```java
package hyperDap.guiPres.charts;

import hyperDap.base.types.dataSet.ValueDataSet;
import javafx.application.Platform;
import javafx.scene.Node;
import javafx.scene.chart.LineChart;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.scene.layout.VBox;

/**
 * This display element is used to show the values of a {@link ValueDataSet} using a
 * {@link LineChart}.
 * <p>
 * It retains a reference to the {@link ValueDataSet} that it displays, allowing the set to be
 * altered externally, as long as {@link #show()} is not executed at the same time.
 *
 * @author soenk
 *
 */
public class DisplayDataSet extends VBox {

  private static String assertionErrorMessage = String
      .format("%s are not editable and no children may be added to it.",
      DisplayDataSet.class);

  private int counter = 0;
  private boolean displayRunning = false;

  private ValueDataSet<? extends Number> set;

  private LineChart<Number, Number> setChart;
  private LineChart<Number, Number> derivChart;
  private NumberAxis xSetAxis;
  private NumberAxis xDerivAxis;
  private NumberAxis ySetAxis;
  private NumberAxis yDerivAxis;

  private XYChart.Series<Number, Number> setSeries;
  private XYChart.Series<Number, Number> constSeries;
  private XYChart.Series<Number, Number> linearSeries;
  private XYChart.Series<Number, Number> SquareSeries;
  private XYChart.Series<Number, Number> cubicSeries;
  private XYChart.Series<Number, Number> expSeries;
  private XYChart.Series<Number, Number> sinSeries;
  private XYChart.Series<Number, Number> changeSeries;
  private XYChart.Series<Number, Number> undefinedSeries;

  // Constructors
  //
  ************************************************************************************
  ************************************

  public DisplayDataSet() {
    super();
    this.setUp();
  }

  public DisplayDataSet(double spacing) {
    super(spacing);
    this.setUp();
  }

  public DisplayDataSet(double spacing, Node... children) throws AssertionError {
    throw new AssertionError(assertionErrorMessage);
  }

  public DisplayDataSet(Node... children) throws AssertionError {
    throw new AssertionError(assertionErrorMessage);
  }

  public DisplayDataSet(ValueDataSet<Number> dataSet) {
```

```
70        this.set = dataSet;
71        this.setUp();
72        this.show();
73      }
74
75      /**
76       * This setup is performed independently of the constructor that is called.
77       *
78       * @category helper
79       * @category constructor
80       */
81      private void setUp() {
82        this.xSetAxis = new NumberAxis();
83        this.ySetAxis = new NumberAxis();
84        this.setChart = new LineChart<>(xSetAxis, ySetAxis);
85
86        this.addToChildren(this.setChart);
87
88        this.setSeries = new XYChart.Series<>();
89        this.setSeries.setName("DataSet");
90        this.setChart.getData().add(this.setSeries);
91
92        this.xDerivAxis = new NumberAxis();
93        this.yDerivAxis = new NumberAxis();
94        this.derivChart = new LineChart<>(this.xDerivAxis, this.yDerivAxis);
95
96        this.yDerivAxis.setLowerBound(-5.0);
97        this.yDerivAxis.setUpperBound(5.0);
98        this.yDerivAxis.setTickUnit(1.0);
99        this.yDerivAxis.setAutoRanging(false);
100       this.yDerivAxis.setMinorTickVisible(false);
101
102       this.addToChildren(this.derivChart);
103
104       this.constSeries = new XYChart.Series<>();
105       this.linearSeries = new XYChart.Series<>();
106       this.SquareSeries = new XYChart.Series<>();
107       this.cubicSeries = new XYChart.Series<>();
108       this.expSeries = new XYChart.Series<>();
109       this.sinSeries = new XYChart.Series<>();
110       this.changeSeries = new XYChart.Series<>();
111       this.undefinedSeries = new XYChart.Series<>();
112
113       this.constSeries.setName("Constant: 1");
114       this.linearSeries.setName("Linear: 2");
115       this.SquareSeries.setName("Square: 3");
116       this.cubicSeries.setName("Cubic: 4");
117       this.expSeries.setName("Exponential: -1");
118       this.sinSeries.setName("Trigonometric: -2");
119       this.changeSeries.setName("Point of Interest: 0");
120       this.undefinedSeries.setName("Undefined: -5");
121
122       this.derivChart.getData().add(this.constSeries);
123       this.derivChart.getData().add(this.linearSeries);
124       this.derivChart.getData().add(this.SquareSeries);
125       this.derivChart.getData().add(this.cubicSeries);
126       this.derivChart.getData().add(this.expSeries);
127       this.derivChart.getData().add(this.sinSeries);
128       this.derivChart.getData().add(this.changeSeries);
129       this.derivChart.getData().add(this.undefinedSeries);
130
131       this.setChart.setPrefHeight(250.0);
132       this.derivChart.setPrefHeight(250.0);
133     }
134
135     // setters
136     //
      ********************************************************************************
      **********************************
137
138     /**
139      * A private helper that allows internally editing children {@link Node Nodes}
       without exposing
```

```
140       * this to external classes.
