Étape 1: Améliorations

- En fait, du point de vue encapsulation, il vaudrait mieux disposer d'un constructeur pour initialiser les coordonnées et de supprimer les manipulateurs.
- D'autre part, du point de vue modélisation ce serait mieux de disposer d'une boule qui sache
  - se déplacer soi-même avec ses propres attributs,
  - se dessiner soi-même.
- Pour effectuer des déplacements plus précis, les coordonnées seront mémorisées comme réels (double) et ce sera seulement lors du dessin qu'on les convertira en entiers.

Nous allons donc changer la classe **`Ball`** comme décrit ci-dessous et créer une nouvelle classe **`MovingBall`** qui saura se déplacer et se dessiner soi-même.

MovingBall
− x : double − y : double − radius : int − xStep : double − yStep : double
+ MovingBall(pX : double, pY : double, pRadius : int, pXStep : double, pYStep : double) + getXStep() : double + getYStep() : double + doStep(width : int, height : int) : void + getX() : double + getY() : double + getRadius() : int + draw(g : Graphics) : void

Cette nouvelle classe représente donc une boule en mouvement. Elle sait se déplacer elle-même à l'aide de la méthode **`doStep(...)`**. En lui envoyant la largeur et la hauteur du canevas sur lequel elle se trouve, elle peut rebondir sur les bords sans jamais sortir du canevas.

La balle possède deux attributs **`xStep`** et **`yStep`** (type **`double`**) qui définissent les pas à effectuer en horizontale et en verticale à chaque appel de **`doStep`**. Ensemble avec le délai du **`timer`**, ces deux attributs définissent ainsi les vitesses horizontale et verticale des balles à l'écran.

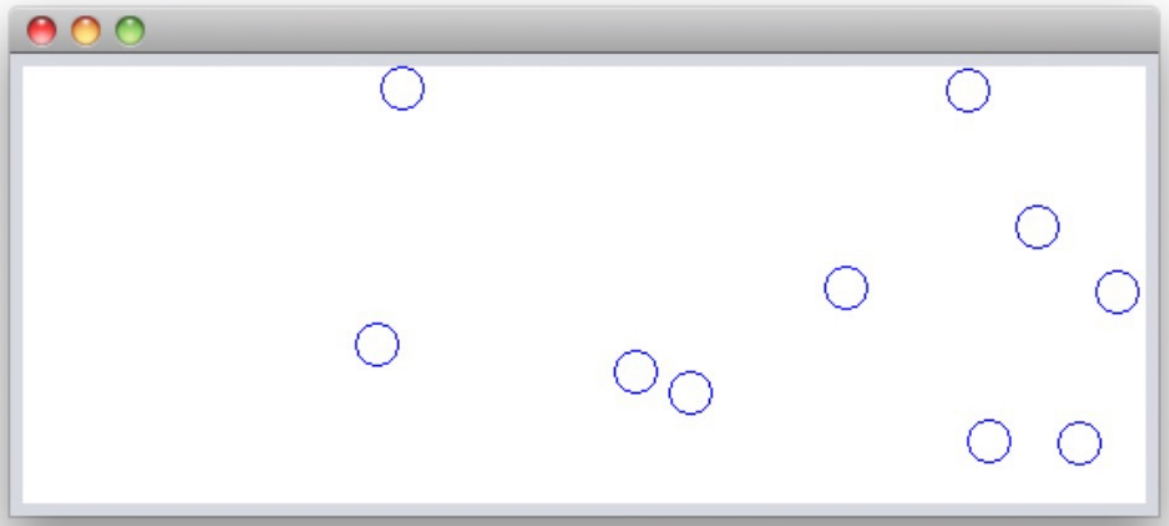
`xStep`	positif ↔ déplacement vers la droite;	négatif ↔ vers la gauche
`yStep`	positif ↔ déplacement vers le bas;	négatif ↔ vers le haut

Comme **`xStep`** et **`yStep`** sont des réels, les balles peuvent avoir une infinité de trajectoires différentes. Désormais, afin de disposer d'une meilleure précision, les coordonnées de la boule seront aussi des nombres réels.

Veillez à ce que la balle retourne automatiquement à l'intérieur du canevas lorsqu'on rétrécit brusquement le canevas.

