```java
import java.awt.Cursor;

/**
 * View class.
 *
 * @author Micah Casey-Fusco
 */
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.
     */
    private State currentState;

    /**
     * Text areas.
     */
    private final JTextArea txtTop, txtBottom;

    /**
     * Operator and related buttons.
     */
    private final JButton btnClear, btnSwap, btnEnter, btnAdd, btnSubtract,
            btnMultiply, btnDivide, btnPower, btnRoot;

    /**
     * Digit entry buttons.
     */
    private final JButton[] btnDigits;

    /**
     * Useful constants.
     */
    private static final int TEXT_AREA_HEIGHT = 5, TEXT_AREA_WIDTH = 20,
            DIGIT_BUTTONS = 10, MAIN_BUTTON_PANEL_GRID_ROWS = 4,
            MAIN_BUTTON_PANEL_GRID_COLUMNS = 4, SIDE_BUTTON_PANEL_GRID_ROWS = 3,
            SIDE_BUTTON_PANEL_GRID_COLUMNS = 1, CALC_GRID_ROWS = 3,
            CALC_GRID_COLUMNS = 1;

    /**
```

```
 69      * Default 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 
 89         /*
 90          * Create widgets
 91          */
 92         this.txtTop = new JTextArea("0", TEXT_AREA_HEIGHT, TEXT_AREA_WIDTH);
 93         this.txtBottom = new JTextArea("0", TEXT_AREA_HEIGHT, TEXT_AREA_WIDTH);
 94         this.btnClear = new JButton("Clear");
 95         this.btnSwap = new JButton("Swap");
 96         this.btnEnter = new JButton("Enter");
 97         this.btnAdd = new JButton("+");
 98         this.btnSubtract = new JButton("-");
 99         this.btnMultiply = new JButton("*");
100         this.btnDivide = new JButton("/");
101         this.btnPower = new JButton("Power");
102         this.btnRoot = new JButton("Root");
103         this.btnDigits = new JButton[DIGIT_BUTTONS];
104         for (int j = 0; j < DIGIT_BUTTONS; j++) {
105             this.btnDigits[j] = new JButton(Integer.toString(j));
106         }
107 
108         // Set up the GUI widgets --------------------------------------------
109 
110         /*
111          * Text areas should wrap lines, and should be read-only; they cannot be
112          * edited because allowing keyboard entry would require checking whether
113          * entries are digits, which we don't want to have to do
114          */
115         this.txtTop.setEditable(false);
116         this.txtTop.setLineWrap(true);
117         this.txtTop.setWrapStyleWord(true);
118         this.txtBottom.setEditable(false);
119         this.txtBottom.setLineWrap(true);
120         this.txtBottom.setWrapStyleWord(true);
121 
122         /*
123          * Initially, the following buttons should be disabled: divide (divisor
124          * must not be 0) and root (root must be at least 2) -- hint: see the
125          * JButton method setEnabled
```

```
126          */
127         this.btnDivide.setEnabled(false);
128         this.btnRoot.setEnabled(false);
129 
130         /*
131          * Create scroll panes for the text areas in case number is long enough
132          * to require scrolling
133          */
134         JScrollPane txtTopScrollPane = new JScrollPane(this.txtTop);
135         JScrollPane txtBottomScrollPane = new JScrollPane(this.txtBottom);
136 
137         /*
138          * Create main button panel
139          */
140         JPanel primaryButtons = new JPanel(new GridLayout(
141                 MAIN_BUTTON_PANEL_GRID_ROWS, MAIN_BUTTON_PANEL_GRID_COLUMNS));
142 
143         /*
144          * Add the buttons to the main button panel, from left to right and top
145          * to bottom
146          */
147         primaryButtons.add(this.btnDigits[7]);
148         primaryButtons.add(this.btnDigits[8]);
149         primaryButtons.add(this.btnDigits[9]);
150         primaryButtons.add(this.btnAdd);
151         primaryButtons.add(this.btnDigits[4]);
152         primaryButtons.add(this.btnDigits[5]);
153         primaryButtons.add(this.btnDigits[6]);
154         primaryButtons.add(this.btnSubtract);
155         primaryButtons.add(this.btnDigits[1]);
156         primaryButtons.add(this.btnDigits[2]);
157         primaryButtons.add(this.btnDigits[3]);
158         primaryButtons.add(this.btnMultiply);
159         primaryButtons.add(this.btnDigits[0]);
160         primaryButtons.add(this.btnPower);
161         primaryButtons.add(this.btnRoot);
162         primaryButtons.add(this.btnDivide);
163 
164         /*
165          * Create side button panel
166          */
167         JPanel sideButtonPanel = new JPanel(new GridLayout(
168                 SIDE_BUTTON_PANEL_GRID_ROWS, SIDE_BUTTON_PANEL_GRID_COLUMNS));
169 
170         /*
171          * Add the buttons to the side button panel, from left to right and top
172          * to bottom
173          */
174         sideButtonPanel.add(this.btnClear);
175         sideButtonPanel.add(this.btnSwap);
176         sideButtonPanel.add(this.btnEnter);
177 
178         /*
179          * Create combined button panel organized using flow layout, which is
180          * simple and does the right thing: sizes of nested panels are natural,
181          * not necessarily equal as with grid layout
182          */
```

```
183         JPanel mainCalcPanel = new JPanel(new FlowLayout());
184 
185         /*
186          * Add the other two button panels to the combined button panel
187          */
188         mainCalcPanel.add(primaryButtons);
189         mainCalcPanel.add(sideButtonPanel);
190 
191         /*
192          * Organize main window
193          */
194         this.setLayout(new GridLayout(CALC_GRID_ROWS, CALC_GRID_COLUMNS));
195 
196         /*
197          * Add scroll panes and button panel to main window, from left to right
198          * and top to bottom
199          */
200         this.add(txtTopScrollPane);
201         this.add(txtBottomScrollPane);
