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()` est marquée comme `Override` ?

2 Compléter le code ci-dessous en ajoutant une méthode `translate` permettant de translater un point.
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 ecrirez votre code dans la classe `Main` vu précédemment.
Écrire une classe `Circle`, un cercle étant définie 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 la rayon.

3 Écrire la méthode `translate(int dx,int dy)` qui translate le cercle. 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);
```

Expliquer le concept de mutabilité.
Que doit-on faire pour que cela n'arrive pas ?

4 Écrire la méthode `equals()` qui renvoie vrai si deux cercles ont le même centre et le même rayon.

5 Écrire la méthode `surface()` qui renvoie la surface du disque.
Modifier la méthode `toString` pour quelle affiche aussi la surface.
6 Écrire la méthode `contains()` qui renvoie vrai si un point est contenu dans un cercle.
7 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, vous ecrirez votre code dans 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. Faîte 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 Quel est le problème avec le code suivant :

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

Que doit-on faire pour le corriger ?

Ensuite,

1 Implanter une méthode `contains(Point)`. Quels problèmes cela pose ?
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.