```
 1 // Java Program to Implement the RSA
   Algorithm
 2
 3 import java.math.*;
 4 import java.util.*;
 5
 6 ///https://www.javatpoint.com/rsa-
   encryption-algorithm
 7
 8 /*
 9 Generating Public Key
10
11 1. Select two prime no's. Suppose P = 53
    and Q = 59.
12 Now First part of the Public key  : n =
   P*Q = 3127.
13
14 2. We also need a small exponent say e :
15    But e Must be
16
17     -An integer.
18
19     -Not be a factor of n.
20
21     -1 < e < Φ(n) [Φ(n) is discussed
   below],
22      Let us now consider it to be equal
   to 3.
23
24 The public key has been made of n and e
25
26 Generating Private Key
27
28 1. We need to calculate Φ(n) :
29    Such that Φ(n) = (P-1)(Q-1)
30       so,  Φ(n) = 3016
31
32 2. Now calculate Private Key, d :
```

```
33     d = (k*Φ(n) + 1) / e for some integer
     k
34
35  3. For k = 2, value of d is 2011.
36
37  The private key has been made of d
38      Consider two prime numbers p and q.
39      Compute n = p*q
40      Compute ϕ(n) = (p − 1) ∗ (q − 1)
41      Choose e such gcd(e , ϕ(n) ) = 1
42      Calculate d such e*d mod ϕ(n) = 1
43      Public Key {e,n} Private Key {d,n}
44      Cipher text C = Pe mod n where P =
    plaintext
45      For Decryption D = Dd mod n where D
    will refund the plaintext.
46   */
47
48  class RSA {
49      public static void main(String args
    []) {
50          int p, q, n, phi, d = 0, e, i;
51
52          // The number to be encrypted
    and decrypted
53          int msg = 2;
54          double c;
55          BigInteger msgback;
56
57          // 1st prime number p
58          p = 2;
59
60          // 2nd prime number q
61          q = 7;
62
63          // Value of N
64          n = p * q;
65          System.out.println("the value of
```

```
65   N = " + n);
66
67           // value of phi
68           phi = (p - 1) * (q - 1);
69           System.out.println("the value
     of phi = " + phi);
70
71           for (e = 2; e < phi; e++) {
72
73               // e is for public key
     exponent
74               if (gcd(e, phi) == 1) {
75                   break;
76               }
77           }
78
79           System.out.println("the value
     of e = " + e);
80           for (i = 0; i <= 9; i++) {
81               int x = 1 + (i * phi);
82
83               // d is for private key
     exponent
84               if (x % e == 0 && d != e
      ) {
85                   d = x / e;
86                   break;
87               }
88           }
89           System.out.println("the value
     of d = " + d);
90           System.out.println("the message
      in clear= " + msg);
91           /*
92               C = me mod n
93               Here, m must be less than n
     .
94            */
```

```
 95
 96          c = (Math.pow(msg, e)) % n;
 97          System.out.println("Encrypted
    message is : " + c);
 98
 99          // converting int value of n to
     BigInteger
100          BigInteger N = BigInteger.
    valueOf(n);
101
102          // converting float value of c
    to BigInteger
103          BigInteger C = BigDecimal.
    valueOf(c).toBigInteger();
104          msgback = (C.pow(d)).mod(N);
105          System.out.println("Decrypted
    message is : "
106                  + msgback);
107      }
108
109      static int gcd(int e, int z) {
110          if (e == 0)
111              return z;
112          else
113              return gcd(z % e, e);
114      }
115 }
116
```