|  |  |  |
| --- | --- | --- |
| Names: | Roll no: | Registration no: |
| Nihaal Panda | 1 | 150905097 |
| Pallabi Chandra | 11 | 160905011 |
| Jayashree K | 40 | 160905452 |
| Manjusha K Shetty | 60 | 160905459 |
| Abirami Byju | 31 | 160905097 |

OOP ASSIGNMENT 3

CSE- SEC A

**INDEX:**

1. Introduction
2. Cryptography
3. One time Pad
4. Explanation of code

* Ciphering
* Deciphering

1. Codes

- Ciphering

- Deciphering

6. Outputs

**INTRODUCTION:**

In this project we have decided to implement the One-Time pad using static and instance methods of java.

Static vs Instance :

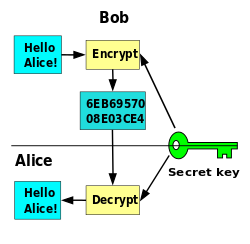
|  |  |
| --- | --- |
| Static | Instance |
| Static methods are the methods in Java that can be called without creating an object of class. They are referenced by the **class name itself** or reference to the Object of that class. | Instance method are methods which require an object of its class to be created before it can be called. To invoke a instance method, we have to create an Object of the class in within which it defined. |
| * Static method(s) are associated to the class in which they reside i.e. they can be called even without creating an instance of theclass ie, ClassName.methodName(args). * They are designed with aim to be shared among all Objects created from the same class. * Static methods can not be overridden. But can be overloaded since they are resolved using **static binding** by compiler at compile time. | * Instance method(s) belong to the Object of the class not to the class i.e. they can be called after creating the Object of the class. * Every individual Object created from the class has its own copy of the instance method(s) of that class. * They can be overridden since they are resolved using **dynamic binding** at run time. |
| * Static methods can access the static variables and static methods directly. * Static methods can’t access instance methods and instance variables directly. They must use reference to object. And static method can’t use [this](http://quiz.geeksforgeeks.org/this-reference-in-java/) keyword as there is no instance for ‘this’ to refer to. | * Instance method can access the instance methods and instance variables directly.   Instance method can access static variables and static methods directly. |
| * public void geek(String name) * { * // code to be executed.... * } * // Return type can be int, float String or user defined data type. | * public static void geek(String name) * { * // code to be executed.... * } * // Must have static modifier in their declaration. * // Return type can be int, float, String or user defined data type. |

**Cryptography**:

Human being from ages had two inherent needs − (a) to communicate and share information and (b) to communicate selectively. These two needs gave rise to the art of coding the messages in such a way that only the intended people could have access to the information. Unauthorized people could not extract any information, even if the scrambled messages fell in their hand.

*The art and science of concealing the messages to introduce secrecy in information security is recognized as cryptography.*

The word ‘cryptography’ was coined by combining two Greek words, ‘Krypto’ meaning hidden and ‘graphene’ meaning writing.



This is how real world public-key encryption is often done.

* Bob generates a key pair, consisting of his public key (red padlock) and private key (red key).
* Bob then publishes his public key, and Alice fetches it (Bob mails his padlock to Alice).
* Alice then generates a temporary symmetric key (the pair of orange keys) and uses Bob’s public key (red padlock) to securely send it to Bob.
* Bob then uses his private key (red key) to unlock his copy of the symmetric key (orange key).
* Bob and Alice can then use those symmetric keys to securely send messages back and forth.

**One-Time Pad**:

In [cryptography](https://en.wikipedia.org/wiki/Cryptography), the one-time pad (OTP) is an [encryption](https://en.wikipedia.org/wiki/Encryption) technique that cannot be [cracked](https://en.wikipedia.org/wiki/Cryptanalysis), but requires the use of a one-time pre-shared key the same size as, or longer than, the message being sent. In this technique, a [plaintext](https://en.wikipedia.org/wiki/Plaintext) is paired with a random secret [key](https://en.wikipedia.org/wiki/Key_(cryptography)) (also referred to as a one-time pad). Then, each bit or character of the plaintext is encrypted by combining it with the corresponding bit or character from the pad using modular addition. If the key is truly [random](https://en.wikipedia.org/wiki/Random), is at least as long as the plaintext, is never reused in whole or in part, and is kept completely [secret](https://en.wikipedia.org/wiki/Secret), then the resulting [cipher text](https://en.wikipedia.org/wiki/Ciphertext) will be impossible to decrypt or break. It has also been proven that any cipher with the perfect secrecy property must use keys with effectively the same requirements as OTP keys.

