

```
1 import components.naturalnumber.NaturalNumber;
2 import components.naturalnumber.NaturalNumber2;
3
4 /**
5  * Controller class.
6  *
7  * @author Shafin Alam
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
27     /**
28      * Updates this.view to display this.model, and to
       allow only operations
29      * that are legal given this.model.
30      *
```

```
31     * @param model
32     *           the model
33     * @param view
34     *           the view
35     * @ensures [view has been updated to be consistent
    with model]
36     */
37     private static void updateViewToMatchModel(NNCalcModel
    model,
38         NNCalcView view) {
39         //Save the top and bottom of the model to
    NaturalNumbers
40         NaturalNumber top = model.top();
41         NaturalNumber bottom = model.bottom();
42         /*
43         * Should check if subtraction is allowed, (bottom
    must be smaller than
44         * top). Otherwise it's allowed.
45         */
46         if (bottom.compareTo(top) > 0) {
47             view.updateSubtractAllowed(false);
48         } else {
49             view.updateSubtractAllowed(true);
50         }
51         /*
52         * Should check if division is allowed, (division
    by 0 = illegal).
53         * Otherwise it's allowed.
54         */
55         if (bottom.isZero()) {
56             view.updateDivideAllowed(false);
57         } else {
58             view.updateDivideAllowed(true);
59         }
```

```
60      /*
61      * Should check if power is allowed, (bottom can't
    be larger than the
62      * maximum of any integer). Otherwise it is
    allowed.
63      */
64      if (bottom.compareTo(INT_LIMIT) <= 0) {
65          view.updatePowerAllowed(true);
66      } else {
67          view.updatePowerAllowed(false);
68      }
69      /*
70      * Should check if root is allowed, (bottom must
    be larger than 2).
71      * Otherwise it is allowed.
72      */
73      if (bottom.compareTo(TWO) >= 0 && bottom.compareTo
    (INT_LIMIT) <= 0) {
74          view.updateRootAllowed(true);
75      } else {
76          view.updateRootAllowed(false);
77      }
78      /*
79      * View should change accordingly.
80      */
81      view.updateTopDisplay(top);
82      view.updateBottomDisplay(bottom);
83
84  }
85
86  /**
87   * Constructor.
88   *
89   * @param model
```

```
90      *           model to connect to
91      * @param view
92      *           view to connect to
93      */
94      public NNCalcController1(NNCalcModel model, NNCalcView
view) {
95          this.model = model;
96          this.view = view;
97          updateViewToMatchModel(model, view);
98      }
99
100     @Override
101     public void processClearEvent() {
102         /*
103          * Get alias to bottom from model
104          */
105         NaturalNumber bottom = this.model.bottom();
106         /*
107          * Update model in response to this event
108          */
109         bottom.clear();
110         /*
111          * Update view to reflect changes in model
112          */
113         updateViewToMatchModel(this.model, this.view);
114     }
115
116     @Override
117     public void processSwapEvent() {
118         /*
119          * Get aliases to top and bottom from model
120          */
121         NaturalNumber top = this.model.top();
122         NaturalNumber bottom = this.model.bottom();
```

```
123         /*
124         * Update model in response to this event
125         */
126         NaturalNumber temp = top.newInstance();
127         temp.transferFrom(top);
128         top.transferFrom(bottom);
129         bottom.transferFrom(temp);
130         /*
131         * Update view to reflect changes in model
132         */
133         updateViewToMatchModel(this.model, this.view);
134     }
135
136     @Override
137     public void processEnterEvent() {
138         /*
139         * Should set the top equal to the bottom or copy
140         it. Update model in
141         * response to this event
142         */
143         this.model.top().copyFrom(this.model.bottom());
144         /*
145         * Update view to reflect changes in model
146         */
147         updateViewToMatchModel(this.model, this.view);
148     }
149
150     @Override
151     public void processAddEvent() {
152         /*
153         * Should add the top and the bottom, and then
154         clear the top; saving the
155         * answer to the bottom. Update model in response
```

```
    to this event
155         */
156         this.model.bottom().add(this.model.top());
157         this.model.top().clear();
158         /*
159         * Update view to reflect changes in model
160         */
161         updateViewToMatchModel(this.model, this.view);
162
163     }
164
165     @Override
166     public void processSubtractEvent() {
167         /*
168         * Should subtract the bottom from the top, and
169         * then save that number to
170         * the bottom; afterwards, clearing the top.
171         * Update model in response to
172         * this event
173         */
174         this.model.top().subtract(this.model.bottom());
175         this.model.bottom().copyFrom(this.model.top());
176         this.model.top().clear();
177         /*
178         * Update view to reflect changes in model
179         */
180         updateViewToMatchModel(this.model, this.view);
181
182     }
183
184     @Override
185     public void processMultiplyEvent() {
186         /*
187         * Should multiply the top and bottom together,
```

```
        and clear the top;
186         * saving the answer to the bottom. Update model
        in response to this
187         * event
188         */
189         this.model.bottom().multiply(this.model.top());
190         this.model.top().clear();
191         /*
192         * Update view to reflect changes in model
193         */
194         updateViewToMatchModel(this.model, this.view);
195
196     }
197
198     @Override
199     public void processDivideEvent() {
200         /*
201         * Should divide the top by the bottom, and save
        the answer to the
202         * bottom, clearing the top afterwards. Update
        model in response to this
203         * event
204         */
205         NaturalNumber r = this.model.top().divide
        (this.model.bottom());
206         this.model.bottom().copyFrom(this.model.top());
207         this.model.top().copyFrom(r);
208         /*
209         * Update view to reflect changes in model
210         */
211         updateViewToMatchModel(this.model, this.view);
212
213     }
214
```

```
215     @Override
216     public void processPowerEvent() {
217         /*
218         * Should take the top to the power of the bottom,
219         and save the answer
220         * to the bottom, clearing the top afterwards.
221         Update model in response
222         * to this event
223         */
224         this.model.top().power(this.model.bottom().toInt
225         ());
226         this.model.bottom().copyFrom(this.model.top());
227         this.model.top().clear();
228         /*
229         * Update view to reflect changes in model
230         */
231         updateViewToMatchModel(this.model, this.view);
232     }
233
234     @Override
235     public void processRootEvent() {
236         /*
237         * Should take the bottom"th" root of the top,
238         saving the answer to the
239         * bottom, clearing the top afterwards. Update
240         model in response to this
241         * event
242         */
243         this.model.top().root(this.model.bottom().toInt
244         ());
245         this.model.bottom().copyFrom(this.model.top());
246         this.model.top().clear();
247         /*
```



```
243         * Update view to reflect changes in model
244         */
245         updateViewToMatchModel(this.model, this.view);
246
247     }
248
249     @Override
250     public void processAddNewDigitEvent(int digit) {
251         /*
252         * This will update the bottom when the user
253         inputs a number, by
254         * multiplying by 10, and then adding the digit.
255         Update model in
256         * response to this event
257         */
258         this.model.bottom().multiplyBy10(digit);
259         /*
260         * Update view to reflect changes in model
261         */
262         updateViewToMatchModel(this.model, this.view);
263     }
264 }
265
```