Palestine Technical University (Kadoorie)

Faculty of Engineering and Technology

Department of Computer Systems Engineering

Cryptosystem and Network Security

- Triple S-DES System -

Prepared By:

Ahmad Haj-Qasem - 201810048

Mays Najjar - 201811598

Sondos Jarrar -201811910

Supervised by:

Dr. Anas Melhem

Tulkarm – Palestine

April.18th. 2022

Task definition :

In this project we have to develop a network application to test the 'Triple S-DESA'.
for sure we need to implement this project to allow the encryption and decryption modes. So communicating parts must be represented as running on two separate computers connected via some channel.

Definition of an algorithm and mode of operation to be implemented:

Before starting understanding the code, let's check how the 'S-DES' works from the figure below :

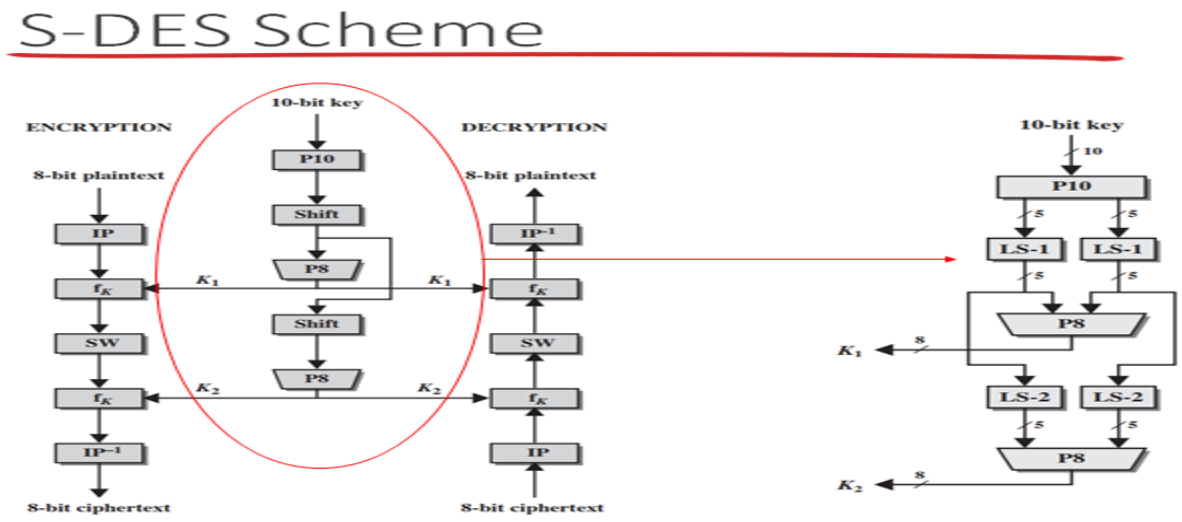


Figure 1: S-DES encryption and decryption modes

From the figure: 1 we can see the operations that we must be done before having the cipher text during the encryption, or the plain text in the decryption in S-DES . So first the 10-bits key enter P10 to do the permutation in a specific way and after that it enter a shift-1 operator which from this stage a copy from the key go to another shift-2 and P8 to wait for the key 1 be ready and enter together to the final function.
After the first shift for the key and to shrink it, it enters to P8 and becomes a 8-bits key, from this stage we can have our first key which enter to a specific function and then there's a swap happens between its bits and enters the function again with the second key.
The last step is to do IP^{-1} which is the inverse of the initial permutation and after this stage we have our cipher text.

But what we have to do is to implement the Triple S-DES, so we need to repeat some operations from the normal S-DES to do It in a more complex way.

Figure 2 shows the number of bits for the key and the plain text after the encryption for S-DES in a simple way :

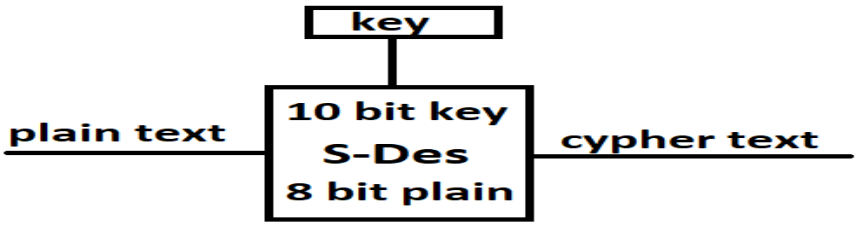


Figure 2: S-DES encryption

In the figure below we can notice how the Triple S-DES works in both, encryption and decryption modes .

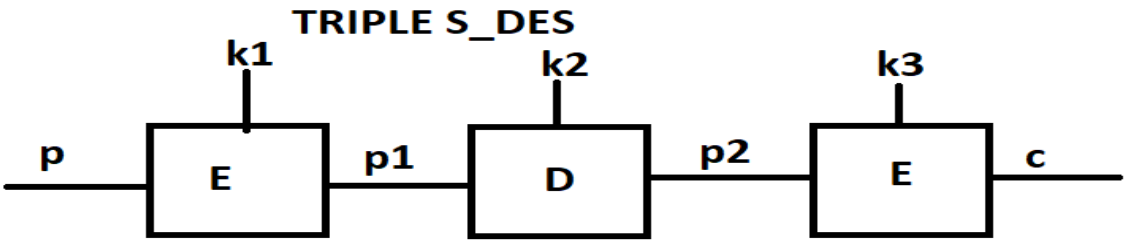


Figure 3: Triple S-DES encryption

As we can see from the figure above there's a special operation to do to have our cipher text which is : **Triple S-DES: 3S-DES(x)= E(K3,(D(K2 (E(K1,x)))))**
We can notice that we need 3 keys with 56 bits for each, in order to have our cipher text and the same thing for the decryption operation to have the plain text as we can see from the figure 4 below.