**Real-world uses of one-time pad:**

1.Military applications:

The [World War II](https://en.wikipedia.org/wiki/World_War_II) voice [scrambler](https://en.wikipedia.org/wiki/Scrambler) [SIGSALY](https://en.wikipedia.org/wiki/SIGSALY) was also a form of one-time system. It added noise to the signal at one end and removed it at the other end. The noise was distributed to the channel ends in the form of large shellac records which were manufactured in unique pairs. There were both starting synchronization and longer-term phase drift problems which arose and were solved before the system could be used. U.S. Army Special Forces used one-time pads in Vietnam. By using Morse code with one-time pads and continuous wave radio transmission (the carrier for Morse code), they achieved both secrecy and reliable communications.

2.Diplomatic communication:

The [Weimar Republic](https://en.wikipedia.org/wiki/Weimar_Republic) Diplomatic Service began using the method in about 1920. The breaking of poor [Soviet](https://en.wikipedia.org/wiki/Union_of_Soviet_Socialist_Republics) cryptography by the [British](https://en.wikipedia.org/wiki/United_Kingdom), with messages made public for political reasons in two instances in the 1920s, appear to have induced the U.S.S.R. to adopt one-time pads for some purposes by around 1930. [KGB](https://en.wikipedia.org/wiki/KGB) spies are also known to have used pencil and paper one-time pads more recently.

3.QKD applications:

The algorithm most commonly associated with [quantum key distribution](https://en.wikipedia.org/wiki/Quantum_key_distribution) is the one-time pad. QKD works on the principle that the quantum state of the information passed changes immediately when intercepted by a user. Thus if the information has been intercepted before the recipient receives it, by checking the quantum state the recipient knows that the message has been breached.

**EXPLANATION OF THE CODE IMPLEMENTED:**

To use this encryption method, we take a *plaintext* string, for example THISISMYSECRETCODE, and encrypt it using a *key*, for example, SUPERSECRETKEYKEEPMESAFE. For a one-time pad to work, we need the key to be longer than the plaintext.

| Letter | Number | Letter | Number |
| --- | --- | --- | --- |
| A | 0 | P | 15 |
| B | 1 | Q | 16 |
| C | 2 | R | 17 |
| D | 3 | S | 18 |
| E | 4 | T | 19 |
| F | 5 | U | 20 |
| G | 6 | V | 21 |
| H | 7 | W | 22 |
| I | 8 | X | 23 |
| J | 9 | Y | 24 |
| K | 10 | Z | 25 |
| L | 11 |  |  |
| M | 12 |  |  |
| N | 13 |  |  |
| O | 14 |  |  |

First we consider mapping letters to numbers:

This allows us to convert both our key and plaintext into a list of numbers. For example:

key= SUPERSECRETKEYKEEPMESAFE =18 20 15 4 17…

plaintext= THISISMYSECRETCODE = 19 7 8 18 8 …

To encipher, using the one-time pad, we add up the numbers:

37 27 23 22 25…

Take the modulo of each number around 26:

11 1 23 22 25…

and then convert these numbers back to letters:

ciphertext = lbxwz…

**Ciphering (explanation of the code):**

To implement the ciphering we have created a class ‘OneTimePadEncipher’.

**The skeleton for the code is below:**

public class OneTimePadEncipher {

public static int charToInt(char l) {

// ADD CODE HERE

//To convert a character to an integer, for example 'a' -> 0, 'b' -> 1

}

public static char intToChar(int i) {

// ADD CODE HERE

// Should convert an integer to a character, for example 0 -> 'a', b -> '1'

// it should always return lower case char

}

public static boolean isAlpha(char c) {

// You do not need to implement this method, but you may find it useful.

}

public static String encipher(String original, String onetimepad) {

// ADD CODE HERE

}

public static void main(String[] args) {

String ciphertext = encipher(“HELLO EVERYBODY”, “MYSECRETKETMYSECRETKEY”);

System.out.print(ciphertext);

}

}

We have used the convention that key and plaintext should be in uppercase, while ciphertexts should be lowercase for testing and we have used the String method toUpperCase() and toLowerCase() to convert a string to upper or lowercase. The key will not contain spaces, but the plaintext and the ciphertext may.  Note that this method returns a new string, so it must be assigned. E.g., mystring = mystring.toLowerCase;.

To extract a char from a String at a certain index, we have used the String.charAt() method on the String object. For example, char c = "hello".charAt(2);, will set c=l.

If everything is successful, one should be able to encrypt the following:

| **Plaintext** | **Key** | **Ciphertext** |
| --- | --- | --- |
| HELLO | XMCKL | eqnvz |
| SUPERSECRETMESSAGE | MYSUPERDUPERSECRETCKEY | eshygwvfltxdwwurkx |
| IS THIS SECURE | KEEPMEVERYVERYSAFE | sw itmn jcxyic |

**DECIPHERING:**

Created a new class OneTimePadDecipher and implemented a method to decrypt *ciphertext* given a key. To decrypt a code, you need to **subtract** the key from the ciphertext representation, instead of adding it.

