**DES CODE**

import java.util.\*;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.security.spec.KeySpec;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESedeKeySpec;

import sun.misc.BASE64Decoder;

import sun.misc.BASE64Encoder;

public class DES {

private static final String UNICODE\_FORMAT = "UTF8";

public static final String DESEDE\_ENCRYPTION\_SCHEME = "DESede";

privateKeySpecmyKeySpec;

privateSecretKeyFactorymySecretKeyFactory;

private Cipher cipher;

byte[] keyAsBytes;

private String myEncryptionKey;

private String myEncryptionScheme;

SecretKey key;

static BufferedReader br = new BufferedReader(new

InputStreamReader(System.in)); public DES() throws Exception {

myEncryptionScheme =DESEDE\_ENCRYPTION\_SCHEME; keyAsBytes =

myEncryptionKey.getBytes(UNICODE\_FORMAT); myKeySpec

= new DESedeKeySpec(keyAsBytes);

mySecretKeyFactory = SecretKeyFactory.getInstance(myEncryptionScheme);

cipher = Cipher.getInstance(myEncryptionScheme);

key = mySecretKeyFactory.generateSecret(myKeySpec);

}

public String encrypt(String unencryptedString)

{ String encryptedString = null;

try {

cipher.init(Cipher.ENCRYPT\_MODE, key);

byte[] plainText = unencryptedString.getBytes(UNICODE\_FORMAT);

byte[] encryptedText = cipher.doFinal(plainText);

BASE64Encoder base64encoder = new BASE64Encoder();

encryptedString = base64encoder.encode(encryptedText); }

catch (Exception e) {

e.printStackTrace(); }

returnencryptedString; }

public String decrypt(String encryptedString)

{ String decryptedText=null;

try {

cipher.init(Cipher.DECRYPT\_MODE, key);

BASE64Decoder base64decoder = new BASE64Decoder();

byte[] encryptedText = base64decoder.decodeBuffer(encryptedString);

byte[] plainText = cipher.doFinal(encryptedText); decryptedText=

bytes2String(plainText); }

catch (Exception e) {

e.printStackTrace(); }

returndecryptedText; }

private static String bytes2String(byte[] bytes)

{ StringBufferstringBuffer = new

StringBuffer(); for (int i = 0; i <bytes.length;

i++) { stringBuffer.append((char) bytes[i]); }

returnstringBuffer.toString(); }

public static void main(String args []) throws Exception

{ System.out.print("Enter the string: ");

DES myEncryptor= new DES();

String stringToEncrypt = br.readLine();

String encrypted = myEncryptor.encrypt(stringToEncrypt); String decrypted =

myEncryptor.decrypt(encrypted); System.out.println("\nString To Encrypt: "

+stringToEncrypt); System.out.println("\nEncrypted Value : " +encrypted);

System.out.println("\nDecrypted Value : " +decrypted);

System.out.println("");

}

}

OUTPUT

**DES CODE**

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

public class DES

{

public static void main(String[] argv) {

try{

System.out.println("Message Encryption Using DES Algorithm\n-------");

KeyGenerator keygenerator = KeyGenerator.getInstance("DES");

SecretKey myDesKey = keygenerator.generateKey();

Cipher desCipher;

desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

desCipher.init(Cipher.ENCRYPT\_MODE, myDesKey);

byte[] text = "Secret Information ".getBytes();

System.out.println("Message [Byte Format] : " + text);

System.out.println("Message : " + new String(text));

byte[] textEncrypted = desCipher.doFinal(text);

System.out.println("Encrypted Message: " + textEncrypted);

desCipher.init(Cipher.DECRYPT\_MODE, myDesKey);

byte[] textDecrypted = desCipher.doFinal(textEncrypted);

System.out.println("Decrypted Message: " + new

String(textDecrypted));

}catch(NoSuchAlgorithmException e){

e.printStackTrace();

}catch(NoSuchPaddingException e){

e.printStackTrace();

}catch(InvalidKeyException e){

e.printStackTrace();

}catch(IllegalBlockSizeException e){

e.printStackTrace();

}catch(BadPaddingException e){

e.printStackTrace();

}

}

}

OUTPUT