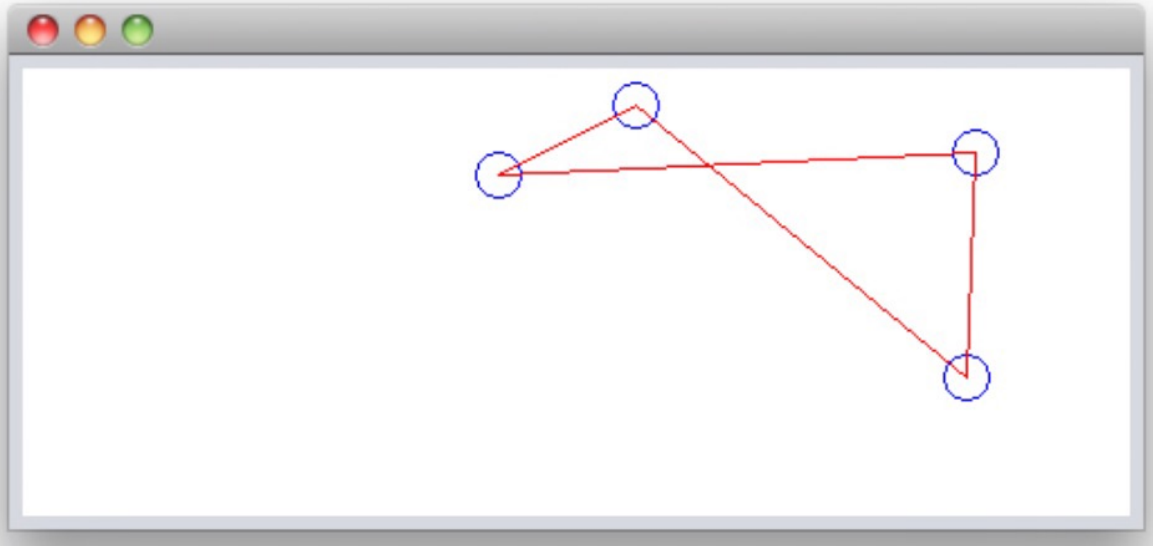
Étape 2

Modifiez ensuite votre programme de manière à ce qu'il devienne possible d'ajouter plusieurs boules en mouvement. Leurs positions de départ sont aléatoires. Leurs trajectoires sont aléatoires (-5.0 ≤ **xStep** ≤ 5.0 et -5.0 ≤ **yStep** ≤ 5.0). La liste des balles est gérée par la classe **MovingBalls**.