202         this.add(mainCalcPanel);
203 
204         // Set up the observers -----------------------------------------------
205 
206         /*
207          * Register this object as the observer for all GUI events
208          */
209         this.btnAdd.addActionListener(this);
210         this.btnSubtract.addActionListener(this);
211         this.btnMultiply.addActionListener(this);
212         this.btnDivide.addActionListener(this);
213         this.btnClear.addActionListener(this);
214         this.btnSwap.addActionListener(this);
215         this.btnEnter.addActionListener(this);
216         this.btnRoot.addActionListener(this);
217         this.btnPower.addActionListener(this);
218         for (int i = 0; i < DIGIT_BUTTONS; i++) {
219             this.btnDigits[i].addActionListener(this);
220         }
221 
222         // Set up the main application window --------------------------------
223 
224         this.pack();
225         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
226         this.setVisible(true);
227         /*
228          * Make sure the main window is appropriately sized, exits this program
229          * on close, and becomes visible to the user
230          */
231 
232     }
233 
234     @Override
235     public void registerObserver(NNCalcController controller) {
236 
237         this.controller = controller;
238 
239     }
```

```
240
241     @Override
242     public void updateTopDisplay(NaturalNumber n) {
243
244         this.txtTop.setText(n.toString());
245
246     }
247
248     @Override
249     public void updateBottomDisplay(NaturalNumber n) {
250
251         this.txtBottom.setText(n.toString());
252
253     }
254
255     @Override
256     public void updateSubtractAllowed(boolean allowed) {
257
258         this.btnSubtract.setEnabled(allowed);
259
260     }
261
262     @Override
263     public void updateDivideAllowed(boolean allowed) {
264
265         this.btnDivide.setEnabled(allowed);
266
267     }
268
269     @Override
270     public void updatePowerAllowed(boolean allowed) {
271
272         this.btnPower.setEnabled(allowed);
273
274     }
275
276     @Override
277     public void updateRootAllowed(boolean allowed) {
278
279         this.btnRoot.setEnabled(allowed);
280
281     }
282
283     @Override
284     public void actionPerformed(ActionEvent event) {
285         /*
286          * Set cursor to indicate computation on-going; this matters only if
287          * processing the event might take a noticeable amount of time as seen
288          * by the user
289          */
290         this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
291         /*
292          * Determine which event has occurred that we are being notified of by
293          * this callback; in this case, the source of the event (i.e, the widget
294          * calling actionPerformed) is all we need because only buttons are
295          * involved here, so the event must be a button press; in each case,
296          * tell the controller to do whatever is needed to update the model and
```

```
297          * to refresh the view
298          */
299         Object source = event.getSource();
300         if (source == this.btnClear) {
301             this.controller.processClearEvent();
302             this.currentState = State.SAW_CLEAR;
303         } else if (source == this.btnSwap) {
304             this.controller.processSwapEvent();
305             this.currentState = State.SAW_ENTER_OR_SWAP;
306         } else if (source == this.btnEnter) {
307             this.controller.processEnterEvent();
308             this.currentState = State.SAW_ENTER_OR_SWAP;
309         } else if (source == this.btnAdd) {
310             this.controller.processAddEvent();
311             this.currentState = State.SAW_OTHER_OP;
312         } else if (source == this.btnSubtract) {
313             this.controller.processSubtractEvent();
314             this.currentState = State.SAW_OTHER_OP;
315         } else if (source == this.btnMultiply) {
316             this.controller.processMultiplyEvent();
317             this.currentState = State.SAW_OTHER_OP;
318         } else if (source == this.btnDivide) {
319             this.controller.processDivideEvent();
320             this.currentState = State.SAW_OTHER_OP;
321         } else if (source == this.btnPower) {
322             this.controller.processPowerEvent();
323             this.currentState = State.SAW_OTHER_OP;
324         } else if (source == this.btnRoot) {
325             this.controller.processRootEvent();
326             this.currentState = State.SAW_OTHER_OP;
327         } else {
328             for (int i = 0; i < DIGIT_BUTTONS; i++) {
329                 if (source == this.btnDigits[i]) {
330                     switch (this.currentState) {
331                         case SAW_ENTER_OR_SWAP:
332                             this.controller.processClearEvent();
333                             break;
334                         case SAW_OTHER_OP:
335                             this.controller.processEnterEvent();
336                             this.controller.processClearEvent();
337                             break;
338                         default:
339                             break;
340                     }
341                     this.controller.processAddNewDigitEvent(i);
342                     this.currentState = State.SAW_DIGIT;
343                     break;
344                 }
345             }
346         }
347         /*
348          * Set the cursor back to normal (because we changed it at the beginning
349          * of the method body)
350          */
351         this.setCursor(Cursor.getDefaultCursor());
352     }
353
```

```
354 }
355
```