```java
import components.naturalnumber.NaturalNumber;

/**
 * Controller class.
 *
 * @author Steven Masilonis
 */
public final class NNCalcController1 implements NNCalcController {

    /**
     * Model object.
     */
    private final NNCalcModel model;

    /**
     * View object.
     */
    private final NNCalcView view;

    /**
     * Useful constants.
     */
    private static final NaturalNumber TWO = new NaturalNumber2(2),
            INT_LIMIT = new NaturalNumber2(Integer.MAX_VALUE);

    /**
     * Updates this.view to display this.model, and to allow only operations
     * that are legal given this.model.
     *
     * @param model
     *            the model
     * @param view
     *            the view
     * @ensures [view has been updated to be consistent with model]
     */
    private static void updateViewToMatchModel(NNCalcModel model,
            NNCalcView view) {
        final NaturalNumber output = model.top();
        final NaturalNumber input = model.bottom();
        view.updateTopDisplay(output);
        view.updateBottomDisplay(input);
```

```
43          view.updateSubtractAllowed(output.compareTo(input) >= 0);
44          view.updateDivideAllowed(!input.isZero());
45          view.updatePowerAllowed(
46                  input.compareTo(NNCalcController1.INT_LIMIT) <=
    0);
47
    view.updateRootAllowed(input.compareTo(NNCalcController1.TWO) >= 0
48                  && input.compareTo(NNCalcController1.INT_LIMIT) <=
    0);
49
50      }
51
52      /**
53       * Constructor.
54       *
55       * @param model
56       *             model to connect to
57       * @param view
58       *             view to connect to
59       */
60      public NNCalcController1(NNCalcModel model, NNCalcView view) {
61          this.model = model;
62          this.view = view;
63          updateViewToMatchModel(model, view);
64      }
65
66      @Override
67      public void processClearEvent() {
68          /*
69           * Get alias to bottom from model
70           */
71          NaturalNumber bottom = this.model.bottom();
72          /*
73           * Update model in response to this event
74           */
75          bottom.clear();
76          /*
77           * Update view to reflect changes in model
78           */
79          updateViewToMatchModel(this.model, this.view);
80      }
81
82      @Override
83      public void processSwapEvent() {
```

```
 84         /*
 85          * Get aliases to top and bottom from model
 86          */
 87         NaturalNumber top = this.model.top();
 88         NaturalNumber bottom = this.model.bottom();
 89         /*
 90          * Update model in response to this event
 91          */
 92         NaturalNumber temp = top.newInstance();
 93         temp.transferFrom(top);
 94         top.transferFrom(bottom);
 95         bottom.transferFrom(temp);
 96         /*
 97          * Update view to reflect changes in model
 98          */
 99         updateViewToMatchModel(this.model, this.view);
100     }
101
102     @Override
103     public void processEnterEvent() {
104         this.model.top().copyFrom(this.model.bottom());
105         updateViewToMatchModel(this.model, this.view);
106
107     }
108
109     @Override
110     public void processAddEvent() {
111
112         NaturalNumber answer = this.model.top().newInstance();
113         answer.transferFrom(this.model.top());
114         answer.add(this.model.bottom());
115         this.model.bottom().transferFrom(answer);
116         updateViewToMatchModel(this.model, this.view);
117     }
118
119     @Override
120     public void processSubtractEvent() {
121
122         NaturalNumber answer = this.model.top().newInstance();
123         answer.transferFrom(this.model.top());
124         answer.subtract(this.model.bottom());
125         this.model.bottom().transferFrom(answer);
126         updateViewToMatchModel(this.model, this.view);
127
```

```
128     }
129
130     @Override
131     public void processMultiplyEvent() {
132
133         NaturalNumber answer = this.model.top().newInstance();
134         answer.transferFrom(this.model.top());
135         answer.multiply(this.model.bottom());
136         this.model.bottom().transferFrom(answer);
137         updateViewToMatchModel(this.model, this.view);
138
139     }
140
141     @Override
142     public void processDivideEvent() {
143
144         NaturalNumber answer = this.model.top().newInstance();
145         answer.transferFrom(this.model.top());
146         answer.divide(this.model.bottom());
147         this.model.bottom().transferFrom(answer);
148         updateViewToMatchModel(this.model, this.view);
149
150     }
151
152     @Override
153     public void processPowerEvent() {
154
155         NaturalNumber answer = this.model.top().newInstance();
156         answer.transferFrom(this.model.top());
157         answer.power(this.model.bottom().toInt());
158         this.model.bottom().transferFrom(answer);
159         updateViewToMatchModel(this.model, this.view);
160
161     }
162
163     @Override
164     public void processRootEvent() {
165
166         NaturalNumber answer = this.model.top().newInstance();
167         answer.transferFrom(this.model.top());
168         answer.root(this.model.bottom().toInt());
169         this.model.bottom().transferFrom(answer);
170         updateViewToMatchModel(this.model, this.view);
171
```

```
172     }
173
174     @Override
175     public void processAddNewDigitEvent(int digit) {
176
177         this.model.bottom().multiplyBy10(digit);
178         updateViewToMatchModel(this.model, this.view);
179
180     }
181
182 }
183
```