Practice 6: PUBLIC ENCRYPTION RSA

6.1 OVEVIEW

6.1.1 Introduction

- Lab 6: Public Encryption RSA
- Practice time: class: 3 study hours, self-study: 3 study hours.
- Requirements: Students using Netbeans Software

6.1.2 Objective

- This course provides students with knowledge of cryptographic algorithms and how they are used in today's world.
- The content emphasizes the principles, topics, approaches, and problem solving related to the underlying technologies and architectures of the field.

6.2 CONTENTS

6.2.1 Basic knowledge

RSA is a public key cryptographic algorithm. This is the first algorithm that can produce digital signatures while also encrypting data. It represents a significant development in the use of public keys in cryptography.

Let's say Bob wishes to send Alice some information. If they choose to use RSA, Bob will need access to Alice's public key in order to encrypt the message, and Alice will need to use her private key in order to decode it. In order to allow Bob to send his encrypted communications, Alice sends Bob her public key (n, e) through a trustworthy but not necessarily private channel. The private key (d) of Alice is never shared.

After Bob obtains Alice's public key, he can send a message M to Alice. He first turns M (strictly speaking, the un-padded plaintext) into an integer m (strictly speaking, the padded plaintext), such that $0 \leq m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c, using Alice's public key e, corresponding to:

$$m^e \equiv c \pmod{n}$$

Alice can recover m from c by using her private key exponent d by computing:

$$c^d \equiv (m^e)^d \equiv m \pmod{n}$$

❖ **Description of the key generation:**

– Choose two distinct prime numbers p and q.

– Compute n = pq.

– Compute Euler

$$\phi(n) = (p-1)(q-1)$$

– Choose an integer e such that:

$$1 < e < \phi(n)$$

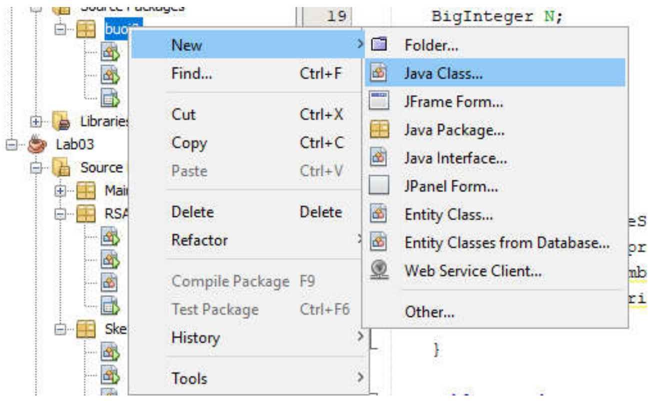
– Determine d as:

