```java
package Serveur.fenetre;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.Closeable;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.function.Function;

import javax.swing.JFrame;
import javax.swing.JPanel;

import Serveur.Main;
import Serveur.maths.Utils;

/**
    Sert Ã  tracer des courbes sur une fenÃªtre java.
*/
public class Graph extends JFrame {
    private static final long serialVersionUID = 1L;
    public int avancementPixel = 1;
    public ArrayList<Function<Float, Float>> foncts = new ArrayList<Function<Float, Float>>();
    public ArrayList<Color> colors = new ArrayList<Color>();
    public volatile boolean painted = true;
    private Closeable closer = null;

    public Graph(float xmin, float xmax, float ymin, float ymax) {
        super();
        super.setTitle("Graph");
        super.setSize(1600, 900);
        super.setContentPane(new JPanel() {
            private static final long serialVersionUID = 1L;

            public void paintComponent(Graphics g) {
                Rectangle rect = g.getClipBounds();
                Drawer drawer = new Drawer(g);
                drawer.setAntiAliasing(true);

                int zero_x = (int) Utils.map(0, xmin, xmax, rect.x, rect.x + rect.width - 1);
                int zero_y = (int) Utils.map(0, ymin, ymax, rect.y + rect.height - 1, rect.y);

                g.setColor(Color.WHITE);
                g.fillRect(0, 0, rect.width, rect.height);
                g.setColor(Color.BLACK);
                g.drawLine(0, zero_y, rect.width - 1, zero_y);
                g.drawLine(zero_x, 0, zero_x, rect.height - 1);

                double delta_x = (double) rect.width / (xmax - xmin);
                double delta_y = (double) rect.height / (ymax - ymin);

                for (double x = zero_x; x < rect.x + rect.width; x += delta_x)
                    g.drawLine((int) x, zero_y + 5, (int) x, zero_y - 5);

                for (double x = zero_x; x > rect.x; x -= delta_x)
                    g.drawLine((int) x, zero_y + 5, (int) x, zero_y - 5);
```

```
60
61                  for (double y = zero_y; y < rect.y + rect.height; y += delta_y)
62                      g.drawLine(zero_x + 5, (int) y, zero_x - 5, (int) y);
63
64                  for (double y = zero_y; y > rect.y; y -= delta_y)
65                      g.drawLine(zero_x + 5, (int) y, zero_x - 5, (int) y);
66
67                  int lasty = 0;
68                  float angle = 0;
69                  Iterator<Color> colorsIt = colors.iterator();
70
71                  for (Function<Float, Float> f : foncts)
72                  {
73                      if (colorsIt.hasNext())
74                          g.setColor(colorsIt.next());
75                      else
76                      {
77                          g.setColor(Color.getHSBColor((float) (angle / (2 * Math.PI)), 1,
  0.9f));
78                          angle = (float) ((angle + 3.6) % (2 * Math.PI));
79                      }
80
81                      for (int x = 0; x < rect.width; x += avancementPixel) {
82                          float xmap = (float) Utils.map(x, 0, rect.width - 1, xmin, xmax);
83                          float ymap = f.apply(xmap);
84                          int y = (int) Utils.map(ymap, ymin, ymax, rect.height - 1, 0);
85
86                          if (x > 0)
87                              drawer.drawBigLine(x - 1, lasty, x, y, 3);
88
89                          lasty = y;
90                      }
91
92                  }
93
94                  painted = true;
95                  Main.main.painted = true;
96              }
97          });
98
99          super.addWindowListener(new WindowListener() {
100             public void windowOpened(WindowEvent e) {
101             }
102             public void windowIconified(WindowEvent e) {
103             }
104             public void windowDeiconified(WindowEvent e) {
105             }
106             public void windowDeactivated(WindowEvent e) {
107             }
108
109             public void windowClosing(WindowEvent e) {
110                 foncts.clear();
111                 colors.clear();
112
113                 if (Graph.this.closer != null)
114                     try {
115                         Graph.this.closer.close();
116                     } catch (IOException ex) {
117                         ex.printStackTrace();
118                     }
119                 System.exit(0);
120             }
```

```
121
122             public void windowClosed(WindowEvent e) {
123             }
124             public void windowActivated(WindowEvent e) {
125             }
126         });
127         super.setVisible(true);
128     }
129
130     public void repaint() {
131         painted = false;
132         super.repaint();
133         while (!painted);
134     }
135
136     public void setCloser(Closeable closer) {
137         this.closer = closer;
138     }
139 }
```