Étape 3

Ensuite nous pouvons ajouter le code qui permet de tracer une ligne entre les différentes boules. Quelle classe faut-il charger de cette tâche ?



```
1
2   import java.awt.Color;
3   import java.awt.Graphics;
4
5   /*
5    * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.
6   txt to change this license
6    * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JPanel.java to
7   edit this template
8    */
9
10  /**
11   *
12   * @author luxformel
13   */
14  public class DrawPanel extends javax.swing.JPanel {
15      private MovingBalls movingBalls = null;
16
17      public void setMovingBalls(MovingBalls movingBalls) {
18          this.movingBalls = movingBalls;
19      }
20
21
22      /**
23       * Creates new form DrawPanel
24       */
25      public DrawPanel() {
26          initComponents();
27      }
28
29      @Override
30      protected void paintComponent(Graphics g) {
30          super.paintComponent(g); // Generated from nbfs:
31  //nbhost/SystemFileSystem/Templates/Classes/Code/OverriddenMethodBody
32          g.setColor(Color.white);
33          g.fillRect(0, 0, getWidth(), getHeight());
34          if (movingBalls != null){
35              movingBalls.draw(g);
36          }
37      }
38
39
40
41      /**
41       * This method is called from within the constructor to initialize the
42  form.
42       * WARNING: Do NOT modify this code. The content of this method is
43  always
44       * regenerated by the Form Editor.
45       */
48      @SuppressWarnings("unchecked")
60
```

```
61
62
63      // Variables declaration - do not modify//GEN-BEGIN:variables
64      // End of variables declaration//GEN-END:variables
65  }
66
```

```
1
2   import javax.swing.Timer;
3
4   /*
4    * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.
5   txt to change this license
5    * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to
6   edit this template
7    */
8
9   /**
10   *
11   * @author luxformel
12   */
13  public class MainFrame extends javax.swing.JFrame {
14
15
16      private MovingBalls movingBalls = new MovingBalls();
17
18      private int step = 5;
19      private Timer timer;
20
21      /**
22       * Creates new form MainFrame
23       */
24      public MainFrame() {
25          initComponents();
26
27
28          drawPanel.setMovingBalls(movingBalls);
29          drawPanel.repaint();
30          timer = new Timer(100, stepButton.getActionListeners()[0]);
31      }
32
33      /**
33       * This method is called from within the constructor to initialize the
34  form.
34       * WARNING: Do NOT modify this code. The content of this method is
35  always
36       * regenerated by the Form Editor.
37       */
40      @SuppressWarnings("unchecked")
45
47
58
65
72
79
109
112
113
113     private void stepButtonActionPerformed(java.awt.event.ActionEvent evt)
```

```
114 {
115         movingBalls.move(drawPanel.getWidth(), drawPanel.getHeight());
116 
117         drawPanel.repaint();
118     }
119 
119     private void startStopButtonActionPerformed(java.awt.event.ActionEvent
120 evt) {
121         if (timer.isRunning()){
122             timer.stop();
123             startStopButton.setText("Start");
124         }
125         else{
126             timer.start();
127             startStopButton.setText("Stop");
128         }
129     }
130 
131     private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
132 
133         int radius = (int)(Math.random() * (40 - 10 + 1)) + 10;
134 
135         int minX = radius;
136         int maxX = drawPanel.getWidth() - radius;
137         int minY = radius;
138         int maxY = drawPanel.getHeight() - radius;
139 
140         int x = (int)(Math.random() * (maxX - minX + 1)) + minX;
141         int y = (int)(Math.random() * (maxY - minY + 1)) + minY;
142 
143         int stepX = (int)(Math.random() * (5 - (-5) + 1)) + (-5);
144         int stepY = (int)(Math.random() * (5 - (-5) + 1)) + (-5);
145 
146         MovingBall ball = new MovingBall(x, y, radius, stepX, stepY);
147 
148         movingBalls.add(ball);
149         drawPanel.repaint();
150     }
151 
152     /**
153      * @param args the command line arguments
154      */
155     public static void main(String args[]) {
177         /* Set the Nimbus look and feel */
178 
179         /* Create and display the form */
180         java.awt.EventQueue.invokeLater(new Runnable() {
181             public void run() {
182                 new MainFrame().setVisible(true);
183             }
184         });
185     }
```

```
186
187     // Variables declaration - do not modify//GEN-BEGIN:variables
188     private javax.swing.JButton addButton;
189     private DrawPanel drawPanel;
190     private javax.swing.JButton startStopButton;
191     private javax.swing.JButton stepButton;
192     // End of variables declaration//GEN-END:variables
193 }
194
```

```java

import java.awt.Color;
import java.awt.Graphics;

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.
txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */

/**
 *
 * @author luxformel
 */
public class MovingBall {
    private int x;
    private int y;
    private int radius;
    private int stepX = 5;
    private int stepY = 5;

    public MovingBall(int x, int y, int radius, int stepX, int stepY) {
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.stepX = stepX;
        this.stepY = stepY;
    }



    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public int getRadius() {
        return radius;
    }

```

```
51      public void setRadius(int radius) {
52          this.radius = radius;
53      }
54
55      public void draw(Graphics g){
56          g.setColor(Color.black);
57          g.drawOval(x - radius, y - radius, 2 * radius, 2 * radius);
58      }
59
60      public void move(int width, int height){
61          int oldX = getX();
62          int oldY = getY();
63
64          setX(oldX + stepX);
65
66          if (getX() + getRadius() > width){
67              setX(width - getRadius());
68              stepX = -5;
69          }
70
71          if (getX() - getRadius() < 0){
72              setX(getRadius());
73              stepX = 5;
74          }
75
76
77          setY(oldY + stepY);
78
79          if (getY() + getRadius() > height){
80              setY(height - getRadius());
81              stepY = -5;
82          }
83
84          if (getY() - getRadius() < 0){
85              setY(getRadius());
86              stepY = 5;
87          }
88
89      }
90  }
91
```

```java

import java.awt.Graphics;
import java.util.ArrayList;

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.
txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */

/**
 *
 * @author luxformel
 */
public class MovingBalls {
    private ArrayList<MovingBall> alMovingBalls = new ArrayList<>();

    public boolean add(MovingBall e) {
        return alMovingBalls.add(e);
    }

    public void draw(Graphics g){
        for (int i = 0; i < alMovingBalls.size(); i++){
            alMovingBalls.get(i).draw(g);
        }
    }

    public void move(int width, int height){
        for (int i = 0; i < alMovingBalls.size(); i++){
            alMovingBalls.get(i).move(width, height);
        }
    }
}

```