Practical No. 9

AES DES in Java

**Q.1]** Write a Java program for implementing AES and DES algorithms.

**AES**

**Code:**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.security.InvalidKeyException;

import java.security.Key;

import java.security.NoSuchAlgorithmException;

import java.util.Date;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.spec.SecretKeySpec;

public class AesExample {

static void fileProcessor(int cipherMode,String key,File inputFile,File outputFile){

try {

Key secretKey = new SecretKeySpec(key.getBytes(), "AES");

Cipher cipher = Cipher.getInstance("AES");

cipher.init(cipherMode, secretKey);

FileInputStream inputStream = new FileInputStream(inputFile);

byte[] inputBytes = new byte[(int) inputFile.length()];

inputStream.read(inputBytes);

byte[] outputBytes = cipher.doFinal(inputBytes);

FileOutputStream outputStream = new FileOutputStream(outputFile);

outputStream.write(outputBytes);

inputStream.close();

outputStream.close();

}

catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException | BadPaddingException | IllegalBlockSizeException | IOException e) {

e.printStackTrace();

}

}

public static void main(String[] args)

{

long beforeUsedMem = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();

long lStartTime = new Date().getTime();

String key = "This is a secret";

File inputFile = new File("text.txt");

File encryptedFile = new File("aes\_text.encrypted");

File decryptedFile = new File("aes\_decrypted-text.txt");

try

{

AesExample.fileProcessor(Cipher.ENCRYPT\_MODE,key,inputFile,encryptedFile);

AesExample.fileProcessor(Cipher.DECRYPT\_MODE,key,encryptedFile,decryptedFile);

System.out.println("Success");

}

catch (Exception ex) {

System.out.println(ex.getMessage());

ex.printStackTrace();

}

long lEndTime = new Date().getTime();

long difference = lEndTime - lStartTime;

System.out.println("Elapsed milliseconds: " + difference);

long afterUsedMem = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();

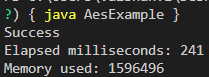
long actualMemUsed = afterUsedMem - beforeUsedMem;

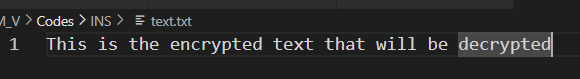
System.out.println("Memory used: " + actualMemUsed);

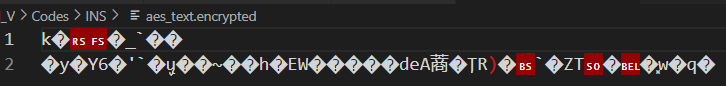
}

}

**Output:**

****

****

****

****

**DES**

**Code:**

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.util.Date;

import javax.crypto.Cipher;

import javax.crypto.CipherInputStream;

import javax.crypto.CipherOutputStream;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESKeySpec;

public class DesExample {

public static void main(String[] args)

{

long beforeUsedMem = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();

long lStartTime = new Date().getTime();

try

{

String key = "squirrel123"; // needs to be at least 8 characters for DES

FileInputStream fis = new FileInputStream("text.txt");

FileOutputStream fos = new FileOutputStream("des\_encrypted.txt");

encrypt(key, fis, fos);

FileInputStream fis2 = new FileInputStream("des\_encrypted.txt");

FileOutputStream fos2 = new FileOutputStream("des\_decrypted.txt");

decrypt(key, fis2, fos2);

System.out.println("Success");

}

catch (Throwable e)

{

e.printStackTrace();

}

long lEndTime = new Date().getTime();

long difference = lEndTime - lStartTime;

System.out.println("Elapsed milliseconds: " + difference);

long afterUsedMem = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();

long actualMemUsed = afterUsedMem - beforeUsedMem;

System.out.println("Memory used: " + actualMemUsed);

}

public static void encrypt(String key, InputStream is, OutputStream os) throws Throwable

{

encryptOrDecrypt(key, Cipher.ENCRYPT\_MODE, is, os);

}

public static void decrypt(String key, InputStream is, OutputStream os) throws Throwable

{

encryptOrDecrypt(key, Cipher.DECRYPT\_MODE, is, os);

}

public static void encryptOrDecrypt(String key, int mode, InputStream is, OutputStream os) throws Throwable

{

DESKeySpec dks = new DESKeySpec(key.getBytes());

SecretKeyFactory skf = SecretKeyFactory.getInstance("DES");

SecretKey desKey = skf.generateSecret(dks);

Cipher cipher = Cipher.getInstance("DES"); // DES/ECB/PKCS5Padding for SunJCE

if (mode == Cipher.ENCRYPT\_MODE) {

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

CipherInputStream cis = new CipherInputStream(is,cipher);

doCopy(cis, os);

}

else if (mode == Cipher.DECRYPT\_MODE) {

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

CipherOutputStream cos = new CipherOutputStream(os,cipher);

doCopy(is, cos);

}

}

public static void doCopy(InputStream is, OutputStream os) throws IOException

{

byte[] bytes = new byte[64];

int numBytes;

while((numBytes = is.read(bytes)) != -1){

os.write(bytes, 0, numBytes);

}

os.flush();

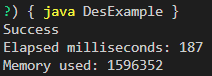
os.close();

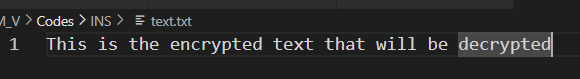
is.close();

}

}

**Output:**

****

****

****

****