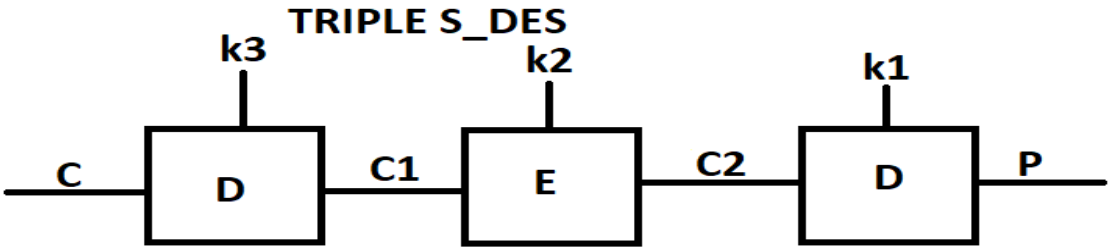


Figure 4: Triple S-DES decryption

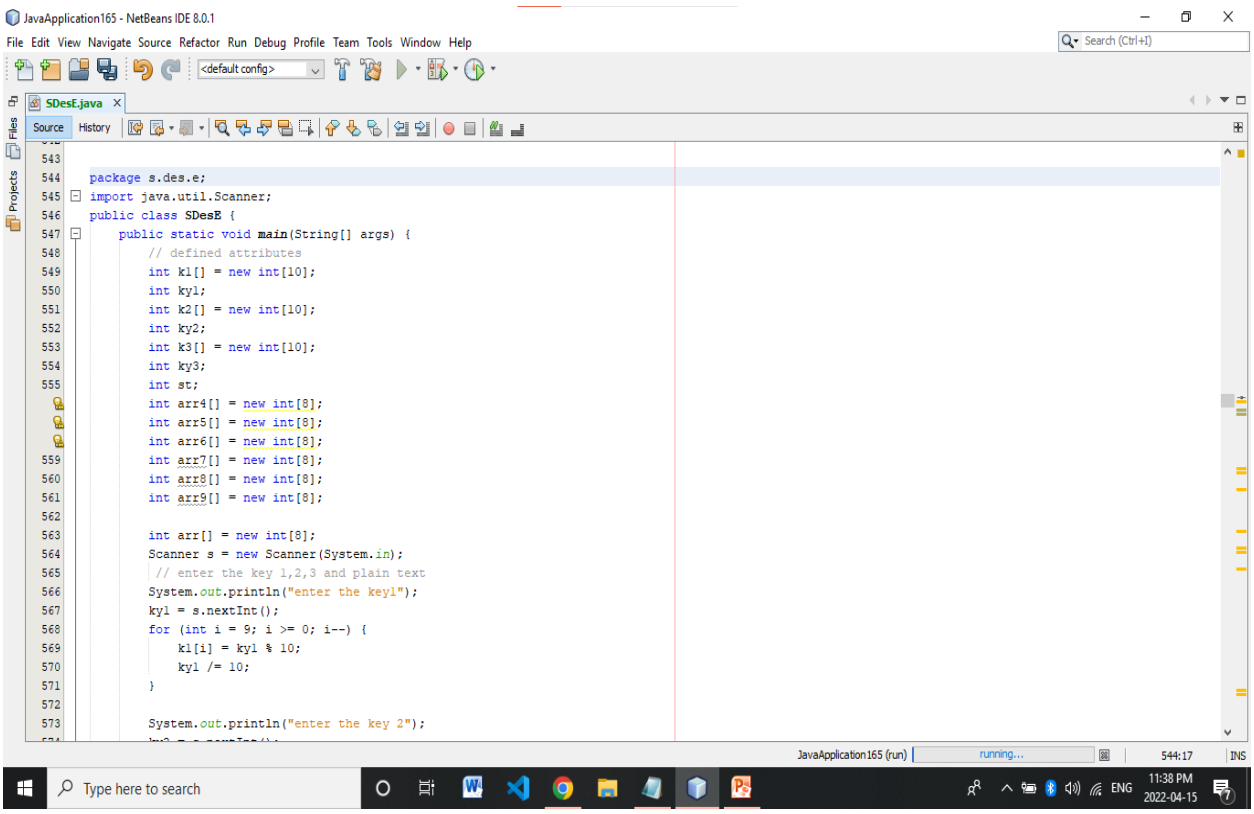
After all those steps all what we need is to make them work in two devices using a network with some codes to make it work and clear which we will see in some screenshots after a while in this project.

