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

/**
 * @author Yiming Cheng
 *
 */
public class CryptoUtilitiesTest {

    /*
     * Tests of reduceToGCD
     */

    @Test
    public void testReduceToGCD_0_0() {
        NaturalNumber n = new NaturalNumber2(0);
        NaturalNumber nExpected = new NaturalNumber2(0);
        NaturalNumber m = new NaturalNumber2(0);
        NaturalNumber mExpected = new NaturalNumber2(0);
        CryptoUtilities.reduceToGCD(n, m);
        assertEquals(nExpected, n);
        assertEquals(mExpected, m);
    }

    @Test
    public void testReduceToGCD_30_21() {
        NaturalNumber n = new NaturalNumber2(30);
        NaturalNumber nExpected = new NaturalNumber2(3);
        NaturalNumber m = new NaturalNumber2(21);
        NaturalNumber mExpected = new NaturalNumber2(0);
        CryptoUtilities.reduceToGCD(n, m);
        assertEquals(nExpected, n);
        assertEquals(mExpected, m);
    }

    @Test
    public void testReduceToGCD_11_121() {
        NaturalNumber n = new NaturalNumber2(11);
        NaturalNumber nExpected = new NaturalNumber2(11);
        NaturalNumber m = new NaturalNumber2(121);
        NaturalNumber mExpected = new NaturalNumber2(0);
        CryptoUtilities.reduceToGCD(n, m);
        assertEquals(nExpected, n);
        assertEquals(mExpected, m);
    }

    /*
     * Tests of isEven
     */

    @Test
    public void testIsEven_0() {
        NaturalNumber n = new NaturalNumber2(0);
        NaturalNumber nExpected = new NaturalNumber2(0);
        boolean result = CryptoUtilities.isEven(n);
        assertEquals(nExpected, n);
        assertEquals(true, result);
    }

    @Test
    public void testIsEven_1() {
        NaturalNumber n = new NaturalNumber2(1);
        NaturalNumber nExpected = new NaturalNumber2(1);
        boolean result = CryptoUtilities.isEven(n);
        assertEquals(nExpected, n);
        assertEquals(false, result);
    }

    /*
     * Tests of powerMod
     */

    @Test
    public void testPowerMod_0_0_2() {
        NaturalNumber n = new NaturalNumber2(0);
        NaturalNumber nExpected = new NaturalNumber2(1);
        NaturalNumber p = new NaturalNumber2(0);
        NaturalNumber pExpected = new NaturalNumber2(0);
        NaturalNumber m = new NaturalNumber2(2);
        NaturalNumber mExpected = new NaturalNumber2(2);
        CryptoUtilities.powerMod(n, p, m);
        assertEquals(nExpected, n);
        assertEquals(pExpected, p);
        assertEquals(mExpected, m);
    }

    @Test
    public void testPowerMod_17_18_19() {
        NaturalNumber n = new NaturalNumber2(17);
        NaturalNumber nExpected = new NaturalNumber2(1);
        NaturalNumber p = new NaturalNumber2(18);
        NaturalNumber pExpected = new NaturalNumber2(18);
        NaturalNumber m = new NaturalNumber2(19);
        NaturalNumber mExpected = new NaturalNumber2(19);
        CryptoUtilities.powerMod(n, p, m);
        assertEquals(nExpected, n);
        assertEquals(pExpected, p);
        assertEquals(mExpected, m);
    }

    /*
     * Tests of isWitnessToCompositeness
     */
    @Test
    public void isWitnessToCompositeness_11_29() {
        NaturalNumber w = new NaturalNumber2(11);
        NaturalNumber n = new NaturalNumber2(29);
        boolean result = CryptoUtilities.isWitnessToCompositeness(w, n);
        assertEquals(false, result);
    }

    @Test
    public void isWitnessToCompositeness_7_35() {
        NaturalNumber w = new NaturalNumber2(7);
        NaturalNumber n = new NaturalNumber2(35);
        boolean result = CryptoUtilities.isWitnessToCompositeness(w, n);
        assertEquals(true, result);
    }

    /*
     * Tests of isPrime1
     */
    @Test
    public void testisPrime1_5() {
        NaturalNumber n = new NaturalNumber2(5);
        NaturalNumber nExpected = new NaturalNumber2(5);
        boolean result = CryptoUtilities.isPrime1(n);
        assertEquals(n, nExpected);
        assertEquals(result, true);
    }

    @Test
    public void testisPrime1_8() {
        NaturalNumber n = new NaturalNumber2(8);
        boolean result = CryptoUtilities.isPrime1(n);
        assertEquals(false, result);
    }

    /*
     * Tests of isPrime2
     */
    @Test
    public void testisPrime2_24() {
        NaturalNumber n = new NaturalNumber2(24);
        boolean result = CryptoUtilities.isPrime2(n);
        assertEquals(false, result);
    }

    @Test
    public void testisPrime2_29() {
        NaturalNumber n = new NaturalNumber2(29);
        boolean result = CryptoUtilities.isPrime2(n);
        assertEquals(true, result);
    }

    /*
     * Tests of generateNextLikelyPrime
     */
    @Test
    public void testgenerateNextLikelyPrime2() {
        NaturalNumber n = new NaturalNumber2(2);
        NaturalNumber nExpected = new NaturalNumber2(2);
        CryptoUtilities.generateNextLikelyPrime(n);
        assertEquals(nExpected, n);
    }

    @Test
    public void testgenerateNextLikelyPrime11() {
        NaturalNumber n = new NaturalNumber2(11);
        NaturalNumber nExpected = new NaturalNumber2(11);
        CryptoUtilities.generateNextLikelyPrime(n);
        assertEquals(nExpected, n);
    }

}
```