

# Overview:

The objective of this lab was to develop an implementation of the One-Time Pad (OTP) cipher using Java. The task was to create two functions: Encrypt(plaintext, key) and Decrypt(ciphertext, key). These functions should respectively encrypt and decrypt the input message using a randomly generated key that is at least as long as the plaintext.

## Analysis:

**Implementation:** For this lab, I implemented the One-Time Pad cipher in Java. The encryption and decryption methods use simple character shifts based on the positions of letters in the alphabet. The key is randomly generated using Java's Random class, and it must be of the same length as the plaintext for the encryption and decryption to work properly.

The basic process I followed for both encryption and decryption was as follows:

1. **Key Generation:** I created a random key using `Random.nextInt(26)`, which ensures the key consists of uppercase letters from 'A' to 'Z'.
2. **Encryption:** The encryption is performed by shifting each character of the plaintext by the corresponding character in the key. The shift is computed using modular arithmetic (mod 26) to keep the result within the alphabet's range.
3. **Decryption:** The decryption process works similarly but in reverse. Each character of the ciphertext is shifted back using the corresponding character from the key to retrieve the original plaintext.

The encryption and decryption functions were implemented as follows:

```
public static String encrypt(String plaintext, String key) {  
    String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
    String ciph = "";  
    for (int i = 0; i < plaintext.length(); i++) {  
        char plain = plaintext.charAt(i);  
        char k = key.charAt(i);
```

```
int pval = chars.indexOf(plain);
```

```
int kval = chars.indexOf(k);
```

```
int comb = (pval + kval) % 26;
```

```
ciph = ciph + chars.charAt(comb);
```

```
}
```

```
return ciph;
```

```
}
```

```
public static String decrypt(String ciph, String key) {
```

```
String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
String plain = "";
```

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

```
char cipher = ciph.charAt(i);
```

```
char k = key.charAt(i);
```

```
int cval = chars.indexOf(cipher);
```

```
int kval = chars.indexOf(k);
```

```
int comb = (cval - kval) % 26;
```

```
comb = (comb < 0) ? comb + 26 : comb;
```

```
plain = plain + chars.charAt(comb);
```

```
}
```

