SOLUTION DES EXERCICES

1/ L'instruction en ligne 4 peut être évitée. Prouvez-le !

```
1  public static void main(String[] args) {
2     Point pA = new Point(10,10);
3     Point pB = new Point();
4     pA.setRectangular(10, 10);
5     if (UtilPoint.equalsXY(pA, pB))
6        System.out.println("les points sont égaux en X et Y.");
7     else
8        System.out.println("les points sont différents en X et Y.");
11    System.out.println(pA);
10 }
```

Finalement, c'est bien le rôle du constructeur d'initialiser l'instance !

2/ Entre la ligne 4 et 5, ajouter une instruction qui augmente la position en X du point pB de 20 unités, **sans** utiliser la méthode `setRectangular`.

```
pB.offset(20, 0);  // translation en X de 20
```

ou

```
pB.setX(pB.getX() + 20);  // translation en X de 20
```

3/ Suite à votre solution, à l'exécution, quelle sera la sortie produite ?

```
les points sont différents en X et Y.
bean.Point : (10, 10)
```

Appel automatique à la méthode toString (classe Point)

Certaines parties du projet nécessitent de travailler avec des points en trois dimensions (x, y et z pour la profondeur).

1. Concevoir une classe Point3D répondant à ce problème, qui soit **compatible** avec la classe Point.

   voir page suivante

2. Que faudrait-il changer **à minima** au programme exemple pour qu'il travaille avec des instances de la nouvelle classe ?

   Juste changer les constructeurs ! En effet les objets de la classe Point3D ont la capacité de se comporter également comme des instances de Point qu'ils sont également (en effet, vue l'héritage, un point3D peut être vu comme un Pont 2D, sa dimension en z est tout simplement ignorée). Exemple :

```
public static void main(String[] args) {
      Point pA = new Point3D(10,10);
      Point pB = new Point3D();
      pB.offset(20,0);
      if (UtilPoint.equalsXY(pA, pB))
            System.out.println("les points sont égaux en X et Y.");
      else
            System.out.println("les points sont différents en X et Y.");
      System.out.println(pA);
}
```

   sortie :

```
      les points sont différents en X et Y.
      bean.Point3D : (10, 10)
```

La classe Point3D - à minima !

```
public class Point3D extends Point {

	private int z;

	/**
	 *
	 */
	public Point3D() {
		super();
		z = 0;
	}

	/**
	 * @param x
	 * @param y
	 */
	public Point3D(int x, int y) {
		super(x, y);
		this.z = 0;
	}

	public Point3D(int x, int y, int z) {
		super(x, y);
		this.z = z;
	}


	@Override
	public boolean equals(Object obj) { /*TODO*/ return false; }


}
```