```
import java.util.*;
import java.math.*;

class RSA
{
 public static void main(String args[])
 {
 Scanner sc=new Scanner(System.in);
 int p,q,n,z,d=0,e,i;
 System.out.println("Enter the number to be encrypted and decrypted");
 int msg=sc.nextInt();
 double c;
 BigInteger msgback;
 System.out.println("Enter 1st prime number p");
 p=sc.nextInt();
 System.out.println("Enter 2nd prime number q");
 q=sc.nextInt();

 n=p*q;
 z=(p-1)*(q-1);
 System.out.println("the value of z = "+z);

 for(e=2;e<z;e++)
 {
 if(gcd(e,z)==1)            // e is for public key exponent
{
break;
}
}
System.out.println("the value of e = "+e);
for(i=0;i<=9;i++)
{
int x=1+(i*z);
if(x%e==0)      //d is for private key exponent
{
d=x/e;
break;
}
}
System.out.println("the value of d = "+d);
c=(Math.pow(msg,e))%n;
System.out.println("Encrypted message is : -");
System.out.println(c);
            //converting int value of n to BigInteger
BigInteger N = BigInteger.valueOf(n);
//converting float value of c to BigInteger
BigInteger C = BigDecimal.valueOf(c).toBigInteger();
msgback = (C.pow(d)).mod(N);
System.out.println("Derypted message is : -");
System.out.println(msgback);
```

```
 }


 static int gcd(int e, int z)

 {

 if(e==0)

 return z;

 else

 return gcd(z%e,e);

 }

}
```