$$de \equiv 1 \pmod{\phi(n)}$$

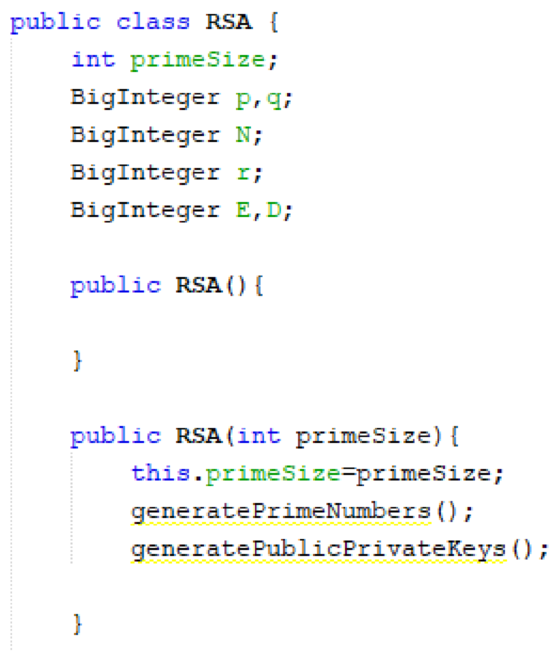
6.2.2 Public Encryption RSA

Write a program to implement RSA encryption algorithm

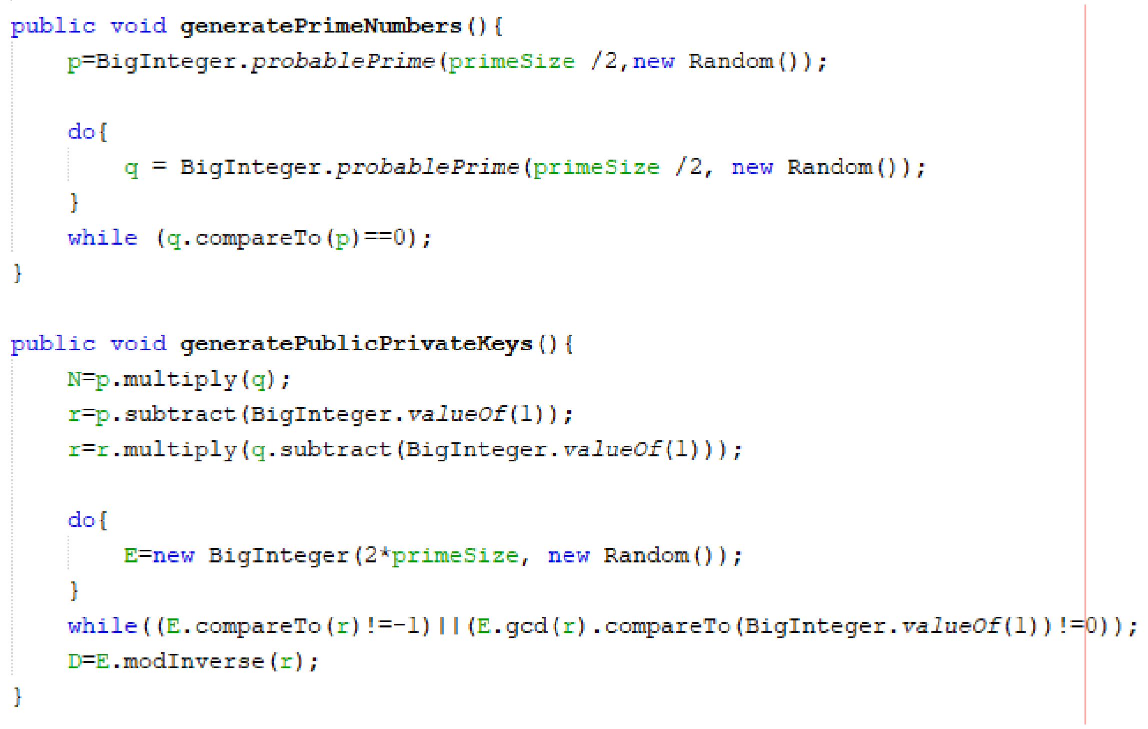
❖ **Step 1: Create Java Class with name RSA.java:**



