TASK-1A

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Scanner;

public class Task1a

{

static Scanner sc = new Scanner(System.in);

static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

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

{

System.out.print("Enter any String: ");

String str = br.readLine();

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

int key = sc.nextInt();

String encrypted = encrypt(str, key);

System.out.println("\nEncrypted String is: " + encrypted);

String decrypted = decrypt(encrypted, key);

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

System.out.println();

}

public static String encrypt(String str, int key)

{

String encrypted = "";

for (int i = 0; i < str.length(); i++)

{

int c = str.charAt(i);

if (Character.isUpperCase(c))

{

c = c + (key % 26);

if (c > 'Z')

c = c - 26;

}

else if (Character.isLowerCase(c))

{

c = c + (key % 26);

if (c > 'z')

c = c - 26;

}

encrypted += (char) c;

}

return encrypted;

}

public static String decrypt(String str, int key)

{

String decrypted = "";

for (int i = 0; i < str.length(); i++)

{

int c = str.charAt(i);

if (Character.isUpperCase(c))

{

c = c - (key % 26);

if (c < 'A')

c = c + 26;

}

else if (Character.isLowerCase(c))

{

c = c - (key % 26);

if (c < 'a')

c = c + 26;

}

decrypted += (char) c;

}

return decrypted;

}

}

-----------------------------------------------------------------------------------------------------

TASK 1B

import java.io.\*;

import java.util.\*;

public class Task1b

{

static Scanner sc = new Scanner(System.in);

static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

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

{

String encrypt = encrypt();

String decrypt = decrypt(encrypt);

System.out.println("The encrypted data is: " +encrypt);

System.out.println("The decrypted data is: " +decrypt);

}

public static String encrypt()throws IOException

{

String encrypt = "";

String a = "abcdefghijklmnopqrstuvwxyz";

String b = "zyxwvutsrqponmlkjihgfedcba";

System.out.print("Enter any string: ");

String str = br.readLine();

char c;

for(int i=0;i < str.length();i++)

{

c = str.charAt(i);

int j = a.indexOf(c);

encrypt = encrypt+b.charAt(j);

}

return encrypt;

}

public static String decrypt(String encrypt)

{

String a = "abcdefghijklmnopqrstuvwxyz";

String b = "zyxwvutsrqponmlkjihgfedcba";

String decrypt = "";

char c;

for(int i=0;i < encrypt.length();i++)

{

c = encrypt.charAt(i);

int j = a.indexOf(c);

decrypt = decrypt+b.charAt(j);

}

return decrypt;

}

}

-----------------------------------------------------------------------------------------------------

TASK 2

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 Task2 {

private static Cipher encryptCipher;

private static Cipher decryptCipher;

public static void main(String[] args) {

try {

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

SecretKey secretKey = keygenerator.generateKey();

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

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

byte[] encryptedData = encryptData("Classified Information!");

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

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

decryptData(encryptedData);

} 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();

}

}

/\*\*

\* Encrypt Data

\* @param data

\* @return

\* @throws IllegalBlockSizeException

\* @throws BadPaddingException

\*/

private static byte[] encryptData(String data)

throws IllegalBlockSizeException, BadPaddingException {

System.out.println("Data Before Encryption :" + data);

byte[] dataToEncrypt = data.getBytes();

byte[] encryptedData = encryptCipher.doFinal(dataToEncrypt);

System.out.println("Encryted Data: " + encryptedData);

return encryptedData;

}

/\*\*

\* Decrypt Data

\* @param data

\* @throws IllegalBlockSizeException

\* @throws BadPaddingException

\*/

private static void decryptData(byte[] data)

throws IllegalBlockSizeException, BadPaddingException {

byte[] textDecrypted = decryptCipher.doFinal(data);

System.out.println("Decryted Data: " + new String(textDecrypted));

}

}

-----------------------------------------------------------------------------------------------------

TASK 3

import java.io.\*;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.security.Key;

import javax.crypto.Cipher;

import javax.crypto.CipherOutputStream;

import javax.crypto.KeyGenerator;

import java.util.Base64; // Use Base64 from java.util

public class Task3 {

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

// Generate a Blowfish key

KeyGenerator keyGenerator = KeyGenerator.getInstance("Blowfish");

keyGenerator.init(128);

Key secretKey = keyGenerator.generateKey();

// Initialize the cipher in encryption mode

Cipher cipherOut = Cipher.getInstance("Blowfish/CFB/NoPadding");

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

// Encode and display the Initialization Vector (IV)

byte iv[] = cipherOut.getIV();

if (iv != null) {

String encodedIV = Base64.getEncoder().encodeToString(iv); // Encode using java.util.Base64

System.out.println("Initialization Vector of the Cipher: " + encodedIV);

