableau aura un nombre d'entiers maximum défini à la création, mais pouvant varier d'une instance à l'autre. Ecrire le constructeur de *nbPremier*.
2. Ecrire une méthode **add** ajoutant un entier à *nbPremier.*
3. Créer une nouvelle méthode **initRandom(n, m)** pour initialiser *nbPremier* avec n entiers tirés aléatoirement entre 1 et m. (Pour générer des nombres aléatoires, utiliser java.util.Random).
4. Ajouter une méthode privée **isPrime(p)** qui retourne *true* si l'entier p passé en paramètre est premier ou *false* sinon.
5. Ecrire une méthode **printPrimes** qui affiche uniquement les nombres premiers de la collection.

6. Ecrire une méthode **main** pour qui génère une collection de 100 entiers aléatoires tirés entre 1 et 100, et affiche ceux qui sont premiers grâce aux méthodes précédentes.