

## Task 2

```
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
class Main{
    public static SecretKey gen() throws Exception{
        KeyGenerator kg = KeyGenerator.getInstance("DES");
        kg.init(56);
        return kg.generateKey();
    }
    public static String enc(String inp,SecretKey key) throws Exception{
        Cipher cipher= Cipher.getInstance("DES");
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] b=cipher.doFinal(inp.getBytes());
        return Base64.getEncoder().encodeToString(b);
    }
    public static String dec(String ctext,SecretKey sk) throws Exception{
        Cipher cipher = Cipher.getInstance("DES");
        cipher.init(Cipher.DECRYPT_MODE, sk);
        byte[] b1 = Base64.getDecoder().decode(ctext);
        byte[] b2 = cipher.doFinal(b1);
        return new String(b2);
    }
    public static void main (String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        SecretKey sk = gen();
        System.out.println("Enter a text");
        String pt = sc.nextLine();
        String et = enc(pt,sk);
        System.out.println("The encrypted text is ");
        System.out.println(et);
        System.out.println("The decrypted text is");
        String dt = dec(et,sk);
        System.out.println(dt);
    }
}
```

## Task 3

```
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
class Main{
    public static SecretKey gen() throws Exception{
        KeyGenerator kg = KeyGenerator.getInstance("Blowfish");
        kg.init(128);
        return kg.generateKey();
    }
    public static String enc(String inp,SecretKey key) throws Exception{
        Cipher cipher= Cipher.getInstance("Blowfish");
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] b=cipher.doFinal(inp.getBytes());
        return Base64.getEncoder().encodeToString(b);
    }
    public static String dec(String ctext,SecretKey sk) throws Exception{
        Cipher cipher = Cipher.getInstance("Blowfish");
    }
```

```

        cipher.init(Cipher.DECRYPT_MODE, sk);
        byte[] b1 = Base64.getDecoder().decode(ctext);
        byte[] b2 = cipher.doFinal(b1);
        return new String(b2);
    }
    public static void main (String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        SecretKey sk = gen();
        System.out.println("Enter a text");
        String pt = sc.nextLine();
        String et = enc(pt, sk);
        System.out.println("The encrypted text is ");
        System.out.println(et);
        System.out.println("The decrypted text is");
        String dt = dec(et, sk);
        System.out.println(dt);
    }
}

```

#### Task 5

```

import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
class Main {
    public static SecretKey gen() throws Exception {
        KeyGenerator kg = KeyGenerator.getInstance("RC4");
        kg.init(128);
        return kg.generateKey();
    }
    public static String enc(String inp, SecretKey key) throws Exception {
        Cipher cipher= Cipher.getInstance("RC4");
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] b=cipher.doFinal(inp.getBytes());
        return Base64.getEncoder().encodeToString(b);
    }
    public static String dec(String ctext, SecretKey sk) throws Exception {
        Cipher cipher = Cipher.getInstance("RC4");
        cipher.init(Cipher.DECRYPT_MODE, sk);
        byte[] b1 = Base64.getDecoder().decode(ctext);
        byte[] b2 = cipher.doFinal(b1);
        return new String(b2);
    }
    public static void main (String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        SecretKey sk = gen();
        System.out.println("Enter a text");
        String pt = sc.nextLine();
        String et = enc(pt, sk);
        System.out.println("The encrypted text is ");
        System.out.println(et);
        System.out.println("The decrypted text is");
        String dt = dec(et, sk);
        System.out.println(dt);
    }
}

```

#### Task 7

```

import java.security.*;
import javax.crypto.Cipher;

```

```

import java.util.Base64;

public class SimpleRSA {

    // Generate RSA Key Pair
    public static KeyPair generateKeyPair() throws Exception {
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("RSA");
        keyPairGen.initialize(2048); // 2048-bit key size
        return keyPairGen.generateKeyPair();
    }

    // Encrypt message using public key
    public static String encrypt(String message, PublicKey publicKey) throws
Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        byte[] encrypted = cipher.doFinal(message.getBytes());
        return Base64.getEncoder().encodeToString(encrypted); // Convert
encrypted bytes to Base64 string
    }

    // Decrypt message using private key
    public static String decrypt(String encryptedMessage, PrivateKey privateKey)
throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] decrypted =
cipher.doFinal(Base64.getDecoder().decode(encryptedMessage)); // Decode Base64
to bytes
        return new String(decrypted);
    }

    public static void main(String[] args) throws Exception {
        // Generate RSA Key Pair
        KeyPair keyPair = generateKeyPair();
        PublicKey publicKey = keyPair.getPublic();
        PrivateKey privateKey = keyPair.getPrivate();

        // Message to encrypt
        String originalMessage = "Hello, RSA!";
        System.out.println("Original Message: " + originalMessage);

        // Encrypt the message
        String encryptedMessage = encrypt(originalMessage, publicKey);
        System.out.println("Encrypted Message: " + encryptedMessage);

        // Decrypt the message
        String decryptedMessage = decrypt(encryptedMessage, privateKey);
        System.out.println("Decrypted Message: " + decryptedMessage);
    }
}

```

#### Task 8

```

import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.math.BigInteger;
import java.security.*;
class Main{
    public static final BigInteger p = new BigInteger("23");

```

```

public static final BigInteger g = new BigInteger("5");
public static BigInteger gen(){
    SecureRandom random = new SecureRandom();
    return new BigInteger(256,random);
}
private static BigInteger calpub(BigInteger pk) {
    return g.modPow(pk, p);
}

private static BigInteger calsh(BigInteger opuk, BigInteger prk) {
    return opuk.modPow(prk, p); // Shared secret =
otherPublicKey^privateKey mod p
}
public static void main (String[] args) {
    BigInteger aliceprivate = gen();
    BigInteger bobprivate = gen();

    BigInteger alicepublic = calpub(aliceprivate);
    BigInteger bobpublic = calpub(bobprivate);

    BigInteger aliceshared = calsh(bobpublic,aliceprivate);
    BigInteger bobshared = calsh(alicepublic,bobprivate);

    System.out.println(aliceshared);
    System.out.println(bobshared);
}
}
Task 9

```

```

import java.security.MessageDigest;
import java.util.Base64;
import java.util.Scanner;

public class Main {
    public static String sha1Hash(String input) throws Exception {
        // Initialize SHA-1 MessageDigest
        MessageDigest md = MessageDigest.getInstance("SHA-1");

        // Compute hash
        byte[] hashBytes = md.digest(input.getBytes());

        // Convert byte array to a Base64-encoded string for readability
        return Base64.getEncoder().encodeToString(hashBytes);
    }

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a String:");
        String input = sc.nextLine();

        // Hash input and display
        String hashedOutput = sha1Hash(input);
        System.out.println("SHA-1 Hash: " + hashedOutput);
    }
}

```

Task 10

```

import java.security.*;
import java.util.*;

```

```

public class Main {
    public static String sha1Hash(String input) throws Exception {
        // Initialize SHA-1 MessageDigest
        MessageDigest md = MessageDigest.getInstance("md5");

        // Compute hash
        byte[] hashBytes = md.digest(input.getBytes());

        // Convert byte array to a Base64-encoded string for readability
        return Base64.getEncoder().encodeToString(hashBytes);
    }

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a String:");
        String input = sc.nextLine();

        // Hash input and display
        String hashedOutput = sha1Hash(input);
        System.out.println("MD5 Hash: " + hashedOutput);
    }
}

```