M2 PISE 2020

Interrogation n°1 - Objets

La production finale devra être rendue sous la forme du code. Le plus simple est de *zipper* le répertoire du code source, de le renommer de votre nom (ex.: `yunes.zip` si votre nom de famille est yunes) et de l'envoyer par mail à `Jean-Baptiste.Yunes@u-paris`. Vous avez 24 heures pour répondre.

1. Créer une classe `Point` ayant pour attributs deux nombres, respectivement `x` et `y`, réprésentant une position dans un plan. Ces nombres seront de type `double`.

   - les attributs devront avoir une valeur fixée au départ (via un contructeur donc),
   - on pourra consulter la valeur de ces attributs (via les getters adéquats `getX`, `getY`),
   - on pourra obtenir l'affichage d'un point par une simple instruction comme `System.out.println(p)` qui devra générer quelque chose comme `[x:4.0;y:5.5]` (via la méthode `toString`).

   Exemple de petit programme qui doit fonctionner avec votre classe `Point`:

   ```
   Point p = new Point(4,5.5);
   System.out.println("Le point a pour valeur de son attribut x : "+p.getX());
   System.out.println(p);
   ```

   et dont l'exécution pourrait donner :

   ```
   Le point a pour valeur de son attribut x : 4.0
   [x:4.0,y:5.5]
   ```

2. Créer une classe `Polygone` qui représentera une collection de `Point`s et ayant pour attribut une couleur. La couleur ne pourra prendre que deux valeurs (vrai pour représenter le blanc, et faux pour représenter le noir). La collections de `Point`s pourra être aussi grande que l'on souhaite, mais devra contenir au moins un point.

   - la couleur et le premier point devront être initialisés à la construction,
   - la couleur pourra être obtenue sous la forme d'une chaîne de caractères `"blanc"` ou `"noir"` via une méthode `String getCouleur()`,
   - on pourra ajouter autant de `Point`s que l'on souhaite via la méthode `add(Point)`, (le stockage de ces `Point`s pourra s'effectuer à l'aide d'une `ArrayList<Point>`).
   - on pourra obtenir l'affichage correct via la méthode `toString`.

   Voici un exemple de programme utilisant cette classe :

   ```
   Point p1 = new Point(10,10);
   Polygone poly = new Polygone(true,p1);
   System.out.println(poly);
   Point p2 = new Point(10,20);
   poly.add(p2);
   System.out.println(poly);

   Polygone poly2 = new Polygone(false,p2);
   for (int i=10; i<13; i++) {
     poly2.add(new Point(i,2*i));
   }
   System.out.println(poly2);
   ```

   qui devra générer une sortie comme :

   ```
   Polygone blanc : [x:10.0,y:10.0]
   Polygone blanc : [x:10.0,y:10.0][x:10.0,y:20.0]
   Polygone noir : [x:10.0,y:20.0][x:10.0,y:20.0][x:11.0,y:22.0][x:12.0,y:24.0]
   ```

3. Créer une classe `Figure` qui soit une simple collection (`ArrayList`) de `Polygone`s additionnée d'un attribut qui sera le nom de la figure. Une figure peut être «vide» (ce sera le cas au départ).

- le nom de la figure devra être initialisé à la construction de la figure vide,
- l'ajout d'un `Polygone` s'effectuera via la méthode `add(Polygone)`,
- on pourra obtenir l'affichage d'une figure via ma méthode `toString`.

Ainsi:

```
Figure f = new Figure("Joli dessin");
System.out.println(f);
f.add(poly);
System.out.println(f);
f.add(poly2);
System.out.println(f);
```

pourra générer la sortie suivante :

```
Figure Joli dessin {
}
Figure Joli dessin {
    Polygone blanc : [x:10.0,y:10.0][x:10.0,y:20.0]
}
Figure Joli dessin {
    Polygone blanc : [x:10.0,y:10.0][x:10.0,y:20.0]
    Polygone noir : [x:10.0,y:20.0][x:10.0,y:20.0][x:11.0,y:22.0][x:12.0,y:24.0]
}
```