```
 1 import java.awt.Cursor;
13
14 /**
15  * View class.
16  *
17  * @author Vaishnavi Kasabwala
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      * 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         this.tTop = new JTextArea("", TEXT_AREA_HEIGHT, TEXT_AREA_WIDTH);
 83         this.tBottom = new JTextArea("", TEXT_AREA_HEIGHT, TEXT_AREA_WIDTH);
 84 
 85         this.bClear = new JButton("Clear");
 86         this.bSwap = new JButton("Swap");
 87         this.bEnter = new JButton("Enter");
 88 
 89         this.bAdd = new JButton("+");
 90         this.bSubtract = new JButton("-");
 91         this.bMultiply = new JButton("*");
 92         this.bDivide = new JButton("/");
 93 
 94         this.bPower = new JButton("Power");
 95         this.bRoot = new JButton("Root");
 96 
 97         this.bDigits = new JButton[11];
 98 
 99         for (int count = 0; count <= DIGIT_BUTTONS; count++) {
100             JButton numbers = new JButton(Integer.toString(count));
101             this.bDigits[count] = numbers;
102         }
103 
104         /*
105          * Set up initial state of GUI to behave like last event was "Clear";
106          * currentState is not a GUI widget per se, but is needed to process
107          * digit button events appropriately
108          */
109         this.currentState = State.SAW_CLEAR;
110 
111         // Set up the GUI widgets --------------------------------------------
112 
113         /*
114          * Text areas should wrap lines, and should be read-only; they cannot be
115          * edited because allowing keyboard entry would require checking whether
116          * entries are digits, which we don't want to have to do
117          */
118 
119         this.tTop.setEditable(false);
120         this.tTop.setLineWrap(true);
121         this.tTop.setWrapStyleWord(true);
122 
123         this.tBottom.setEditable(false);
124         this.tBottom.setLineWrap(true);
125         this.tBottom.setWrapStyleWord(true);
```

```
126
127         /*
128          * Initially, the following buttons should be disabled: divide (divisor
129          * must not be 0) and root (root must be at least 2) -- hint: see the
130          * JButton method setEnabled
131          */
132
133         this.bDivide.setEnabled(false);
134         this.bRoot.setEnabled(false);
135
136         /*
137          * Create scroll panes for the text areas in case number is long enough
138          * to require scrolling
139          */
140
141         JScrollPane inputScrollPane = new JScrollPane(this.tTop);
142         JScrollPane outputScrollPane = new JScrollPane(this.tBottom);
143
144         /*
145          * Create main button panel
146          */
147
148         JPanel mainPanel = new JPanel(new GridLayout(
149                 MAIN_BUTTON_PANEL_GRID_ROWS, MAIN_BUTTON_PANEL_GRID_COLUMNS));
150
151         /*
152          * Add the buttons to the main button panel, from left to right and top
153          * to bottom
154          */
155
156         mainPanel.add(this.bDigits[7]);
157         mainPanel.add(this.bDigits[8]);
158         mainPanel.add(this.bDigits[9]);
159         mainPanel.add(this.bAdd);
160
161         mainPanel.add(this.bDigits[4]);
162         mainPanel.add(this.bDigits[5]);
163         mainPanel.add(this.bDigits[6]);
164         mainPanel.add(this.bSubtract);
165
166         mainPanel.add(this.bDigits[1]);
167         mainPanel.add(this.bDigits[2]);
168         mainPanel.add(this.bDigits[3]);
169         mainPanel.add(this.bMultiply);
170
171         mainPanel.add(this.bDigits[0]);
172         mainPanel.add(this.bPower);
173         mainPanel.add(this.bRoot);
174         mainPanel.add(this.bDivide);
175
176         /*
177          * Create side button panel
178          */
179
180         JPanel sidePanel = new JPanel(new GridLayout(
181                 SIDE_BUTTON_PANEL_GRID_ROWS, SIDE_BUTTON_PANEL_GRID_COLUMNS));
182
```

```
183         /*
184          * Add the buttons to the side button panel, from left to right and top
185          * to bottom
186          */
187
188         sidePanel.add(this.bClear);
189         sidePanel.add(this.bSwap);
190         sidePanel.add(this.bEnter);
191
192         /*
193          * Create combined button panel organized using flow layout, which is
