```java
import components.naturalnumber.NaturalNumber;

/**
 * Controller class.
 *
 * @author Feras Akileh
 */
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) {

        // derive top and bottom number from model
        NaturalNumber topNum = model.top();
        NaturalNumber bottomNum = model.bottom();

        // updates the availability of the subtract button
        if (bottomNum.compareTo(topNum) > 0) {
            view.updateSubtractAllowed(false);
        } else {
            view.updateSubtractAllowed(true);
        }

        // updates the availability of the divide button
        if (bottomNum.isZero()) {
            view.updateDivideAllowed(false);
        } else {
            view.updateDivideAllowed(true);
        }

        // updates the availability of the power button
        if (bottomNum.compareTo(INT_LIMIT) <= 0) {
            view.updatePowerAllowed(true);
```

```
61          } else {
62              view.updatePowerAllowed(false);
63          }
64
65          // updates the availability of the root button
66          if (bottomNum.compareTo(TWO) >= 0
67                  && bottomNum.compareTo(INT_LIMIT) <= 0) {
68              view.updateRootAllowed(true);
69          } else {
70              view.updateRootAllowed(false);
71          }
72
73          // updates the view
74          view.updateTopDisplay(topNum);
75          view.updateBottomDisplay(bottomNum);
76
77      }
78
79      /**
80       * Constructor.
81       *
82       * @param model
83       *            model to connect to
84       * @param view
85       *            view to connect to
86       */
87      public NNCalcController1(NNCalcModel model, NNCalcView view) {
88
89          // initializes model and view
90          this.model = model;
91          this.view = view;
92
93          // calls updateViewToMatchModel
94          updateViewToMatchModel(model, view);
95      }
96
97      @Override
98      public void processClearEvent() {
99          /*
100          * Get alias to bottom from model
101          */
102         NaturalNumber bottom = this.model.bottom();
103         /*
104          * Update model in response to this event
105          */
106         bottom.clear();
107         /*
108          * Update view to reflect changes in model
109          */
110         updateViewToMatchModel(this.model, this.view);
111     }
112
113     @Override
114     public void processSwapEvent() {
115         /*
116          * Get aliases to top and bottom from model
117          */
118         NaturalNumber top = this.model.top();
119         NaturalNumber bottom = this.model.bottom();
```

```
120         /*
121          * Update model in response to this event
122          */
123         NaturalNumber temp = top.newInstance();
124         temp.transferFrom(top);
125         top.transferFrom(bottom);
126         bottom.transferFrom(temp);
127         /*
128          * Update view to reflect changes in model
129          */
130         updateViewToMatchModel(this.model, this.view);
131     }
132
133     @Override
134     public void processEnterEvent() {
135
136         // derive numbers from model
137         NaturalNumber topNum = this.model.top();
138         NaturalNumber bottomNum = this.model.bottom();
139
140         // updates the numbers
141         topNum.copyFrom(bottomNum);
142
143         // updates view
144         updateViewToMatchModel(this.model, this.view);
145     }
146
147     @Override
148     public void processAddEvent() {
149
150         // derive numbers from model
151         NaturalNumber topNum = this.model.top();
152         NaturalNumber bottomNum = this.model.bottom();
153
154         // performs the addition
155         bottomNum.add(topNum);
156         topNum.clear();
157
158         // updates view
159         updateViewToMatchModel(this.model, this.view);
160
161     }
162
163     @Override
164     public void processSubtractEvent() {
165
166         // derive numbers from model
167         NaturalNumber topNum = this.model.top();
168         NaturalNumber bottomNum = this.model.bottom();
169
170         // performs the subtraction
171         topNum.subtract(bottomNum);
172         bottomNum.transferFrom(topNum);
173
174         // updates view
175         updateViewToMatchModel(this.model, this.view);
176
177     }
178
```

```
179     @Override
180     public void processMultiplyEvent() {
181
182         // derive numbers from model
183         NaturalNumber topNum = this.model.top();
184         NaturalNumber bottomNum = this.model.bottom();
185
186         // performs the multiplication
187         topNum.multiply(bottomNum);
188         bottomNum.transferFrom(topNum);
189
190         // updates view
191         updateViewToMatchModel(this.model, this.view);
192
193     }
194
195     @Override
196     public void processDivideEvent() {
197
198         // derive numbers from model
199         NaturalNumber topNum = this.model.top();
200         NaturalNumber bottomNum = this.model.bottom();
201
202         // performs the division
203         NaturalNumber dividend = topNum.divide(bottomNum);
204         bottomNum.transferFrom(topNum);
205         topNum.transferFrom(dividend);
206
207         // updates view
208         updateViewToMatchModel(this.model, this.view);
209
210     }
211
212     @Override
213     public void processPowerEvent() {
214
215         // derive numbers from model
216         NaturalNumber topNum = this.model.top();
217         NaturalNumber bottomNum = this.model.bottom();
218
219         // performs the power function
220         topNum.power(bottomNum.toInt());
221         bottomNum.transferFrom(topNum);
222
223         // updates view
224         updateViewToMatchModel(this.model, this.view);
225
226     }
227
228     @Override
229     public void processRootEvent() {
230
231         // derive numbers from model
232         NaturalNumber topNum = this.model.top();
233         NaturalNumber bottomNum = this.model.bottom();
234
235         // performs the root function
236         topNum.root(bottomNum.toInt());
237         bottomNum.transferFrom(topNum);
```

```
238
239         // updates view
240         updateViewToMatchModel(this.model, this.view);
241
242     }
243
244     @Override
245     public void processAddNewDigitEvent(int digit) {
246
247         // derives the number
248         NaturalNumber bottomNum = this.model.bottom();
249
250         // adds the digit
251         bottomNum.multiplyBy10(digit);
252
253         // updates view
254         updateViewToMatchModel(this.model, this.view);
255
256     }
257
258 }
259
```