

```
1 import components.naturalnumber.NaturalNumber;
2
3
4 /**
5  * Controller class.
6  *
7  * @author Yakob Getu
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
40         ///// Retrieve and display top and bottom numbers
    from the model.
41         NaturalNumber top = model.top();
42         NaturalNumber bottom = model.bottom();
43         view.updateTopDisplay(top);
44         view.updateBottomDisplay(bottom);
45
46         ///// Update UI controls based on codes logic and
    model conditions.
47
48         view.updateRootAllowed(model.bottom().compareTo(INT_LIMIT
    ) <= 0
49             && model.bottom().compareTo(TWO) >= 0);
50
51         view.updatePowerAllowed(model.bottom().compareTo(INT_LIMI
    T) <= 0);
52
53         view.updateSubtractAllowed(model.bottom().compareTo(top)
    <= 0);
54
55         view.updateDivideAllowed(!
    model.bottom().isZero());
56     }
57
58     /**
59     * Constructor.
60     *
61     * @param model
62     *             model to connect to
63     * @param view
64     *             view to connect to
65     */
66     public NNCalcController1(NNCalcModel model,
    NNCalcView view) {
```

```
64         this.model = model;
65         this.view = view;
66         updateViewToMatchModel(model, view);
67     }
68
69     @Override
70     public void processClearEvent() {
71         /*
72          * Get alias to bottom from model
73          */
74         NaturalNumber bottom = this.model.bottom();
75         /*
76          * Update model in response to this event
77          */
78         bottom.clear();
79         /*
80          * Update view to reflect changes in model
81          */
82         updateViewToMatchModel(this.model, this.view);
83     }
84
85     @Override
86     public void processSwapEvent() {
87         /*
88          * Get aliases to top and bottom from model
89          */
90         NaturalNumber top = this.model.top();
91         NaturalNumber bottom = this.model.bottom();
92         /*
93          * Update model in response to this event
94          */
95         NaturalNumber temp = top.newInstance();
96         temp.transferFrom(top);
97         top.transferFrom(bottom);
98         bottom.transferFrom(temp);
99         /*
100          * Update view to reflect changes in model
101          */
102         updateViewToMatchModel(this.model, this.view);
```

```
103     }
104
105     @Override
106     public void processEnterEvent() {
107         // Copy the bottom number to the top in the
108         model.
109         NaturalNumber top = this.model.top();
110         NaturalNumber bottom = this.model.bottom();
111         top.copyFrom(bottom);
112         // Update the view to match the current state of
113         the model.
114         updateViewToMatchModel(this.model, this.view);
115     }
116
117     @Override
118     public void processAddEvent() {
119         // Add the bottom number to the top and update
120         both in the model.
121         NaturalNumber top = this.model.top();
122         NaturalNumber bottom = this.model.bottom();
123         top.add(bottom);
124         bottom.transferFrom(top);
125         // Refresh the view to reflect the updated model.
126         updateViewToMatchModel(this.model, this.view);
127     }
128
129     @Override
130     public void processSubtractEvent() {
131         // Subtract the bottom number from the top and
132         update both.
133         NaturalNumber top = this.model.top();
134         NaturalNumber bottom = this.model.bottom();
135         top.subtract(bottom);
136         bottom.transferFrom(top);
137         // Refresh the view to reflect changes.
138         updateViewToMatchModel(this.model, this.view);
139     }
140
141     @Override
```

```
138     public void processMultiplyEvent() {
139         // Multiply the top number by the bottom and
update both.
140         NaturalNumber top = this.model.top();
141         NaturalNumber bottom = this.model.bottom();
142         top.multiply(bottom);
143         bottom.transferFrom(top);
144         // Update the view to match the new model state.
145         updateViewToMatchModel(this.model, this.view);
146     }
147
148     @Override
149     public void processDivideEvent() {
150         // Divide the top number by the bottom, handle
the remainder.
151         NaturalNumber top = this.model.top();
152         NaturalNumber bottom = this.model.bottom();
153         NaturalNumber remain = top.divide(bottom);
154         bottom.transferFrom(top);
155         top.transferFrom(remain);
156         // Refresh the view with the new values.
157         updateViewToMatchModel(this.model, this.view);
158     }
159
160     @Override
161     public void processPowerEvent() {
162         // Raise the top number to the power of the
bottom's integer value.
163         NaturalNumber top = this.model.top();
164         NaturalNumber bottom = this.model.bottom();
165         top.power(bottom.toInt());
166         bottom.transferFrom(top);
167         // Update the view to display the results of the
power operation.
168         updateViewToMatchModel(this.model, this.view);
169     }
170
171     @Override
172     public void processRootEvent() {
```

```
173         // Calculate the root of the top number based on
        the bottom's value.
174         NaturalNumber top = this.model.top();
175         NaturalNumber bottom = this.model.bottom();
176         top.root(bottom.toInt());
177         bottom.transferFrom(top);
178         // Update the view to show the results of the
        root operation.
179         updateViewToMatchModel(this.model, this.view);
180     }
181
182     @Override
183     public void processAddNewDigitEvent(int digit) {
184         // Append a new digit to the bottom number.
185         NaturalNumber bottom = this.model.bottom();
186         bottom.multiplyBy10(digit);
187         // Update the view to reflect the new number.
188         updateViewToMatchModel(this.model, this.view);
189     }
190
191 }
192
```