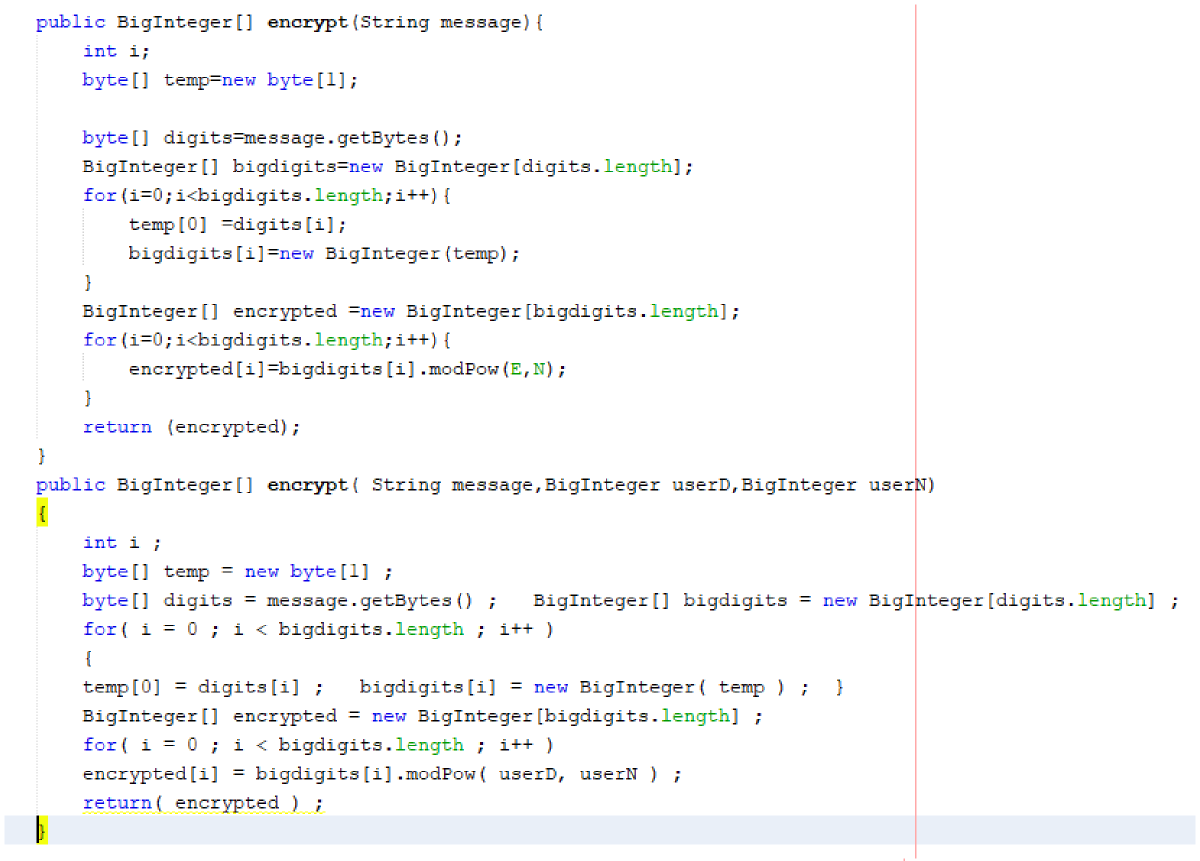
❖ **Step 2: Write code for class RSA:**



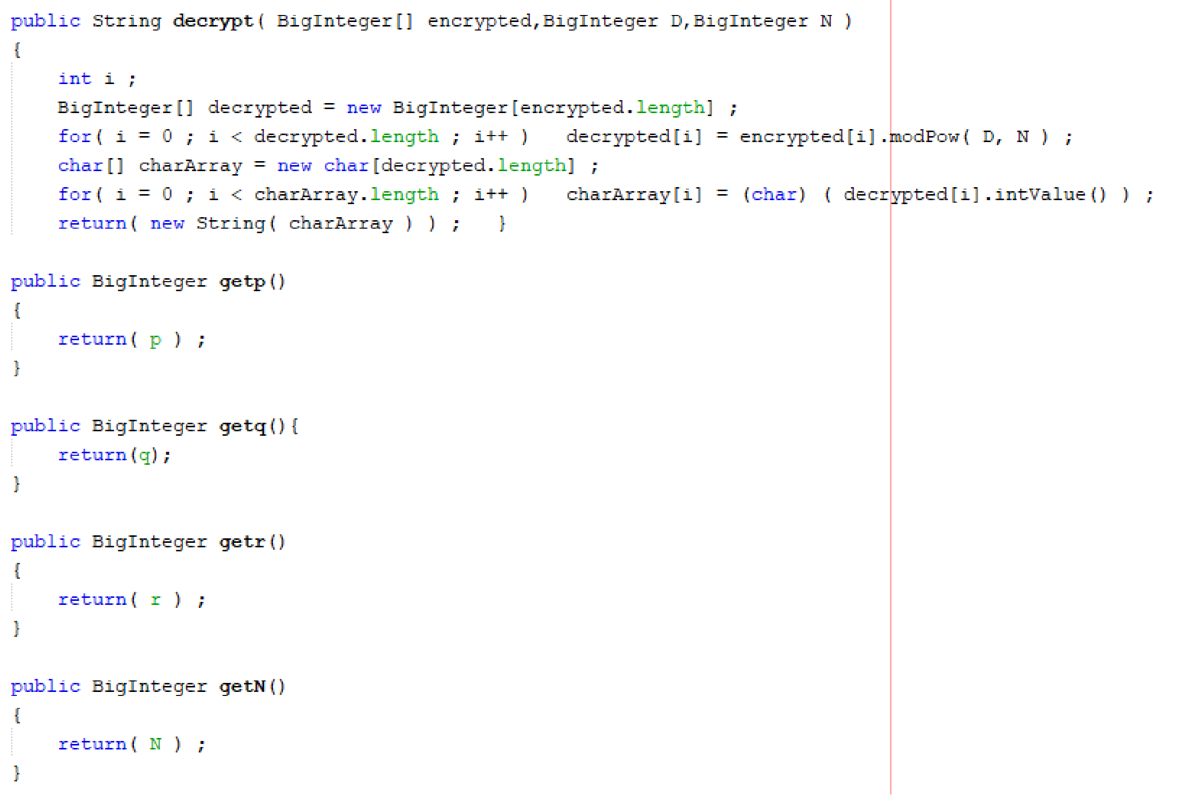
```java
public class RSA {
    int primeSize;
    BigInteger p,q;
    BigInteger N;
    BigInteger r;
    BigInteger E,D;

    public RSA(){

    }

    public RSA(int primeSize){
        this.primeSize=primeSize;
        generatePrimeNumbers();
        generatePublicPrivateKeys();

    }
```



```java
public void generatePrimeNumbers(){
    p=BigInteger.probablePrime(primeSize /2,new Random());

    do{
        q = BigInteger.probablePrime(primeSize /2, new Random());
    }
    while (q.compareTo(p)==0);
}

public void generatePublicPrivateKeys(){
    N=p.multiply(q);
    r=p.subtract(BigInteger.valueOf(1));
    r=r.multiply(q.subtract(BigInteger.valueOf(1)));

    do{
        E=new BigInteger(2*primeSize, new Random());
    }
    while((E.compareTo(r)!=-1)||(E.gcd(r).compareTo(BigInteger.valueOf(1))!=0));
    D=E.modInverse(r);
}
```



```
public BigInteger[] encrypt(String message){
    int i;
    byte[] temp=new byte[1];

    byte[] digits=message.getBytes();
    BigInteger[] bigdigits=new BigInteger[digits.length];
    for(i=0;i<bigdigits.length;i++){
        temp[0] =digits[i];
        bigdigits[i]=new BigInteger(temp);
    }
    BigInteger[] encrypted =new BigInteger[bigdigits.length];
    for(i=0;i<bigdigits.length;i++){
        encrypted[i]=bigdigits[i].modPow(E,N);
    }
    return (encrypted);
}
public BigInteger[] encrypt( String message,BigInteger userD,BigInteger userN)
{
    int i ;
    byte[] temp = new byte[1] ;
    byte[] digits = message.getBytes() ;   BigInteger[] bigdigits = new BigInteger[digits.length] ;
    for( i = 0 ; i < bigdigits.length ; i++ )
    {
    temp[0] = digits[i] ;   bigdigits[i] = new BigInteger( temp ) ;  }
    BigInteger[] encrypted = new BigInteger[bigdigits.length] ;
    for( i = 0 ; i < bigdigits.length ; i++ )
    encrypted[i] = bigdigits[i].modPow( userD, userN ) ;
    return( encrypted ) ;
}
```



```
public String decrypt( BigInteger[] encrypted,BigInteger D,BigInteger N )
{
    int i ;
    BigInteger[] decrypted = new BigInteger[encrypted.length] ;
    for( i = 0 ; i < decrypted.length ; i++ )   decrypted[i] = encrypted[i].modPow( D, N ) ;
    char[] charArray = new char[decrypted.length] ;
    for( i = 0 ; i < charArray.length ; i++ )   charArray[i] = (char) ( decrypted[i].intValue() ) ;
    return( new String( charArray ) ) ;   }

public BigInteger getp()
{
    return( p ) ;
}

public BigInteger getq(){
    return(q);
}

public BigInteger getr()
{
    return( r ) ;
}

public BigInteger getN()
{
    return( N ) ;
}
```

```java
public BigInteger getE()
{
    return( E ) ;
}

public BigInteger getD()
{
    return( D ) ;
}
```

