The Ohio State University

PROJECT 2: NATURALNUMBER ON STRING

Daniil Gofman

Ansh Pachauri

SW 2: Dev & Dsgn

Paolo Bucci

Yiyang Chen

Shivam Gupta

September 12, 2023

```
 1 import static org.junit.Assert.assertEquals;
 2
 3 import org.junit.Test;
 4
 5 import components.naturalnumber.NaturalNumber;
 6
 7 /**
 8  * JUnit test fixture for {@code NaturalNumber}'s constructors and kernel
 9  * methods.
10  *
11  * @author Ansh Pachauri
12  *
13  */
14 public abstract class NaturalNumberTest {
15
16     /**
17      * Invokes the appropriate {@code NaturalNumber} constructor for the
18      * implementation under test and returns the result.
19      *
20      * @return the new number
21      * @ensures constructorTest = 0
22      */
23     protected abstract NaturalNumber constructorTest();
24
25     /**
26      * Invokes the appropriate {@code NaturalNumber} constructor for the
27      * implementation under test and returns the result.
28      *
29      * @param i
30      *            {@code int} to initialize from
31      * @return the new number
32      * @requires i >= 0
33      * @ensures constructorTest = i
34      */
35     protected abstract NaturalNumber constructorTest(int i);
36
37     /**
38      * Invokes the appropriate {@code NaturalNumber} constructor for the
39      * implementation under test and returns the result.
40      *
41      * @param s
42      *            {@code String} to initialize from
43      * @return the new number
44      * @requires there exists n: NATURAL (s = TO_STRING(n))
45      * @ensures s = TO_STRING(constructorTest)
46      */
47     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     /**
105      * Test for Empty constructor.
106      */
107     @Test
108     public final void testConstructorEmpty() {
109         NaturalNumber s = this.constructorTest();
110         NaturalNumber sExpected = this.constructorRef();
111
112         assertEquals(s, sExpected);
113     }
114
115     /**
116      * Test for constructor with a String zero.
117      */
118     @Test
119     public final void testConstructorStringZero() {
120         NaturalNumber s = this.constructorTest("0");
121         NaturalNumber sExpected = this.constructorRef("0");
122
123         assertEquals(s, sExpected);
124     }
```

```
125
126     /**
127      * Test for constructor with String.
128      */
129     @Test
130     public final void testConstructorString1() {
131         NaturalNumber s = this.constructorTest("123");
132         NaturalNumber sExpected = this.constructorRef("123");
133
134         assertEquals(s, sExpected);
135     }
136
137     /**
138      * Test for constructor with String.
139      */
140     @Test
141     public final void testConstructorString2() {
142         NaturalNumber s = this.constructorTest("123456789");
143         NaturalNumber sExpected = this.constructorRef("123456789");
144
145         assertEquals(s, sExpected);
146     }
147
148     /**
149      * Test for constructor with integer zero.
150      */
151     @Test
152     public final void testConstructorInteger1() {
153         NaturalNumber s = this.constructorTest(0);
154         NaturalNumber sExpected = this.constructorRef(0);
155
156         assertEquals(s, sExpected);
157     }
158
159     /**
160      * Test for constructor with integer.
161      */
162     @Test
163     public final void testConstructorInteger2() {
164         final int testNum = 12345;
165         NaturalNumber s = this.constructorTest(testNum);
166         NaturalNumber sExpected = this.constructorRef(testNum);
167
168         assertEquals(s, sExpected);
169     }
170
171     /**
172      * Test for constructor with MAX value of integer.
173      */
174     @Test
175     public final void testConstructorInteger3() {
176         NaturalNumber s = this.constructorTest(Integer.MAX_VALUE);
177         NaturalNumber sExpected = this.constructorRef(Integer.MAX_VALUE);
178
179         assertEquals(s, sExpected);
180     }
181
182     /**
183      * Test for constructor with Natural Number zero.
184      */
185     @Test
186     public final void testConstructorNN1() {
```

```
187         NaturalNumber test = this.constructorTest(0);
188         NaturalNumber s = this.constructorTest(test);