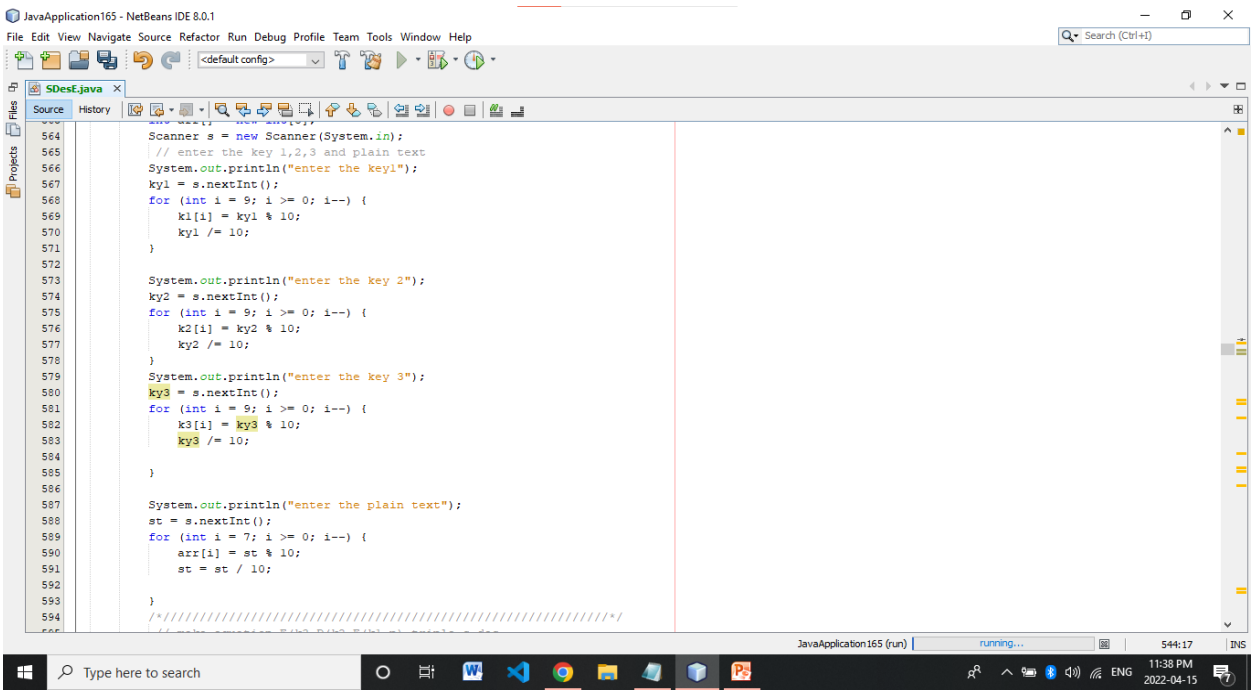
Description of developed application and its source codes:

Well, about the code , we wrote it from a to z ourselves without any source from the internet, and now we're gonna explain everything about the code and provide some screenshots to make it clear.

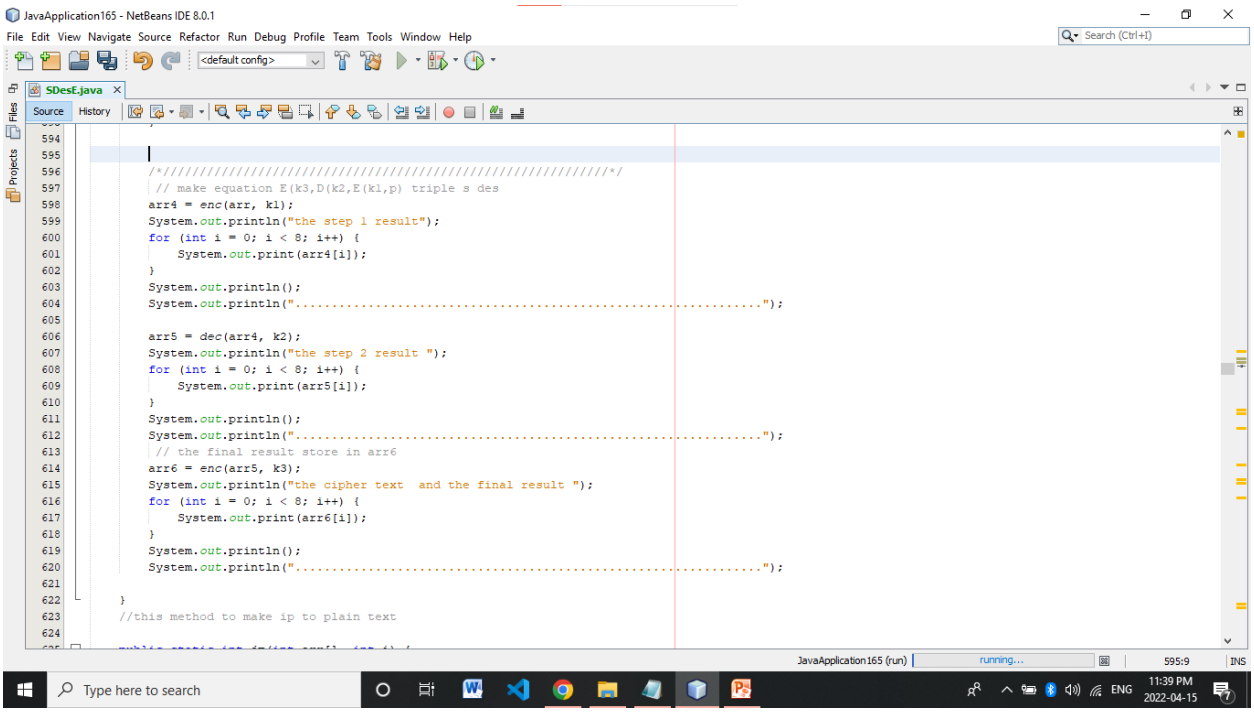
First of all let's start with the encryption operation to do **E(k3,D(k2,E(k1,p) .**

It starts with entering 8-bits plain text with 3 keys those we use in a different stages to do the triple S-Des .



