

NNCalcView1.java

```

1 import java.awt.Cursor;
9
10 /**
11  * View class.
12  *
13  * @author Put your name here
14  */
15 public final class NNCalcView1 extends JFrame implements NNCalcView {
16
17     /**
18      * Controller object registered with this view to observe user-interaction
19      * events.
20      */
21     private NNCalcController controller;
22
23     /**
24      * State of user interaction: last event "seen".
25      */
26     private enum State {
27         /**
28          * Last event was clear, enter, another operator, or digit entry, resp.
29          */
30         SAW_CLEAR, SAW_ENTER, SAW_OTHER_OP, SAW_DIGIT
31     }
32
33     /**
34      * State variable to keep track of which event happened last; needed to
35      * prepare for digit to be added to bottom operand.
36      */
37     private State currentState;
38
39     /**
40      * Text areas.
41      */
42     private final JTextArea tTop, tBottom;
43
44     /**
45      * Operator and related buttons.
46      */
47     private final JButton bClear, bSwap, bEnter, bAdd, bSubtract, bMultiply,
48         bDivide, bPower, bRoot;
49
50     /**
51      * Digit entry buttons.
52      */
53     private final JButton[] bDigits;
54
55     /**
56      * Useful constants.
57      */
58     private static final int TEXT_AREA_HEIGHT = 5, TEXT_AREA_WIDTH = 20,
59         DIGIT_BUTTONS = 10, MAIN_BUTTON_PANEL_GRID_ROWS = 4,
60         MAIN_BUTTON_PANEL_GRID_COLUMNS = 4, SIDE_BUTTON_PANEL_GRID_ROWS = 3,
61         SIDE_BUTTON_PANEL_GRID_COLUMNS = 1, CALC_GRID_ROWS = 3,
62         CALC_GRID_COLUMNS = 1;
63
64     /**

```

NNCalcView1.java

```

65     * Default constructor.
66     */
67     public NNCalcView1() {
68         // Create the JFrame being extended
69
70         /*
71          * Call the JFrame (superclass) constructor with a String parameter to
72          * name the window in its title bar
73          */
74         super("Natural Number Calculator");
75
76         // Set up the GUI widgets -----
77
78         /*
79          * Set up initial state of GUI to behave like last event was "Clear";
80          * currentState is not a GUI widget per se, but is needed to process
81          * digit button events appropriately
82          */
83         this.currentState = State.SAW_CLEAR;
84
85         // TODO: fill in rest of body, following outline in comments
86
87         /*
88          * Create widgets
89          */
90
91         // Set up the GUI widgets -----
92
93         /*
94          * Text areas should wrap lines, and should be read-only; they cannot be
95          * edited because allowing keyboard entry would require checking whether
96          * entries are digits, which we don't want to have to do
97          */
98
99         /*
100          * Initially, the following buttons should be disabled: divide (divisor
101          * must not be 0) and root (root must be at least 2) -- hint: see the
102          * JButton method setEnabled
103          */
104
105         /*
106          * Create scroll panes for the text areas in case number is long enough
107          * to require scrolling
108          */
109
110         /*
111          * Create main button panel
112          */
113
114         /*
115          * Add the buttons to the main button panel, from left to right and top
116          * to bottom
117          */
118
119         /*
120          * Create side button panel
121          */