Step 3: Code for Void main():

```java
  public static void main( String[] args ) throws IOException  {

      int primeSize =8;
// Generate Public and Private Keys
      RSA rsa = new RSA( primeSize ) ;
      System.out.println( "Key Size: [" + primeSize + "]" );
      System.out.println( "" ) ;
      System.out.println( "Generated prime numbers p and q" );
      System.out.println( "p: [" + rsa.getp().toString( 16 ).toUpperCase() + "]" );
      System.out.println( "q: [" + rsa.getq().toString( 16 ).toUpperCase() + "]" );
      System.out.println( "" ) ;
      System.out.println( "The public key is the pair (N, E) which will be published." ) ;
      System.out.println( "N: [" + rsa.getN().toString( 16 ).toUpperCase() + "]" ) ;
      System.out.println( "E: [" + rsa.getE().toString( 16 ).toUpperCase() + "]" ) ;
      System.out.println( "" ) ;
      System.out.println( "The private key is the pair (N, D) which will be kept private." ) ;
      System.out.println( "N: [" + rsa.getN().toString( 16 ).toUpperCase() + "]" ) ;
      System.out.println( "D: [" + rsa.getD().toString( 16 ).toUpperCase() + "]" ) ;
      System.out.println( "" ) ;
// Get message (plaintext) from user
      System.out.println( "Please enter message (plaintext):" ) ;
      String plaintext = ( new BufferedReader( new InputStreamReader( System.in ) ) ).readLine() ;
      System.out.println( "" ) ;

// Encrypt Message
      BigInteger[] ciphertext = rsa.encrypt( plaintext ) ;
      System.out.print( "Ciphertext: [" ) ;
      for( int i = 0 ; i < ciphertext.length ; i++ )
      {
          System.out.print( ciphertext[i].toString( 16 ).toUpperCase() ) ;
          if( i != ciphertext.length - 1 )
              System.out.print( " " ) ;
      }
      System.out.println( "]" ) ;   System.out.println( "" ) ;
      RSA rsa1 = new RSA(8);
      String recoveredPlaintext = rsa1.decrypt( ciphertext ,rsa.getD(),rsa.getN()) ;
      System.out.println( "Recovered plaintext: [" + recoveredPlaintext + "]" ) ;
  }

}
```

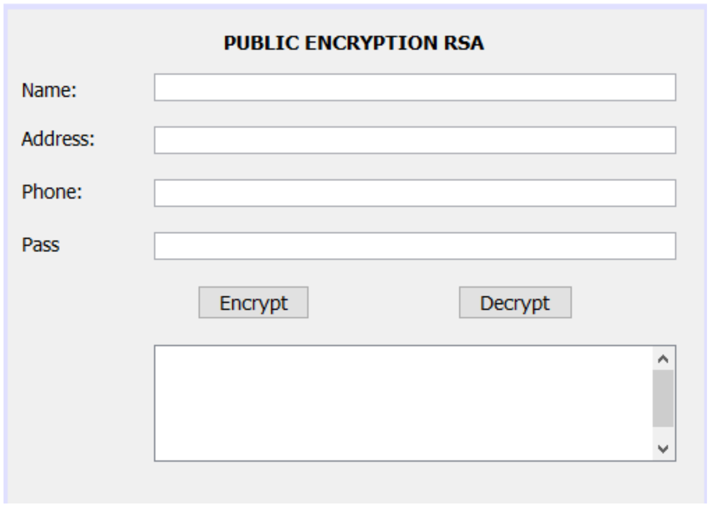
Result:

