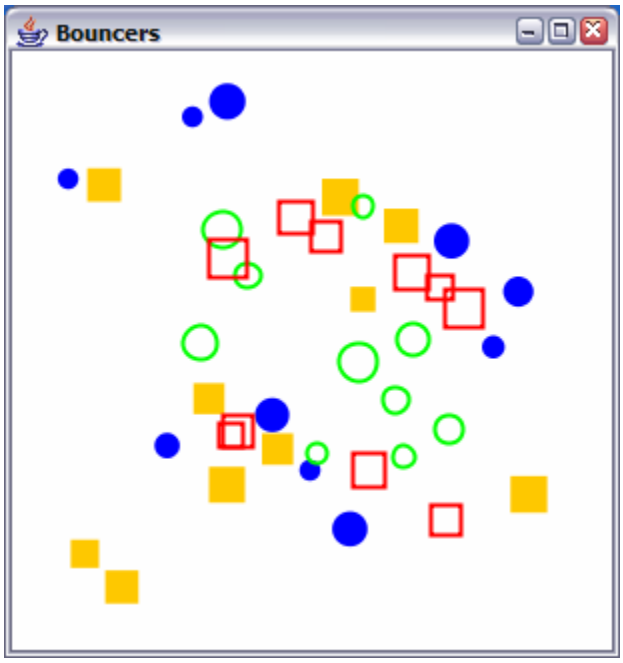
Laboratoire 3: Fabrique abstraite

Durée du laboratoire: 2 séances. A rendre pour le mardi 12 avril 2016, au début de la séance de laboratoire.



1. Objectifs

- Concevoir une application en utilisant les modèles singleton, fabrication et fabrique abstraite.

2. Indications

Définir une application graphique qui permette d'instancier des cercles et des carrés et de les déplacer dans un espace d'affichage graphique commun.

- Deux familles de cercles et de carrés (c.f. interface `Bouncable` donnée en annexe) doivent pouvoir être générées : les cercles et carrés pleins de couleur respectivement bleue et orange et les cercles et carrés possédant une bordure de 2 pixels (c.f. classe `BasicStroke`) de couleur respectivement verte et rouge.
- Ces objets géométriques devront être initialisées aléatoirement (taille, position, vecteur de déplacement).
- Si un objet rencontre un bord, il doit rebondir.
- La classe représentant l'affichage doit mettre en oeuvre un singleton et implémenter l'interface `Displayer` donnée en annexe. Pour gérer l'affichage il peut être intéressant d'utiliser les méthodes `createImage` de la classe `JPanel`, `getGraphics` de la classe `Image`, ainsi que la méthodes `fill` et `drawImage` de la classe `Graphics2D`.
- L'affichage d'un objet `Bouncable` est délégué à un objet spécialisé implémentant l'interface `Renderable`.
- Gestion du clavier (c.f. classe `KeyAdapter`):
  - Touche 'e' : effacer l'affichage.
  - Touche 'b' : générer 10 cercles et 10 carres possédant une bordure.

  - Touche 'f' : générer 10 cercles et 10 pleins.
  - Touche 'q' : quitter le programme.
- La fenêtre contenant l'affichage doit pouvoir être redimensionnée.

Veiller à soigner l'implémentation et effectuer tous les refactorings nécessaires afin que le code produit soit le plus élégant possible selon les standards de programmation POO. Veiller en particulier à ce qu'un nombre minimal d'objets soit instancié.

3. Travail à rendre

- Diagramme des classes.
- Sources du programme.

Annexe

```
public interface Displayer
{
  int getWidth();
  int getHeight();
  Graphics2D getGraphics();
  void repaint();
  void setTitle(String s);
  void addKeyListener(KeyAdapter ka);
}

public interface Bouncable
{
  void draw();
  void move();
  Renderable getRenderer();
  Color getColor();
  Shape getShape();
}

public interface Renderable
{
  void display(Graphics2D g, Bouncable b);
}

public class BounceApp
{
  private LinkedList<Bouncable> bouncers;
  // Autres attributs

  public BounceApp() {
    /* … */
  }

  public void loop() {
    /* … */
  }

  public static void main(String ... args) {
    new BounceApp().loop();
  }
}
```