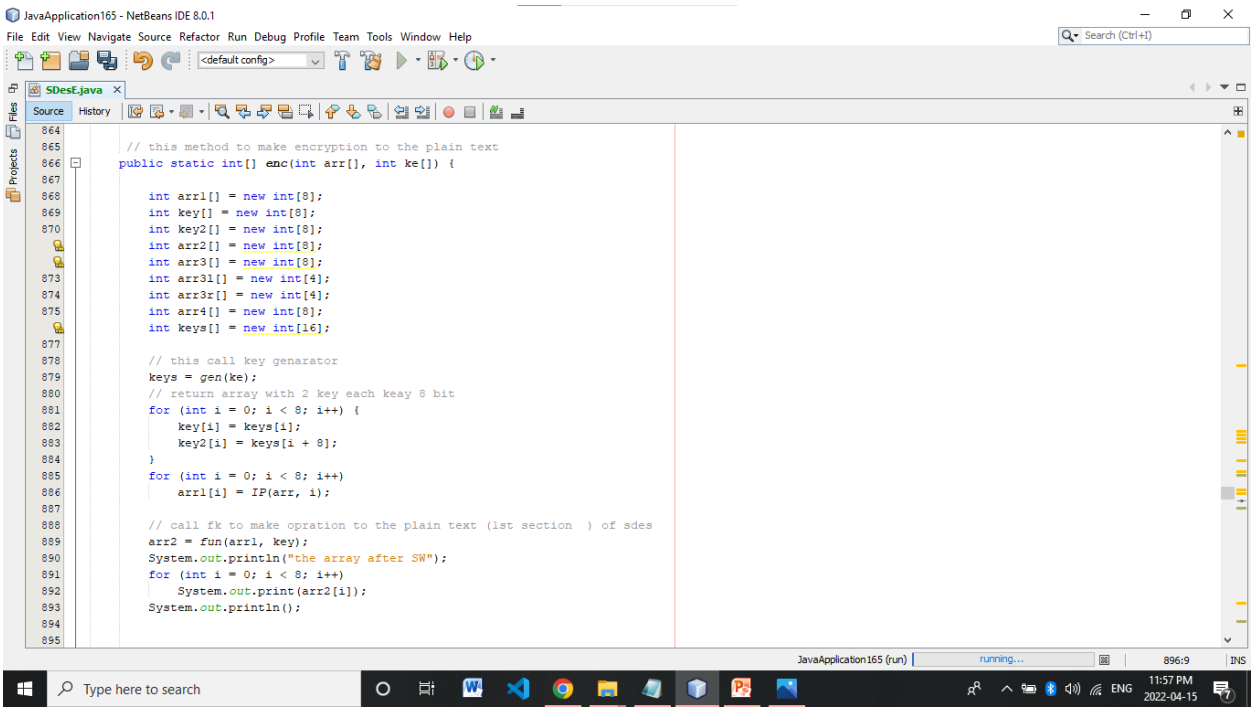


The first stage is to encrypt the key1 with the plain text , then the next one is to decrypt the k2 with the output from the first stage , and then the last one is to encrypt the k3 with the output from the second stage as follow .