6.2.3 Public Encryption RSA (Cont.)

Write a program to encrypt and decrypt text with RSA encryption algorithm. The program can perform the following functions:

- Allow input name, address, phone and password.
- Write code for Encrypt and Decrypt.

❖ **Design Form:**



❖ **Step 1: Write an event handler function:**

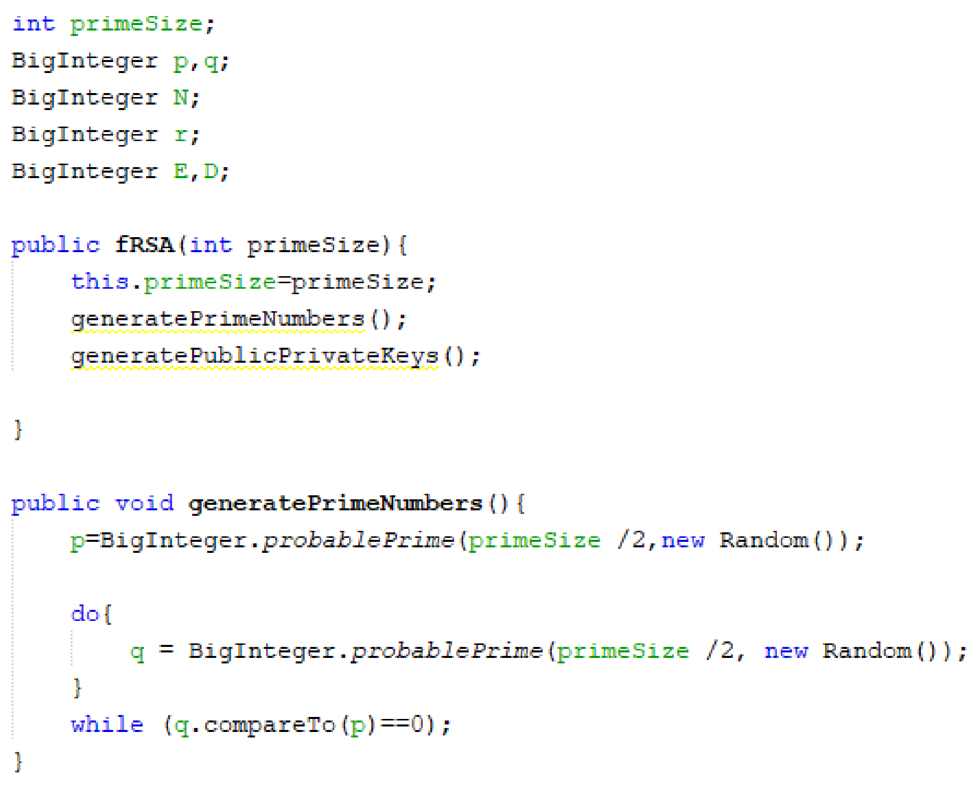
3.1 Button Encrypt

```
private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    Scanner in=new Scanner(System.in);
    String nhash;
    BigInteger[] ciphertext=null;
    BigInteger n=null;
    BigInteger d=null;
    String password="";

    password=txtPass.getText();

    PwdEncryption rsa =new PwdEncryption(8);
    n=rsa.getN();
    d=rsa.getD();
    ciphertext=rsa.encrypt(password);
    StringBuffer bf =new StringBuffer();
    for(int i=0;i<ciphertext.length;i++){
        bf.append(ciphertext[i].toString(16).toUpperCase());
        if(i!=ciphertext.length-1){
            System.out.print("");
        }
    }
    String message=bf.toString();
    txtCipherText.append("Pass encrypted is: "+message);
}
```

