```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 Jeng Zhuang
 */
@SuppressWarnings("serial")
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
    }
```

```
37
38      /**
39       * State variable to keep track of which event happened
   last; needed to
40       * prepare for digit to be added to bottom operand.
41       */
42      private State currentState;
43
44      /**
45       * Text areas.
46       */
47      private final JTextArea tTop, tBottom;
48
49      /**
50       * Operator and related buttons.
51       */
52      private final JButton bClear, bSwap, bEnter, bAdd,
   bSubtract, bMultiply, bDivide,
53              bPower, bRoot;
54
55      /**
56       * Digit entry buttons.
57       */
58      private final JButton[] bDigits;
59
60      /**
61       * Useful constants.
62       */
63      private static final int TEXT_AREA_HEIGHT = 5,
   TEXT_AREA_WIDTH = 20,
64              DIGIT_BUTTONS = 10, MAIN_BUTTON_PANEL_GRID_ROWS =
   4,
65              MAIN_BUTTON_PANEL_GRID_COLUMNS = 4,
   SIDE_BUTTON_PANEL_GRID_ROWS = 3,
66              SIDE_BUTTON_PANEL_GRID_COLUMNS = 1, CALC_GRID_ROWS
   = 3, CALC_GRID_COLUMNS = 1,
67              SEVEN = 7, TEN = 10, FOUR = 4, ONE = 1;
68
69      /**
```

```
 70      * No argument constructor.
 71      */
 72     public NNCalcView1() {
 73         // Create the JFrame being extended
 74 
 75         /*
 76          * Call the JFrame (superclass) constructor with a
    String parameter to
 77          * name the window in its title bar
 78          */
 79         super("Natural Number Calculator");
 80 
 81         // Set up the GUI widgets
    ----------------------------------------
 82 
 83         /*
 84          * Set up initial state of GUI to behave like last
    event was "Clear";
 85          * currentState is not a GUI widget per se, but is
    needed to process
 86          * digit button events appropriately
 87          */
 88         this.currentState = State.SAW_CLEAR;
 89 
 90         /*
 91          * Create widgets
 92          */
 93 
 94         this.tTop = new JTextArea("", TEXT_AREA_HEIGHT,
    TEXT_AREA_WIDTH);
 95         this.tBottom = new JTextArea("", TEXT_AREA_HEIGHT,
    TEXT_AREA_WIDTH);
 96         this.bAdd = new JButton("+");
 97         this.bClear = new JButton("Clear");
 98         this.bDivide = new JButton("/");
 99         this.bEnter = new JButton("Enter");
100         this.bMultiply = new JButton("*");
101         this.bPower = new JButton("Power");
102         this.bRoot = new JButton("Root");
```

```
103         this.bSubtract = new JButton("-");
104         this.bSwap = new JButton("Swap");
105         this.bDigits = new JButton[DIGIT_BUTTONS];
106         for (int i = 0; i < this.bDigits.length; i++) {
107             this.bDigits[i] = new JButton(Integer.toString(i));
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.tTop.setLineWrap(true);
119         this.tTop.setWrapStyleWord(true);
120         this.tBottom.setEditable(false);
121         this.tBottom.setLineWrap(true);
122         this.tBottom.setWrapStyleWord(true);
123         /*
124          * Initially, the following buttons should be disabled:
    divide (divisor
125          * must not be 0) and root (root must be at least 2) --
    hint: see the
126          * JButton method setEnabled
127          */
128         this.bDivide.setEnabled(false);
129         this.bRoot.setEnabled(false);
130 
131         /*
132          * Create scroll panes for the text areas in case
    number is long enough
133          * to require scrolling
134          */
```

```
135
136         JScrollPane scrollTop = new JScrollPane(this.tTop);
137         JScrollPane scrollBottom = new
    JScrollPane(this.tBottom);
138         /*
139          * Create main button panel
140          */
141
142         JPanel mainButtonPanel = new JPanel(new
    GridLayout(MAIN_BUTTON_PANEL_GRID_ROWS,
143                 MAIN_BUTTON_PANEL_GRID_COLUMNS));
144         /*
145          * Add the buttons to the main button panel, from left
    to right and top
146          * to bottom
147          */
148
149         for (int i = SEVEN; i < TEN; i++) {
150             mainButtonPanel.add(this.bDigits[i]);
151         }
152         mainButtonPanel.add(this.bAdd);
153         for (int i = FOUR; i < SEVEN; i++) {
154             mainButtonPanel.add(this.bDigits[i]);
155         }
156         mainButtonPanel.add(this.bSubtract);
157         for (int i = ONE; i < FOUR; i++) {
158             mainButtonPanel.add(this.bDigits[i]);
159         }
160         mainButtonPanel.add(this.bMultiply);
161         mainButtonPanel.add(this.bDigits[0]);
162         mainButtonPanel.add(this.bDivide);
163         mainButtonPanel.add(this.bPower);
164         mainButtonPanel.add(this.bRoot);
165         /*
166          * Create side button panel
167          */
168         JPanel sideButtonPanel = new JPanel(new
    GridLayout(SIDE_BUTTON_PANEL_GRID_ROWS,
169                 SIDE_BUTTON_PANEL_GRID_COLUMNS));
```

```
170
171         /*
172          * Add the buttons to the side button panel, from left
    to right and top
173          * to bottom
174          */
175
176         sideButtonPanel.add(this.bClear);
177         sideButtonPanel.add(this.bSwap);
178         sideButtonPanel.add(this.bEnter);
179
180         /*
181          * Create combined button panel organized using flow
    layout, which is
182          * simple and does the right thing: sizes of nested
    panels are natural,
183          * not necessarily equal as with grid layout
184          */
185         JPanel combinedButtonPanel = new JPanel(new
    FlowLayout());
186
187         /*
188          * Add the other two button panels to the combined
    button panel
189          */
190
191         combinedButtonPanel.add(mainButtonPanel);
192         combinedButtonPanel.add(sideButtonPanel);
193         /*
194          * Organize main window
195          */
196         this.setLayout(new GridLayout(CALC_GRID_ROWS,
    CALC_GRID_COLUMNS));
197
198         /*
199          * Add scroll panes and button panel to main window,
    from left to right
200          * and top to bottom
201          */
```

```
202         this.add(scrollTop);
203         this.add(scrollBottom);
204         this.add(combinedButtonPanel);
205 
206         // Set up the observers
    ----------------------------------------------
207 
208         /*
209          * Register this object as the observer for all GUI
    events
210          */
211         this.bClear.addActionListener(this);
212         this.bSwap.addActionListener(this);
213         this.bEnter.addActionListener(this);
214         this.bAdd.addActionListener(this);
215         this.bSubtract.addActionListener(this);
216         this.bMultiply.addActionListener(this);
217         this.bDivide.addActionListener(this);
218         this.bPower.addActionListener(this);
219         this.bRoot.addActionListener(this);
220 
221         for (int i = 0; i < DIGIT_BUTTONS; i++) {
222             this.bDigits[i].addActionListener(this);
223         }
224 
225         // Set up the main application window
    --------------------------------
226 
227         /*
228          * Make sure the main window is appropriately sized,
    exits this program
229          * on close, and becomes visible to the user
230          */
231         this.pack();
232         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
233         this.setVisible(true);
234 
235     }
236 
```

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

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

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

```
342
343 }
344
```