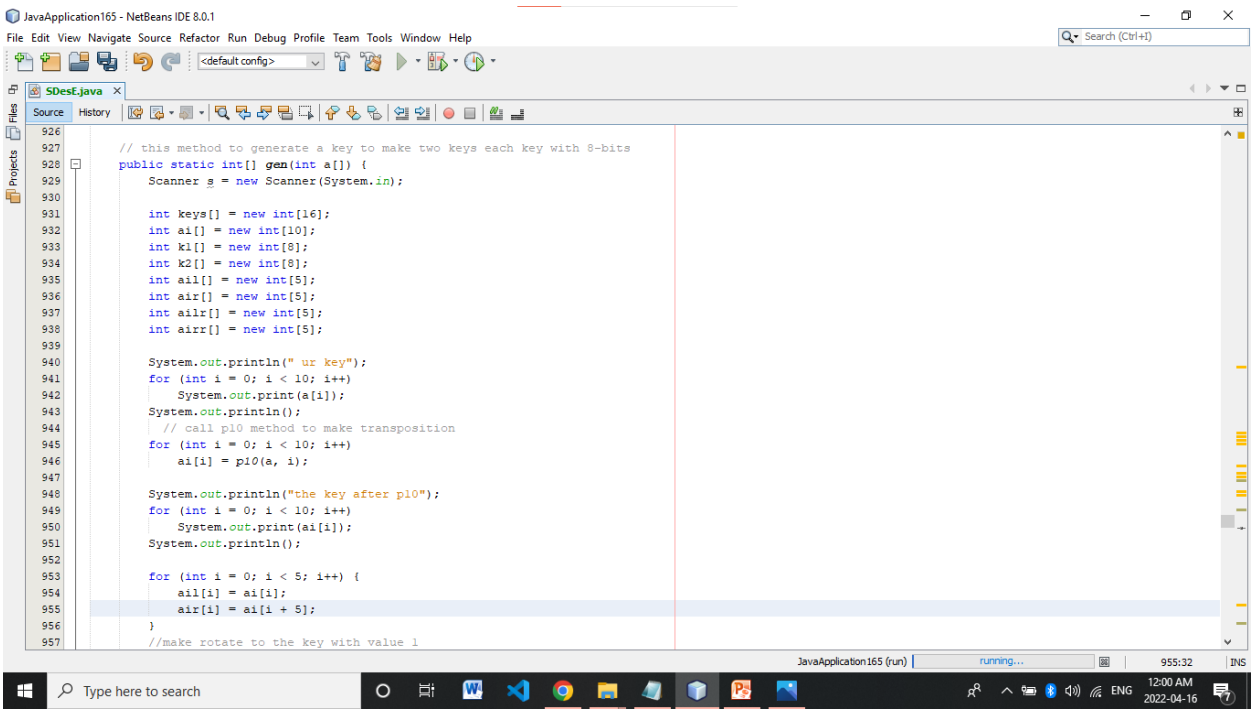


But we must know how each works, so let's explain every stage , the encryption and the decryption Proseccos.

First let’s see how the encryption will work :

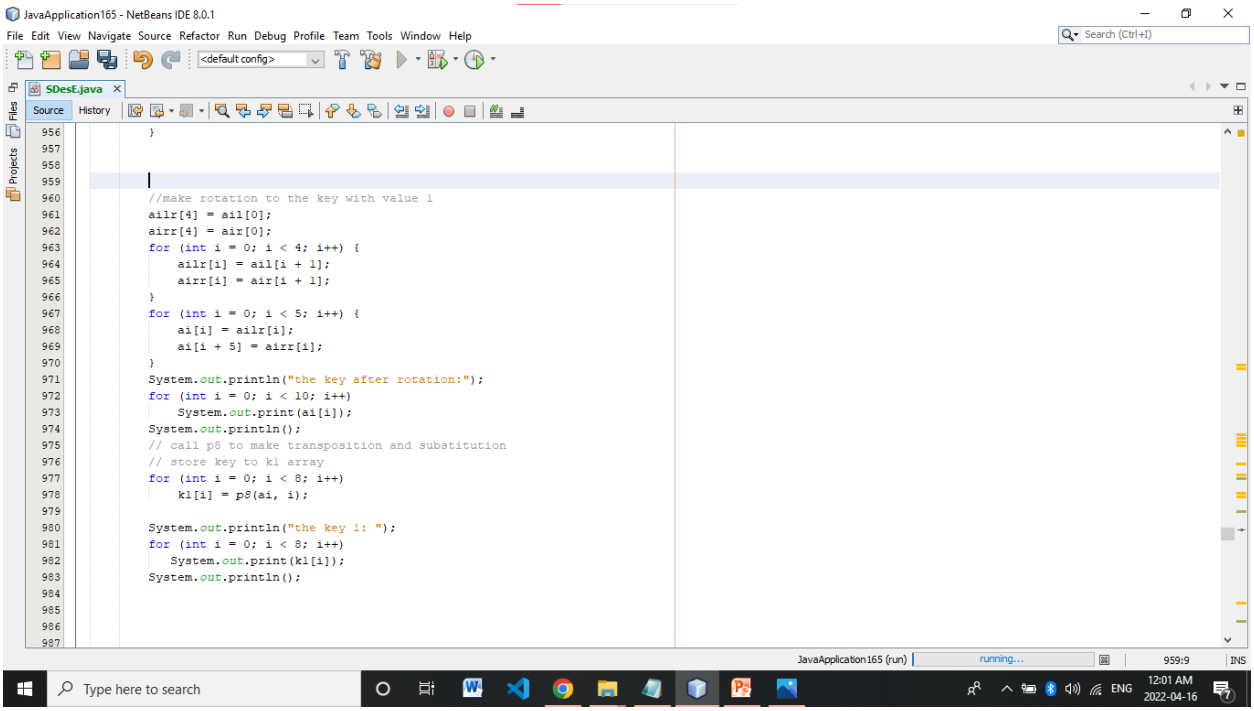


From the figure above we can see that the process starts with generating the keys (2) , but before continuing let’s check the code of generating the 2 keys :



