```
 1 import java.awt.Cursor;
13
14 /**
15  * View class.
16  *
17  * @author Feras Akileh
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
 84        this.tBottom = new JTextArea("", TEXT_AREA_HEIGHT, TEXT_AREA_WIDTH);
 85
 86        /*
 87         * Set up initial state of GUI to behave like last event was "Clear";
 88         * currentState is not a GUI widget per se, but is needed to process
 89         * digit button events appropriately
 90         */
 91        this.currentState = State.SAW_CLEAR;
 92
 93        /*
 94         * Create widgets
 95         */
 96
 97        // Set up the GUI widgets --------------------------------------------
 98
 99        /*
100         * Text areas should wrap lines, and should be read-only; they cannot be
101         * edited because allowing keyboard entry would require checking whether
102         * entries are digits, which we don't want to have to do
103         */
104
105        /*
106         * Initially, the following buttons should be disabled: divide (divisor
107         * must not be 0) and root (root must be at least 2) -- hint: see the
108         * JButton method setEnabled
109         */
110
111        this.bClear = new JButton("Clear");
112        this.bSwap = new JButton("Swap");
113        this.bEnter = new JButton("Enter");
114        this.bAdd = new JButton("+");
115        this.bSubtract = new JButton("-");
116        this.bMultiply = new JButton("*");
117        this.bDivide = new JButton("/");
118        this.bPower = new JButton("Power");
119        this.bRoot = new JButton("Root");
120
121        this.bDigits = new JButton[10];
122
123        int i = 0;
124
125        while (i < DIGIT_BUTTONS) {
126
127            this.bDigits[i] = new JButton(Integer.toString(i));
128            i++;
129
```

```
130         }
131
132         this.bDivide.setEnabled(false);
133         this.bRoot.setEnabled(false);
134
135         /*
136          * Create scroll panes for the text areas in case number is long enough
137          * to require scrolling
138          */
139
140         JScrollPane inPut = new JScrollPane(this.tTop);
141         JScrollPane outPut = new JScrollPane(this.tBottom);
142
143         /*
144          * Create main button panel
145          */
146
147         JPanel buttonPanel = new JPanel(new GridLayout(
148                 MAIN_BUTTON_PANEL_GRID_ROWS, MAIN_BUTTON_PANEL_GRID_COLUMNS));
149
150         /*
151          * Add the buttons to the main button panel, from left to right and top
152          * to bottom
153          */
154
155         buttonPanel.add(this.bDigits[7]);
156         buttonPanel.add(this.bDigits[8]);
157         buttonPanel.add(this.bDigits[9]);
158         buttonPanel.add(this.bAdd);
159
160         buttonPanel.add(this.bDigits[4]);
161         buttonPanel.add(this.bDigits[5]);
162         buttonPanel.add(this.bDigits[6]);
163         buttonPanel.add(this.bSubtract);
164
165         buttonPanel.add(this.bDigits[1]);
166         buttonPanel.add(this.bDigits[2]);
167         buttonPanel.add(this.bDigits[3]);
168         buttonPanel.add(this.bMultiply);
169
170         buttonPanel.add(this.bDigits[0]);
171         buttonPanel.add(this.bPower);
172         buttonPanel.add(this.bRoot);
173         buttonPanel.add(this.bDivide);
174
175         /*
176          * Create side button panel
177          */
178
179         JPanel sidePanel = new JPanel(new GridLayout(
180                 SIDE_BUTTON_PANEL_GRID_ROWS, SIDE_BUTTON_PANEL_GRID_COLUMNS));
181
182         /*
183          * Add the buttons to the side button panel, from left to right and top
184          * to bottom
185          */
186
187         sidePanel.add(this.bClear);
188         sidePanel.add(this.bSwap);
```

```
189         sidePanel.add(this.bEnter);
190 
191         /*
192          * Create combined button panel organized using flow layout, which is
193          * simple and does the right thing: sizes of nested panels are natural,
194          * not necessarily equal as with grid layout
195          */
196 
197         JPanel combinedPanel = new JPanel(new FlowLayout());
198 
199         /*
200          * Add the other two button panels to the combined button panel
201          */
202 
203         combinedPanel.add(buttonPanel);
204         combinedPanel.add(sidePanel);
205 
206         /*
207          * Organize main window
208          */
209 
210         this.setLayout(new GridLayout(CALC_GRID_ROWS, CALC_GRID_COLUMNS));
211 
212         /*
213          * Add scroll panes and button panel to main window, from left to right
214          * and top to bottom
215          */
216 
217         this.add(inPut);
218         this.add(outPut);
219         this.add(combinedPanel);
220 
221         // Set up the observers ----------------------------------------------
222 
223         this.bAdd.addActionListener(this);
224         this.bSubtract.addActionListener(this);
225         this.bDivide.addActionListener(this);
226         this.bMultiply.addActionListener(this);
227         this.bClear.addActionListener(this);
228         this.bSwap.addActionListener(this);
229         this.bEnter.addActionListener(this);
230         this.bPower.addActionListener(this);
231         this.bRoot.addActionListener(this);