194          * simple and does the right thing: sizes of nested panels are natural,
195          * not necessarily equal as with grid layout
196          */
197
198         JPanel combinedPanel = new JPanel(new FlowLayout());
199
200         /*
201          * Add the other two button panels to the combined button panel
202          */
203
204         combinedPanel.add(mainPanel);
205         combinedPanel.add(sidePanel);
206
207         /*
208          * Organize main window
209          */
210
211         this.setLayout(new GridLayout(CALC_GRID_ROWS, CALC_GRID_COLUMNS));
212
213         /*
214          * Add scroll panes and button panel to main window, from left to right
215          * and top to bottom
216          */
217
218         this.add(inputScrollPane);
219         this.add(outputScrollPane);
220         this.add(combinedPanel);
221
222         // Set up the observers -------------------------------------------------
223
224         /*
225          * Register this object as the observer for all GUI events
226          */
227
228         this.bDigits[9].addActionListener(this);
229         this.bDigits[8].addActionListener(this);
230         this.bDigits[7].addActionListener(this);
231         this.bDigits[6].addActionListener(this);
232         this.bDigits[5].addActionListener(this);
233         this.bDigits[4].addActionListener(this);
234         this.bDigits[3].addActionListener(this);
235         this.bDigits[2].addActionListener(this);
236         this.bDigits[1].addActionListener(this);
237         this.bDigits[0].addActionListener(this);
238
239         this.bClear.addActionListener(this);
```

```
240         this.bSwap.addActionListener(this);
241         this.bEnter.addActionListener(this);
242 
243         this.bAdd.addActionListener(this);
244         this.bSubtract.addActionListener(this);
245         this.bMultiply.addActionListener(this);
246         this.bDivide.addActionListener(this);
247 
248         this.bPower.addActionListener(this);
249         this.bRoot.addActionListener(this);
250 
251         // Set up the main application window ------------------------------
252 
253         /*
254          * Make sure the main window is appropriately sized, exits this program
255          * on close, and becomes visible to the user
256          */
257 
258         this.pack();
259         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
260         this.setVisible(true);
261 
262     }
263 
264     @Override
265     public void registerObserver(NNCalcController controller) {
266 
267         this.controller = controller;
268     }
269 
270     @Override
271     public void updateTopDisplay(NaturalNumber n) {
272 
273         this.tTop.setText(n.toString());
274     }
275 
276     @Override
277     public void updateBottomDisplay(NaturalNumber n) {
278 
279         this.tBottom.setText(n.toString());
280     }
281 
282     @Override
283     public void updateSubtractAllowed(boolean allowed) {
284 
285         this.bSubtract.setEnabled(allowed);
286     }
287 
288     @Override
289     public void updateDivideAllowed(boolean allowed) {
290 
291         this.bDivide.setEnabled(allowed);
292     }
293 
294     @Override
295     public void updatePowerAllowed(boolean allowed) {
296 
```

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

```
354                         case SAW_ENTER_OR_SWAP:
355                             this.controller.processClearEvent();
356                             break;
357                         case SAW_OTHER_OP:
358                             this.controller.processEnterEvent();
359                             this.controller.processClearEvent();
360                             break;
361                         default:
362                             break;
363                     }
364                     this.controller.processAddNewDigitEvent(i);
365                     this.currentState = State.SAW_DIGIT;
366                     break;
367                 }
368             }
369         }
370         /*
371          * Set the cursor back to normal (because we changed it at the beginning
372          * of the method body)
373          */
374         this.setCursor(Cursor.getDefaultCursor());
375     }
376 
377 }
378 
```