

```
1 import java.awt.Cursor;
2 import java.awt.FlowLayout;
3 import java.awt.GridLayout;
4 import java.awt.event.ActionEvent;
5
6 import javax.swing.JButton;
7 import javax.swing.JFrame;
8 import javax.swing.JPanel;
9 import javax.swing.JScrollPane;
10 import javax.swing.JTextArea;
11
12 import components.naturalnumber.NaturalNumber;
13
14 /**
15  * View class.
16  *
17  * @author David Park
18  */
19 public final class NNCalcView1 extends JFrame implements NNCalcView {
20
21     /**
22      * Controller object registered with this view to observe user-interaction
23      * events.
24      */
25     private NNCalcController controller;
26
27     /**
28      * State of user interaction: last event "seen".
29      */
30     private enum State {
31         /**
32          * Last event was clear, enter, another operator, or digit entry, resp.
33          */
34         SAW_CLEAR, SAW_ENTER_OR_SWAP, SAW_OTHER_OP, SAW_DIGIT
35     }
36
37     /**
38      * State variable to keep track of which event happened last; needed to
39      * prepare for digit to be added to bottom operand.
40      */
41     private State currentState;
42
43     /**
44      * Text areas.
45      */
46     private final JTextArea tTop, tBottom;
47
48     /**
49      * Operator and related buttons.
50      */
51     private final JButton bClear, bSwap, bEnter, bAdd, bSubtract, bMultiply,
52         bDivide, bPower, bRoot;
53
54     /**
55      * Digit entry buttons.
56      */
57     private final JButton[] bDigits;
```

```

58
59  /**
60   * Useful constants.
61   */
62  private static final int TEXT_AREA_HEIGHT = 5, TEXT_AREA_WIDTH = 20,
63      DIGIT_BUTTONS = 10, MAIN_BUTTON_PANEL_GRID_ROWS = 4,
64      MAIN_BUTTON_PANEL_GRID_COLUMNS = 4, SIDE_BUTTON_PANEL_GRID_ROWS = 3,
65      SIDE_BUTTON_PANEL_GRID_COLUMNS = 1, CALC_GRID_ROWS = 3,
66      CALC_GRID_COLUMNS = 1;
67
68  /**
69   * No argument constructor.
70   */
71  public NNCalcView1() {
72      // Create the JFrame being extended
73
74      /*
75       * Call the JFrame (superclass) constructor with a String parameter to
76       * name the window in its title bar
77       */
78      super("Natural Number Calculator");
79
80      // Set up the GUI widgets -----
81
82      /*
83       * Set up initial state of GUI to behave like last event was "Clear";
84       * currentState is not a GUI widget per se, but is needed to process
85       * digit button events appropriately
86       */
87      this.currentState = State.SAW_CLEAR;
88      this.tTop = new JTextArea("", TEXT_AREA_HEIGHT, TEXT_AREA_WIDTH);
89      this.tBottom = new JTextArea("", TEXT_AREA_HEIGHT, TEXT_AREA_WIDTH);
90      this.bClear = new JButton("Clear");
91      this.bSwap = new JButton("Swap");
92      this.bEnter = new JButton("Enter");
93      this.bAdd = new JButton("+");
94      this.bSubtract = new JButton("-");
95      this.bMultiply = new JButton("*");
96      this.bDivide = new JButton("/");
97      this.bPower = new JButton("Power");
98      this.bRoot = new JButton("Root");
99
100     /*
101      * Create widgets
102      */
103
104     this.bDigits = new JButton[DIGIT_BUTTONS];
105     for (int i = 0; i < DIGIT_BUTTONS; i++) {
106         this.bDigits[i] = new JButton(Integer.toString(i));
107     }
108
109     // Set up the GUI widgets -----
110
111     /*
112      * Text areas should wrap lines, and should be read-only; they cannot be
113      * edited because allowing keyboard entry would require checking whether
114      * entries are digits, which we don't want to have to do

```

```
115         */
116
117         this.tTop.setEditable(false);
118         this.tBottom.setEditable(false);
119         this.tTop.setLineWrap(true);
120         this.tBottom.setLineWrap(true);
121         this.tTop.setWrapStyleWord(true);
122         this.tBottom.setWrapStyleWord(true);
123
124         /*
125          * Initially, the following buttons should be disabled: divide (divisor
126          * must not be 0) and root (root must be at least 2) -- hint: see the
127          * JButton method setEnabled
128          */
129
130         this.bDivide.setEnabled(false);
131         this.bRoot.setEnabled(false);