232 
233         int n = 0;
234         while (n < DIGIT_BUTTONS) {
235             this.bDigits[n].addActionListener(this);
236             n++;
237         }
238 
239         /*
240          * Register this object as the observer for all GUI events
241          */
242 
243         // Set up the main application window --------------------------------
244 
245         /*
246          * Make sure the main window is appropriately sized, exits this program
247          * on close, and becomes visible to the user
```

```
248          */
249
250         this.pack();
251         this.setDefaultCloseOperation(EXIT_ON_CLOSE);
252         this.setVisible(true);
253
254     }
255
256     @Override
257     public void registerObserver(NNCalcController controller) {
258
259         this.controller = controller;
260
261     }
262
263     @Override
264     public void updateTopDisplay(NaturalNumber n) {
265
266         String userNum = n.toString();
267         this.tTop.setText(userNum);
268
269     }
270
271     @Override
272     public void updateBottomDisplay(NaturalNumber n) {
273
274         String userNum = n.toString();
275         this.tBottom.setText(userNum);
276
277     }
278
279     @Override
280     public void updateSubtractAllowed(boolean allowed) {
281
282         this.bSubtract.setEnabled(allowed);
283
284     }
285
286     @Override
287     public void updateDivideAllowed(boolean allowed) {
288
289         this.bDivide.setEnabled(allowed);
290
291     }
292
293     @Override
294     public void updatePowerAllowed(boolean allowed) {
295
296         this.bPower.setEnabled(allowed);
297
298     }
299
300     @Override
301     public void updateRootAllowed(boolean allowed) {
302
303         this.bRoot.setEnabled(allowed);
304
305     }
306
```

```
307     @Override
308     public void actionPerformed(ActionEvent event) {
309         /*
310          * Set cursor to indicate computation on-going; this matters only if
311          * processing the event might take a noticeable amount of time as seen
312          * by the user
313          */
314         this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
315         /*
316          * Determine which event has occurred that we are being notified of by
317          * this callback; in this case, the source of the event (i.e, the widget
318          * calling actionPerformed) is all we need because only buttons are
319          * involved here, so the event must be a button press; in each case,
320          * tell the controller to do whatever is needed to update the model and
321          * to refresh the view
322          */
323         Object source = event.getSource();
324         if (source == this.bClear) {
325             this.controller.processClearEvent();
326             this.currentState = State.SAW_CLEAR;
327         } else if (source == this.bSwap) {
328             this.controller.processSwapEvent();
329             this.currentState = State.SAW_ENTER_OR_SWAP;
330         } else if (source == this.bEnter) {
331             this.controller.processEnterEvent();
332             this.currentState = State.SAW_ENTER_OR_SWAP;
333         } else if (source == this.bAdd) {
334             this.controller.processAddEvent();
335             this.currentState = State.SAW_OTHER_OP;
336         } else if (source == this.bSubtract) {
337             this.controller.processSubtractEvent();
338             this.currentState = State.SAW_OTHER_OP;
339         } else if (source == this.bMultiply) {
340             this.controller.processMultiplyEvent();
341             this.currentState = State.SAW_OTHER_OP;
342         } else if (source == this.bDivide) {
343             this.controller.processDivideEvent();
344             this.currentState = State.SAW_OTHER_OP;
345         } else if (source == this.bPower) {
346             this.controller.processPowerEvent();
347             this.currentState = State.SAW_OTHER_OP;
348         } else if (source == this.bRoot) {
349             this.controller.processRootEvent();
350             this.currentState = State.SAW_OTHER_OP;
351         } else {
352             for (int i = 0; i < DIGIT_BUTTONS; i++) {
353                 if (source == this.bDigits[i]) {
354                     switch (this.currentState) {
355                         case SAW_ENTER_OR_SWAP:
356                             this.controller.processClearEvent();
357                             break;
358                         case SAW_OTHER_OP:
359                             this.controller.processEnterEvent();
360                             this.controller.processClearEvent();
361                             break;
362                         default:
363                             break;
364                     }
365                     this.controller.processAddNewDigitEvent(i);
```

```
366                     this.currentState = State.SAW_DIGIT;
367                     break;
368                 }
369             }
370         }
371         /*
372          * Set the cursor back to normal (because we changed it at the beginning
373          * of the method body)
374          */
375         this.setCursor(Cursor.getDefaultCursor());
376     }
377
378 }
379
```