GHARDA FOUNDATION

GHARDA FOUNDATION'S
GHARDA INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

GIT GHARDA INSTITUTE OF TECHNOLOGY

Evaluation Sheet

Class: T.E Computer Engineering **Sem:** VI

Subject: Cryptography and System Security

Experiment No: 5

Date: 14/02/2023

Title of Experiment: Design and Implementation of Data Encryption Standard (DES).

Sr. No.	Evaluation Criteria	Max Marks	Marks Obtained
1	Practical Performance	12	
2	Oral	2	
3	Timely Submission	1	
	Total	15	

Signature of Subject Teacher
[Vijesh M.Nair]

Program Code –

```
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random;
import java.util.*;

public class DES {
    byte[] skey = new byte[1000];
    String skeyString;
    static byte[] raw;
    String inputMessage, encryptedData, decryptedMessage;

    public DES() {
        try {
            generateSymmetricKey();
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter Plaintext: ");
            inputMessage = sc.nextLine();
            byte[] ibyte = inputMessage.getBytes();
            byte[] ebyte = encrypt(raw, ibyte);
            String encryptedData = new String(ebyte);
            System.out.println("Encrypted Message: " + encryptedData);
            byte[] dbyte = decrypt(raw, ebyte);
            String decryptedMessage = new String(dbyte);
            System.out.println("Decrypted Message: " + decryptedMessage);
        } catch (Exception e) {
            System.out.println(e);
        }
```

```
    }
    void generateSymmetricKey() {
        try {
            Random r = new Random();
            int num = r.nextInt(1000);
            String knum = String.valueOf(num);
            byte[] knumb = knum.getBytes();
            skey = getRawKey(knumb);
            skeyString = new String(skey);
            System.out.println("DES Symmetric Key: " + skeyString);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
    public static byte[] getRawKey(byte[] seed) throws Exception{
        KeyGenerator kgen = KeyGenerator.getInstance("DES");
        SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
        sr.setSeed(seed);
        kgen.init(56, sr);
        SecretKey skey = kgen.generateKey();
        raw = skey.getEncoded();
        return raw;
    }
    private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception{
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");
        Cipher cipher = Cipher.getInstance("DES");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);
        return encrypted;
    }
```

```
    private static byte[] decrypt(byte[] raw,byte[] encrypted) throws
Exception{
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");
        Cipher cipher = Cipher.getInstance("DES");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return decrypted;
    }

    public static void main(String[] args) {
        DES des = new DES();
    }
}
```

Output –