```java
// To implement RSA Algorithm for providing confidentiality

import java.math.BigInteger;
import java.util.*;
class prac5_a
{
    static int checkPrime(int n)
    {
        // checking prime or not
        Scanner scan2 = new Scanner(system.in);
        boolean prime = false;
        while(prime == false)
        {
            if(n%6==1 || n%6==5 || n==2 || n==3)
            {
                prime = true;
            }
            else
            {
                System.out.print("Number is not prime. Enter again:: ");
                n = scan2.nextInt();
            }
        }
        return n;
    }

    static void displayEncryptionKey(int n)
    {
        int temp = 2, element = 0, gcd = 1, count = 0;
        System.out.print("\n(");
        while (temp != n)
        {
            for(int i = 1; i < n; i++)
            {
                if(temp%i==0 && n%i==0)
                {
                    gcd = i;
                    //storing number if gcd is 1
                    if(gcd == 1)
                    {
```

```
                    element = temp;
                }
            }
        }
        if(gcd == 1)
        {
            System.out.print(element+" ");
            count++;

            if(count%10==0)
            {
                System.out.print("\n");
            }
        }
        temp = temp + 1;
    }
    System.out.print("\b)");
}
   public static void main(String args[])
{
    Scanner scan = new Scanner(System.in);
    // input prime numbers and store in p and q
    System.out.print("\nEnter two prime numbers p and q.");
    System.out.print("\np = ");
    int p = scan.nextInt();
    p = checkPrime(p);
    System.out.print("q = ");
    int q = scan.nextInt();
    q = checkPrime(q);

    // computing system modulus
    int n = p*q;
    int pN = (p - 1) * (q - 1);

    // finding encryption key
    System.out.print("\nSelect one of the encryption key.");
    displayEncryptionKey(pN);
    System.out.print("\n\nEncryption key:: ");
    int e = scan.nextInt();
```

```
        // calculate decryption key

            //finding multiplicative inverse
            int d = 1;
            for (int i=1; i<pN; i++)
            {
                if ( (i*e % pN) == 1)
                {
                    d = i;
                }
            }

        d = d % pN;
        System.out.println("Decryption key:: "+d);

        System.out.print("\nEnter integer to be encrypted (less than "+ pN
+" ):: ");
        int m = scan.nextInt();

        // find ciphertext
            BigInteger result = new BigInteger("1");
            // finding m to the power e
            for (int i = 1; i <= e; i++)
            {
                result = result.multiply(BigInteger.valueOf(m));
            }
            BigInteger c = result.mod(BigInteger.valueOf(n));
            System.out.print("Encrypted message:: "+c);

        // find deciphered text
            BigInteger result2 = new BigInteger("1");
            // finding c to the power d
            for (int i = 1; i <= d; i++)
            {
                result2 = result2.multiply(c);
            }
            BigInteger message = result2.mod(BigInteger.valueOf(n));
            System.out.println("\nDecryped message:: "+message+"\n");
    }
}
```