189         NaturalNumber sExpected = this.constructorRef(test);
190
191         assertEquals(s, sExpected);
192     }
193
194     /**
195      * Test for constructor with Natural Number.
196      */
197     @Test
198     public final void testConstructorNN2() {
199         final int val = 12345;
200         NaturalNumber test = this.constructorTest(val);
201         NaturalNumber s = this.constructorTest(test);
202         NaturalNumber sExpected = this.constructorRef(test);
203
204         assertEquals(s, sExpected);
205     }
206
207     /**
208      * Test for constructor with MAX of Natural Number.
209      */
210     @Test
211     public final void testConstructorNN3() {
212         NaturalNumber test = this.constructorTest(Integer.MAX_VALUE);
213         NaturalNumber s = this.constructorTest(test);
214         NaturalNumber sExpected = this.constructorRef(test);
215
216         assertEquals(s, sExpected);
217     }
218
219     /**
220      * Test for MultiplyBy10 with Zero.
221      */
222     @Test
223     public final void testMultiplyBy10Zero() {
224         final int val = 7;
225         NaturalNumber s = this.constructorTest(0);
226         NaturalNumber sExpected = this.constructorRef(val);
227         s.multiplyBy10(val);
228
229         assertEquals(s, sExpected);
230     }
231
232     /**
233      * Test for Non-Empty MultiplyBy10.
234      */
235     @Test
236     public final void testMultiplyBy10NonEmpty1() {
237         final int val = 123;
238         final int valEx = 1234;
239         final int num = 4;
240         NaturalNumber s = this.constructorTest(val);
241         NaturalNumber sExpected = this.constructorRef(valEx);
242         s.multiplyBy10(num);
243
244         assertEquals(s, sExpected);
245     }
246
247     /**
248      * Test for Non-Empty MultiplyBy10.
```

```
249      */
250     @Test
251     public final void testMultiplyBy10NonEmpty2() {
252         final int val = 12345678;
253         final int valEx = 123456789;
254         final int num = 9;
255         NaturalNumber s = this.constructorTest(val);
256         NaturalNumber sExpected = this.constructorRef(valEx);
257
258         s.multiplyBy10(num);
259
260         assertEquals(s, sExpected);
261     }
262
263     /**
264      * Test for DivideBy10 with Zero.
265      */
266     @Test
267     public final void testDivideBy10Zero() {
268         final int val = 7;
269         NaturalNumber s = this.constructorTest(val);
270         NaturalNumber sExpected = this.constructorRef(0);
271
272         s.divideBy10();
273
274         assertEquals(s, sExpected);
275     }
276
277     /**
278      * Test for Non-Empty DivideBy10.
279      */
280     @Test
281     public final void testDivideBy10NonEmpty1() {
282         final int val = 1234;
283         final int valEx = 123;
284         NaturalNumber s = this.constructorTest(val);
285         NaturalNumber sExpected = this.constructorRef(valEx);
286         s.divideBy10();
287
288         assertEquals(s, sExpected);
289     }
290
291     /**
292      * Test for Non-Empty DivideBy10.
293      */
294     @Test
295     public final void testDivideBy10NonEmpty2() {
296         final int val = 123456789;
297         final int valEx = 12345678;
298         NaturalNumber s = this.constructorTest(val);
299         NaturalNumber sExpected = this.constructorRef(valEx);
300         s.divideBy10();
301
302         assertEquals(s, sExpected);
303     }
304
305     /**
306      * Test for isZero with zero.
307      */
308     @Test
309     public final void testIsZeroZero() {
310         NaturalNumber s = this.constructorTest(0);
```

```
311         NaturalNumber sExpected = this.constructorRef(0);
312
313         boolean test = s.isZero();
314
315         assertEquals(s, sExpected);
316         assertEquals(test, true);
317     }
318
319     /**
320      * Test for Non-Empty isZero.
321      */
322     @Test
323     public final void testNonEmptyIsZero() {
324         final int val = 123;
325         NaturalNumber s = this.constructorTest(val);
326         NaturalNumber sExpected = this.constructorRef(val);
327
328         boolean test = s.isZero();
329
330         assertEquals(s, sExpected);
331         assertEquals(test, false);
332     }
333
334 }
335
```