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

/**
 * JUnit test fixture for {@code NaturalNumber}'s constructors and kernel
 * methods.
 *
 * @author Zhuoyang Li + Xinci Ma
 *
 */
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))
     * @ensures s = TO_STRING(constructorTest)
     */
    protected abstract NaturalNumber constructorTest(String
```

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

```
 89      * @ensures s = TO_STRING(constructorRef)
 90      */
 91     protected abstract NaturalNumber constructorRef(String
    s);
 92
 93     /**
 94      * Invokes the appropriate {@code NaturalNumber}
    constructor for the
 95      * reference implementation and returns the result.
 96      *
 97      * @param n
 98      *            {@code NaturalNumber} to initialize from
 99      * @return the new number
100      * @ensures constructorRef = n
101      */
102     protected abstract NaturalNumber
    constructorRef(NaturalNumber n);
103
104     // TODO - add test cases for four constructors,
    multiplyBy10, divideBy10, isZero
105
106     @Test
107     /**
108      * Test constructor().
109      */
110     public final void testConstructor() {
111         /*
112          * Set up variables
113          */
114         NaturalNumber n = this.constructorTest();
115         NaturalNumber nExpected = this.constructorRef();
116         /*
117          * Assert that values of n and nExpected are equal
118          */
119         assertEquals(nExpected, n);
120     }
121
122     @Test
123     public final void testConstructorInt5() {
124         /*
125          * Set up variables
126          */
127         NaturalNumber n = this.constructorTest(5);
128         NaturalNumber nExpected = this.constructorRef(5);
129         /*
130          * Assert that values of n and nExpected are equal
131          */
```

```
132         assertEquals(nExpected, n);
133     }
134
135     @Test
136     public final void testConstructorString5() {
137         /*
138          * Set up variables
139          */
140         NaturalNumber n = this.constructorTest("0");
141         NaturalNumber nExpected = this.constructorRef(0);
142         /*
143          * Assert that values of n and nExpected are equal
144          */
145         assertEquals(nExpected, n);
146     }
147
148     @Test
149     public final void testConstructorNaturalNumber5() {
150         /*
151          * Set up variables
152          */
153         NaturalNumber n =
    this.constructorTest(this.constructorRef(5));
154         NaturalNumber nExpected = this.constructorRef(5);
155         /*
156          * Assert that values of n and nExpected are equal
157          */
158         assertEquals(nExpected, n);
159     }
160
161     @Test
162     public final void test5MultiplyBy10() {
163         /*
164          * Set up variables
165          */
166         NaturalNumber n = this.constructorTest(5);
167         NaturalNumber nExpected = this.constructorRef(55);
168         /*
169          * Call method under test
170          */
171         n.multiplyBy10(5);
172         /*
173          * Assert that values of n and nExpected are equal
174          */
175         assertEquals(nExpected, n);
176     }
177
```

```
178     @Test
179     public final void test0MultiplyBy10() {
180         /*
181          * Set up variables
182          */
183         NaturalNumber n = this.constructorTest(0);
184         NaturalNumber nExpected = this.constructorRef(5);
185         /*
186          * Call method under test
187          */
188         n.multiplyBy10(5);
189         /*
190          * Assert that values of n and nExpected are equal
191          */
192         assertEquals(nExpected, n);
193     }
194
195     @Test
196     public final void test5DivideBy10() {
197         /*
198          * Set up variables
199          */
200         NaturalNumber n = this.constructorTest(5);
201         NaturalNumber nExpected = this.constructorRef(0);
202         /*
203          * Call method under test
204          */
205         int a = n.divideBy10();
206         /*
207          * Assert that values of n and nExpected are equal
208          */
209         assertEquals(nExpected, n);
210         assertEquals(5, a);
211     }
212
213     @Test
214     public final void test0DivideBy10() {
215         /*
216          * Set up variables
217          */
218         NaturalNumber n = this.constructorTest(0);
219         NaturalNumber nExpected = this.constructorRef(0);
220         /*
221          * Call method under test
222          */
223         int a = n.divideBy10();
224         /*
```

```
225          * Assert that values of n and nExpected are equal
226          */
227         assertEquals(nExpected, n);
228         assertEquals(0, a);
229     }
230
231     @Test
232     public final void test14DivideBy10() {
233         /*
234          * Set up variables
235          */
236         NaturalNumber n = this.constructorTest(14);
237         NaturalNumber nExpected = this.constructorRef(1);
238         /*
239          * Call method under test
240          */
241         int a = n.divideBy10();
242         /*
243          * Assert that values of n and nExpected are equal
244          */
245         assertEquals(nExpected, n);
246         assertEquals(4, a);
247     }
248
249     @Test
250     public final void test0IsZero() {
251         /*
252          * Set up variables
253          */
254         NaturalNumber n = this.constructorTest(0);
255         boolean b = n.isZero();
256         /*
257          * Assert that values of n and nExpected are equal
258          */
259         assertEquals(true, b);
260     }
261
262     @Test
263     public final void test5IsZero() {
264         /*
265          * Set up variables
266          */
267         NaturalNumber n = this.constructorTest(5);
268         boolean b = n.isZero();
269         /*
270          * Assert that values of n and nExpected are equal
271          */
```

```
272         assertEquals(false, b);
273     }
274
275     @Test
276     public final void test10IsZero() {
277         /*
278          * Set up variables
279          */
280         NaturalNumber n = this.constructorTest(10);
281         boolean b = n.isZero();
282         /*
283          * Assert that values of n and nExpected are equal
284          */
285         assertEquals(false, b);
286     }
287
288 }
289
```