Code:

```
package information_security._05_RSA;

import java.math.BigInteger;

public class _RSA {

  public static void main(String args[]) {
    // Initialize prime numbers
    int p = 3;
    int q = 11;

    // Calculate n and z
    int n = p * q;
    int z = (p - 1) * (q - 1);
    System.out.println("n = " + n);
    System.out.println("z = " + z);

    // Find public key exponent e
    int e = findPublicKeyExponent(z);
    System.out.println("e = " + e);

    // Find private key exponent d
    int d = findPrivateKeyExponent(e, z);
    System.out.println("d = " + d);

    // Message to be encrypted and decrypted
    int msg = 12;

    // Encrypt the message
    double encryptedMsg = encrypt(msg, e, n);
    System.out.println("Encrypted message: " + encryptedMsg);

    // Decrypt the message
    BigInteger decryptedMsg = decrypt(encryptedMsg, d, n);
    System.out.println("Decrypted message: " + decryptedMsg);
  }

  // Method to find public key exponent
```

```java
static int findPublicKeyExponent(int z) {
    for (int e = 2; e < z; e++) {
        if (gcd(e, z) == 1) {
            return e;
        }
    }
    return -1; // Public key exponent not found
}

// Method to find private key exponent
static int findPrivateKeyExponent(int e, int z) {
    for (int i = 0; i <= 9; i++) {
        int x = 1 + (i * z);
        if (x % e == 0) {
            return x / e;
        }
    }
    return -1; // Private key exponent not found
}

// Method to calculate greatest common divisor (gcd)
static int gcd(int e, int z) {
    if (e == 0) {
        return z;
    } else {
        return gcd(z % e, e);
    }
}

// Method to encrypt the message
static double encrypt(int msg, int e, int n) {
    return Math.pow(msg, e) % n;
}

// Method to decrypt the message
static BigInteger decrypt(double encryptedMsg, int d, int n) {
    BigInteger N = BigInteger.valueOf(n);
    BigInteger C = BigInteger.valueOf((long) encryptedMsg);
    return C.pow(d).mod(N);
}
```

```
}
```

OUTPUT:

```
n = 33
z = 20
e = 3
d = 7
Encrypted message: 12.0
Decrypted message: 12
```