

```
1 import components.naturalnumber.NaturalNumber;
2
3
4 /**
5  * Controller class.
6  *
7  * @author Shyam Sai Bethina
8  */
9 public final class NNCalcController1 implements NNCalcController {
10
11     /**
12      * Model object.
13      */
14     private final NNCalcModel model;
15
16     /**
17      * View object.
18      */
19     private final NNCalcView view;
20
21     /**
22      * Useful constants.
23      */
24     private static final NaturalNumber TWO = new
        NaturalNumber2(2),
25         INT_LIMIT = new NaturalNumber2(Integer.MAX_VALUE),
26         ZERO = new NaturalNumber2();
27
28     /**
29      * Updates this.view to display this.model, and to allow only
        operations
30      * that are legal given this.model.
31      *
32      * @param model
33      *         the model
34      * @param view
35      *         the view
36      * @ensures [view has been updated to be consistent with
        model]
37      */
38     private static void updateViewToMatchModel(NNCalcModel model,
39         NNCalcView view) {
40
41         //Gets the values from the top and bottom displays
42         NaturalNumber input = model.bottom();
```

```
43     NaturalNumber output = model.top();
44
45     //Updates the bottom and top display based model values
46     view.updateBottomDisplay(input);
47     view.updateTopDisplay(output);
48
49     /*
50     * If the input is greater than 0, then the divide
function is allowed,
51     * but if it is 0 or less than 0, the divide button
becomes disabled to
52     * avoid divide by 0 and negative numbers
53     */
54     if (!input.isZero()) {
55         view.updateDivideAllowed(true);
56     } else {
57         view.updateDivideAllowed(false);
58     }
59
60     /*
61     * If the input is less than 0, then the subtract function
is allowed,
62     * but if it less than the output, the subtract button
becomes disabled
63     * to avoid negative numbers
64     */
65     if (input.compareTo(output) < 0) {
66         view.updateSubtractAllowed(true);
67     } else {
68         view.updateSubtractAllowed(false);
69     }
70
71     /*
72     * If the input is less than the integer limit, then the
power function
73     * is allowed, but if it greater than the integer limit,
the power
74     * button becomes disabled to avoid outOfBounds error
because the
75     * processPowerMethod uses the toInt(), which cannot
convert any object
76     * that is greater than the integer limit
77     */
78     if (input.compareTo(INT_LIMIT) < 0) {
```

```
79         view.updatePowerAllowed(true);
80     } else {
81         view.updatePowerAllowed(false);
82     }
83
84     /*
85     * If the input is greater than or equal to two, and is
86     less than the
87     * integer limit, then the root function is allowed. But
88     if it greater
89     * than the integer limit or is less than two, the power
90     button becomes
91     * disabled to avoid outOfBounds error because the
92     processRootMethod
93     * uses the toInt(), which cannot convert any object that
94     is greater
95     * than the integer limit
96     */
97     if (input.compareTo(TWO) >= 0 &&
98         input.compareTo(INT_LIMIT) < 0) {
99         view.updateRootAllowed(true);
100     } else {
101         view.updateRootAllowed(false);
102     }
103 }
104
105 /**
106  * Constructor.
