```java
package hyperDap.guiPres.views.honoursMainView;

import java.util.HashMap;
import java.util.Map;
import hyperDap.base.types.dataSet.ValueDataSet;
import hyperDap.guiPres.charts.DisplayDataSet;
import hyperDap.guiPres.fxEncapsulation.GUIMainForFX;
import javafx.beans.property.BooleanProperty;
import javafx.beans.property.SimpleBooleanProperty;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.CheckBox;
import javafx.scene.control.TextField;
import javafx.scene.layout.VBox;

/**
 * This is the {@code Controller} to the honoursMainView, to demonstrate application features at the
 * completion of the author's Honours Project.
 *
 * @author soenk
 *
 */
public class HonoursMainController {

  GUIMainForFX main;

  @FXML
  TextField baseField;
  @FXML
  Button baseRandButton;
  @FXML
  TextField stepField;
  @FXML
  Button stepRandButton;
  @FXML
  TextField lengthField;
  @FXML
  Button lengthRandButton;
  @FXML
  TextField precisionField;
  @FXML
  Button precisionRandomField;

  @FXML
  CheckBox didi1;
  @FXML
  CheckBox didi2;
  @FXML
  CheckBox didi3;
  @FXML
  CheckBox didi4;
  @FXML
  CheckBox didi5;
  @FXML
  CheckBox didi6;
  @FXML
  CheckBox didi7;
  @FXML
  CheckBox didi8;
  CheckBox randomBox;

  Map<CheckBox, String> didiMap;

  @FXML
  VBox graphBox;

  @FXML
  Button executeButton;
  @FXML
  Button executeButton2;
  @FXML
  Button exitButton;
```

```
73
74      private DisplayDataSet setChart;
75
76      /**
77       * Constructor.
78       * <p>
79       * Remember that JavaFX elements can only be accessed later, in {@link
          #initialize()}
80       */
81      public HonoursMainController() {
82        System.out.println(String.format("%s has been instantiated",
          HonoursMainController.class));
83      }
84
85      /**
86       * Called after JavaFX elements have been initialised and can be accessed by the
87       * {@code Controller}.
88       */
89      public void initialize() {
90
91        // for reading in whcih functions should be plotted
92        this.didiMap = new HashMap<CheckBox, String>();
93        this.didiMap.put(didi1, "constant");
94        this.didiMap.put(didi2, "linear");
95        this.didiMap.put(didi3, "square");
96        this.didiMap.put(didi4, "cubic");
97        this.didiMap.put(didi5, "exp");
98        this.didiMap.put(didi6, "rand");
99        this.didiMap.put(didi7, "bias");
100       this.didiMap.put(didi8, "noise");
101       this.randomBox = this.didi6;
102
103       // the graphs used for display
104       this.setChart = new DisplayDataSet();
105       this.graphBox.getChildren().add(this.setChart);
106
107       // a boolean property to help unfocus at startup. Credit:
108       //
          https://stackoverflow.com/questions/29051225/remove-default-focus-from-textfield-j
          avafx
109       final BooleanProperty firstTime = new SimpleBooleanProperty(true);
110       this.baseField.focusedProperty().addListener((observable, oldValue, newValue) -> {
111         if (newValue && firstTime.get()) {
112           firstTime.set(false);
113           this.baseField.getParent().requestFocus();
114         }
115       });
116
117     }
118
119     // fx interface
120     //
          ************************************************************************************
          ************************************
121
122     public void terminate() {
123       this.main.terminate();
124     }
125
126     public void execute() {
127       Map<String, Double> map = new HashMap<String, Double>();
128       Double temp;
129
130       try {
131         temp = Double.valueOf(this.baseField.getText());
132         map.put("base", temp);
133       } catch (NumberFormatException e) {
134         this.baseField.setPromptText("This must be a number e.g. '5.0'");
135         this.baseField.setText("");
136         return;
137       }
138       try {
139         temp = Double.valueOf(this.stepField.getText());
```

```
140         map.put("step", temp);
141       } catch (NumberFormatException e) {
142         this.stepField.setPromptText("This must be a number e.g. '5.0'");
143         this.stepField.setText("");
144         return;
145       }
146       try {
147         temp = Integer.valueOf(this.lengthField.getText()).doubleValue();
148         if (temp < 10) {
149           this.lengthField.setText("");
150           this.lengthField.setPromptText("This must be above 10");
151           return;
152         }
153         map.put("length", temp);
154       } catch (NumberFormatException e) {
155         this.lengthField.setPromptText("This must be a number e.g. '20'");
156         this.lengthField.setText("");
157         return;
158       }
159       try {
160         temp = Double.valueOf(this.precisionField.getText());
161         map.put("precision", temp);
162       } catch (NumberFormatException e) {
163         this.precisionField.setPromptText("This must be a number e.g. '0.01'");
164         this.precisionField.setText("");
165         return;
166       }
167
168       temp = 0.0;
169       for (CheckBox didi : didiMap.keySet()) {
170         if (didi.isSelected() == true) {
171           map.put(this.didiMap.get(didi), 1.0);
172           temp = temp + 1.0;
173         }
174       }
175       if (temp == 0.0) {
176         map.put("constant", 1.0);
177       }
178       if ((map.get("length").doubleValue() / temp) - 2.0 < 10.00) {
179         temp = temp * 12.0;
180         this.lengthField.setPromptText(
181             String.format("Must have at least %s points for these functions",
                temp.intValue()));
182         this.lengthField.setText("");
183         return;
184       }
185
186       this.main.execute(map);
187     }
188
189     public void baseDefault() {
190       this.baseField.setText("0.0");
191     }
192
193     public void stepDefault() {
194       this.stepField.setText("1.0");
195     }
196
197     public void lengthDefault() {
198       this.lengthField.setText("50");
199     }
200
201     public void precisionDefault() {
202       this.precisionField.setText("0.01");
203     }
204
205     // public void setPrecision() {
206     // Double precision;
207     // try {
208     // precision = Double.parseDouble(this.precisionField.getText());
209     // } catch (NumberFormatException ne) {
210     // this.precisionField.setPromptText("Invalid Number Format!");
211     // this.precisionField.setText("");
```

```
212     // return;
213     // } catch (NullPointerException e) {
214     // return;
215     // }
216     // Tangenter.setPrecision(precision);
217     // this.precisionField.setPromptText("Adjust Precision");
218     // this.precisionField.setText("");
219     // System.out
220     // .println(String.format("User set new pprecision of %s in %s!", precision,
        Tangenter.class));
221     // }
222
223     // for GUIMain
224     //
        ************************************************************************************
        ************************************
225
226     public void giveGUIMain(GUIMainForFX guiMain) {
227       this.main = guiMain;
228     }
229
230     public void displayDataSet(ValueDataSet<? extends Number> dataSet) {
231       System.out.println("Displaying new DataSet");
232       this.setChart.setDataSet(dataSet);
233       // this.setChart.showData();
234     }
235
236   }
237
```