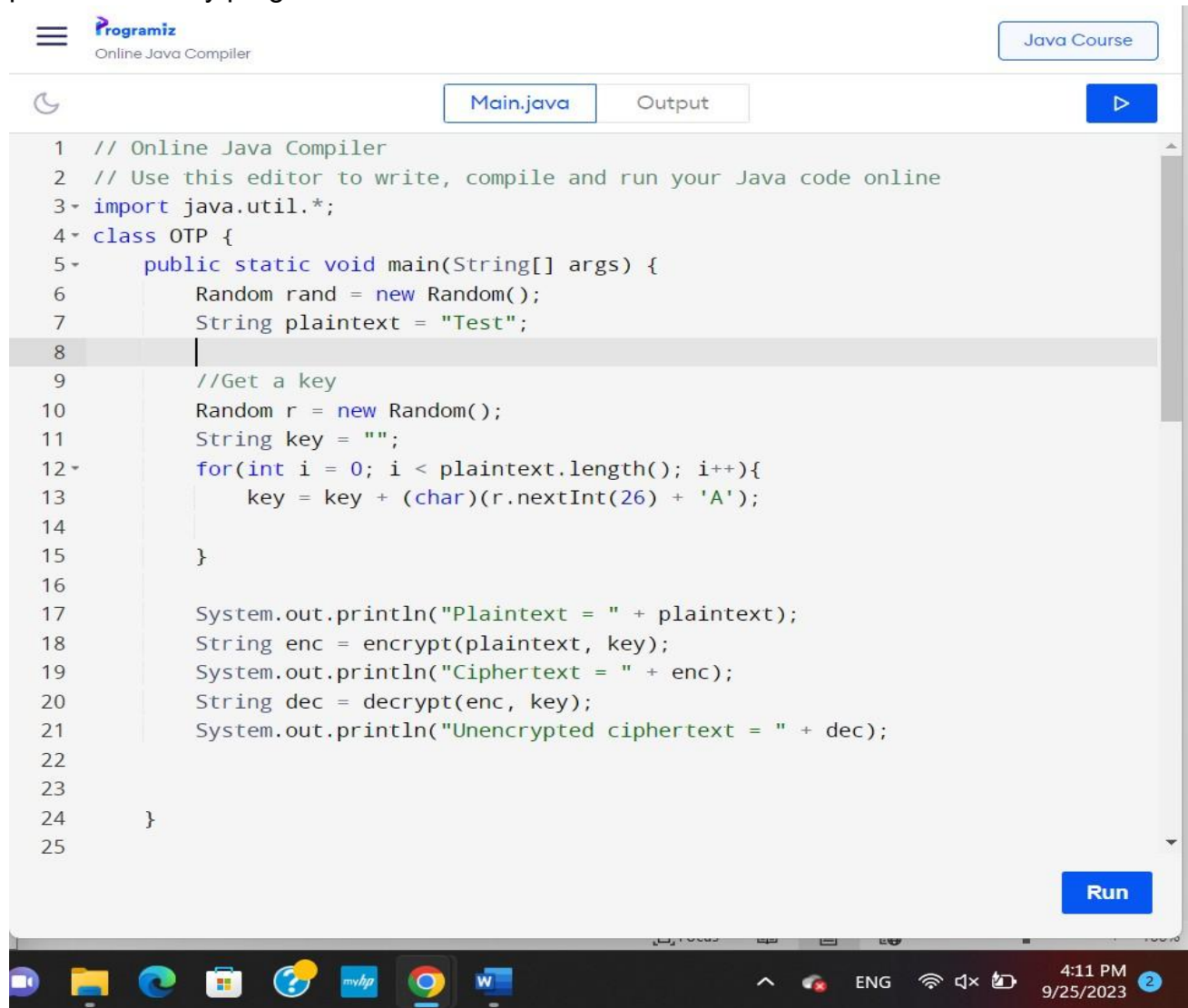
```
return plain;
```

```
}
```

Once I implemented the functions, I tested them by encrypting and decrypting a sample plaintext message. For example, I used the plaintext "TEST" and ran it through both the encryption and decryption processes.

- **Encryption:** The Encrypt() function successfully encrypted the plaintext "TEST" into a ciphertext, and the output was printed in the console.
- **Decryption:** The Decrypt() function was able to take the ciphertext and correctly recover the original plaintext message.

**Screenshots:** Below are screenshots showing the demo of the encryption and decryption processes in my program.



The screenshot displays the Programiz Online Java Compiler interface. At the top, there is a logo for Programiz and the text "Online Java Compiler". On the right, there is a button labeled "Java Course". Below the header, there is a tab labeled "Main.java" and a button with a play icon. The main area contains a Java code editor with the following code:

```
1 // Online Java Compiler
2 // Use this editor to write, compile and run your Java code online
3 import java.util.*;
4 class OTP {
5     public static void main(String[] args) {
6         Random rand = new Random();
7         String plaintext = "Test";
8
9         //Get a key
10        Random r = new Random();
11        String key = "";
12        for(int i = 0; i < plaintext.length(); i++){
13            key = key + (char)(r.nextInt(26) + 'A');
14        }
15
16        System.out.println("Plaintext = " + plaintext);
17        String enc = encrypt(plaintext, key);
18        System.out.println("Ciphertext = " + enc);
19        String dec = decrypt(enc, key);
20        System.out.println("Unencrypted ciphertext = " + dec);
21
22    }
23
24 }
25
```

At the bottom right of the code editor, there is a blue button labeled "Run". The bottom of the image shows a Windows taskbar with various application icons and a system clock displaying "4:11 PM 9/25/2023".



Main.java

Output



```
26 public static String encrypt(String plaintext, String key) {
27     String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
28     String ciph = "";
29     for(int i = 0; i < plaintext.length(); i++){
30         char plain = plaintext.charAt(i);
31         char k = key.charAt(i);
32
33         int pval = chars.indexOf(plain);
34         int kval = chars.indexOf(k);
35
36         int comb = (pval + kval) % 26;
37
38         ciph = ciph + chars.charAt(comb);
39     }
40
41     return ciph;
42 }
43
44 public static String decrypt(String ciph, String key){
45
46     String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
47     String plain = "";
48     for(int i = 0; i < ciph.length(); i++){
49         char cipher = ciph.charAt(i);
50         char k = key.charAt(i);
```

Run



ENG



4:11 PM  
9/25/2023





Main.java

Output



```
39     }
40
41     return ciph;
42 }
43
44 public static String decrypt(String ciph, String key){
45
46     String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
47     String plain = "";
48     for(int i = 0; i < ciph.length(); i++){
49         char cipher = ciph.charAt(i);
50         char k = key.charAt(i);
51
52         int cval = chars.indexOf(cipher);
53         int kval = chars.indexOf(k);
54
55         int comb = (cval - kval) % 26;
56         comb = (comb < 0)? comb + 26 : comb;
57
58         plain = plain + chars.charAt(comb);
59     }
60     return plain;
61 }
62 }
63 }
```

Run



ENG

4:11 PM  
9/25/2023

2



Main.java

Output



```
java -cp /tmp/qPEYhhu0aU OTP  
Plaintext = Test  
Ciphertext = YDXD  
Unencrypted ciphertext = Test
```



ENG



4:09 PM  
9/25/2023

2