```java
import static org.junit.Assert.assertEquals;

/**
 * JUnit test fixture for {@code NaturalNumber}'s constructors and kernel
 * methods.
 *
 * @author Shyam Sai Bethina and Yihone Chu
 *
 */
public abstract class NaturalNumberTest {

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @return the new number
     * @ensures constructorTest = 0
     */
    protected abstract NaturalNumber constructorTest();

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @param i
     *            {@code int} to initialize from
     * @return the new number
     * @requires i >= 0
     * @ensures constructorTest = i
     */
    protected abstract NaturalNumber constructorTest(int i);

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @param s
     *            {@code String} to initialize from
     * @return the new number
     * @requires there exists n: NATURAL (s = TO_STRING(n))
```

```
46      * @ensures s = TO_STRING(constructorTest)
47      */
48     protected abstract NaturalNumber constructorTest(String s);
49
50     /**
51      * Invokes the appropriate {@code NaturalNumber} constructor
   for the
52      * implementation under test and returns the result.
53      *
54      * @param n
55      *            {@code NaturalNumber} to initialize from
56      * @return the new number
57      * @ensures constructorTest = n
58      */
59     protected abstract NaturalNumber constructorTest(NaturalNumber
   n);
60
61     /**
62      * Invokes the appropriate {@code NaturalNumber} constructor
   for the
63      * reference implementation and returns the result.
64      *
65      * @return the new number
66      * @ensures constructorRef = 0
67      */
68     protected abstract NaturalNumber constructorRef();
69
70     /**
71      * Invokes the appropriate {@code NaturalNumber} constructor
   for the
72      * reference implementation and returns the result.
73      *
74      * @param i
75      *            {@code int} to initialize from
76      * @return the new number
77      * @requires i >= 0
78      * @ensures constructorRef = i
79      */
80     protected abstract NaturalNumber constructorRef(int i);
81
82     /**
83      * Invokes the appropriate {@code NaturalNumber} constructor
   for the
84      * reference implementation and returns the result.
```

```
 85      *
 86      * @param s
 87      *            {@code String} to initialize from
 88      * @return the new number
 89      * @requires there exists n: NATURAL (s = TO_STRING(n))
 90      * @ensures s = TO_STRING(constructorRef)
 91      */
 92     protected abstract NaturalNumber constructorRef(String s);
 93
 94     /**
 95      * Invokes the appropriate {@code NaturalNumber} constructor
    for the
 96      * reference implementation and returns the result.
 97      *
 98      * @param n
 99      *            {@code NaturalNumber} to initialize from
100      * @return the new number
101      * @ensures constructorRef = n
102      */
103     protected abstract NaturalNumber constructorRef(NaturalNumber
    n);
104
105     /**
106      * Test the no argument constructor.
107      */
108     @Test
109     public void testNoArgConstructor() {
110         NaturalNumber result = this.constructorTest();
111
112         NaturalNumber expected = this.constructorRef();
113
114         assertEquals(expected, result);
115     }
116
117     /**
118      * Test the int argument constructor with 0.
119      */
120     @Test
121     public void testIntConstructor0() {
122         NaturalNumber result = this.constructorTest(0);
123
124         NaturalNumber expected = this.constructorRef(0);
125
126         assertEquals(expected, result);
```

```
127     }
128
129     /**
130      * Test the int argument constructor.
131      */
132     @Test
133     public void testIntConstructor() {
134         final int number = 5;
135         NaturalNumber result = this.constructorTest(number);
136
137         NaturalNumber expected = this.constructorRef(number);
138
139         assertEquals(expected, result);
140     }
141
142     /**
143      * Test the int argument constructor with long input.
144      */
145     @Test
146     public void testIntConstructorLong() {
147         final int number = 5;
148         NaturalNumber result = this.constructorTest(123456789);
149
150         NaturalNumber expected = this.constructorRef(123456789);
151
152         assertEquals(expected, result);
153     }
154
155     /**
156      * Test the String argument constructor.
157      */
158     @Test
159     public void testStringConstructor() {
160         NaturalNumber result = this.constructorTest("5");
161
162         NaturalNumber expected = this.constructorRef("5");
163
164         assertEquals(expected, result);
165
166     }
167
168     /**
169      * Test the String argument constructor with "0".
170      */
```

```
171     @Test
172     public void testStringConstructor0() {
173         NaturalNumber result = this.constructorTest("0");
174
175         NaturalNumber expected = this.constructorRef("0");
176
177         assertEquals(expected, result);
178
179     }
180
181     /**
182      * Test the String argument constructor with really long
    input.
183      */
184     @Test
185     public void testStringConstructorLong() {
186         NaturalNumber result = this.constructorTest("123456789");
187
188         NaturalNumber expected = this.constructorRef("123456789");
189
190         assertEquals(expected, result);
191
192     }
193
194     /**
195      * Test the Natural Number argument constructor.
196      */
197     @Test
198     public void testNNConstructor() {
199         final int number = 5;
200         NaturalNumber result = this.constructorTest(new
    NaturalNumber2(number));
201
202         NaturalNumber expected = this
203                 .constructorRef(new NaturalNumber2(number));
204
205         assertEquals(expected, result);
206     }
207
208     /**
209      * Test the Natural Number argument constructor with 0.