132
133         /*
134          * Create scroll panes for the text areas in case number is long enough
135          * to require scrolling
136          */
137
138         JScrollPane scrollTTop = new JScrollPane(this.tTop);
139         JScrollPane scrollTBottom = new JScrollPane(this.tBottom);
140
141         /*
142          * Create main button panel
143          */
144
145         JPanel buttonPanel = new JPanel(new GridLayout(
146             MAIN_BUTTON_PANEL_GRID_ROWS, MAIN_BUTTON_PANEL_GRID_COLUMNS));
147
148         /*
149          * Add the buttons to the main button panel, from left to right and top
150          * to bottom
151          */
152
153         buttonPanel.add(this.bDigits[Integer.parseInt("7")]);
154         buttonPanel.add(this.bDigits[Integer.parseInt("8")]);
155         buttonPanel.add(this.bDigits[Integer.parseInt("9")]);
156         buttonPanel.add(this.bAdd);
157         buttonPanel.add(this.bDigits[Integer.parseInt("4")]);
158         buttonPanel.add(this.bDigits[Integer.parseInt("5")]);
159         buttonPanel.add(this.bDigits[Integer.parseInt("6")]);
160         buttonPanel.add(this.bSubtract);
161         buttonPanel.add(this.bDigits[Integer.parseInt("1")]);
162         buttonPanel.add(this.bDigits[Integer.parseInt("2")]);
163         buttonPanel.add(this.bDigits[Integer.parseInt("3")]);
164         buttonPanel.add(this.bMultiply);
165         buttonPanel.add(this.bDigits[0]);
166         buttonPanel.add(this.bPower);
167         buttonPanel.add(this.bRoot);
168         buttonPanel.add(this.bDivide);
169
170         /*
171          * Create side button panel
```

```
172     */
173
174     JPanel sideButtonPanel = new JPanel(new GridLayout(
175         SIDE_BUTTON_PANEL_GRID_ROWS, SIDE_BUTTON_PANEL_GRID_COLUMNS));
176
177     /*
178     * Add the buttons to the side button panel, from left to right and top
179     * to bottom
180     */
181
182     sideButtonPanel.add(this.bClear);
183     sideButtonPanel.add(this.bSwap);
184     sideButtonPanel.add(this.bEnter);
185
186     /*
187     * Create combined button panel organized using flow layout, which is
188     * simple and does the right thing: sizes of nested panels are natural,
189     * not necessarily equal as with grid layout
190     */
191
192     JPanel combinedButtonPanel = new JPanel(new FlowLayout());
193
194     /*
195     * Add the other two button panels to the combined button panel
196     */
197
198     combinedButtonPanel.add(buttonPanel);
199     combinedButtonPanel.add(sideButtonPanel);
200
201     /*
202     * Organize main window
203     */
204
205     this.setLayout(new GridLayout(CALC_GRID_ROWS, CALC_GRID_COLUMNS));
206
207     /*
208     * Add scroll panes and button panel to main window, from left to right
209     * and top to bottom
210     */
211
212     this.add(scrollTTop);
213     this.add(scrollTBottom);
214     this.add(combinedButtonPanel);
215
216     // Set up the observers -----
217
218     this.bClear.addActionListener(this);
219     this.bSwap.addActionListener(this);
220     this.bEnter.addActionListener(this);
221     this.bAdd.addActionListener(this);
222     this.bSubtract.addActionListener(this);
223     this.bMultiply.addActionListener(this);
224     this.bDivide.addActionListener(this);
225     this.bPower.addActionListener(this);
226     this.bRoot.addActionListener(this);
227
228     for (int i = 0; i < DIGIT_BUTTONS; i++) {
```

```
229         this.bDigits[i].addActionListener(this);
230     }
231
232     /*
233     * Register this object as the observer for all GUI events
234     */
235
236     // Set up the main application window -----
237
238     this.pack();
239     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
240     this.setVisible(true);
241     // this.tTop.setText("0");
242     // this.tBottom.setText("0");
243
244     /*
245     * Make sure the main window is appropriately sized, exits this program
246     * on close, and becomes visible to the user
247     */
248
249 }
250
251 @Override
252 public void registerObserver(NNCalcController controller) {
253     // Set controller for controller
254     this.controller = controller;
