```java
import java.awt.Cursor;

/**
 * View class.
 *
 * @author Shyam Sai Bethina
 */
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 tTop, tBottom;

    /**
     * Operator and related buttons.
     */
    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, SEVEN = 7, FOUR = 4;
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         /*
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          // Set up the GUI widgets
     ----------------------------------------
 90
 91          /*
 92           * Create widgets
 93           */
 94          /*
 95           * Creates the top and bottom display using the giving
     dimensions
 96           */
 97          this.tTop = new JTextArea("", this.TEXT_AREA_HEIGHT,
 98                  this.TEXT_AREA_WIDTH);
 99
100          this.tBottom = new JTextArea("", this.TEXT_AREA_HEIGHT,
101                  this.TEXT_AREA_WIDTH);
102
103          /*
104           * Creates the clear, swap and enter buttons
105           */
106          this.bClear = new JButton("Clear");
107          this.bSwap = new JButton("Swap");
108          this.bEnter = new JButton("Enter");
109
110          /*
111           * Creates the addition, subtraction, multiplication,
     division, power,
112           * and root buttons and enter buttons
113           */
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          /*
122           * Adds the 0-9 number buttons to the bDigits array
123           */
124          this.bDigits = new JButton[DIGIT_BUTTONS];
125          for (int i = 0; i < DIGIT_BUTTONS; i++) {
126              this.bDigits[i] = new JButton(Integer.toString(i));
127          }
128
```

```
129         // Set up the GUI widgets
    ------------------------------------------
130 
131         /*
132          * Text areas should wrap lines, and should be read-only;
    they cannot be
133          * edited because allowing keyboard entry would require
    checking whether
134          * entries are digits, which we don't want to have to do
135          */
136         this.tTop.setEditable(false);
137         this.tTop.setLineWrap(true);
138         this.tTop.setWrapStyleWord(true);
139         this.tBottom.setEditable(false);
140         this.tBottom.setLineWrap(true);
141         this.tBottom.setWrapStyleWord(true);
142 
143         /*
144          * Initially, the following buttons should be disabled:
    divide (divisor
145          * must not be 0) and root (root must be at least 2) --
    hint: see the
146          * JButton method setEnabled
147          */
148         this.bDivide.setEnabled(false);
149         this.bRoot.setEnabled(false);
150 
151         /*
152          * Create scroll panes for the text areas in case number
    is long enough
153          * to require scrolling
154          */
155         JScrollPane inputTextScrollPaneTop = new
    JScrollPane(this.tTop);
156         JScrollPane inputTextScrollPaneBottom = new
    JScrollPane(this.tBottom);
157 
158         /*
159          * Create main button panel
160          */
161         JPanel mainButtonPanel = new JPanel(
162                 new GridLayout(this.MAIN_BUTTON_PANEL_GRID_ROWS,
163                         this.MAIN_BUTTON_PANEL_GRID_COLUMNS));
164 
```

```
165         /*
166          * Add the buttons to the main button panel, from left to
    right and top
167          * to bottom
168          */
169 
170         /*
171          * Adds the first row of numbers 7-9, and then adds the
    addition button
172          * to complete the first row of buttons
173          */
174         for (int i = SEVEN; i < DIGIT_BUTTONS; i++) {
175             mainButtonPanel.add(this.bDigits[i]);
176         }
177         mainButtonPanel.add(this.bAdd);
178 
179         /*
180          * Adds the second row of numbers 4-6, and then adds the
    subtraction
181          * button to complete the second row of buttons
182          */
183         for (int i = FOUR; i < SEVEN; i++) {
184             mainButtonPanel.add(this.bDigits[i]);
185         }
186         mainButtonPanel.add(this.bSubtract);
187 
188         /*
189          * Adds the third row of numbers 0-3, and then adds the
    multiplication,
190          * power, root and division buttons to complete the third
    row of buttons
191          * button to complete the second row of buttons
192          */
193         for (int i = 1; i < FOUR; i++) {
194             mainButtonPanel.add(this.bDigits[i]);
195         }
196         mainButtonPanel.add(this.bMultiply);
197         mainButtonPanel.add(this.bDigits[0]);
198         mainButtonPanel.add(this.bPower);
199         mainButtonPanel.add(this.bRoot);
200         mainButtonPanel.add(this.bDivide);
201 
202         /*
203          * Create side button panel
```

```
204          */
205         JPanel sideButtonPanel = new JPanel(
206                 new GridLayout(this.SIDE_BUTTON_PANEL_GRID_ROWS,
207                         this.SIDE_BUTTON_PANEL_GRID_COLUMNS));
208 
209         /*
210          * Add the buttons to the side button panel, from left to
    right and top
211          * to bottom
212          */
213 
214         /*
215          * Adds the clear, swap and enter buttons to the
    sideButtonPanel
216          */
217         sideButtonPanel.add(this.bClear);
218         sideButtonPanel.add(this.bSwap);
219         sideButtonPanel.add(this.bEnter);
220 
221         /*
222          * Create combined button panel organized using flow
    layout, which is
223          * simple and does the right thing: sizes of nested panels
    are natural,
224          * not necessarily equal as with grid layout
225          */
226         JPanel combinedButtonPanel = new JPanel(new FlowLayout());
227 
228         /*
229          * Add the other two button panels to the combined button
    panel
230          */
231         combinedButtonPanel.add(mainButtonPanel);
232         combinedButtonPanel.add(sideButtonPanel);
233 
234         /*
235          * Organize main window
236          */
237         this.setLayout(
238                 new GridLayout(this.CALC_GRID_ROWS,
    this.CALC_GRID_COLUMNS));
239 
240         /*
241          * Add scroll panes and button panel to main window, from
```

```
   left to right
242         * and top to bottom
243         */
244        this.add(inputTextScrollPaneTop);
245        this.add(inputTextScrollPaneBottom);
246        this.add(combinedButtonPanel);
247
248        // Set up the observers
   ---------------------------------------------