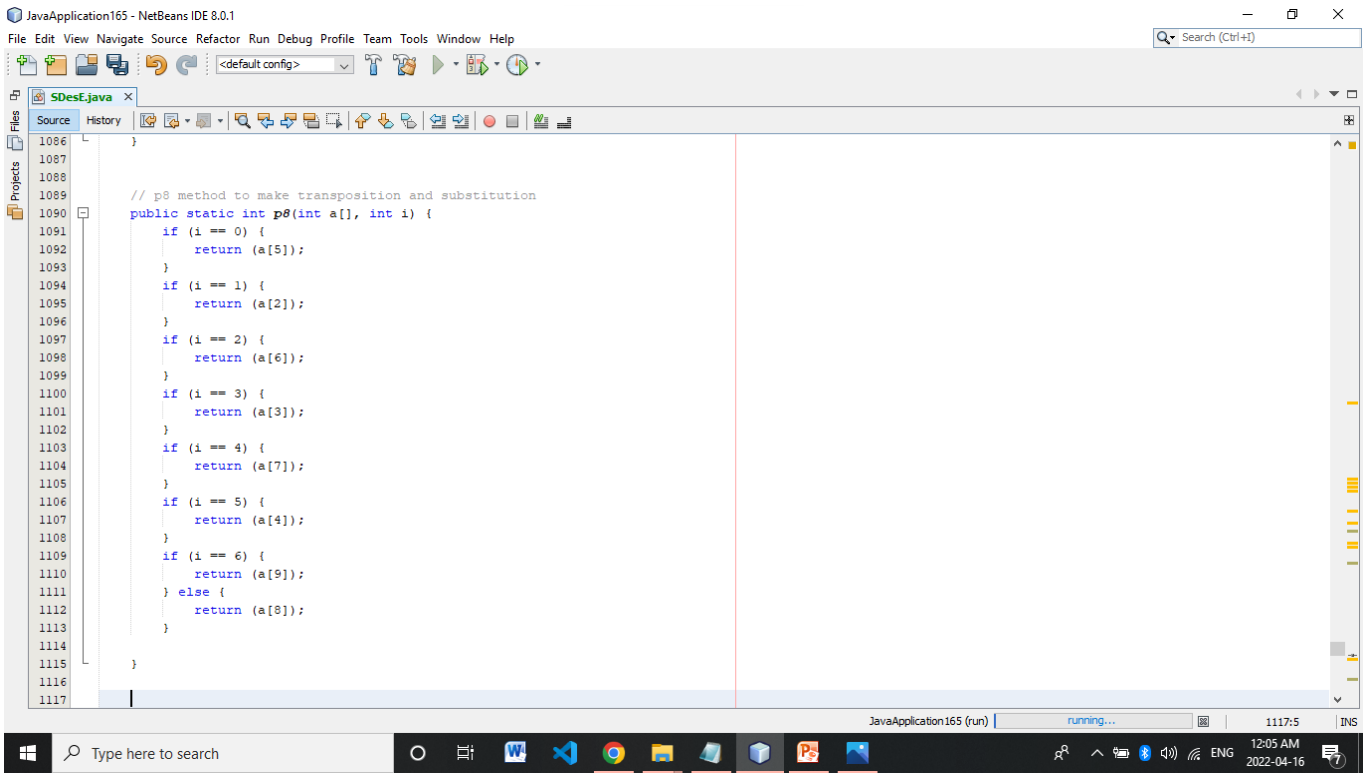
First a 10-bits key enters a P10 to make some permutation on it and then the output key from the p10 divided into 2 parts , left and right , as in the code above .

After that the two parts of the key rotate with 1-bit to the left then to the P8 which we will see its code soon, then after P8 , key 1 is ready ! which we can see in the code bellow.



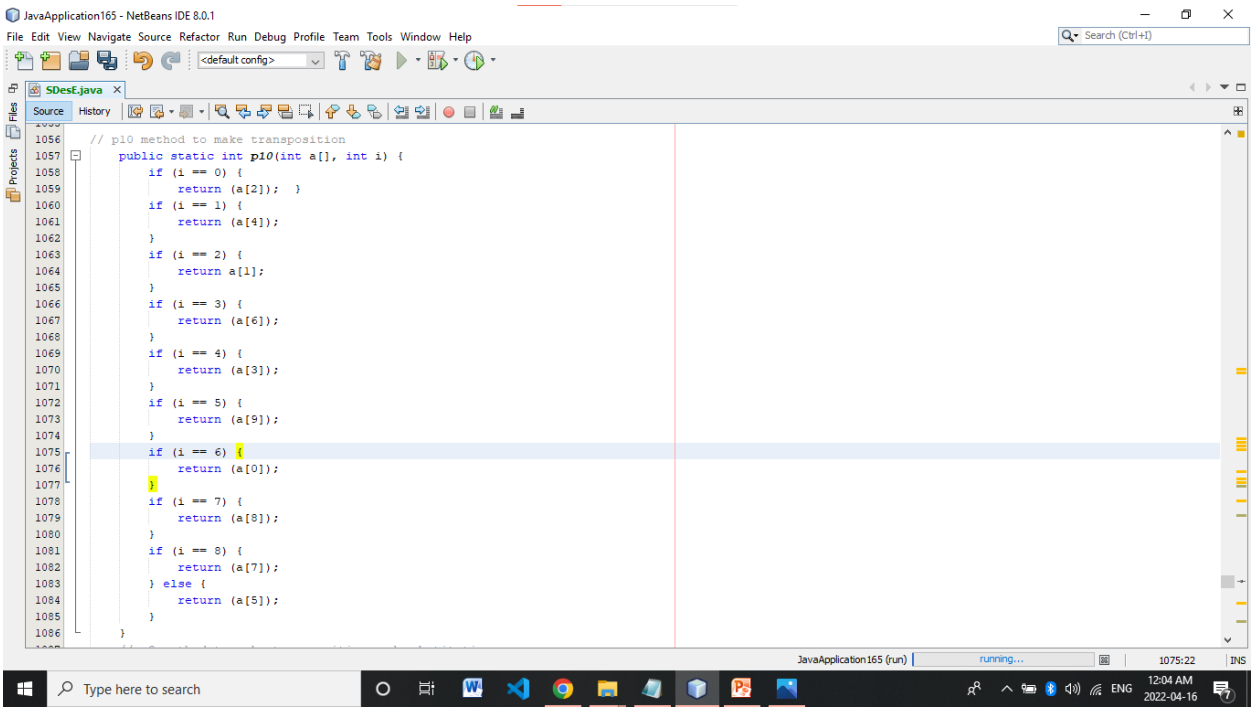
```
        }


        //make rotation to the key with value 1
        ailr[4] = ail[0];
        airr[4] = air[0];
        for (int i = 0; i < 4; i++) {
            ailr[i] = ail[i + 1];
            airr[i] = air[i + 1];
        }
        for (int i = 0; i < 5; i++) {
            ai[i] = ailr[i];
            ai[i + 5] = airr[i];
        }
        System.out.println("the key after rotation:");
        for (int i = 0; i < 10; i++)
            System.out.print(ai[i]);
        System.out.println();
        // call p8 to make transposition and substitution
        // store key to k1 array
        for (int i = 0; i < 8; i++)
            k1[i] = p8(ai, i);

        System.out.println("the key 1: ");
        for (int i = 0; i < 8; i++)
           System.out.print(k1[i]);
        System.out.println();
```

