```java
import java.awt.Cursor;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

import components.naturalnumber.NaturalNumber;

/**
 * View class.
 *
 * @author Steven Masilonis
 */
public final class NNCalcView1 extends JFrame implements
NNCalcView {

    /**
     * Controller object registered with this view to observe
user-interaction
     * events.
     */
    private NNCalcController controller;

    /**
     * State of user interaction: last event "seen".
     */
    private enum State {
        /**
         * Last event was clear, enter, another operator, or digit
entry, resp.
         */
        SAW_CLEAR, SAW_ENTER_OR_SWAP, SAW_OTHER_OP, SAW_DIGIT
    }

    /**
     * State variable to keep track of which event happened last;
needed to
     * prepare for digit to be added to bottom operand.
     */
```

```
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       * Default constructor.
70       */
71      public NNCalcView1() {
72          // Create the JFrame being extended
73          /*
74           * Call the JFrame (superclass) constructor with a String
    parameter to
75           * name the window in its title bar
76           */
77          super("Natural Number Calculator");
78
79          // Set up the GUI widgets
```

```
   ------------------------------------------
 80
 81         /*
 82          * Set up initial state of GUI to behave like last event
   was "Clear";
 83          * currentState is not a GUI widget per se, but is needed
   to process
 84          * digit button events appropriately
 85          */
 86         this.currentState = State.SAW_CLEAR;
 87
 88         /*
 89          * Create widgets
 90          */
 91         this.tTop = new JTextArea("", TEXT_AREA_HEIGHT,
   TEXT_AREA_WIDTH);
 92         this.tBottom = new JTextArea("", TEXT_AREA_HEIGHT,
   TEXT_AREA_WIDTH);
 93         this.bClear = new JButton("Clear");
 94         this.bSwap = new JButton("Swap");
 95         this.bEnter = new JButton("Enter");
 96         this.bAdd = new JButton("Add");
 97         this.bSubtract = new JButton("Subtract");
 98         this.bMultiply = new JButton("Multiply");
 99         this.bDivide = new JButton("Divide");
100         this.bPower = new JButton("Power");
101         this.bRoot = new JButton("Root");
102         this.bDigits = new JButton[DIGIT_BUTTONS];
103         for (int i = 0; i < DIGIT_BUTTONS; i++) {
104             this.bDigits[i] = new JButton("" + i);
105         }
106         // Set up the GUI widgets
   ------------------------------------------
107
108         /*
109          * Text areas should wrap lines, and should be read-only;
   they cannot be
110          * edited because allowing keyboard entry would require
   checking whether
111          * entries are digits, which we don't want to have to do
112          */
113         this.tTop.setEditable(false);
114         this.tTop.setLineWrap(true);
115         this.tTop.setWrapStyleWord(true);
```

```
116         this.tBottom.setEditable(false);
117         this.tBottom.setLineWrap(true);
118         this.tBottom.setWrapStyleWord(true);
119 
120         /*
121          * Initially, the following buttons should be disabled:
    divide (divisor
122          * must not be 0) and root (root must be at least 2) --
    hint: see the
123          * JButton method setEnabled
124          */
125         this.bDivide.setEnabled(false);
126         this.bRoot.setEnabled(false);
127         /*
128          * Create scroll panes for the text areas in case number
    is long enough
129          * to require scrolling
130          */
131         JScrollPane scrollPaneforTop = new JScrollPane(this.tTop);
132         JScrollPane scrollPaneforBottom = new
    JScrollPane(this.tBottom);
133         /*
134          * Create main button panel
135          */
136         JPanel mainpane1 = new JPanel(new GridLayout(
137                 MAIN_BUTTON_PANEL_GRID_ROWS,
    MAIN_BUTTON_PANEL_GRID_COLUMNS));
138         /*
139          * Add the buttons to the main button panel, from left to
    right and top
140          * to bottom
141          */
142         mainpane1.add(this.bDigits[7]);
143         mainpane1.add(this.bDigits[8]);
144         mainpane1.add(this.bDigits[9]);
145         mainpane1.add(this.bAdd);
146         mainpane1.add(this.bDigits[4]);
147         mainpane1.add(this.bDigits[5]);
148         mainpane1.add(this.bDigits[6]);
149         mainpane1.add(this.bSubtract);
150         mainpane1.add(this.bDigits[1]);
151         mainpane1.add(this.bDigits[2]);
152         mainpane1.add(this.bDigits[3]);
153         mainpane1.add(this.bMultiply);
```

```
154         mainpane1.add(this.bDigits[0]);
155         mainpane1.add(this.bPower);
156         mainpane1.add(this.bRoot);
157         mainpane1.add(this.bDivide);
158         /*
159          * Create side button panel
160          */
161         JPanel sidepane1 = new JPanel(new GridLayout(
162                 SIDE_BUTTON_PANEL_GRID_ROWS,
    SIDE_BUTTON_PANEL_GRID_COLUMNS));
163         /*
164          * Add the buttons to the side button panel, from left to
    right and top
165          * to bottom
166          */
167         sidepane1.add(this.bClear);
168         sidepane1.add(this.bSwap);
169         sidepane1.add(this.bEnter);
170         /*
171          * Create combined button panel organized using flow
    layout, which is
172          * simple and does the right thing: sizes of nested panels
    are natural,
173          * not necessarily equal as with grid layout
174          */
175         JPanel combpane1 = new JPanel(new FlowLayout());
176         /*
177          * Add the other two button panels to the combined button
    panel
178          */
179         combpane1.add(mainpane1);
180         combpane1.add(sidepane1);
181         /*
182          * Organize main window
183          */
184         this.setLayout(new GridLayout(CALC_GRID_ROWS,
    CALC_GRID_COLUMNS));
185         /*
