```
 1 import components.naturalnumber.NaturalNumber;
 2 import components.naturalnumber.NaturalNumber2;
 3
 4 /**
 5  * Controller class.
 6  *
 7  * @author Ansh Pachauri
 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         NaturalNumber top = model.top();
40         NaturalNumber bottom = model.bottom();
41
42         if (bottom.compareTo(top) > 0) {
43             view.updateSubtractAllowed(false);
44         } else {
45             view.updateSubtractAllowed(true);
46         }
47
48         if (bottom.isZero()) {
49             view.updateDivideAllowed(false);
50         } else {
51             view.updateDivideAllowed(true);
52         }
53
54         if (bottom.compareTo(INT_LIMIT) <= 0) {
55             view.updatePowerAllowed(true);
56         } else {
57             view.updatePowerAllowed(false);
58         }
59
60         if (bottom.compareTo(TWO) >= 0 &&
   bottom.compareTo(INT_LIMIT) <= 0) {
61             view.updateRootAllowed(true);
62         } else {
63             view.updateRootAllowed(false);
64         }
65
66         view.updateTopDisplay(top);
67         view.updateBottomDisplay(bottom);
68
69     }
70
71     /**
72      * Constructor.
```

```
 73      *
 74      * @param model
 75      *              model to connect to
 76      * @param view
 77      *              view to connect to
 78      */
 79     public NNCalcController1(NNCalcModel model, NNCalcView
    view) {
 80         this.model = model;
 81         this.view = view;
 82         updateViewToMatchModel(model, view);
 83     }
 84
 85     @Override
 86     public void processClearEvent() {
 87         /*
 88          * Get alias to bottom from model
 89          */
 90         NaturalNumber bottom = this.model.bottom();
 91         /*
 92          * Update model in response to this event
 93          */
 94         bottom.clear();
 95         /*
 96          * Update view to reflect changes in model
 97          */
 98         updateViewToMatchModel(this.model, this.view);
 99     }
100
101     @Override
102     public void processSwapEvent() {
103         /*
104          * Get aliases to top and bottom from model
105          */
106         NaturalNumber top = this.model.top();
107         NaturalNumber bottom = this.model.bottom();
108         /*
109          * Update model in response to this event
110          */
```

```
111         NaturalNumber temp = top.newInstance();
112         temp.transferFrom(top);
113         top.transferFrom(bottom);
114         bottom.transferFrom(temp);
115         /*
116          * Update view to reflect changes in model
117          */
118         updateViewToMatchModel(this.model, this.view);
119     }
120
121     @Override
122     public void processEnterEvent() {
123         /*
124          * Get aliases to top and bottom from model
125          */
126         NaturalNumber top = this.model.top();
127         NaturalNumber bottom = this.model.bottom();
128         /*
129          * Update model in response to this event
130          */
131         top.copyFrom(bottom);
132         /*
133          * Update view to reflect changes in model
134          */
135         updateViewToMatchModel(this.model, this.view);
136
137     }
138
139     @Override
140     public void processAddEvent() {
141         /*
142          * Get aliases to top and bottom from model
143          */
144         NaturalNumber top = this.model.top();
145         NaturalNumber bottom = this.model.bottom();
146         /*
147          * Update model in response to this event
148          */
149         bottom.add(top);
```

```
150         top.clear();
151         /*
152          * Update view to reflect changes in model
153          */
154         updateViewToMatchModel(this.model, this.view);
155 
156     }
157 
158     @Override
159     public void processSubtractEvent() {
160         /*
161          * Get aliases to top and bottom from model
162          */
163         NaturalNumber top = this.model.top();
164         NaturalNumber bottom = this.model.bottom();
165         /*
166          * Update model in response to this event
167          */
168         top.subtract(bottom);
169         bottom.transferFrom(top);
170         /*
171          * Update view to reflect changes in model
172          */
173         updateViewToMatchModel(this.model, this.view);
174 
175     }
176 
177     @Override
178     public void processMultiplyEvent() {
179         /*
180          * Get aliases to top and bottom from model
181          */
182         NaturalNumber top = this.model.top();
183         NaturalNumber bottom = this.model.bottom();
184         /*
185          * Update model in response to this event
186          */
187         top.multiply(bottom);
188         bottom.transferFrom(top);
```

```
189         /*
190          * Update view to reflect changes in model
191          */
192         updateViewToMatchModel(this.model, this.view);
193 
194     }
195 
196     @Override
197     public void processDivideEvent() {
198         /*
199          * Get aliases to top and bottom from model
200          */
201         NaturalNumber top = this.model.top();
202         NaturalNumber bottom = this.model.bottom();
203         /*
204          * Update model in response to this event
205          */
206         NaturalNumber remainder = top.divide(bottom);
207         bottom.transferFrom(top);
208         top.transferFrom(remainder);
209         /*
210          * Update view to reflect changes in model
211          */
212         updateViewToMatchModel(this.model, this.view);
213 
214     }
215 
216     @Override
217     public void processPowerEvent() {
218         /*
219          * Get aliases to top and bottom from model
220          */
221         NaturalNumber top = this.model.top();
222         NaturalNumber bottom = this.model.bottom();
223         /*
224          * Update model in response to this event
225          */
226         top.power(bottom.toInt());
227         bottom.transferFrom(top);
```

```
228         /*
229          * Update view to reflect changes in model
230          */
231         updateViewToMatchModel(this.model, this.view);
232 
233     }
234 
235     @Override
236     public void processRootEvent() {
237         /*
238          * Get aliases to top and bottom from model
239          */
240         NaturalNumber top = this.model.top();
241         NaturalNumber bottom = this.model.bottom();
242         /*
243          * Update model in response to this event
244          */
245         top.root(bottom.toInt());
246         bottom.transferFrom(top);
247         /*
248          * Update view to reflect changes in model
249          */
250         updateViewToMatchModel(this.model, this.view);
251 
252     }
253 
254     @Override
255     public void processAddNewDigitEvent(int digit) {
256         /*
257          * Get aliases to top and bottom from model
258          */
259         NaturalNumber bottom = this.model.bottom();
260         /*
261          * Update model in response to this event
262          */
263         bottom.multiplyBy10(digit);
264         /*
265          * Update view to reflect changes in model
266          */
```

```
267         updateViewToMatchModel(this.model, this.view);
268
269     }
270
271 }
272
```