

```
1 import static org.junit.Assert.assertEquals;
2
3 import org.junit.Test;
4
5 import components.naturalnumber.NaturalNumber;
6 import components.naturalnumber.NaturalNumber2;
7
8 /**
9  * @author David Park
10  *
11  */
12 public class CryptoUtilitiesTest {
13
14     /*
15      * Tests of reduceToGCD
16      */
17
18     @Test
19     public void testReduceToGCD_0_0() {
20         NaturalNumber n = new NaturalNumber2(0);
21         NaturalNumber nExpected = new NaturalNumber2(0);
22         NaturalNumber m = new NaturalNumber2(0);
23         NaturalNumber mExpected = new NaturalNumber2(0);
24         CryptoUtilities.reduceToGCD(n, m);
25         assertEquals(nExpected, n);
26         assertEquals(mExpected, m);
27     }
28
29     @Test
30     public void testReduceToGCD_30_21() {
31         NaturalNumber n = new NaturalNumber2(30);
32         NaturalNumber nExpected = new NaturalNumber2(3);
33         NaturalNumber m = new NaturalNumber2(21);
34         NaturalNumber mExpected = new NaturalNumber2(0);
35         CryptoUtilities.reduceToGCD(n, m);
36         assertEquals(nExpected, n);
37         assertEquals(mExpected, m);
38     }
39
40     /*
41      * Tests of isEven
42      */
43
44     @Test
45     public void testIsEven_0() {
46         NaturalNumber n = new NaturalNumber2(0);
47         NaturalNumber nExpected = new NaturalNumber2(0);
48         boolean result = CryptoUtilities.isEven(n);
49         assertEquals(nExpected, n);
50         assertEquals(true, result);
51     }
52
53     @Test
54     public void testIsEven_1() {
55         NaturalNumber n = new NaturalNumber2(1);
56         NaturalNumber nExpected = new NaturalNumber2(1);
57         boolean result = CryptoUtilities.isEven(n);
```

```
58         assertEquals(nExpected, n);
59         assertEquals(false, result);
60     }
61
62     /*
63     * Tests of powerMod
64     */
65
66     @Test
67     public void testPowerMod_0_0_2() {
68         NaturalNumber n = new NaturalNumber2(0);
69         NaturalNumber nExpected = new NaturalNumber2(1);
70         NaturalNumber p = new NaturalNumber2(0);
71         NaturalNumber pExpected = new NaturalNumber2(0);
72         NaturalNumber m = new NaturalNumber2(2);
73         NaturalNumber mExpected = new NaturalNumber2(2);
74         CryptoUtilities.powerMod(n, p, m);
75         assertEquals(nExpected, n);
76         assertEquals(pExpected, p);
77         assertEquals(mExpected, m);
78     }
79
80     @Test
81     public void testPowerMod_17_18_19() {
82         NaturalNumber n = new NaturalNumber2(17);
83         NaturalNumber nExpected = new NaturalNumber2(1);
84         NaturalNumber p = new NaturalNumber2(18);
85         NaturalNumber pExpected = new NaturalNumber2(18);
86         NaturalNumber m = new NaturalNumber2(19);
87         NaturalNumber mExpected = new NaturalNumber2(19);
88         CryptoUtilities.powerMod(n, p, m);
89         assertEquals(nExpected, n);
90         assertEquals(pExpected, p);
91         assertEquals(mExpected, m);
92     }
93
94     @Test
95     public void isPrime2_50() {
96         NaturalNumber n = new NaturalNumber2(50);
97         NaturalNumber nExpected = new NaturalNumber2(50);
98         boolean result = CryptoUtilities.isPrime2(n);
99         assertEquals(nExpected, n);
100        assertEquals(false, result);
101    }
102
103    @Test
104    public void isWitness2_30() {
105        NaturalNumber two = new NaturalNumber2(2);
106        NaturalNumber twoExpected = new NaturalNumber2(2);
107        NaturalNumber thirty = new NaturalNumber2(30);
108        NaturalNumber thirtyExpected = new NaturalNumber2(30);
109        boolean result = CryptoUtilities.isWitnessToCompositeness(two, thirty);
110        assertEquals(twoExpected, two);
111        assertEquals(thirtyExpected, thirty);
112        assertEquals(true, result);
113    }
114
```

```
115     @Test
116     public void testReduceToGCD_1_1() {
117         NaturalNumber n = new NaturalNumber2(1);
118         NaturalNumber nExpected = new NaturalNumber2(1);
119         NaturalNumber m = new NaturalNumber2(1);
120         NaturalNumber mExpected = new NaturalNumber2(0);
121         CryptoUtilities.reduceToGCD(n, m);
