```java
import components.naturalnumber.NaturalNumber;

/**
 * Controller class.
 *
 * @author David Park
 */
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) {

        NaturalNumber in = model.top();
        NaturalNumber out = model.bottom();

        // Update the top and bottom display to show top n bottom number

        view.updateTopDisplay(in);
        view.updateBottomDisplay(out);
        // Enable the subtract button if >= top number
        view.updateSubtractAllowed(out.compareTo(in) <= 0);
        // Enable the divide button if the bottom number is not zero
        view.updateDivideAllowed(!out.isZero());
        // Enable the root button if the bottom number is 2-limit
        view.updateRootAllowed(
                out.compareTo(TWO) >= 0 && out.compareTo(INT_LIMIT) <= 0);
        view.updatePowerAllowed(out.compareTo(INT_LIMIT) <= 0);
    }

    /**
     * Constructor.
```

```
 59      *
 60      * @param model
 61      *            model to connect to
 62      * @param view
 63      *            view to connect to
 64      */
 65     public NNCalcController1(NNCalcModel model, NNCalcView view) {
 66         this.model = model;
 67         this.view = view;
 68         updateViewToMatchModel(model, view);
 69     }
 70
 71     @Override
 72     public void processClearEvent() {
 73         /*
 74          * Get alias to bottom from model
 75          */
 76         NaturalNumber bottom = this.model.bottom();
 77         /*
 78          * Update model in response to this event
 79          */
 80         bottom.clear();
 81         /*
 82          * Update view to reflect changes in model
 83          */
 84         updateViewToMatchModel(this.model, this.view);
 85     }
 86
 87     @Override
 88     public void processSwapEvent() {
 89         /*
 90          * Get aliases to top and bottom from model
 91          */
 92         NaturalNumber top = this.model.top();
 93         NaturalNumber bottom = this.model.bottom();
 94         /*
 95          * Update model in response to this event
 96          */
 97         NaturalNumber temp = top.newInstance();
 98         temp.transferFrom(top);
 99         top.transferFrom(bottom);
100         bottom.transferFrom(temp);
101         /*
102          * Update view to reflect changes in model
103          */
104         updateViewToMatchModel(this.model, this.view);
105     }
106
107     @Override
108     public void processEnterEvent() {
109         NaturalNumber in = this.model.top();
110         NaturalNumber out = this.model.bottom();
111
112         //Processes an "Enter" event which copies the bottom number to the top.
113
114         in.copyFrom(out);
115
```

```
116         updateViewToMatchModel(this.model, this.view);
117     }
118 
119     @Override
120     public void processAddEvent() {
121         NaturalNumber in = this.model.top();
122         NaturalNumber out = this.model.bottom();
123 
124         // Processes an "Add" event where the top number is added to the
125         // bottom number, and the top is then cleared.
126 
127         out.add(in);
128         in.clear();
129 
130         updateViewToMatchModel(this.model, this.view);
131     }
132 
133     @Override
134     public void processSubtractEvent() {
135         NaturalNumber in = this.model.top();
136         NaturalNumber out = this.model.bottom();
137 
138         // Processes a "Subtract" event where the top number is subtracted
139         // from the bottom number, then the result is transferred to the bottom.
140 
141         in.subtract(out);
142         out.transferFrom(in);
143 
144         updateViewToMatchModel(this.model, this.view);
145     }
146 
147     @Override
148     public void processMultiplyEvent() {
149         NaturalNumber in = this.model.top();
150         NaturalNumber out = this.model.bottom();
151 
152         // Processes a "Multiply" event where the top number is multiplied
153         // with the bottom number, and the top is then cleared.
154 
155         out.multiply(in);
156         in.clear();
157 
158         updateViewToMatchModel(this.model, this.view);
159     }
160 
161     @Override
162     public void processDivideEvent() {
163         NaturalNumber in = this.model.top();
164         NaturalNumber out = this.model.bottom();
165         NaturalNumber remainder = in.newInstance();
166         // Processes a "Divide" event, dividing the top number by the bottom,
167         // storing the quotient in bottom, and remainder in top.
168         remainder = in.divide(out);
169         out.transferFrom(in);
170         in.transferFrom(remainder);
171 
172         updateViewToMatchModel(this.model, this.view);
```

```
173     }
174
175     @Override
176     public void processPowerEvent() {
177         NaturalNumber in = this.model.top();
178         NaturalNumber out = this.model.bottom();
179         int ex = out.toInt();
180
181         //Processes a "Power" event where the bottom number is treated as the
182         // exponent to raise the top number.
183
184         in.power(ex);
185         out.transferFrom(in);
186
187         updateViewToMatchModel(this.model, this.view);
188     }
189
190     @Override
191     public void processRootEvent() {
192         NaturalNumber in = this.model.top();
193         NaturalNumber out = this.model.bottom();
194
195         //Processes a "Root" event where the bottom number is treated as
196         // the 'n' in an nth-root operation on the top number.
197
198         int rootNum = out.toInt();
199         in.root(rootNum);
200         out.transferFrom(in);
201
202         updateViewToMatchModel(this.model, this.view);
203     }
204
205     @Override
206     public void processAddNewDigitEvent(int digit) {
207         //Processes an event to add a new digit to the bottom number by
208         // multiplying the bottom number by 10 and adding the digit.
209         NaturalNumber out = this.model.bottom();
210         out.multiplyBy10(digit);
211         updateViewToMatchModel(this.model, this.view);
212     }
213
214 }
215
```