

```
1 import static org.junit.Assert.assertEquals;
2
3 /**
4  * JUnit test fixture for {@code NaturalNumber}'s constructors and kernel
5  * methods.
6  *
7  * @author Put your name here
8  */
9 public abstract class NaturalNumberTest {
10
11     /**
12      * Invokes the appropriate {@code NaturalNumber} constructor for the
13      * implementation under test and returns the result.
14      *
15      * @return the new number
16      * @ensures constructorTest = 0
17      */
18     protected abstract NaturalNumber constructorTest();
19
20     /**
21      * Invokes the appropriate {@code NaturalNumber} constructor for the
22      * implementation under test and returns the result.
23      *
24      * @param i
25      *        {@code int} to initialize from
26      * @return the new number
27      * @requires i >= 0
28      * @ensures constructorTest = i
29      */
30     protected abstract NaturalNumber constructorTest(int i);
31
32     /**
33      * Invokes the appropriate {@code NaturalNumber} constructor for the
34      * implementation under test and returns the result.
35      *
36      * @param s
37      *        {@code String} to initialize from
38      * @return the new number
39      * @requires there exists n: NATURAL (s = TO_STRING(n))
40      * @ensures s = TO_STRING(constructorTest)
41      */
42     protected abstract NaturalNumber constructorTest(String s);
43
44     /**
45      * Invokes the appropriate {@code NaturalNumber} constructor for the
46      * implementation under test and returns the result.
47      *
48      * @param n
49      *        {@code NaturalNumber} to initialize from
50      * @return the new number
51      * @ensures constructorTest = n
52      */
53     protected abstract NaturalNumber constructorTest(NaturalNumber n);
54
55     /**
56      * Invokes the appropriate {@code NaturalNumber} constructor for the
57      * reference implementation and returns the result.
58      */
59 }
```

```

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    @Test
108    public final void testForDefaultConstructor() {
109        NaturalNumber n = this.constructorTest();
110        NaturalNumber nExp = this.constructorRef();
111
112        assertEquals(n, nExp);
113    }
114
115    @Test
116    public final void testForIntConstructorZero() {
117        int i = 0;
118
119        NaturalNumber n = this.constructorTest(i);
120        NaturalNumber nExp = this.constructorRef(i);
121
122        assertEquals(n, nExp);
123    }

```

```
124
125 @Test
126 public final void testForIntConstructor() {
127     int i = 5;
128
129     NaturalNumber n = this.constructorTest(i);
130     NaturalNumber nExp = this.constructorRef(i);
131
132     assertEquals(n, nExp);
133 }
134
135 @Test
136 public final void testForStringConstructor() {
137     String s = "5";
138
139     NaturalNumber n = this.constructorTest(s);
140     NaturalNumber nExp = this.constructorRef(s);
141
142     assertEquals(n, nExp);
143 }
144
145 @Test
146 public final void testForNaturalNumberConstructor() {
147     NaturalNumber natN = new NaturalNumber1L(3);
148
149     NaturalNumber n = this.constructorTest(natN);
150     NaturalNumber nExp = this.constructorRef(natN);
151
152     assertEquals(n, nExp);
153 }
154
155 @Test
156 public final void testForMultiplyBy10Zero() {
157
158     NaturalNumber n = this.constructorTest(0);
159     NaturalNumber nExp = this.constructorRef(0);
160
161     n.multiplyBy10(0);
162
163     assertEquals(n, nExp);
164 }
165
166 @Test
167 public final void testForMultiplyBy10LeadingZero() {
168
169     NaturalNumber n = this.constructorTest(0);
170     NaturalNumber nExp = this.constructorRef(7);
171
172     n.multiplyBy10(7);
173
174     assertEquals(n, nExp);
175 }
176
177 @Test
178 public final void testForMultiplyBy10Int() {
179
180     NaturalNumber n = this.constructorTest(123);
181     NaturalNumber nExp = this.constructorRef(1234);
182
```

```
183         n.multiplyBy10(4);
184
185         assertEquals(n, nExp);
186     }
187
188     @Test
189     public final void testFormultiplyBy10MaxInt() {
190
191         NaturalNumber n = this.constructorTest(Integer.MAX_VALUE);
192         NaturalNumber nExp = this.constructorTest("21474836470");
193
194         n.multiplyBy10(0);
195
196         assertEquals(n, nExp);
197     }
198
199     @Test
200     public final void testFormultiplyBy10String() {
201
202         NaturalNumber n = this.constructorTest("678");
203         NaturalNumber nExp = this.constructorRef("6789");
204
205         n.multiplyBy10(9);
206
207         assertEquals(n, nExp);
208     }
209
210     @Test
211     public final void testFormultiplyBy10NN() {
212
213         NaturalNumber nInitial = this.constructorRef(4567);
214         NaturalNumber nFinal = this.constructorRef(45678);
215
216         NaturalNumber n = this.constructorTest(nInitial);
217         NaturalNumber nExp = this.constructorRef(nFinal);
218         n.multiplyBy10(8);
219
220         assertEquals(n, nExp);
221     }
222
223     @Test
224     public final void testDivideBy10Zero() {
225         NaturalNumber n = this.constructorTest(0);
226         NaturalNumber nExp = this.constructorRef(0);
227
228         int dig = n.divideBy10();
229
230         assertEquals(n, nExp);
231         assertEquals(0, dig);
232     }
233
234     @Test
235     public final void testDivideBy10OneDigit() {
236         NaturalNumber n = this.constructorTest(7);
237         NaturalNumber nExp = this.constructorRef(0);
238
239         int dig = n.divideBy10();
240
241         assertEquals(n, nExp);
```

```
242     assertEquals(7, dig);
243 }
244
245 @Test
246 public final void testDivideBy10Int() {
247     NaturalNumber n = this.constructorTest(12345);
248     NaturalNumber nExp = this.constructorRef(1234);
249
250     int dig = n.divideBy10();
251
252     assertEquals(n, nExp);
253     assertEquals(5, dig);
254 }
255
256 @Test
257 public final void testDivideBy10String() {
258     NaturalNumber n = this.constructorTest("98765");
259     NaturalNumber nExp = this.constructorRef("9876");
260
261     int dig = n.divideBy10();
262
263     assertEquals(n, nExp);
264     assertEquals(5, dig);
265 }
266
267 @Test
268 public final void testDivideBy10NN() {
269     NaturalNumber nInitial = this.constructorRef(45678);
270     NaturalNumber nFinal = this.constructorRef(4567);
271
272     NaturalNumber n = this.constructorTest(nInitial);
273     NaturalNumber nExp = this.constructorRef(nFinal);
274
275     int dig = n.divideBy10();
276
277     assertEquals(n, nExp);
278     assertEquals(8, dig);
279 }
280
281 @Test
282 public final void testForIsZeroWhenZero() {
283     NaturalNumber n = this.constructorTest();
284
285     assertEquals(true, n.isZero());
286 }
287
288 @Test
289 public final void testForIsZeroWhenNonZero() {
290     NaturalNumber n = this.constructorTest(13);
291
292     assertEquals(false, n.isZero());
293 }
294
295 }
296
```