122         assertEquals(nExpected, n);
123         assertEquals(mExpected, m);
124     }
125
126     @Test
127     public void testReduceToGCD_100_10() {
128         NaturalNumber n = new NaturalNumber2(100);
129         NaturalNumber nExpected = new NaturalNumber2(10);
130         NaturalNumber m = new NaturalNumber2(10);
131         NaturalNumber mExpected = new NaturalNumber2(0);
132         CryptoUtilities.reduceToGCD(n, m);
133         assertEquals(nExpected, n);
134         assertEquals(mExpected, m);
135     }
136
137     @Test
138     public void testReduceToGCD_25_15() {
139         NaturalNumber n = new NaturalNumber2(25);
140         NaturalNumber nExpected = new NaturalNumber2(5);
141         NaturalNumber m = new NaturalNumber2(15);
142         NaturalNumber mExpected = new NaturalNumber2(0);
143         CryptoUtilities.reduceToGCD(n, m);
144         assertEquals(nExpected, n);
145         assertEquals(mExpected, m);
146     }
147
148     @Test
149     public void testIsEven_2() {
150         NaturalNumber n = new NaturalNumber2(2);
151         boolean result = CryptoUtilities.isEven(n);
152         assertEquals(true, result);
153     }
154
155     @Test
156     public void testIsEven_999() {
157         NaturalNumber n = new NaturalNumber2(999);
158         boolean result = CryptoUtilities.isEven(n);
159         assertEquals(false, result);
160     }
161
162     @Test
163     public void testIsEven_22() {
164         NaturalNumber n = new NaturalNumber2(22);
165         boolean result = CryptoUtilities.isEven(n);
166         assertEquals(true, result);
167     }
168
169     @Test
170     public void testIsEven_45() {
171         NaturalNumber n = new NaturalNumber2(45);
```

```
172     boolean result = CryptoUtilities.isEven(n);
173     assertEquals(false, result);
174 }
175
176 @Test
177 public void testPowerMod_2_5_13() {
178     NaturalNumber n = new NaturalNumber2(2);
179     NaturalNumber p = new NaturalNumber2(5);
180     NaturalNumber m = new NaturalNumber2(13);
181     CryptoUtilities.powerMod(n, p, m);
182     NaturalNumber nExpected = new NaturalNumber2(6); //  $2^5 \bmod 13 = 6$ 
183     assertEquals(nExpected, n);
184 }
185
186 @Test
187 public void testPowerMod_7_3_10() {
188     NaturalNumber n = new NaturalNumber2(7);
189     NaturalNumber p = new NaturalNumber2(3);
190     NaturalNumber m = new NaturalNumber2(10);
191     CryptoUtilities.powerMod(n, p, m);
192     NaturalNumber nExpected = new NaturalNumber2(3); //  $7^3 \bmod 10 = 3$ 
193     assertEquals(nExpected, n);
194 }
195
196 @Test
197 public void testPowerMod_3_4_5() {
198     NaturalNumber n = new NaturalNumber2(3);
199     NaturalNumber p = new NaturalNumber2(4);
200     NaturalNumber m = new NaturalNumber2(5);
201     CryptoUtilities.powerMod(n, p, m);
202     NaturalNumber nExpected = new NaturalNumber2(1); //  $3^4 \bmod 5 = 1$ 
203     assertEquals(nExpected, n);
204 }
205
206 @Test
207 public void testPowerMod_5_3_23() {
208     NaturalNumber n = new NaturalNumber2(5);
209     NaturalNumber p = new NaturalNumber2(3);
210     NaturalNumber m = new NaturalNumber2(23);
211     CryptoUtilities.powerMod(n, p, m);
212     NaturalNumber nExpected = new NaturalNumber2(10); //  $5^3 \bmod 23 = 10$ 
213     assertEquals(nExpected, n);
214 }
215
216 @Test
217 public void testPowerMod_6_2_7() {
218     NaturalNumber n = new NaturalNumber2(6);
219     NaturalNumber p = new NaturalNumber2(2);
220     NaturalNumber m = new NaturalNumber2(7);
221     CryptoUtilities.powerMod(n, p, m);
222     NaturalNumber nExpected = new NaturalNumber2(1); //  $6^2 \bmod 7 = 1$ 
223     assertEquals(nExpected, n);
224 }
225
226 @Test
227 public void isWitnessToCompositeness_2_4() {
228     NaturalNumber two = new NaturalNumber2(2);
```

```
229     NaturalNumber four = new NaturalNumber2(4);
230     boolean result = CryptoUtilities.isWitnessToCompositeness(two, four);
231     assertEquals(true, result); // 2 is a witness to 4 being composite
232 }
233
234 @Test
235 public void isWitnessToCompositeness_3_7() {
236     NaturalNumber three = new NaturalNumber2(3);
237     NaturalNumber seven = new NaturalNumber2(7);
238     boolean result = CryptoUtilities.isWitnessToCompositeness(three, seven);