141       *
142       * @param node The {@link Node} that is to be added.
143       * @category helper
144       */
145      private void addToChildren(Node node) {
146        super.getChildren().add(node);
147      }
148
149      /**
150       * Allows setting or altering the {@link ValueDataSet} that is displayed in this
          chart.
151       * <p>
152       * If a reference to this set is retained it can be edited without using this
          method again, as
153       * long as no race conditions are provoked with {@link #showData()}.
154       *
155       * @param dataSet
156       */
157      public void setDataSet(ValueDataSet<? extends Number> dataSet) {
158        this.set = dataSet;
159        this.showData();
160      }
161
162      /**
163       * Clear the current data display and display the current data points from the
          currently stored
164       * {@link ValueDataSet}.
165       * <p>
166       * If the {@link ValueDataSet} is undefined a warning is printed and no data is
          displayed.
167       */
168      @Deprecated
169      public void show() {
170        if (this.set == null) {
171          System.err.println(String.format("%s.set is undefined!", DisplayDataSet.class));
172          boolean first = true;
173          for (StackTraceElement element : Thread.currentThread().getStackTrace()) {
174            if (first == true) {
175              first = false;
176              continue;
177            }
178            System.err.println(element.toString());
179          }
180          return;
181        }
182
183        this.set.calcDerivDepths();
184        this.setSeries.getData().clear();
185        double xVal;
186        int depth;
187        for (int i = 0; i < this.set.size(); i++) {
188          xVal = this.set.getIndependentValue(i);
189          this.setSeries.getData().add(new XYChart.Data<Number, Number>(xVal,
190              this.set.getByIndex(i)/* ,this.set.getDerivDepthsByIndex(i) */));
191          depth = this.set.getDerivDepthsByIndex(i);
192          this.switchSeries(depth).getData().add(new XYChart.Data<Number, Number>(xVal,
              depth));
193        }
194      }
195
196      private XYChart.Series<Number, Number> switchSeries(int derivDepth) {
197        XYChart.Series<Number, Number> ser;
198        switch (derivDepth) {
199          case 0:
200            ser = this.constSeries;
201            break;
202          case 1:
203            ser = this.linearSeries;
204            break;
205          case 2:
206            ser = this.SquareSeries;
207            break;
```

```
208         case 3:
209           ser = this.cubicSeries;
210           break;
211         case -1:
212           ser = this.changeSeries;
213           break;
214         case -2:
215           ser = this.expSeries;
216           break;
217         case -3:
218           ser = this.sinSeries;
219           break;
220         default:
221           ser = this.undefinedSeries;
222       }
223       return ser;
224     }
225
226     private void clearSeries() {
227       Platform.runLater(new Runnable() {
228         @Override
229         public void run() {
230           resetAllSeries();
231         }
232       });
233     }
234
235     private void resetAllSeries() {
236       double base = set.getBase();
237       double max = set.size() * set.getStep() + base;
238       setSeries.getData().clear();
239       resetSeries(constSeries, base, max, 1);
240       resetSeries(linearSeries, base, max, 2);
241       resetSeries(SquareSeries, base, max, 3);
242       resetSeries(cubicSeries, base, max, 4);
243       resetSeries(expSeries, base, max, -1);
244       resetSeries(sinSeries, base, max, -2);
245       resetSeries(changeSeries, base, max, 0);
246       resetSeries(undefinedSeries, base, max, -5);
247     }
248
249     private void resetSeries(XYChart.Series<Number, Number> series, double base,
       double max,
250         double val) {
251       // series.getData().clear();
252       // series.getData().add(new XYChart.Data<Number, Number>(base, val));
253       // series.getData().add(new XYChart.Data<Number, Number>(max, val));
254       series.getData().clear();
255     }
256
257     /**
258      * Initiates the recursive addition of data points from the internal {@link
        ValueDataSet} to the
259      * two displayed graphs. The {@link #runLaterCall()} and {@link #displayPoints()}
        methods are
260      * called recursively to count through the entire {@link ValueDataSet}.