}

// File input/output streams

FileInputStream fin = new FileInputStream("inputFile.txt");

FileOutputStream fout = new FileOutputStream("outputFile.txt");

CipherOutputStream cout = new CipherOutputStream(fout, cipherOut);

// Read from input file and write encrypted data to output file

int input = 0;

while ((input = fin.read()) != -1) {

cout.write(input);

}

// Close the streams

fin.close();

cout.close();

}

}

-----------------------------------------------------------------------------------------------------

TASK 5

import java.io.\*;

public class Task5 {

public static void main(String args[]) throws IOException {

int temp = 0;

String ptext;

String key;

int s[] = new int[256];

int k[] = new int[256];

// Input handling

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

System.out.println("\nENTER PLAIN TEXT:");

ptext = in.readLine();

System.out.println("\nENTER KEY TEXT:");

key = in.readLine();

// Conversion of input to character arrays

char ptextc[] = ptext.toCharArray();

char keyc[] = key.toCharArray();

int cipher[] = new int[ptext.length()];

int decrypt[] = new int[ptext.length()];

int ptexti[] = new int[ptext.length()];

int keyi[] = new int[key.length()];

// Convert plaintext and key to integer arrays

for (int i = 0; i < ptext.length(); i++) {

ptexti[i] = (int) ptextc[i];

}

for (int i = 0; i < key.length(); i++) {

keyi[i] = (int) keyc[i];

}

// Initialize S and K arrays

for (int i = 0; i < 256; i++) {

s[i] = i;

k[i] = keyi[i % key.length()];

}

// Key scheduling algorithm (KSA)

int j = 0;

for (int i = 0; i < 256; i++) {

j = (j + s[i] + k[i]) % 256;

temp = s[i];

s[i] = s[j];

s[j] = temp;

}

// Pseudo-random generation algorithm (PRGA)

int i = 0;

j = 0;

int z = 0;

for (int l = 0; l < ptext.length(); l++) {

i = (i + 1) % 256;

j = (j + s[i]) % 256;

temp = s[i];

s[i] = s[j];

s[j] = temp;

z = s[(s[i] + s[j]) % 256];

cipher[l] = z ^ ptexti[l]; // Encryption

decrypt[l] = z ^ cipher[l]; // Decryption

}

// Display results

System.out.println("\nENCRYPTED:");

display(cipher);

System.out.println("\nDECRYPTED:");

display(decrypt);

}

// Helper method to display an array as a string

static void display(int disp[]) {

char convert[] = new char[disp.length];

for (int l = 0; l < disp.length; l++) {

convert[l] = (char) disp[l];

System.out.print(convert[l]);

}

}

}

TASK 7

import java.math.BigInteger;

import java.util.Random;

import java.io.\*;

public class RSA {

private BigInteger p;

private BigInteger q;

private BigInteger N;

private BigInteger phi;

private BigInteger e;

private BigInteger d;

private int bitlength = 1024;

private Random r;

// Default constructor to generate keys

public RSA() {

r = new Random();

p = BigInteger.probablePrime(bitlength, r);

q = BigInteger.probablePrime(bitlength, r);

N = p.multiply(q);

phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

e = BigInteger.probablePrime(bitlength / 2, r);

while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0) {

e = e.add(BigInteger.ONE);

}

d = e.modInverse(phi);

}

// Constructor for custom keys

public RSA(BigInteger e, BigInteger d, BigInteger N) {

this.e = e;

this.d = d;

this.N = N;

}

public static void main(String[] args) throws IOException {

RSA rsa = new RSA();

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

String testString;

// Get plaintext input

System.out.println("Enter the plain text:");

testString = in.readLine();

// Display original text

System.out.println("Encrypting String: " + testString);

System.out.println("String in Bytes: " + bytesToString(testString.getBytes()));

// Encrypt

byte[] encrypted = rsa.encrypt(testString.getBytes());

System.out.println("Encrypted String in Bytes: " + bytesToString(encrypted));

// Decrypt

byte[] decrypted = rsa.decrypt(encrypted);

System.out.println("Decrypted String in Bytes: " + bytesToString(decrypted));

System.out.println("Decrypted String: " + new String(decrypted));

}

// Helper method to convert bytes to a readable string

private static String bytesToString(byte[] encrypted) {

StringBuilder result = new StringBuilder();

for (byte b : encrypted) {

result.append(Byte.toString(b)).append(" ");

}

return result.toString().trim();

}

// Encrypt message

public byte[] encrypt(byte[] message) {

return (new BigInteger(message)).modPow(e, N).toByteArray();

}

// Decrypt message

public byte[] decrypt(byte[] message) {

return (new BigInteger(message)).modPow(d, N).toByteArray();

}

}

-----------------------------------------------------------------------------------------------------