249
250        /*
251         * Register this object as the observer for all GUI events
252         */
253        this.bClear.addActionListener(this);
254        this.bSwap.addActionListener(this);
255        this.bEnter.addActionListener(this);
256        this.bAdd.addActionListener(this);
257        this.bSubtract.addActionListener(this);
258        this.bMultiply.addActionListener(this);
259        this.bDivide.addActionListener(this);
260        this.bPower.addActionListener(this);
261        this.bRoot.addActionListener(this);
262
263        /*
264         * Loops through the number buttons 0-9 to add the action
   listener to
265         * each of the buttons
266         */
267        for (int i = 0; i < DIGIT_BUTTONS; i++) {
268            this.bDigits[i].addActionListener(this);
269        }
270
271        // Set up the main application window
   --------------------------------
272
273        /*
274         * Make sure the main window is appropriately sized, exits
   this program
275         * on close, and becomes visible to the user
276         */
277        this.pack();
278        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
279        this.setVisible(true);
280
```

```
281     }
282
283     @Override
284     public void registerObserver(NNCalcController controller) {
285
286         /*
287          * Sets the view's controller to the NNCalcController
    controller
288          */
289         this.controller = controller;
290     }
291
292     @Override
293     public void updateTopDisplay(NaturalNumber n) {
294
295         /*
296          * Sets the top display of the view to n
297          */
298         this.tTop.setText(n.toString());
299     }
300
301     @Override
302     public void updateBottomDisplay(NaturalNumber n) {
303         /*
304          * Sets the top display of the view to n
305          */
306         this.tBottom.setText(n.toString());
307     }
308
309     @Override
310     public void updateSubtractAllowed(boolean allowed) {
311         /*
312          * Disables or enables the subtract button depending on
    the boolean
313          * allowed
314          */
315         this.bSubtract.setEnabled(allowed);
316     }
317
318     @Override
319     public void updateDivideAllowed(boolean allowed) {
320         /*
321          * Disables or enables the divide button depending on the
    boolean
```

```
322          * allowed
323          */
324         this.bDivide.setEnabled(allowed);
325 
326     }
327 
328     @Override
329     public void updatePowerAllowed(boolean allowed) {
330         /*
331          * Disables or enables the power button depending on the
    boolean allowed
332          */
333         this.bPower.setEnabled(allowed);
334 
335     }
336 
337     @Override
338     public void updateRootAllowed(boolean allowed) {
339         /*
340          * Disables or enables the root button depending on the
    boolean allowed
341          */
342         this.bRoot.setEnabled(allowed);
343 
344     }
345 
346     @Override
347     public void actionPerformed(ActionEvent event) {
348         /*
349          * Set cursor to indicate computation on-going; this
    matters only if
350          * processing the event might take a noticeable amount of
    time as seen
351          * by the user
352          */
353 
    this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
354         /*
355          * Determine which event has occurred that we are being
    notified of by
356          * this callback; in this case, the source of the event
    (i.e, the widget
357          * calling actionPerformed) is all we need because only
    buttons are
```

```
358          * involved here, so the event must be a button press; in each case,
359          * tell the controller to do whatever is needed to update the model and
360          * to refresh the view
361          */
362         Object source = event.getSource();
363         if (source == this.bClear) {
364             this.controller.processClearEvent();
365             this.currentState = State.SAW_CLEAR;
366         } else if (source == this.bSwap) {
367             this.controller.processSwapEvent();
368             this.currentState = State.SAW_ENTER_OR_SWAP;
369         } else if (source == this.bEnter) {
370             this.controller.processEnterEvent();
371             this.currentState = State.SAW_ENTER_OR_SWAP;
372         } else if (source == this.bAdd) {
373             this.controller.processAddEvent();
374             this.currentState = State.SAW_OTHER_OP;
375         } else if (source == this.bSubtract) {
376             this.controller.processSubtractEvent();
377             this.currentState = State.SAW_OTHER_OP;
378         } else if (source == this.bMultiply) {
379             this.controller.processMultiplyEvent();
380             this.currentState = State.SAW_OTHER_OP;
381         } else if (source == this.bDivide) {
382             this.controller.processDivideEvent();
383             this.currentState = State.SAW_OTHER_OP;
384         } else if (source == this.bPower) {
385             this.controller.processPowerEvent();
386             this.currentState = State.SAW_OTHER_OP;
387         } else if (source == this.bRoot) {
388             this.controller.processRootEvent();
389             this.currentState = State.SAW_OTHER_OP;
390         } else {
391             for (int i = 0; i < DIGIT_BUTTONS; i++) {
392                 if (source == this.bDigits[i]) {
393                     switch (this.currentState) {
394                         case SAW_ENTER_OR_SWAP:
395                             this.controller.processClearEvent();
396                             break;
397                         case SAW_OTHER_OP:
398                             this.controller.processEnterEvent();
399                             this.controller.processClearEvent();
```

```
400                             break;
401                         default:
402                             break;
403                     }
404                     this.controller.processAddNewDigitEvent(i);
405                     this.currentState = State.SAW_DIGIT;
406                     break;
407                 }
408             }
409         }
410         /*
411          * Set the cursor back to normal (because we changed it at
    the beginning
412          * of the method body)
413          */
414         this.setCursor(Cursor.getDefaultCursor());
415     }
416 
417 }
418 
```