Héritage, mutabilité, varargs, redéfinition, polymorphisme

Exercice 1 - Point

```
public class Point {
    public Point(int x,int y) {
        this.x=x;
        this.y=y;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
    @Override public String toString() {
        return "("+x+', '+y+')';
    }
    private final int x;
    private final int y;
}
```

- 1 Pourquoi la méthode `toString()` utilise l'annotation `@Override` ?
- 2 On souhaite ajouter une méthode `translate` permettant de traduire un point. Discuter des choix d'implémentations et de leurs influences sur la signature de la méthode `translate`.
- 3 Sachant que l'on choisit l'implémentation rendant le `Point` mutable. Écrire le code de la méthode `translate`
Vous écrirez le code de test de `translate` dans un `main` dans une classe séparée (disons `Main`).

Exercice 2 - Circle

Pour tout l'exercice, pour vos tests, vous écrirez votre code dans la classe `Main` vue précédemment.

Écrire une classe `Circle`, un cercle étant défini par un point correspondant au centre et un rayon.

- 1 Écrire le constructeur du `Circle`.
- 2 Écrire la méthode `toString` qui affiche le centre et le rayon. N'oubliez pas ce qu'il ne faut pas oublier.
- 3 Écrire la méthode `translate(int dx,int dy)` qui traduit le cercle.
- 4 Qu'affiche le code suivant :

```
Point p=new Point(1,2);
Circle c=new Circle(p,1);

Circle c2=new Circle(p,2);
c2.translate(1,1);

System.out.println(c+" "+c2);
```

Que doit-on faire pour que cela n'arrive pas ?

- 5 Quel est le problème avec l'accessor `getCenter()` codé de la façon suivante :

```
public Point getCenter() {  
    return center;  
}
```

Que doit-on faire pour que le code soit correct ?

- 6 Écrire la méthode `equals()` qui renvoie vrai si deux cercles ont le même centre et le même rayon.
- 7 Écrire la méthode `surface()` qui renvoie la surface du disque.
Modifier la méthode `toString()` pour quelle affiche aussi la surface.
- 8 Écrire la méthode `contains()` qui renvoie vrai si un point est contenu dans un cercle.
- 9 Écrire la méthode `contains(Point p, Circle... circles)` qui renvoie vrai si un point est contenu dans un des cercles.

Exercice 3 - One Ring for ...

Le but de cet exercice est de construire un anneau comme étant un cercle dont on a évidé une zone circulaire définie par son rayon interne.

Pour tout l'exercice, pour vos tests, réutiliser la classe `Main` vu précédemment.

- 1 Rappeler dans un premier temps, dans quel cas il est judicieux de faire de l'héritage.
- 2 Écrire la classe `Ring` qui hérite de la classe `Circle`.
- 3 Écrire un constructeur de la classe `Ring` prenant en paramètre, un centre, un rayon et un rayon interne. Faites attention à ce que le rayon interne soit inférieur au rayon de l'anneau.

Note: Tous les champs doivent être privés.

- 4 Écrire la méthode `equals()` qui test l'égalité de deux anneaux.
- 5 On souhaite afficher un anneau (centre, rayon, surface, rayon interne). Quel est le problème avec le code suivant :

```
Ring r=...  
System.out.println(r);
```

Que doit-on faire pour le corriger ?

Ensuite,

- 1 Implanter une méthode `contains(Point)` en évitant d'allouer des objets ou de dupliquer du code.
PS: il existe deux solutions dont une plus élégante que l'autre.
- 2 Ecrire la méthode `contains(Point p, Ring... rings)` qui renvoie vrai si un point est contenu dans un des anneaux.