210      */
211     @Test
212     public void testNNConstructor0() {
```

```
213         NaturalNumber result = this.constructorTest(new
    NaturalNumber2());
214
215         NaturalNumber expected = this.constructorRef(new
    NaturalNumber2());
216
217         assertEquals(expected, result);
218     }
219
220     /**
221      * Test the Natural Number argument constructor with long
    input.
222      */
223     @Test
224     public void testNNConstructorLong() {
225         NaturalNumber result = this
226                 .constructorTest(new NaturalNumber2(12345678));
227
228         NaturalNumber expected = this
229                 .constructorRef(new NaturalNumber2(12345678));
230
231         assertEquals(expected, result);
232     }
233
234     /**
235      * Test edge with multiplyBy10 method.
236      */
237     @Test
238     public void testMultiplyBy10() {
239         NaturalNumber result = this.constructorTest();
240         result.multiplyBy10(0);
241
242         NaturalNumber expected = this.constructorRef();
243         expected.multiplyBy10(0);
244         assertEquals(expected, result);
245     }
246
247     /**
248      * Test strange case with multiplyBy10 method.
249      */
250     @Test
251     public void testMultiplyBy102() {
252         NaturalNumber result = this.constructorTest(10);
253         result.multiplyBy10(0);
```

```
254         System.out.println(result);
255
256         NaturalNumber expected = this.constructorRef(10);
257         expected.multiplyBy10(0);
258         System.out.println(expected);
259         assertEquals(expected, result);
260     }
261
262     /**
263      * Test routine case with multiplyBy10 method.
264      */
265     @Test
266     public void testMultiplyBy103() {
267         NaturalNumber result = this.constructorTest(1);
268         result.multiplyBy10(0);
269
270         NaturalNumber expected = this.constructorRef(1);
271         expected.multiplyBy10(0);
272         assertEquals(expected, result);
273     }
274
275     /**
276      * Test routine case with multiplyBy10 method.
277      */
278     @Test
279     public void testMultiplyBy104() {
280         NaturalNumber result = this.constructorTest(10);
281         result.multiplyBy10(7);
282
283         NaturalNumber expected = this.constructorRef(10);
284         expected.multiplyBy10(7);
285         assertEquals(expected, result);
286     }
287
288     /**
289      * Test strange case with divideBy10 method.
290      */
291     @Test
292     public void testDivideBy101() {
293         NaturalNumber result = this.constructorTest("0");
294         result.divideBy10();
295
296         NaturalNumber expected = this.constructorRef("0");
297         expected.divideBy10();
```

```
298         assertEquals(expected, result);
299     }
300
301     /**
302      * Test strange case with divideBy10 method.
303      */
304     @Test
305     public void testDivideBy102() {
306         NaturalNumber result = this.constructorTest();
307         result.divideBy10();
308
309         NaturalNumber expected = this.constructorRef();
310         expected.divideBy10();
311         assertEquals(expected, result);
312     }
313
314     /**
315      * Test edge case with divideBy10 method.
316      */
317     @Test
318     public void testDivideBy103() {
319         NaturalNumber result = this.constructorTest(1);
320         result.divideBy10();
321
322         NaturalNumber expected = this.constructorRef(1);
323         expected.divideBy10();
324         assertEquals(expected, result);
325     }
326
327     /**
328      * Test routine case with divideBy10 method.
329      */
330     @Test
331     public void testDivideBy104() {
332         NaturalNumber result = this.constructorTest(101);
333         result.divideBy10();
334
335         NaturalNumber expected = this.constructorRef(101);
336         expected.divideBy10();
337         assertEquals(expected, result);
338     }
339
340     /**
341      * Test strange case with isZero method.
```

```
342      */
343     @Test
344     public void testisZero1() {
345         NaturalNumber result = this.constructorTest(0);
346         boolean resultTest = result.isZero();
347 
348         NaturalNumber expected = this.constructorRef(0);
349         boolean expectedTest = expected.isZero();
350         assertEquals(expected, result);
351         assertEquals(expectedTest, resultTest);
352     }
353 
354     /**
355      * Test edge case with isZero method.
356      */
357     @Test
358     public void testisZero3() {
359         NaturalNumber result = this.constructorTest();
360         boolean resultTest = result.isZero();
361 
362         NaturalNumber expected = this.constructorRef();
363         boolean expectedTest = expected.isZero();
364 
365         assertEquals(expected, result);
366         assertEquals(expectedTest, resultTest);
367     }
368 
369     /**
370      * Test routine case with isZero method.
371      */
372     @Test
373     public void testisZero4() {
374         NaturalNumber result = this.constructorTest(20);
375         boolean resultTest = result.isZero();
376 
377         NaturalNumber expected = this.constructorRef(20);
378         boolean expectedTest = expected.isZero();
379         assertEquals(expected, result);
380         assertEquals(expectedTest, resultTest);
381     }
382 
383 }
384 
```