

```
1 import java.awt.Cursor;
13
14 /**
15  * View class.
16  *
17  * @author Nicholas Cheong
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,
63        TEXT_AREA_WIDTH = 20,
64        DIGIT_BUTTONS = 10, MAIN_BUTTON_PANEL_GRID_ROWS =
65        4,
66        MAIN_BUTTON_PANEL_GRID_COLUMNS = 4,
67        SIDE_BUTTON_PANEL_GRID_ROWS = 3,
68        SIDE_BUTTON_PANEL_GRID_COLUMNS = 1, CALC_GRID_ROWS
69        = 3,
70        CALC_GRID_COLUMNS = 1;
71
72    /**
73     * Default constructor.
74     */
75    public NNCalcView1() {
76        // Create the JFrame being extended
77
78        /*
79         * Call the JFrame (superclass) constructor with a
80         String parameter to
81         * name the window in its title bar
82         */
83        super("Natural Number Calculator");
84
85        // Set up the GUI widgets
86        -----
87
88        /*
89         * Set up initial state of GUI to behave like last
90         event was "Clear";
91         * currentState is not a GUI widget per se, but is
92         needed to process
93         * digit button events appropriately
94         */
```

```
87         this.currentState = State.SAW_CLEAR;
88
89         // TODO: fill in rest of body, following outline in
comments
90
91         /*
92          * Create widgets
93          */
94
95         this.tTop = new JTextArea("", TEXT_AREA_HEIGHT,
TEXT_AREA_WIDTH);
96         this.tBottom = new JTextArea("", TEXT_AREA_HEIGHT,
TEXT_AREA_WIDTH);
97         this.bClear = new JButton("clear");
98         this.bSwap = new JButton("swap");
99         this.bEnter = new JButton("=");
100        this.bAdd = new JButton("+");
101        this.bSubtract = new JButton("-");
102        this.bMultiply = new JButton("x");
103        this.bDivide = new JButton("/");
104        this.bPower = new JButton("^");
105        this.bRoot = new JButton("root");
106        // ...
107        this.bDigits = new JButton[DIGIT_BUTTONS];
108        for (int i = 0; i < this.bDigits.length; i++) {
109            this.bDigits[i] = new JButton("" + i);
110        }
111
112        // Set up the GUI widgets
-----
113
114        /*
115         * Text areas should wrap lines, and should be read-
only; they cannot be
116         * edited because allowing keyboard entry would require
checking whether
117         * entries are digits, which we don't want to have to
do
118         */
119
120        this.tTop.setLineWrap(true);
121        this.tBottom.setLineWrap(true);
122
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
129         this.bDivide.setEnabled(false);
130         this.bRoot.setEnabled(false);
131
132         /*
133         * Create scroll panes for the text areas in case
        number is long enough
134         * to require scrolling
135         */
136
137         JScrollPane topScrollPane = new JScrollPane(this.tTop);
138         JScrollPane bottomScrollPane = new
        JScrollPane(this.tBottom);
139
140         /*
141         * Create main button panel
142         */
143
144         JPanel mainButtonPanel = new JPanel(new GridLayout(
145             MAIN_BUTTON_PANEL_GRID_ROWS,
            MAIN_BUTTON_PANEL_GRID_COLUMNS));
146
147         /*
148         * Add the buttons to the main button panel, from left
        to right and top
149         * to bottom
150         */
151
152         mainButtonPanel.add(this.bDigits[7]);
153         mainButtonPanel.add(this.bDigits[8]);
154         mainButtonPanel.add(this.bDigits[9]);
155         mainButtonPanel.add(this.bAdd);
156         mainButtonPanel.add(this.bDigits[4]);
157         mainButtonPanel.add(this.bDigits[5]);
158         mainButtonPanel.add(this.bDigits[6]);
159         mainButtonPanel.add(this.bSubtract);
160         mainButtonPanel.add(this.bDigits[1]);
161         mainButtonPanel.add(this.bDigits[2]);
```

```
162         mainButtonPanel.add(this.bDigits[3]);
163         mainButtonPanel.add(this.bMultiply);
164         mainButtonPanel.add(this.bDigits[0]);
165         mainButtonPanel.add(this.bPower);
166         mainButtonPanel.add(this.bRoot);
167         mainButtonPanel.add(this.bDivide);
168
169         /*
170          * Create side button panel
171          */
172
173         JPanel sideButtonPanel = new JPanel(new GridLayout(
174             SIDE_BUTTON_PANEL_GRID_ROWS,
175             SIDE_BUTTON_PANEL_GRID_COLUMNS));
176
177         /*
178          * Add the buttons to the side button panel, from left
179          * to right and top
180          * to bottom
181          */
182         sideButtonPanel.add(this.bClear);
183         sideButtonPanel.add(this.bSwap);
184         sideButtonPanel.add(this.bEnter);
185
186         /*
187          * Create combined button panel organized using flow
188          * layout, which is
189          * simple and does the right thing: sizes of nested
190          * panels are natural,
191          * not necessarily equal as with grid layout
192          */
193         JPanel combinedPanel = new JPanel(new FlowLayout());
194
195         /*
196          * Add the other two button panels to the combined
197          * button panel
198          */
199
200         combinedPanel.add(mainButtonPanel);
201         combinedPanel.add(sideButtonPanel);
202
203         /*
```

```
201         * Organize main window
202         */
203
204         this.setLayout(new GridLayout(CALC_GRID_ROWS,
CALC_GRID_COLUMNS));
205
206         /*
207         * Add scroll panes and button panel to main window,
from left to right
208         * and top to bottom
209         */
210
211         this.add(topScrollPane);