255 }
256
257 @Override
258 public void updateTopDisplay(NaturalNumber n) {
259     // Set text for bTop
260     this.tTop.setText(n.toString());
261 }
262
263 @Override
264 public void updateBottomDisplay(NaturalNumber n) {
265     // Set text for bBottom
266     this.tBottom.setText(n.toString());
267 }
268
269 @Override
270 public void updateSubtractAllowed(boolean allowed) {
271     // Set enable for bSubtract
272     this.bSubtract.setEnabled(allowed);
273 }
274
275 @Override
276 public void updateDivideAllowed(boolean allowed) {
277     // Set enable for bDivide
278     this.bDivide.setEnabled(allowed);
279 }
280
281 @Override
282 public void updatePowerAllowed(boolean allowed) {
283     // Set enable for bPower
284     this.bPower.setEnabled(allowed);
285 }
```

```
286
287     @Override
288     public void updateRootAllowed(boolean allowed) {
289         // Set enable for bRoot
290         this.bRoot.setEnabled(allowed);
291     }
292
293     @Override
294     public void actionPerformed(ActionEvent event) {
295         /*
296          * Set cursor to indicate computation on-going; this matters only if
297          * processing the event might take a noticeable amount of time as seen
298          * by the user
299          */
300         this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
301         /*
302          * Determine which event has occurred that we are being notified of by
303          * this callback; in this case, the source of the event (i.e, the widget
304          * calling actionPerformed) is all we need because only buttons are
305          * involved here, so the event must be a button press; in each case,
306          * tell the controller to do whatever is needed to update the model and
307          * to refresh the view
308          */
309         Object source = event.getSource();
310         if (source == this.bClear) {
311             this.controller.processClearEvent();
312             this.currentState = State.SAW_CLEAR;
313         } else if (source == this.bSwap) {
314             this.controller.processSwapEvent();
315             this.currentState = State.SAW_ENTER_OR_SWAP;
316         } else if (source == this.bEnter) {
317             this.controller.processEnterEvent();
318             this.currentState = State.SAW_ENTER_OR_SWAP;
319         } else if (source == this.bAdd) {
320             this.controller.processAddEvent();
321             this.currentState = State.SAW_OTHER_OP;
322         } else if (source == this.bSubtract) {
323             this.controller.processSubtractEvent();
324             this.currentState = State.SAW_OTHER_OP;
325         } else if (source == this.bMultiply) {
326             this.controller.processMultiplyEvent();
327             this.currentState = State.SAW_OTHER_OP;
328         } else if (source == this.bDivide) {
329             this.controller.processDivideEvent();
330             this.currentState = State.SAW_OTHER_OP;
331         } else if (source == this.bPower) {
332             this.controller.processPowerEvent();
333             this.currentState = State.SAW_OTHER_OP;
334         } else if (source == this.bRoot) {
335             this.controller.processRootEvent();
336             this.currentState = State.SAW_OTHER_OP;
337         } else {
338             for (int i = 0; i < DIGIT_BUTTONS; i++) {
339                 if (source == this.bDigits[i]) {
340                     switch (this.currentState) {
341                         case SAW_ENTER_OR_SWAP:
342                             this.controller.processClearEvent();
```

```
343             break;
344         case SAW_OTHER_OP:
345             this.controller.processEnterEvent();
346             this.controller.processClearEvent();
347             break;
348         default:
349             break;
350     }
351     this.controller.processAddNewDigitEvent(i);
352     this.currentState = State.SAW_DIGIT;
353     break;
354 }
355 }
356 }
357 /*
358  * Set the cursor back to normal (because we changed it at the beginning
359  * of the method body)
360  */
361 this.setCursor(Cursor.getDefaultCursor());
362 }
363 }
364 }
```