107  *
108  * @param model
109  *         model to connect to
110  * @param view
111  *         view to connect to
112  */
113 public NNCalcController1(NNCalcModel model, NNCalcView view) {
114     this.model = model;
115     this.view = view;
116     updateViewToMatchModel(model, view);
117 }
118
119 @Override
120 public void processClearEvent() {
121     /*
122     * Get alias to bottom from model
123     */
124 }
```

```
117         */
118         NaturalNumber bottom = this.model.bottom();
119         /*
120         * Update model in response to this event
121         */
122         bottom.clear();
123         /*
124         * Update view to reflect changes in model
125         */
126         updateViewToMatchModel(this.model, this.view);
127     }
128
129     @Override
130     public void processSwapEvent() {
131         /*
132         * Get aliases to top and bottom from model
133         */
134         NaturalNumber top = this.model.top();
135         NaturalNumber bottom = this.model.bottom();
136         /*
137         * Update model in response to this event
138         */
139         NaturalNumber temp = top.newInstance();
140         temp.transferFrom(top);
141         top.transferFrom(bottom);
142         bottom.transferFrom(temp);
143         /*
144         * Update view to reflect changes in model
145         */
146         updateViewToMatchModel(this.model, this.view);
147     }
148
149     @Override
150     public void processEnterEvent() {
151         /*
152         * Get aliases to top and bottom from model
153         */
154         NaturalNumber bottom = this.model.bottom();
155         NaturalNumber top = this.model.top();
156         /*
157         * Update model in response to this event
158         */
159         top.copyFrom(bottom);
160         /*
```

```
161         * Update view to reflect changes in model
162         */
163         updateViewToMatchModel(this.model, this.view);
164     }
165
166     @Override
167     public void processAddEvent() {
168         /*
169         * Get aliases to top and bottom from model
170         */
171         NaturalNumber top = this.model.top();
172         NaturalNumber bottom = this.model.bottom();
173         /*
174         * Update model in response to this event
175         */
176         bottom.add(top);
177         top.clear();
178         /*
179         * Update view to reflect changes in model
180         */
181         updateViewToMatchModel(this.model, this.view);
182     }
183
184
185     @Override
186     public void processSubtractEvent() {
187         /*
188         * Get aliases to top and bottom from model
189         */
190         NaturalNumber top = this.model.top();
191         NaturalNumber bottom = this.model.bottom();
192         /*
193         * Update model in response to this event
194         */
195         top.subtract(bottom);
196         bottom.transferFrom(top);
197         /*
198         * Update view to reflect changes in model
199         */
200         updateViewToMatchModel(this.model, this.view);
201     }
202
203
204     @Override
```

```
205     public void processMultiplyEvent() {
206         /*
207          * Get aliases to top and bottom from model
208          */
209         NaturalNumber top = this.model.top();
210         NaturalNumber bottom = this.model.bottom();
211         /*
212          * Update model in response to this event
213          */
214         top.multiply(bottom);
215         bottom.transferFrom(top);
216         /*
217          * Update view to reflect changes in model
218          */
219         updateViewToMatchModel(this.model, this.view);
220     }
221
222     @Override
223     public void processDivideEvent() {
224         /*
225          * Get aliases to top and bottom from model
226          */
227         NaturalNumber top = this.model.top();
228         NaturalNumber bottom = this.model.bottom();
229         /*
230          * remainder is the remainder when top is divide by bottom
231          */
232         NaturalNumber remainder = top.divide(bottom);
233         /*
234          * Update model in response to this event, top displays
the remainder
235          */
236         bottom.transferFrom(top);
237         top.transferFrom(remainder);
238         /*
239          * Update view to reflect changes in model
240          */
241         updateViewToMatchModel(this.model, this.view);
242     }
243
244     @Override
245     public void processPowerEvent() {
246         /*
247          * Update model in response to this event
```

```
248         */
249         NaturalNumber top = this.model.top();
250         NaturalNumber bottom = this.model.bottom();
251         /*
252         * bottom is converted to an integer, and top becomes top
to the power
253         * of bottom
254         */
255         top.power(bottom.toInt());
256         /*
257         * Update model in response to this events
258         */
259         bottom.transferFrom(top);
260         /*
261         * Update view to reflect changes in model
262         */
263         updateViewToMatchModel(this.model, this.view);
264     }
265
266     @Override
267     public void processRootEvent() {
268         /*
269         * Update model in response to this event
270         */
271         NaturalNumber top = this.model.top();
272         NaturalNumber bottom = this.model.bottom();
273         /*
274         * bottom is converted to an integer, and top becomes
bottom root of top
276         */
277         top.root(bottom.toInt());
278         /*
279         * Update model in response to this events
280         */
281         bottom.transferFrom(top);
282         /*
283         * Update view to reflect changes in model
284         */
285         updateViewToMatchModel(this.model, this.view);
286     }
287
288     @Override
289     public void processAddNewDigitEvent(int digit) {
```

```
290      /*
291      * Update model in response to this event
292      */
293      NaturalNumber bottom = this.model.bottom();
294      /*
295      * Adds a new digit to the input by multiplying by 10 and
adding the new
296      * digit
297      */
298      bottom.multiplyBy10(digit);
299      /*
300      * Update view to reflect changes in model
301      */
302      updateViewToMatchModel(this.model, this.view);
303  }
304
305 }
306
```