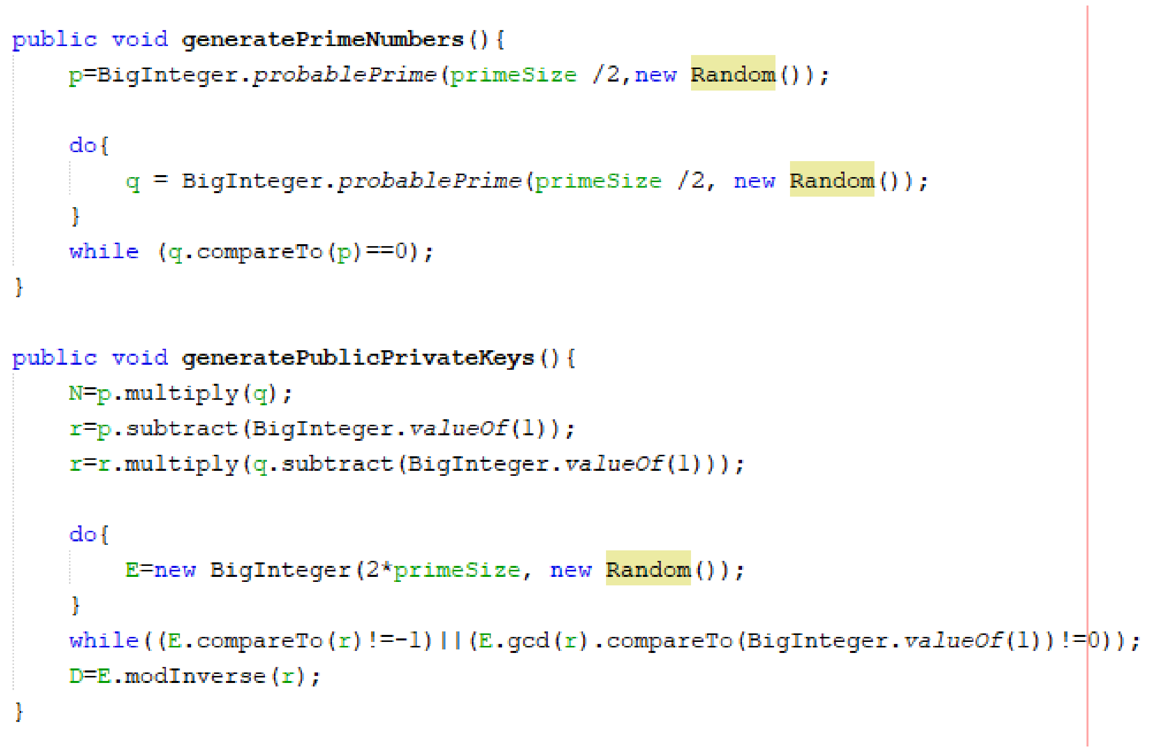
3.2 button Decrypt

```
private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
    Scanner in=new Scanner(System.in);
    String nhash;
    BigInteger[] ciphertext=null;
    BigInteger n=null;
    BigInteger d=null;
    String password="";

    password=txtPass.getText();

    PwdEncryption rsa =new PwdEncryption(8);
    n=rsa.getN();
    d=rsa.getD();
    ciphertext=rsa.encrypt(password);
    String dhash =rsa.decrypt(ciphertext,d,n);
    txtCipherText.append("\nPass after decrypt is: "+dhash);
}
```

3.3 Code funtion:

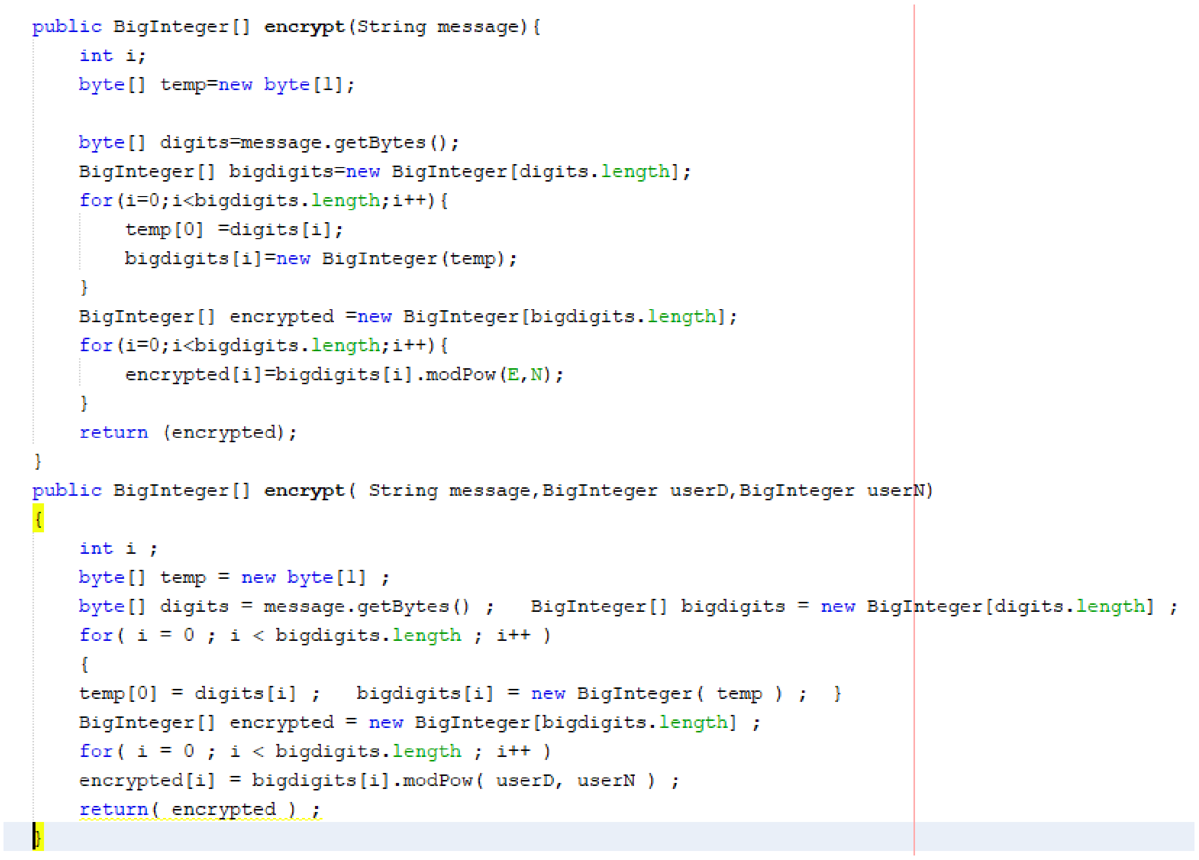


```
int primeSize;
BigInteger p,q;
BigInteger N;
BigInteger r;
BigInteger E,D;

public fRSA(int primeSize){
    this.primeSize=primeSize;
    generatePrimeNumbers();
    generatePublicPrivateKeys();

}

public void generatePrimeNumbers(){
    p=BigInteger.probablePrime(primeSize /2,new Random());

    do{
        q = BigInteger.probablePrime(primeSize /2, new Random());
    }
    while (q.compareTo(p)==0);
}
```