239     assertEquals(false, result); // 3 is not a witness to 7 being composite
240 }
241
242 @Test
243 public void isWitnessToCompositeness_2_9() {
244     NaturalNumber two = new NaturalNumber2(2);
245     NaturalNumber nine = new NaturalNumber2(9);
246     boolean result = CryptoUtilities.isWitnessToCompositeness(two, nine);
247     assertEquals(true, result);
248 }
249
250 @Test
251 public void isWitnessToCompositeness_10_12() {
252     NaturalNumber ten = new NaturalNumber2(10);
253     NaturalNumber twelve = new NaturalNumber2(12);
254     boolean result = CryptoUtilities.isWitnessToCompositeness(ten, twelve);
255     assertEquals(true, result);
256 }
257
258 @Test
259 public void isPrime1_2() {
260     NaturalNumber two = new NaturalNumber2(2);
261     boolean result = CryptoUtilities.isPrime1(two);
262     assertEquals(true, result);
263 }
264
265 @Test
266 public void isPrime1_4() {
267     NaturalNumber four = new NaturalNumber2(4);
268     boolean result = CryptoUtilities.isPrime1(four);
269     assertEquals(false, result);
270 }
271
272 @Test
273 public void isPrime1_5() {
274     NaturalNumber five = new NaturalNumber2(5);
275     boolean result = CryptoUtilities.isPrime1(five);
276     assertEquals(true, result);
277 }
278
279 @Test
280 public void isPrime1_10() {
281     NaturalNumber ten = new NaturalNumber2(10);
282     boolean result = CryptoUtilities.isPrime1(ten);
283     assertEquals(false, result);
284 }
285
```

```
286     @Test
287     public void isPrime1_97() {
288         NaturalNumber ninetySeven = new NaturalNumber2(97);
289         boolean result = CryptoUtilities.isPrime1(ninetySeven);
290         assertEquals(true, result);
291     }
292
293     @Test
294     public void isPrime2_3() {
295         NaturalNumber three = new NaturalNumber2(3);
296         boolean result = CryptoUtilities.isPrime2(three);
297         assertEquals(true, result); // 3 is a prime number
298     }
299
300     @Test
301     public void isPrime2_9() {
302         NaturalNumber nine = new NaturalNumber2(9);
303         boolean result = CryptoUtilities.isPrime2(nine);
304         assertEquals(false, result); // 9 is not a prime number
305     }
306
307     @Test
308     public void isPrime2_11() {
309         NaturalNumber eleven = new NaturalNumber2(11);
310         boolean result = CryptoUtilities.isPrime2(eleven);
311         assertEquals(true, result);
312     }
313
314     @Test
315     public void isPrime2_15() {
316         NaturalNumber fifteen = new NaturalNumber2(15);
317         boolean result = CryptoUtilities.isPrime2(fifteen);
318         assertEquals(false, result);
319     }
320
321     @Test
322     public void isPrime2_101() {
323         NaturalNumber oneHundredOne = new NaturalNumber2(101);
324         boolean result = CryptoUtilities.isPrime2(oneHundredOne);
325         assertEquals(true, result);
326     }
327
328     @Test
329     public void generateNextLikelyPrime_startingFrom2() {
330         NaturalNumber n = new NaturalNumber2(2);
331         CryptoUtilities.generateNextLikelyPrime(n);
332         NaturalNumber expected = new NaturalNumber2(3); // The next prime after 2 is 3
333         assertEquals(expected, n);
334     }
335
336     @Test
337     public void generateNextLikelyPrime_startingFrom14() {
338         NaturalNumber n = new NaturalNumber2(14);
339         CryptoUtilities.generateNextLikelyPrime(n);
340         NaturalNumber expected = new NaturalNumber2(17); // The next prime after 14 is 17
341         assertEquals(expected, n);
342     }
```

```
343
344  @Test
345  public void generateNextLikelyPrime_startingFrom25() {
346      NaturalNumber n = new NaturalNumber2(25);
347      CryptoUtilities.generateNextLikelyPrime(n);
348      NaturalNumber expected = new NaturalNumber2(29); // The next prime after 25 is 29
349      assertEquals(expected, n);
350  }
351
352  @Test
353  public void generateNextLikelyPrime_startingFrom1() {
354      NaturalNumber n = new NaturalNumber2(1);
355      CryptoUtilities.generateNextLikelyPrime(n);
356      NaturalNumber expected = new NaturalNumber2(2); // The next prime after 1 is 2
357      assertEquals(expected, n);
358  }
359
360 }
361
```