**EX.NO.5 ADVANCED ENCRYPTION STANDARD (AES)**

**Code:**

import java.util.\*;

import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec;

import java.security.MessageDigest;

import java.io.UnsupportedEncodingException;

import java.security.NoSuchAlgorithmException;

public class AES {

SecretKeySpec secretKey;

byte[] keyArray;

public void generateKey(String key) {

MessageDigest sha = null;

try {

keyArray = key.getBytes("UTF-8");

sha = MessageDigest.getInstance("SHA-1");

keyArray = sha.digest(keyArray);

keyArray = Arrays.copyOf(keyArray, 16);

secretKey = new SecretKeySpec(keyArray, "AES"); //

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

}

}

String encrypt(String strToEncrypt, String secret) {

try {

generateKey(secret);

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); //

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

return Base64

.getEncoder()

.encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));

} catch (Exception e) {

System.out.println("Error while encrypting: " + e.toString());

}

return null;

}

String decrypt(String strToDecrypt, String secret) {

try {

generateKey(secret);

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");

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

return new String(

cipher.doFinal(Base64.getDecoder().decode(strToDecrypt))

);

} catch (Exception e) {

System.out.println("Error while decrypting: " + e.toString());

}

return null;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String key, plainText, cipherText;

int choice = 0;

AES aes = new AES();

while (true) {

System.out.println("\nADVANCED ENCRYPTION STANDARD - AES");

System.out.println("------------------------------");

System.out.println("\n1.Key Generation");

System.out.println("\n2.Encryption");

System.out.println("\n3.Decryption");

System.out.println("\n4.Exit");

System.out.print("\nEnter your choice(1/2/3/4): ");

choice = sc.nextInt();

sc.nextLine();

if (choice == 1) {

System.out.println("\nKEY - GENERATION");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.print("\nEnter the key: ");

key = sc.nextLine();

aes.generateKey(key);

System.out.println(

"\nSHA1 hash of key (in Base64 format):" +

Base64.getEncoder().encodeToString(aes.keyArray)

);

} else if (choice == 2) {

System.out.println("\nENCRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*");

System.out.print("\nEnter plaintext: ");

plainText = sc.nextLine();

System.out.print("\nEnter the key: ");

key = sc.nextLine();

aes.generateKey(key);

cipherText = aes.encrypt(plainText, key);

System.out.println("\nThe ciphertext (in Base64 format): " + cipherText);

} else if (choice == 3) {

System.out.println("\nDECRYPTION");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*");

System.out.print("\nEnter ciphertext (in Base64 format): ");

cipherText = sc.nextLine();

System.out.print("\nEnter the key: ");

key = sc.nextLine();

aes.generateKey(key);

plainText = aes.decrypt(cipherText, key);

System.out.println("\nThe plaintext is: " + plainText);

} else {

break;

}

}

sc.close();

}

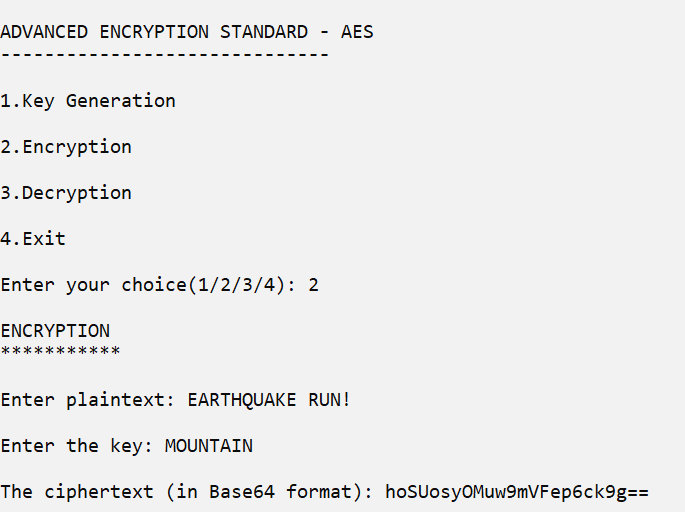
}

**OUTPUT:**

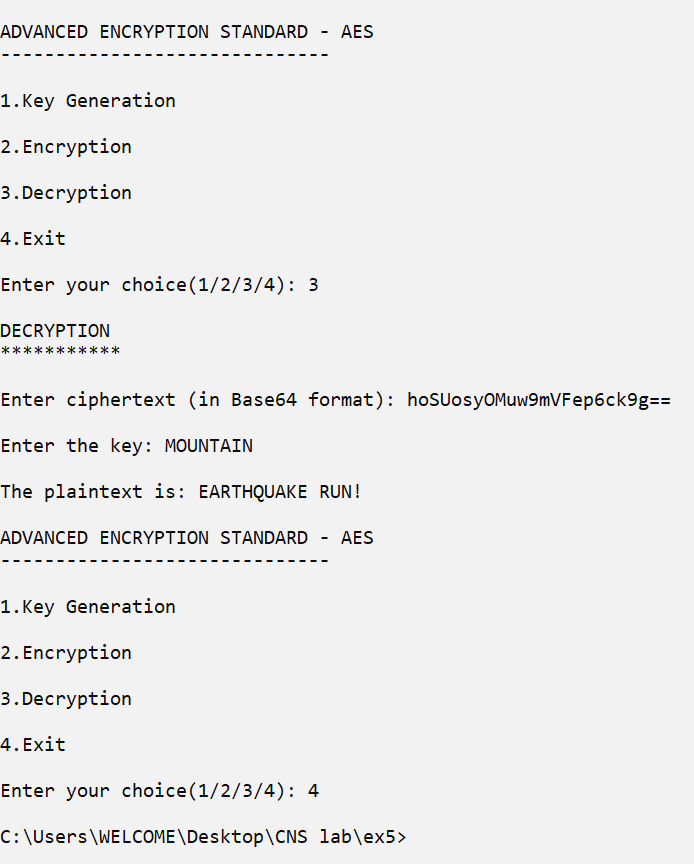
**Key Generation:**



**Encryption:**

****

**Decryption:**

****