The P8 code to do permutation :

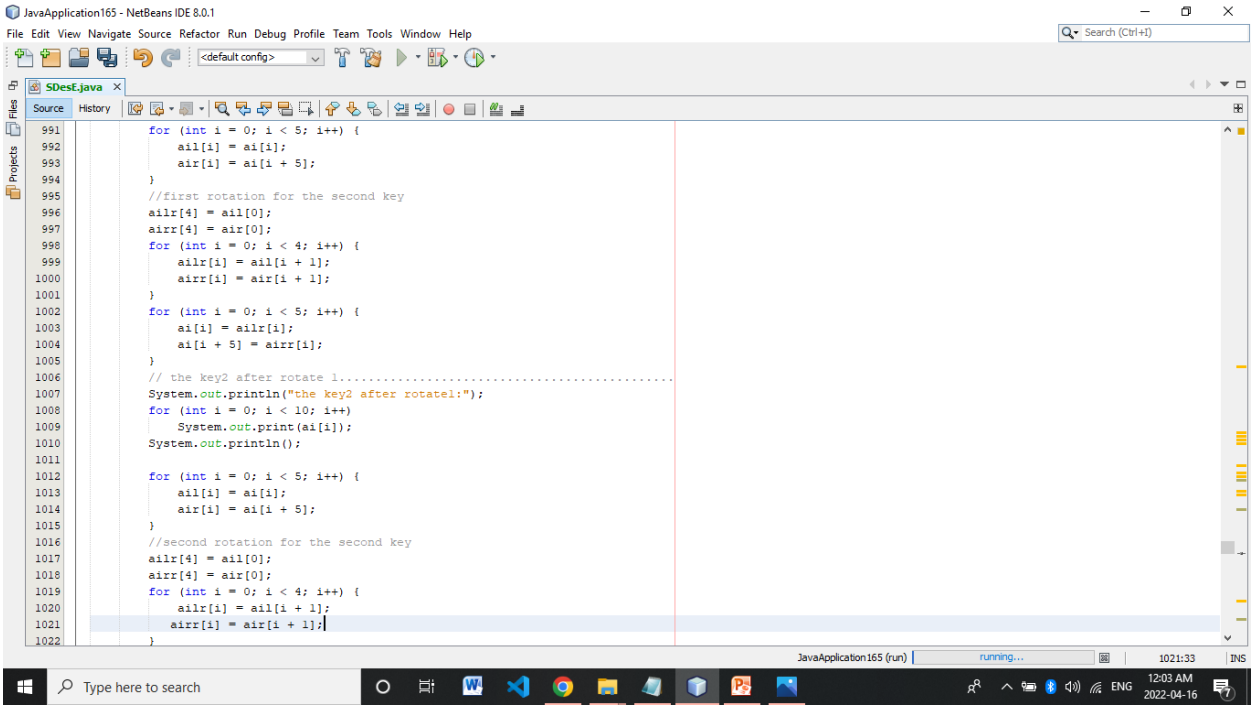


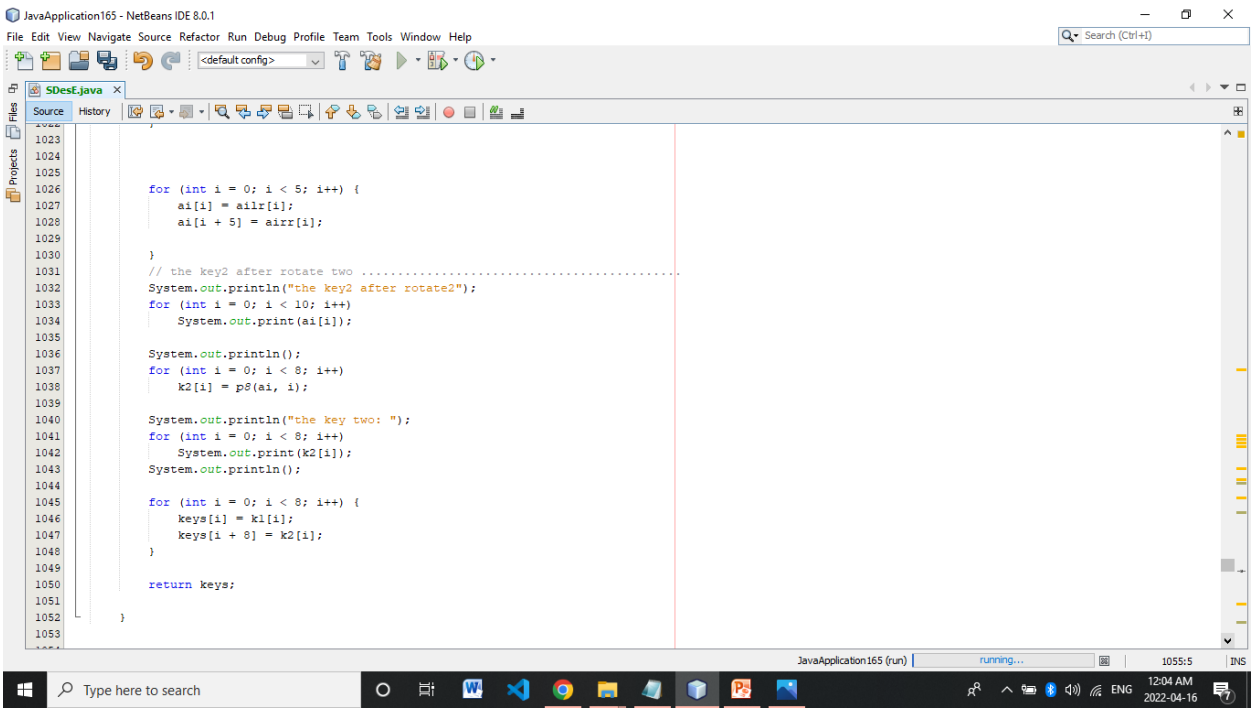
```
    }


    // p8 method to make transposition and substitution
    public static int p8(int a[], int i) {
        if (i == 0) {
            return (a[5]);
        }
        if (i == 1) {
            return (a[2]);
        }
        if (i == 2) {
            return (a[6]);
        }
        if (i == 3) {
            return (a[3]);
        }
        if (i == 4) {
            return (a[7]);
        }
        if (i == 5) {
            return (a[4]);
        }
        if (i == 6) {
            return (a[9]);
        } else {
            return (a[8]);
        }

    }
```