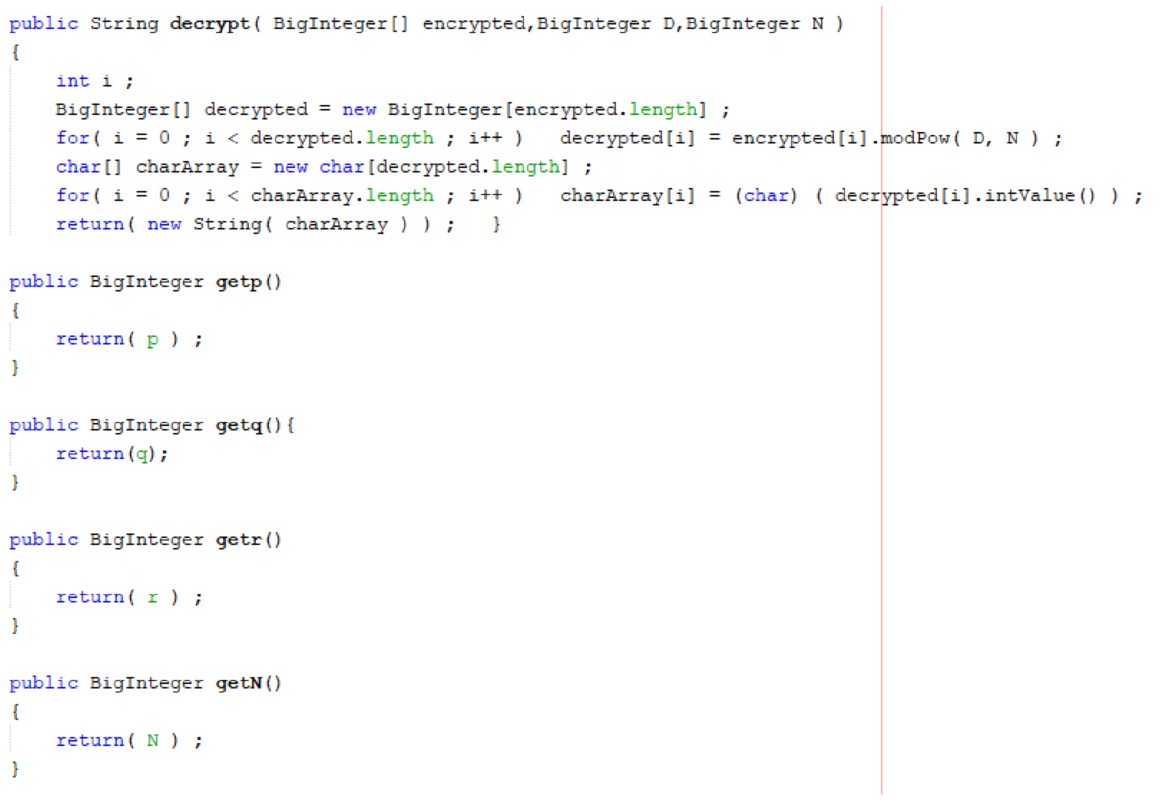


```
public void generatePrimeNumbers(){
    p=BigInteger.probablePrime(primeSize /2,new Random());

    do{
        q = BigInteger.probablePrime(primeSize /2, new Random());
    }
    while (q.compareTo(p)==0);
}
```

```
public void generatePublicPrivateKeys(){
    N=p.multiply(q);
    r=p.subtract(BigInteger.valueOf(1));
    r=r.multiply(q.subtract(BigInteger.valueOf(1)));

    do{
        E=new BigInteger(2*primeSize, new Random());
    }
    while((E.compareTo(r)!=-1)||(E.gcd(r).compareTo(BigInteger.valueOf(1))!=0));
    D=E.modInverse(r);
}
```