186          * Add scroll panes and button panel to main window, from
    left to right
187          * and top to bottom
188          */
189         this.add(scrollPaneforTop);
190         this.add(scrollPaneforBottom);
```

```
191         this.add(combpane1);
192 
193         // Set up the observers
    ------------------------------------------
194 
195         /*
196          * Register this object as the observer for all GUI events
197          */
198         this.bClear.addActionListener(this);
199         this.bSwap.addActionListener(this);
200         this.bEnter.addActionListener(this);
201         this.bAdd.addActionListener(this);
202         this.bSubtract.addActionListener(this);
203         this.bMultiply.addActionListener(this);
204         this.bDivide.addActionListener(this);
205         this.bPower.addActionListener(this);
206         this.bRoot.addActionListener(this);
207         for (int i = 0; i < DIGIT_BUTTONS; i++) {
208             this.bDigits[i].addActionListener(this);
209         }
210         // Set up the main application window
    --------------------------------
211 
212         /*
213          * Make sure the main window is appropriately sized, exits
    this program
214          * on close, and becomes visible to the user
215          */
216         this.pack();
217         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
218         this.setVisible(true);
219 
220     }
221 
222     @Override
223     public void registerObserver(NNCalcController controller) {
224 
225         this.controller = controller;
226 
227     }
228 
229     @Override
230     public void updateTopDisplay(NaturalNumber n) {
231 
```

```
232         this.tTop.setText(n.toString());
233 
234     }
235 
236     @Override
237     public void updateBottomDisplay(NaturalNumber n) {
238 
239         this.tBottom.setText(n.toString());
240 
241     }
242 
243     @Override
244     public void updateSubtractAllowed(boolean allowed) {
245 
246         this.bSubtract.setEnabled(allowed);
247 
248     }
249 
250     @Override
251     public void updateDivideAllowed(boolean allowed) {
252 
253         this.bDivide.setEnabled(allowed);
254 
255     }
256 
257     @Override
258     public void updatePowerAllowed(boolean allowed) {
259 
260         this.bPower.setEnabled(allowed);
261 
262     }
263 
264     @Override
265     public void updateRootAllowed(boolean allowed) {
266 
267         this.bRoot.setEnabled(allowed);
268 
269     }
270 
271     @Override
272     public void actionPerformed(ActionEvent event) {
273         /*
274          * Set cursor to indicate computation on-going; this
    matters only if
```

```
275          * processing the event might take a noticeable amount of
    time as seen
276          * by the user
277          */
278
    this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
279         /*
280          * Determine which event has occurred that we are being
    notified of by
281          * this callback; in this case, the source of the event
    (i.e, the widget
282          * calling actionPerformed) is all we need because only
    buttons are
283          * involved here, so the event must be a button press; in
    each case,
284          * tell the controller to do whatever is needed to update
    the model and
285          * to refresh the view
286          */
287         Object source = event.getSource();
288         if (source == this.bClear) {
289             this.controller.processClearEvent();
290             this.currentState = State.SAW_CLEAR;
291         } else if (source == this.bSwap) {
292             this.controller.processSwapEvent();
293             this.currentState = State.SAW_ENTER_OR_SWAP;
294         } else if (source == this.bEnter) {
295             this.controller.processEnterEvent();
296             this.currentState = State.SAW_ENTER_OR_SWAP;
297         } else if (source == this.bAdd) {
298             this.controller.processAddEvent();
299             this.currentState = State.SAW_OTHER_OP;
300         } else if (source == this.bSubtract) {
301             this.controller.processSubtractEvent();
302             this.currentState = State.SAW_OTHER_OP;
303         } else if (source == this.bMultiply) {
304             this.controller.processMultiplyEvent();
305             this.currentState = State.SAW_OTHER_OP;
306         } else if (source == this.bDivide) {
307             this.controller.processDivideEvent();
308             this.currentState = State.SAW_OTHER_OP;
309         } else if (source == this.bPower) {
310             this.controller.processPowerEvent();
311             this.currentState = State.SAW_OTHER_OP;
```

```
312         } else if (source == this.bRoot) {
313             this.controller.processRootEvent();
314             this.currentState = State.SAW_OTHER_OP;
315         } else {
316             for (int i = 0; i < DIGIT_BUTTONS; i++) {
317                 if (source == this.bDigits[i]) {
318                     switch (this.currentState) {
319                         case SAW_ENTER_OR_SWAP:
320                             this.controller.processClearEvent();
321                             break;
322                         case SAW_OTHER_OP:
323                             this.controller.processEnterEvent();
324                             this.controller.processClearEvent();
325                             break;
326                         default:
327                             break;
328                     }
329                     this.controller.processAddNewDigitEvent(i);
330                     this.currentState = State.SAW_DIGIT;
331                     break;
332                 }
333             }
334         }
335         /*
336          * Set the cursor back to normal (because we changed it at
    the beginning
337          * of the method body)
338          */
339         this.setCursor(Cursor.getDefaultCursor());
340     }
341 
342 }
343 
```