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

/**
 * JUnit test fixture for {@code NaturalNumber}'s constructors and kernel
 * methods.
 *
 * @author Put your name here
 *
 */
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);

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

    /**
     * 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    // TODO - add test cases for four constructors, multiplyBy10, divideBy10, isZero
106
107    /**
108     * Test non-element.
109     */
110    @Test
111    public final void testNonConstructor() {
112        NaturalNumber n = this.constructorTest();
113        NaturalNumber nExpected = this.constructorRef();
114        assertEquals(n, nExpected);
115    }
116
117    /**
118     * Test one digit.
119     */
```

```
120     @Test
121     public final void testConstructorOne() {
122         NaturalNumber n = this.constructorTest(9);
123         NaturalNumber nExpected = this.constructorRef(9);
124         assertEquals(n, nExpected);
125     }
126 
127     /**
128      * Test many digits.
129      */
130     @Test
131     public final void testStringConstructorForMany() {
132         NaturalNumber n = this.constructorTest("12312312");
133         NaturalNumber nExpected = this.constructorRef("12312312");
134         assertEquals(n, nExpected);
135     }
136 
137     /**
138      * Test cases for multiplyBy10
139      */
140 
141     @Test
142     public final void testMultiplyBy10Ints() {
143         /*
144          * Initialize the variables
145          */
146         int first = 2;
147         int ans = 20;
148         NaturalNumber n = this.constructorTest(first);
149         NaturalNumber nExpected = this.constructorRef(ans);
150         n.multiplyBy10(0);
151         assertEquals(nExpected, n);
152     }
153 
154     @Test
155     public final void testMultiplyBy10UsingString() {
156         /*
157          * Initialize the variables
158          */
159         String original = "4564";
160         String changed = "45640";
161         NaturalNumber n = this.constructorTest(original);
162         NaturalNumber nExpected = this.constructorRef(changed);
163         n.multiplyBy10(0);
164         assertEquals(nExpected, n);
165     }
166 
167     @Test
168     public final void testMultiplyBy10MaxInts() {
169         /*
170          * Initialize variables
171          */
172         int first = Integer.MAX_VALUE;
173         NaturalNumber n = this.constructorTest(first);
174         NaturalNumber nExpected = this.constructorRef(first);
175         nExpected.multiplyBy10(0);
176         n.multiplyBy10(0);
```

```
177         assertEquals(nExpected, n);
178     }
179 
180     @Test
181     public final void testMultiplyBy10UsingZero() {
182         /*
183          * Initialize variables
184          */
185         int original = 0;
186         int changed = 0;
187         NaturalNumber n = this.constructorTest(original);
188         NaturalNumber nExpected = this.constructorRef(changed);
189         nExpected.multiplyBy10(0);
190         assertEquals(nExpected, n);
191     }
192 
193     /**
194      * Test for divideBy10.
195      */
196     @Test
197     public final void testDivideBy10on50() {
198         NaturalNumber nn = this.constructorTest(50);
199         NaturalNumber nnExpected = this.constructorRef(5);
200         int r = nn.divideBy10();
201         assertEquals(nn, nnExpected);
202         assertTrue(r == 0);
203     }
204 
205     /**
206      * Test for divideBy10.
207      */
208     @Test
209     public final void testDivideBy10on12300() {
210         NaturalNumber nn = this.constructorTest(12300);
211         NaturalNumber nnExpected = this.constructorRef(1230);
212         int r = nn.divideBy10();
213         assertEquals(nn, nnExpected);
214         assertTrue(r == 0);
215     }
216 
217     /**
218      * Test for divideBy10.
219      */
220     @Test
221     public final void testDivideBy10on63() {
222         NaturalNumber nn = this.constructorTest(63);
223         NaturalNumber nnExpected = this.constructorRef(6);
224         int r = nn.divideBy10();
225         assertEquals(nn, nnExpected);
226         assertTrue(r == 3);
227     }
228 
229     /**
230      * Test for divideBy10.
231      */
232     @Test
233     public final void testDivideBy10on9() {
```

```
234         NaturalNumber nn = this.constructorTest(9);
235         NaturalNumber nnExpected = this.constructorRef(0);
236         int r = nn.divideBy10();
237         assertEquals(nn, nnExpected);
238         assertTrue(r == 9);
239     }
240
241     /**
242      * Test for divideBy10.
243      */
244     @Test
245     public final void testDivideBy10on19InString() {
246         NaturalNumber nn = this.constructorTest("19");
247         NaturalNumber nnExpected = this.constructorRef(1);
248         int r = nn.divideBy10();
249         assertEquals(nn, nnExpected);
250         assertTrue(r == 9);
251     }
252
253     /**
254      * Test for divideBy10.
255      */
256     @Test
257     public final void testDivideBy10on19InNN() {
258         NaturalNumber NN = this.constructorTest(19);
259         NaturalNumber nn = this.constructorTest(NN);
260         NaturalNumber nnExpected = this.constructorRef(1);
261         int r = nn.divideBy10();
262         assertEquals(nn, nnExpected);
263         assertTrue(r == 9);
264     }
265
266     /**
267      * Test for divideBy10.
268      */
269     @Test
270     public final void testDivideBy10on0InNN() {
271         NaturalNumber NN = this.constructorTest(0);
272         NaturalNumber nn = this.constructorTest(NN);
273         NaturalNumber nnExpected = this.constructorRef(0);
274         int r = nn.divideBy10();
275         assertEquals(nn, nnExpected);
276         assertTrue(r == 0);
277     }
278
279     /**
280      * Test for isZero.
281      */
282     @Test
283     public final void testIsZeroTrueNoElement() {
284         NaturalNumber nn = this.constructorTest();
285         boolean b = nn.isZero();
286         assertEquals(b, true);
287     }
288
289     /**
290      * Test for isZero.
```

```
291      */
292     @Test
293     public final void testIsZeroTrueWithInt() {
294         NaturalNumber nn = this.constructorTest(0);
295         boolean b = nn.isZero();
296         assertEquals(b, true);
297     }
298
299     /**
300      * Test for isZero.
301      */
302     @Test
303     public final void testIsZeroTrueWithString() {
304         NaturalNumber nn = this.constructorTest("0");
305         boolean b = nn.isZero();
306         assertEquals(b, true);
307     }
308
309     /**
310      * Test for isZero.
311      */
312     @Test
313     public final void testIsZeroTrueWithNN() {
314         NaturalNumber nnExpect = this.constructorTest(0);
315         NaturalNumber nn = this.constructorTest(nnExpect);
316         boolean b = nn.isZero();
317         assertEquals(b, true);
318     }
319
320     /**
321      * Test for isZero.
322      */
323     @Test
324     public final void testIsZeroFalseWithInt() {
325         NaturalNumber nn = this.constructorTest(1);
326         boolean b = nn.isZero();
327         assertEquals(b, false);
328     }
329
330     /**
331      * Test for isZero.
332      */
333     @Test
334     public final void testIsZeroFalseWithString() {
335         NaturalNumber nn = this.constructorTest("1");
336         boolean b = nn.isZero();
337         assertEquals(b, false);
338     }
339
340     /**
341      * Test for isZero.
342      */
343     @Test
344     public final void testIsZeroFalseWithNN() {
345         NaturalNumber NN = this.constructorTest(1);
346         NaturalNumber nn = this.constructorTest(NN);
347         boolean b = nn.isZero();
```

```
348         assertEquals(b, false);
349     }
350
351 }
352
```