The signature of the new method is as follows:

public static String decipher(String encipheredText, String onetimepad)

If everything is successful, you should be able to decrypt the following:

| **Ciphertext** | **Key** |
| --- | --- |
| uvufsghktdal | WHATSTHEPASSWORD |
| wconlahzgzgleuai | YOULLNEVERREADMYONETIMEPAD |

**CODE FOR CIPHERING:**

import java.util.Scanner;

public class OneTimePadEncipher {

public static int charToInt(char l) {

int c = Character.toLowerCase(l) - 'a';

return c;

}

public static char intToChar(int i) {

return (char) (((i + 26) % 26) + 'a');

}

public static boolean isAlpha(char c) {

int t = charToInt(c);

if (t >= 0 && t < 26) {

return true;

}

return false;

}

public static String encipher(String plaintext, String onetimepad) {

try{

if (plaintext.length() > onetimepad.length())

throw new Exception();

}

catch(Exception ob){

System.out.println("Exception occured");

System.out.println(ob);

System.out.println("The length of the OTP is too short");

System.exit(0);

}

String lcPlaintext = plaintext.toLowerCase();

String lcOnetimepad = onetimepad.toLowerCase();

String newStr = "";

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

char o = lcPlaintext.charAt(i);

char k = lcOnetimepad.charAt(i);

if (isAlpha(o)) {

newStr += intToChar(charToInt(o) + charToInt(k));

}

else {

newStr += o;

}}

return newStr;

}

public static void main(String args[]){

System.out.println("enter the message to be encrypted");

String s;

Scanner in=new Scanner(System.in);

s=in.next();

System.out.println("enter the key");

String key;

key=in.next();

String ciphtext=OneTimePadEncipher.encipher(s,key);

System.out.println("the cipher text is");

System.out.println(ciphtext);

}}

**CODE FOR DECIPHERING:**

import java.util.Scanner;

public class OneTimePadDecipher {

public static String decipher(String ciphertext, String onetimepad) {

try{

if (ciphertext.length() > onetimepad.length())

throw new Exception();

}

catch(Exception ob){

System.out.println("Exception occured");

System.out.println(ob);

System.out.println("The length of the OTP is too short");

System.exit(0);

}

String lcCiphertext = ciphertext.toLowerCase();

String lcOnetimepad = onetimepad.toLowerCase()

String newStr = " ";

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

char o = lcCiphertext.charAt(i);

char k = lcOnetimepad.charAt(i);

if (OneTimePadEncipher.isAlpha(o)) {

newStr += OneTimePadEncipher.intToChar(OneTimePadEncipher.charToInt(o) - OneTimePadEncipher.charToInt(k));

}

else {

newStr += o;

}

}

return newStr.toUpperCase();

}

public static void main(String args[])

{

System.out.println("Decryption process:");

System.out.println("enter the encrypted message");

String ciphtext;

Scanner in=new Scanner(System.in);

ciphtext =in.next();

System.out.println("enter the key");

String key;

key=in.next();

String plaintext=OneTimePadDecipher.decipher(ciphtext,key);

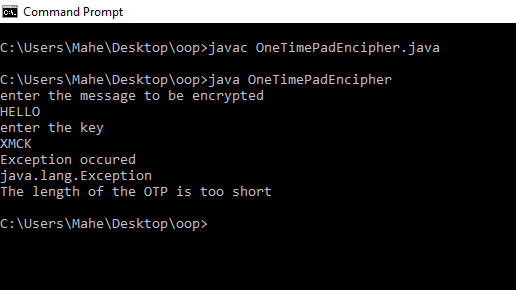
System.out.println("the plaintext after decoding");

System.out.println(plaintext);

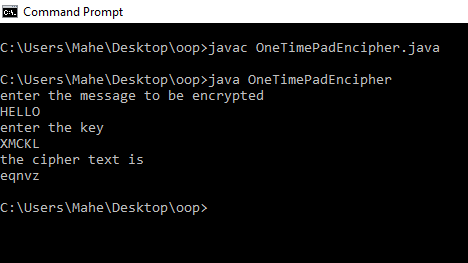
}}

**OUTPUT:**

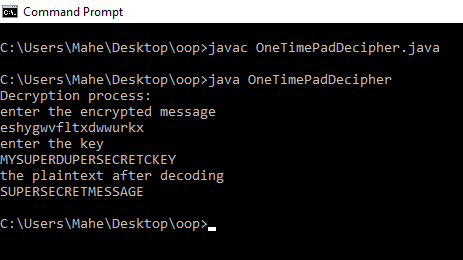
**Case 1 (exception):**

****

**Case2 (ciphering):**

****

**Case3(Deciphering):**

****