```java
import components.naturalnumber.NaturalNumber;

/**
 * Controller class.
 *
 * @author Jeng Zhuang
 */
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]
     */
```

```
37    private static void updateViewToMatchModel(NNCalcModel
   model, NNCalcView view) {
38
39        // Update the natural values in the top display area
40        view.updateTopDisplay(model.top());
41        // Update the natural values in the bottom display area
42        view.updateBottomDisplay(model.bottom());
43        // Set the subtraction button status
44        // (available when the bottom value is <= the top
   value)
45
   view.updateSubtractAllowed(model.bottom().compareTo(model.top()
   ) <= 0);
46        // Set the status of the division button
47        // (available when the bottom value is not 0)
48        view.updateDivideAllowed(!model.bottom().isZero());
49        // Set the status of the exponentiation button
50        // (available when the bottom value is <= the maximum
   integer value)
51
   view.updatePowerAllowed(model.bottom().compareTo(INT_LIMIT) <=
   0);
52        // Set the status of the root operation button
53        // (available when the bottom value is >= 2 and <= the
   maximum integer value)
54        view.updateRootAllowed(model.bottom().compareTo(TWO) >=
   0
55                && model.bottom().compareTo(INT_LIMIT) <= 0);
56
57    }
58
59    /**
60     * Constructor.
61     *
62     * @param model
63     *            model to connect to
64     * @param view
65     *            view to connect to
66     */
```

```
 67     public NNCalcController1(NNCalcModel model, NNCalcView
    view) {
 68         this.model = model;
 69         this.view = view;
 70         updateViewToMatchModel(model, view);
 71     }
 72
 73     @Override
 74     public void processClearEvent() {
 75         /*
 76          * Get alias to bottom from model
 77          */
 78         NaturalNumber bottom = this.model.bottom();
 79         /*
 80          * Update model in response to this event
 81          */
 82         bottom.clear();
 83         /*
 84          * Update view to reflect changes in model
 85          */
 86         updateViewToMatchModel(this.model, this.view);
 87     }
 88
 89     @Override
 90     public void processSwapEvent() {
 91         /*
 92          * Get aliases to top and bottom from model
 93          */
 94         NaturalNumber top = this.model.top();
 95         NaturalNumber bottom = this.model.bottom();
 96         /*
 97          * Update model in response to this event
 98          */
 99         NaturalNumber temp = top.newInstance();
100         temp.transferFrom(top);
101         top.transferFrom(bottom);
102         bottom.transferFrom(temp);
103         /*
104          * Update view to reflect changes in model
```

```
105          */
106         updateViewToMatchModel(this.model, this.view);
107     }
108 
109     @Override
110     public void processEnterEvent() {
111 
112         /*
113          * Copy the bottom value to the top
114          */
115         this.model.top().copyFrom(this.model.bottom());
116         /*
117          * Update view to reflect changes in model
118          */
119         updateViewToMatchModel(this.model, this.view);
120 
121     }
122 
123     @Override
124     public void processAddEvent() {
125 
126         /*
127          * Perform addition operation: bottom = top + bottom
128          */
129         NaturalNumber top = this.model.top();
130         NaturalNumber bottom = this.model.bottom();
131         bottom.add(top); // Add the top value to the bottom
132         top.clear(); // Clear the top
133         updateViewToMatchModel(this.model, this.view);
134 
135     }
136 
137     @Override
138     public void processSubtractEvent() {
139 
140         /*
141          * Perform subtraction operation: bottom = top - bottom
142          */
143         NaturalNumber top = this.model.top();
```

```
144         NaturalNumber bottom = this.model.bottom();
145         top.subtract(bottom); // Subtract the top value to the
    bottom
146         bottom.transferFrom(top);
147         top.clear(); // Clear the top
148         updateViewToMatchModel(this.model, this.view);
149 
150     }
151 
152     @Override
153     public void processMultiplyEvent() {
154 
155         /*
156          * Perform multiply operation: bottom = top * bottom
157          */
158         NaturalNumber top = this.model.top();
159         NaturalNumber bottom = this.model.bottom();
160         top.multiply(bottom); // multiply the top value to the
    bottom
161         bottom.transferFrom(top);
162         top.clear(); // Clear the top
163         updateViewToMatchModel(this.model, this.view);
164 
165     }
166 
167     @Override
168     public void processDivideEvent() {
169 
170         /*
171          * Perform division operations: bottom = top / bottom,
    top = top % bottom
172          */
173         NaturalNumber top = this.model.top();
174         NaturalNumber bottom = this.model.bottom();
175 
176         // Calculate the remainder
177         NaturalNumber remainder = top.divide(bottom);
178 
179         // Exchange result: bottom stores the quotient, top
```

```
    stores the remainder
180         NaturalNumber quotient = top.newInstance();
181         quotient.transferFrom(top);
182         bottom.transferFrom(quotient);
183         top.transferFrom(remainder);
184 
185         updateViewToMatchModel(this.model, this.view);
186 
187     }
188 
189     @Override
190     public void processPowerEvent() {
191 
192         /*
193          * Perform exponentiation operations: bottom = top ^
    bottom
194          */
195         NaturalNumber base = this.model.top();
196         NaturalNumber exponent = this.model.bottom();
197 
198         // Raise base to the power of exponent
199         base.power(exponent.toInt());
200 
201         // The result is stored in the bottom and the top is
    cleared
202         this.model.bottom().transferFrom(base);
203         base.clear();
204 
205         updateViewToMatchModel(this.model, this.view);
206 
207     }
208 
209     @Override
210     public void processRootEvent() {
211 
212         /*
213          * Perform root operations
214          */
215         NaturalNumber radicand = this.model.top();
```

```
216         NaturalNumber degree = this.model.bottom();
217 
218         // Perform the root operation
219         radicand.root(degree.toInt());
220 
221         // The result is stored in the bottom and the top is
    cleared
222         this.model.bottom().transferFrom(radicand);
223         radicand.clear();
224 
225         updateViewToMatchModel(this.model, this.view);
226 
227     }
228 
229     @Override
230     public void processAddNewDigitEvent(int digit) {
231 
232         /*
233          * Add a new number to the end (equivalent to
    multiplying by 10 and then
234          * adding digit)
235          */
236         NaturalNumber bottom = this.model.bottom();
237         bottom.multiplyBy10(1); // Shift one place to the left
    (decimal)
238         bottom.add(new NaturalNumber2(digit)); // Add new
    numbers
239         updateViewToMatchModel(this.model, this.view);
240 
241     }
242 
243 }
244 
```