212         this.add(bottomScrollPane);
213         this.add(combinedPanel);
214
215         // Set up the observers
-----
216
217         /*
218         * Register this object as the observer for all GUI
events
219         */
220
221         this.bClear.addActionListener(this);
222         this.bSwap.addActionListener(this);
223         this.bEnter.addActionListener(this);
224         this.bAdd.addActionListener(this);
225         this.bSubtract.addActionListener(this);
226         this.bMultiply.addActionListener(this);
227         this.bDivide.addActionListener(this);
228         this.bPower.addActionListener(this);
229         this.bRoot.addActionListener(this);
230         for (int i = 0; i < this.bDigits.length; i++) {
231             this.bDigits[i].addActionListener(this);
232         }
233
234         // Set up the main application window
-----
235
236         /*
237         * Make sure the main window is appropriately sized,
exits this program
238         * on close, and becomes visible to the user
```

```
239         */
240
241         this.pack();
242         this.setVisible(true);
243
244     }
245
246     @Override
247     public void registerObserver(NNCalcController controller) {
248
249         this.controller = controller;
250
251     }
252
253     @Override
254     public void updateTopDisplay(NaturalNumber n) {
255         this.tTop.setText(n.toString());
256     }
257
258     @Override
259     public void updateBottomDisplay(NaturalNumber n) {
260
261         this.tBottom.setText(n.toString());
262
263     }
264
265     @Override
266     public void updateSubtractAllowed(boolean allowed) {
267         this.bSubtract.setEnabled(allowed);
268     }
269
270     @Override
271     public void updateDivideAllowed(boolean allowed) {
272
273         this.bDivide.setEnabled(allowed);
274
275     }
276
277     @Override
278     public void updatePowerAllowed(boolean allowed) {
279
280         this.bPower.setEnabled(allowed);
281
282     }
```

```
283
284     @Override
285     public void updateRootAllowed(boolean allowed) {
286
287         this.bRoot.setEnabled(allowed);
288
289     }
290
291     @Override
292     public void actionPerformed(ActionEvent event) {
293         /*
294          * Set cursor to indicate computation on-going; this
295          * matters only if
296          * processing the event might take a noticeable amount
297          * of time as seen
298          * by the user
299          */
300         this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
301         /*
302          * Determine which event has occurred that we are being
303          * notified of by
304          * this callback; in this case, the source of the event
305          * (i.e, the widget
306          * calling actionPerformed) is all we need because only
307          * buttons are
308          * involved here, so the event must be a button press;
309          in each case,
310          * tell the controller to do whatever is needed to
311          update the model and
312          * to refresh the view
313          */
314         Object source = event.getSource();
315         if (source == this.bClear) {
316             this.controller.processClearEvent();
317             this.currentState = State.SAW_CLEAR;
318         } else if (source == this.bSwap) {
319             this.controller.processSwapEvent();
320             this.currentState = State.SAW_ENTER_OR_SWAP;
321         } else if (source == this.bEnter) {
322             this.controller.processEnterEvent();
323             this.currentState = State.SAW_ENTER_OR_SWAP;
324         } else if (source == this.bAdd) {
325             this.controller.processAddEvent();
326         }
```



```
319         this.currentState = State.SAW_OTHER_OP;
320     } else if (source == this.bSubtract) {
321         this.controller.processSubtractEvent();
322         this.currentState = State.SAW_OTHER_OP;
323     } else if (source == this.bMultiply) {
324         this.controller.processMultiplyEvent();
325         this.currentState = State.SAW_OTHER_OP;
326     } else if (source == this.bDivide) {
327         this.controller.processDivideEvent();
328         this.currentState = State.SAW_OTHER_OP;
329     } else if (source == this.bPower) {
330         this.controller.processPowerEvent();
331         this.currentState = State.SAW_OTHER_OP;
332     } else if (source == this.bRoot) {
333         this.controller.processRootEvent();
334         this.currentState = State.SAW_OTHER_OP;
335     } else {
336         for (int i = 0; i < DIGIT_BUTTONS; i++) {
337             if (source == this.bDigits[i]) {
338                 switch (this.currentState) {
339                     case SAW_ENTER_OR_SWAP:
340
341                         this.controller.processClearEvent();
342                         break;
343                     case SAW_OTHER_OP:
344
345                         this.controller.processEnterEvent();
346                         this.controller.processClearEvent();
347                         break;
348                     default:
349                         break;
350                 }
351                 this.controller.processAddNewDigitEvent(i);
352                 this.currentState = State.SAW_DIGIT;
353                 break;
354             }
355         }
356     }
357     /*
358     * Set the cursor back to normal (because we changed it
359     at the beginning
360     * of the method body)
361     */
```

```
359         this.setCursor(Cursor.getDefaultCursor());  
360     }  
361  
362 }  
363
```