Les patrons de *construction*

Exercice 1 – *Affichage de dessins vectoriels*

Dans l'archive associée à ce sujet, vous trouverez un programme permettant d'afficher un dessin sous forme vectoriel. Ces dessins peuvent être formés de lignes, cercles et rectangles. Ces différents concepts sont déjà implémentés. Pour former un dessins, l'on créera une liste contenant des instances des classes `Line, Rectangle, Circle` que l'on passera à une instance de GraphicViewer qui ouvre une fenêtre et affiche le graphique.
Pour ajouter de nouvelles formes, on doit implémenter l'interface :

```
1  import java.awt.Graphics2D;
2
3  public interface IShape {
4    public void draw(Graphics2D screen);
5  }
```

La classe Test contient une méthode `getDemo()` qui renvoie une liste constituant le dessin de test que vous pouvez voir :

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.awt.image.*;
4  import java.awt.geom.*;
5  import java.util.*;
6
7  public class Test {
8    // Code d'initialisation :
9    public static java.util.List<IShape> getDemo() {
10     java.util.List<IShape> ls=new ArrayList<IShape>();
11     ls.add(new Line(0, 500, 800, 500, Color.GREEN));
12     ls.add(new Line(300, 0, 0, 300, Color.YELLOW));
13
14     ls.add(new Line(30, 300, 180, 200, Color.BLUE));
15     ls.add(new Line(330, 300, 180, 200, Color.BLUE));
16     ls.add(new Rectangle(30, 300,330, 500, Color.RED));
17
18     double sunX = 600;
19     double sunY = 120;
20     double sunRad = 60;
21     ls.add(new Circle(sunX, sunY, sunRad, Color.BLACK));
22     int sunRay = 20;
23     for (int i=0; i<sunRay; ++i) {
24       double tau=i*2*Math.PI/sunRay;
25       ls.add(new Line(sunX+(sunRad+5)*Math.cos(tau),
26         sunY-(sunRad+5)*Math.sin(tau),
27         sunX+(1.5*sunRad+5)*Math.cos(tau),
28         sunY-(1.5*sunRad+5)*Math.sin(tau),
29         Color.BLACK));
```

```
30        }
31
32        double manX=600;
33        double manY=450;
34        ls.add(new Line(manX, manY-70, manX-40, manY-110, Color.RED));
35        ls.add(new Line(manX, manY-70, manX+40, manY-110, Color.RED));
36        ls.add(new Circle(manX, manY-120, 20, Color.GRAY));
37        ls.add(new Line(manX, manY, manX, manY-100, Color.BLUE));
38        ls.add(new Line(manX, manY, manX-20, manY+50, Color.BLACK));
39        ls.add(new Line(manX, manY, manX+20, manY+50, Color.BLACK));
40
41
42        return ls;
43      }
44
45      // Code du client :
46      public static void main(String[] args) {
47        GraphicViewer gv = new GraphicViewer();
48        java.util.List<IShape> demo=getDemo();
49        gv.draw(demo);
50      }
51    }
```

On souhaite ajouter un mode de dessin supplémentaire qui donnerait un style de dessin à main levée lors du rendu. Pour cela, on se propose d'ajouter les classes suivantes :

- HandLine : celle-ci utilisera la classe `QuadCurve2D` (courbe de Bézier) pour le tracé tel que : on déplacera aléatoirement de quelques pixels (`getNoise(void)`) l'origine et l'arrivée du segment de droite et on placera le point de contrôle au milieu du segment plus ou moins un nombre de pixels aléatoire de l'ordre de la longueur du segment (`getNoise(longueurX)`).
- HandRectangle : on utilisera simplement quatre HandLine pour cela,
- HandCircle : on ajoutera aléatoirement et indépendamment aux rayons horizontaux et verticaux de l'ellipse quelques pixels de l'ordre du rayon (`getNoise(rayonX)`).

Les méthodes `getNoise()` et `getNoise(double)` vous sont fournies par la classe `Noise`. Évidemment, il sera nécessaire de modifier la méthode `getDemo()` pour utiliser les nouvelles classes. Cependant on souhaite pouvoir faire en sorte que toute méthode fournissant un dessin puisse le faire aussi bien dans le mode classique que dans le mode à main levée ou bien tout autre mode à venir.

Question 1.1 : Quel patron de conception proposez vous d'utiliser. Rappelez son schéma de principe

Question 1.2 : Donnez le schéma UML de votre solution (en identifiant les acteurs).

Question 1.3 : Implémentez la solution correspondante.