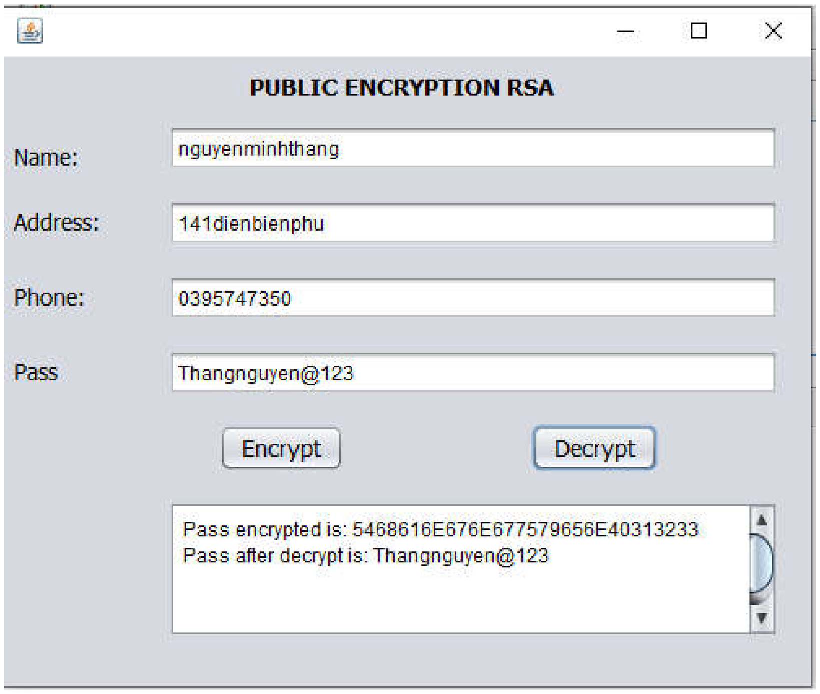
```java
public BigInteger[] encrypt(String message){
    int i;
    byte[] temp=new byte[1];

    byte[] digits=message.getBytes();
    BigInteger[] bigdigits=new BigInteger[digits.length];
    for(i=0;i<bigdigits.length;i++){
        temp[0] =digits[i];
        bigdigits[i]=new BigInteger(temp);
    }
    BigInteger[] encrypted =new BigInteger[bigdigits.length];
    for(i=0;i<bigdigits.length;i++){
        encrypted[i]=bigdigits[i].modPow(E,N);
    }
    return (encrypted);
}
public BigInteger[] encrypt( String message,BigInteger userD,BigInteger userN)
{
    int i ;
    byte[] temp = new byte[1] ;
    byte[] digits = message.getBytes() ;   BigInteger[] bigdigits = new BigInteger[digits.length] ;
    for( i = 0 ; i < bigdigits.length ; i++ )
    {
    temp[0] = digits[i] ;   bigdigits[i] = new BigInteger( temp ) ;  }
    BigInteger[] encrypted = new BigInteger[bigdigits.length] ;
    for( i = 0 ; i < bigdigits.length ; i++ )
    encrypted[i] = bigdigits[i].modPow( userD, userN ) ;
    return( encrypted ) ;
}
```



```java
public String decrypt( BigInteger[] encrypted,BigInteger D,BigInteger N )
{
    int i ;
    BigInteger[] decrypted = new BigInteger[encrypted.length] ;
    for( i = 0 ; i < decrypted.length ; i++ )   decrypted[i] = encrypted[i].modPow( D, N ) ;
    char[] charArray = new char[decrypted.length] ;
    for( i = 0 ; i < charArray.length ; i++ )   charArray[i] = (char) ( decrypted[i].intValue() ) ;
    return( new String( charArray ) ) ;   }

public BigInteger getp()
{
    return( p ) ;
}

public BigInteger getq(){
    return(q);
}

public BigInteger getr()
{
    return( r ) ;
}

public BigInteger getN()
{
    return( N ) ;
}
```

```
public BigInteger getE()
{
    return( E ) ;
}

public BigInteger getD()
{
    return( D ) ;
}
```

Result:



PUBLIC ENCRYPTION RSA

Name: nguyenminhthang

Address: 141dienbienphu

Phone: 0395747350

Pass Thangnguyen@123

```
Pass encrypted is: 5468616E676E677579656E40313233
Pass after decrypt is: Thangnguyen@123
```