```java
package Serveur;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.IOException;
import java.util.function.Function;

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

import Serveur.fenetre.Graph;
import Serveur.maths.Fonction;
import Serveur.maths.Utils;
import Serveur.server.ClientTriangulation;
import Serveur.server.ServerTriangulation;
import Serveur.triangulation.Balle;
import Serveur.triangulation.Capteur;
import Serveur.triangulation.Map;

public class Main {
    public static Main main;
    public static boolean FORCER_ARRET = true;
    public final boolean DEBUG = false;
    public Map map = new Map();
    public final double RAPPORT_HAUTEUR_LONGUEUR = 9. / 16;
    public double xmin = -10 * ClientTriangulation.ZOOM, xmax = 10 * ClientTriangulation.ZOOM;
    public double ymin = xmin * RAPPORT_HAUTEUR_LONGUEUR,
            ymax = xmax * RAPPORT_HAUTEUR_LONGUEUR;
    public float offset = 30;
    public JFrame fenetre;
    public volatile boolean painted = true;
    public volatile boolean continuer = true;
    public volatile boolean fenetreActive = true;

    public void lancerFenetre(ServerTriangulation server) {
        fenetre = new JFrame();
        fenetre.setSize(1600, 900);
        fenetre.setResizable(false);
        fenetre.setLocationRelativeTo(null);
        fenetre.setTitle("Serveur");
        fenetre.addWindowListener(new WindowListener() {
            public void windowOpened(WindowEvent e) {
            }
            public void windowIconified(WindowEvent e) {
            }
            public void windowDeiconified(WindowEvent e) {
            }
            public void windowDeactivated(WindowEvent e) {
            }

            public void windowClosing(WindowEvent e) {
                if (FORCER_ARRET)
                    System.exit(0);

                server.close();
                continuer = false;
                painted = true;
                fenetreActive = false;
            }
```

```
62
63              public void windowClosed(WindowEvent e) {
64              }
65              public void windowActivated(WindowEvent e) {
66              }
67          });
68
69          fenetre.setVisible(true);
70
71          fenetre.setContentPane(new JPanel() {
72              private static final long serialVersionUID = 1L;
73
74              @SuppressWarnings("unused")
75              public void paintComponent(Graphics g) {
76                  int width = getWidth();
77                  int height = getHeight();
78
79                  map.paintComponent(g, xmin, xmax, ymin, ymax);
80
81                  painted = true;
82              }
83          });
84      }
85
86      public void lancerGraph(ServerTriangulation server) {
87          Function<Float, Float> rssi = t -> 0f;
88          fenetre = new Graph(0, 80, offset - 80, offset + 20);
89          ((Graph) fenetre).foncts.add(rssi);
90          ((Graph) fenetre).colors.add(new Color(255, 17, 76));
91          ((Graph) fenetre).colors.add(new Color(50, 80, 255));
92          ((Graph) fenetre).colors.add(new Color(70, 255, 13));
93      }
94
95      @SuppressWarnings("unchecked")
96      public Function<Float, Float>[] createFunctionsRSSI() {
97          if (map.capteurs.size() == 0)
98              return new Function[]{x -> 0f};
99
100         Double[][] listRSSI = {
101                 map.capteurs.get(0).getHistoriqueRSSI(),
102                 map.capteurs.get(0).getHistoriqueSmoothRSSI(),
103                 map.capteurs.get(0).getHistoriqueFiltredRSSI(),
104         };
105
106         Function<Float, Float>[] result = new Function[listRSSI.length];
107         for (int i = 0; i < listRSSI.length; i++)
108         {
109             Double[] rssi = listRSSI[i];
110             result[i] = x -> {
111             switch (rssi.length) {
112                 case 0 :
113                     return offset;
114                 case 1 :
115                     return (float) (double) rssi[0] + offset;
116                 default :
117                     Float t = (float) Utils.map(x, 0, 80, 0, rssi.length - 2);
118                     int index = (int) (float) t;
119                     float blend_x = t - index;
120
121                     return (float) ((1 - blend_x) * rssi[index]
122                             + blend_x * rssi[index + 1]) + offset;
123             }};
```

```
124         }
125
126         return result;
127     }
128
129     public void updateRSSI() {
130         if (fenetre instanceof Graph)
131         {
132             ((Graph) fenetre).foncts.clear();
133
134             for (Function<Float, Float> f : createFunctionsRSSI())
135             ((Graph) fenetre).foncts.add(f);
136         }
137     }
138
139     public void repaint() {
140         doModifications(() -> {
141             updateRSSI();
142             painted = false;
143             fenetre.repaint();
144             while (!painted && fenetreActive);
145             return null;
146         });
147     }
148
149     public void setupMapTest() {
150         Capteur capteur1 = new Capteur(-1 - 3, -2 + 3);
151         capteur1.setDistance(3);
152         map.addCapteur(capteur1);
153
154         Capteur capteur2 = new Capteur(1 - 3, -1.5 + 3);
155         capteur2.setDistance(3);
156         map.addCapteur(capteur2);
157
158         Capteur capteur3 = new Capteur(3 - 3, 1 + 3);
159         capteur3.setDistance(4);
160         map.addCapteur(capteur3);
161
162         for (int i = 0; i <= 16; i++) {
163             double x = Utils.map(i, 0, 16, xmin, xmax);
164
165             for (int j = 0; j <= 9; j++) {
166                 double y = Utils.map(j, 0, 9, ymin, ymax);
167                 map.addBalle(x, y);
168             }
169         }
170     }
171
172     public void resetBalles() {
173         for (Balle balle : map.balles)
174             balle.close();
175
176         for (int i = 0; i <= 16; i++) {
177             double x = Utils.map(i, 0, 16, xmin, xmax);
178
179             for (int j = 0; j <= 9; j++) {
180                 double y = Utils.map(j, 0, 9, ymin, ymax);
181                 map.addBalle(x, y);
182             }
183         }
184     }
185
```

```
186     public void main() {
187         resetBalles();
188 
189         try {
190             // System.setOut(new PrintStream(new File("out.txt")));
191             ServerTriangulation server = new ServerTriangulation(80);
192             server.start();
193             lancerFenetre(server);
194             //lancerGraph(server);
195         } catch (IOException e) {
196             e.printStackTrace();
197         }
198 
199         while (continuer) {
200             map.getSource(1);
201             repaint();
202         }
203 
204         map.close();
205         fenetre.dispose();
206     }
207 
208     public static void main(String[] args) {
209         main = new Main();
210         main.main();
211     }
212 
213     public synchronized Object doModifications(Fonction f) {
214         return f.method();
215     }
216 }
```