```

NNCalcView1.java

```

122
123     /*
124     * Add the buttons to the side button panel, from left to right and top
125     * to bottom
126     */
127
128     /*
129     * Create combined button panel organized using flow layout, which is
130     * simple and does the right thing: sizes of nested panels are natural,
131     * not necessarily equal as with grid layout
132     */
133
134     /*
135     * Add the other two button panels to the combined button panel
136     */
137
138     /*
139     * Organize main window
140     */
141
142     /*
143     * Add scroll panes and button panel to main window, from left to right
144     * and top to bottom
145     */
146
147     // Set up the observers -----
148
149     /*
150     * Register this object as the observer for all GUI events
151     */
152
153     // Set up the main application window -----
154
155     /*
156     * Make sure the main window is appropriately sized, exits this program
157     * on close, and becomes visible to the user
158     */
159
160 }
161
162 @Override
163 public void registerObserver(NNCalcController controller) {
164
165     // TODO: fill in body
166
167 }
168
169 @Override
170 public void updateTopDisplay(NaturalNumber n) {
171
172     // TODO: fill in body
173
174 }
175
176 @Override
177 public void updateBottomDisplay(NaturalNumber n) {
178

```

NNCalcView1.java

```

179 // TODO: fill in body
180
181 }
182
183 @Override
184 public void updateSubtractAllowed(boolean allowed) {
185
186 // TODO: fill in body
187
188 }
189
190 @Override
191 public void updateDivideAllowed(boolean allowed) {
192
193 // TODO: fill in body
194
195 }
196
197 @Override
198 public void updatePowerAllowed(boolean allowed) {
199
200 // TODO: fill in body
201
202 }
203
204 @Override
205 public void updateRootAllowed(boolean allowed) {
206
207 // TODO: fill in body
208
209 }
210
211 @Override
212 public void actionPerformed(ActionEvent event) {
213     /*
214      * Set cursor to indicate computation on-going; this matters only if
215      * processing the event might take a noticeable amount of time as seen
216      * by the user
217      */
218     this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
219     /*
220      * Determine which event has occurred that we are being notified of by
221      * this callback; in this case, the source of the event (i.e, the widget
222      * calling actionPerformed) is all we need because only buttons are
223      * involved here, so the event must be a button press; in each case,
224      * tell the controller to do whatever is needed to update the model and
225      * to refresh the view
226      */
227     Object source = event.getSource();
228     if (source == this.bClear) {
229         this.controller.processClearEvent();
230         this.currentState = State.SAW_CLEAR;
231     } else if (source == this.bSwap) {
232         this.controller.processSwapEvent();
233         this.currentState = State.SAW_OTHER_OP;
234     } else if (source == this.bEnter) {
235         this.controller.processEnterEvent();

```

```

236         this.currentState = State.SAW_ENTER;
237     } else if (source == this.bAdd) {
238         this.controller.processAddEvent();
239         this.currentState = State.SAW_OTHER_OP;
240     } else if (source == this.bSubtract) {
241         this.controller.processSubtractEvent();
242         this.currentState = State.SAW_OTHER_OP;
243     } else if (source == this.bMultiply) {
244         this.controller.processMultiplyEvent();
245         this.currentState = State.SAW_OTHER_OP;
246     } else if (source == this.bDivide) {
247         this.controller.processDivideEvent();
248         this.currentState = State.SAW_OTHER_OP;
249     } else if (source == this.bPower) {
250         this.controller.processPowerEvent();
251         this.currentState = State.SAW_OTHER_OP;
252     } else if (source == this.bRoot) {
253         this.controller.processRootEvent();
254         this.currentState = State.SAW_OTHER_OP;
255     } else {
256         for (int i = 0; i < DIGIT_BUTTONS; i++) {
257             if (source == this.bDigits[i]) {
258                 switch (this.currentState) {
259                     case SAW_ENTER:
260                         this.controller.processClearEvent();
261                         break;
262                     case SAW_OTHER_OP:
263                         this.controller.processEnterEvent();
264                         this.controller.processClearEvent();
265                         break;
266                     default:
267                         break;
268                 }
269                 this.controller.processAddNewDigitEvent(i);
270                 this.currentState = State.SAW_DIGIT;
271                 break;
272             }
273         }
274     }
275     /*
276     * Set the cursor back to normal (because we changed it at the beginning
277     * of the method body)
278     */
279     this.setCursor(Cursor.getDefaultCursor());
280 }
281
282 }
283

```