261      * <p>
262      * This is done successively to prevent a slow-down of the gui while
263      * {@link Platform#runLater(Runnable)} is executed, which is necessary to prevent
        race conditions
264      * in relation to JavaFX elements.
265      */
266     public void showData() {
267       // in case showData is already in progress
268       if (this.displayRunning) {
269         // notify the progress to terminate
270         this.displayRunning = false;
271         // print warning
272         System.err.println(String.format("%s aborting the running display!",
          DisplayDataSet.class));
273         boolean first = true;
274         for (StackTraceElement element : Thread.currentThread().getStackTrace()) {
275           if (first == true) {
```

```
276             first = false;
277             continue;
278           }
279           System.err.println(element.toString());
280         }
281         // wait for progress to terminate
282         while (this.displayRunning == false) {
283           // TODO timeout -> throw runtime-exception
284           try {
285             Thread.sleep(100);
286           } catch (InterruptedException e) {
287             e.printStackTrace();
288           }
289         }
290       } else {
291         System.out.println("Initiation runLater recursion to display DataSet");
292       }
293       // Platform.runLater for every 10 data points
294       this.clearSeries();
295       this.counter = 0;
296       this.displayRunning = true;
297       this.runLaterCall();
298     }
299
300     /**
301      * Calls {@link Platform#runLater(Runnable)} to update data points in charts by
         calling
302      * {@link #displayPoints()} without causing race conditions.
303      */
304     private void runLaterCall() {
305       Platform.runLater(new Runnable() {
306         @Override
307         public void run() {
308           System.out.println(String.format("Executing runLater for DataSet display,
             counter: %s/%s",
309               counter, set.size()));
310           displayPoints();
311         }
312       });
313     }
314
315     /**
316      * This method should only be called by {@link Platform#runLater(Runnable)} to
         prevent race
317      * conditions within JavaFX elements.
318      * <p>
319      * It adds the next 10 data points and their derivDepths to the respective graphs,
         before calling
320      * {@link #runLaterCall()} to recursively continue he process, if necessary. A
         class internal
321      * counter is used to count through the entire set in the recursive method
         invocations.
322      */
323     public void displayPoints() {
324       // terminate if showData() was called again
325       if (this.displayRunning == false) {
326         this.displayRunning = true;
327         return;
328       }
329       // only add the next 10 values, or until the end of the set
330       int max = this.counter + 10;
331       if (max > this.set.size()) {
332         max = this.set.size();
333       }
334       // add values
335       double xVal;
336       int depth;
337       while (this.counter < max) {
338         // add data point to setSeries
339         xVal = this.set.getIndependentValue(counter);
340         if (this.set.getValidByIndex(counter) == true) {
341           this.setSeries.getData()
342               .add(new XYChart.Data<Number, Number>(xVal,
```

```
                this.set.getByIndex(counter)));
343         }
344         // add derivDepth to the correct series
345         depth = this.set.getDerivDepthsByIndex(counter);
346         try {
347           this.switchSeries(depth).getData().add(new XYChart.Data<Number,
              Number>(xVal, depth + 1));
348         } catch (NullPointerException e) {
349           System.err.println(String.format("Undefined derivDepth for %s at index %s!",
350               DisplayDataSet.class, counter));
351           this.undefinedSeries.getData().add(new XYChart.Data<Number, Number>(xVal,
              -5.0));
352         }
353         // TODO do not show last 10 derivDepths on chart. could remove them
354         // counter
355         this.counter++;
356       }
357       // add another runLater if needed
358       if (this.counter < this.set.size() && this.displayRunning == true) {
359         this.runLaterCall();
360       } else {
361         this.displayRunning = false;
362       }
363     }
364
365   }
366
```