TASK 8

import java.io.\*;

import java.math.BigInteger;

public class Task8

{

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

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter prime number:");

BigInteger p=new BigInteger(br.readLine());

System.out.print("Enter primitive root of "+p+":");

BigInteger g=new BigInteger(br.readLine());

System.out.println("Enter value for x less than " +p+":");

BigInteger x=new BigInteger(br.readLine());

BigInteger R1=g.modPow(x,p);

System.out.println("R1="+R1);

System.out.print("Enter value for y less than "+p+":");

BigInteger y=new BigInteger(br.readLine());

BigInteger R2=g.modPow(y,p);

System.out.println("R2="+R2);

BigInteger k1=R2.modPow(x,p);

System.out.println("Key1 calculated :"+k1);

BigInteger k2=R1.modPow(y,p);

System.out.println("Key2 calculated :"+k2);

System.out.println("deffie hellman secret key Encryption has Taken");

}

}

-----------------------------------------------------------------------------------------------------

TASK 9

import java.security.\*;

public class Task9 {

public static void main(String[] args) {

try {

MessageDigest md = MessageDigest.getInstance("SHA1");

System.out.println("Message digest object info:");

System.out.println("Algorithm = " + md.getAlgorithm());

System.out.println("ToString = " + md.toString());

// Test cases

String input;

// Case 1: Empty input

input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println("\nSHA1(\"" + input + "\") = " + bytesToHex(output));

// Case 2: "abc"

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println("\nSHA1(\"" + input + "\") = " + bytesToHex(output));

// Case 3: "abcdefghijklmnopqrstuvwxyz"

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println("\nSHA1(\"" + input + "\") = " + bytesToHex(output));

} catch (Exception e) {

System.out.println("Exception: " + e);

}

}

// Helper method to convert byte array to hex string

public static String bytesToHex(byte[] b) {

char hexDigit[] = {

'0', '1', '2', '3', '4', '5', '6', '7',

'8', '9', 'A', 'B', 'C', 'D', 'E', 'F'

};

StringBuffer buf = new StringBuffer();

for (int j = 0; j < b.length; j++) {

buf.append(hexDigit[(b[j] >> 4) & 0x0f]);

buf.append(hexDigit[b[j] & 0x0f]);

}

return buf.toString();

}

}

-----------------------------------------------------------------------------------------------------

TASK 10

import java.security.\*;

public class Task10 {

public static void main(String[] args) {

try {

// Create MD5 message digest instance

MessageDigest md = MessageDigest.getInstance("MD5");

// Print information about the MessageDigest object

System.out.println("Message digest object info:");

System.out.println("Algorithm = " + md.getAlgorithm());

System.out.println("ToString = " + md.toString());

// Test cases

String input;

// Case 1: Empty input

input = "";

md.update(input.getBytes());

byte[] output = md.digest();

System.out.println("\nMD5(\"" + input + "\") = " + bytesToHex(output));

// Case 2: "abc"

input = "abc";

md.update(input.getBytes());

output = md.digest();

System.out.println("\nMD5(\"" + input + "\") = " + bytesToHex(output));

// Case 3: "abcdefghijklmnopqrstuvwxyz"

input = "abcdefghijklmnopqrstuvwxyz";

md.update(input.getBytes());

output = md.digest();

System.out.println("\nMD5(\"" + input + "\") = " + bytesToHex(output));

} catch (Exception e) {

System.out.println("Exception: " + e);

}

}

// Helper method to convert a byte array to a hexadecimal string

public static String bytesToHex(byte[] b) {

char hexDigit[] = {

'0', '1', '2', '3', '4', '5', '6', '7',

'8', '9', 'A', 'B', 'C', 'D', 'E', 'F'

};

StringBuffer buf = new StringBuffer();

for (int j = 0; j < b.length; j++) {

buf.append(hexDigit[(b[j] >> 4) & 0x0f]);

buf.append(hexDigit[b[j] & 0x0f]);

}

return buf.toString();

}

}

-----------------------------------------------------------------------------------------------------

TASK 12

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.Signature;

import java.util.Base64;

public class Task12 {

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

// Generate a key pair

KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");

kpg.initialize(1024);

KeyPair keyPair = kpg.genKeyPair();

// Input data to sign

byte[] data = "Sample Text".getBytes("UTF8");

// Create and initialize the signature object for signing

Signature sig = Signature.getInstance("MD5WithRSA");

sig.initSign(keyPair.getPrivate());

sig.update(data);

// Generate the digital signature

byte[] signatureBytes = sig.sign();

System.out.println("Signature: \n" + Base64.getEncoder().encodeToString(signatureBytes));

// Verify the signature

sig.initVerify(keyPair.getPublic());

sig.update(data);

System.out.println("Signature verification result: " + sig.verify(signatureBytes));

}

}