The P10 code which help in the first step with the origin key :



After that and to generate the key2 all what we need is to do another 2bits rotate to the left of the two parts that we did a 1-bit rotation in the beginning and then to enter them to the P8 as in the code bellow :

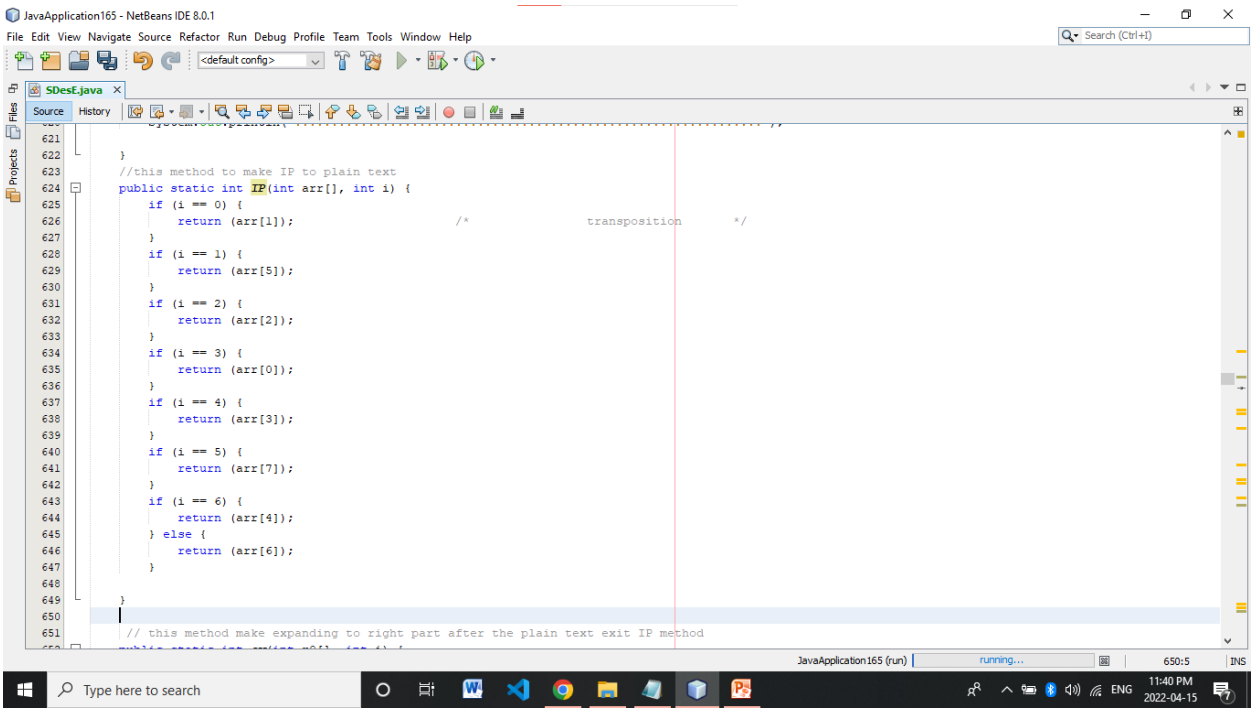




```
        for (int i = 0; i < 5; i++) {
            ai[i] = ailr[i];
            ai[i + 5] = airr[i];

        }
        // the key2 after rotate two .....
        System.out.println("the key2 after rotate2");
        for (int i = 0; i < 10; i++)
            System.out.print(ai[i]);

        System.out.println();
        for (int i = 0; i < 8; i++)
            k2[i] = p8(ai, i);

        System.out.println("the key two: ");
        for (int i = 0; i < 8; i++)
            System.out.print(k2[i]);
        System.out.println();

        for (int i = 0; i < 8; i++) {
            keys[i] = k1[i];
            keys[i + 8] = k2[i];
        }

        return keys;

    }
```

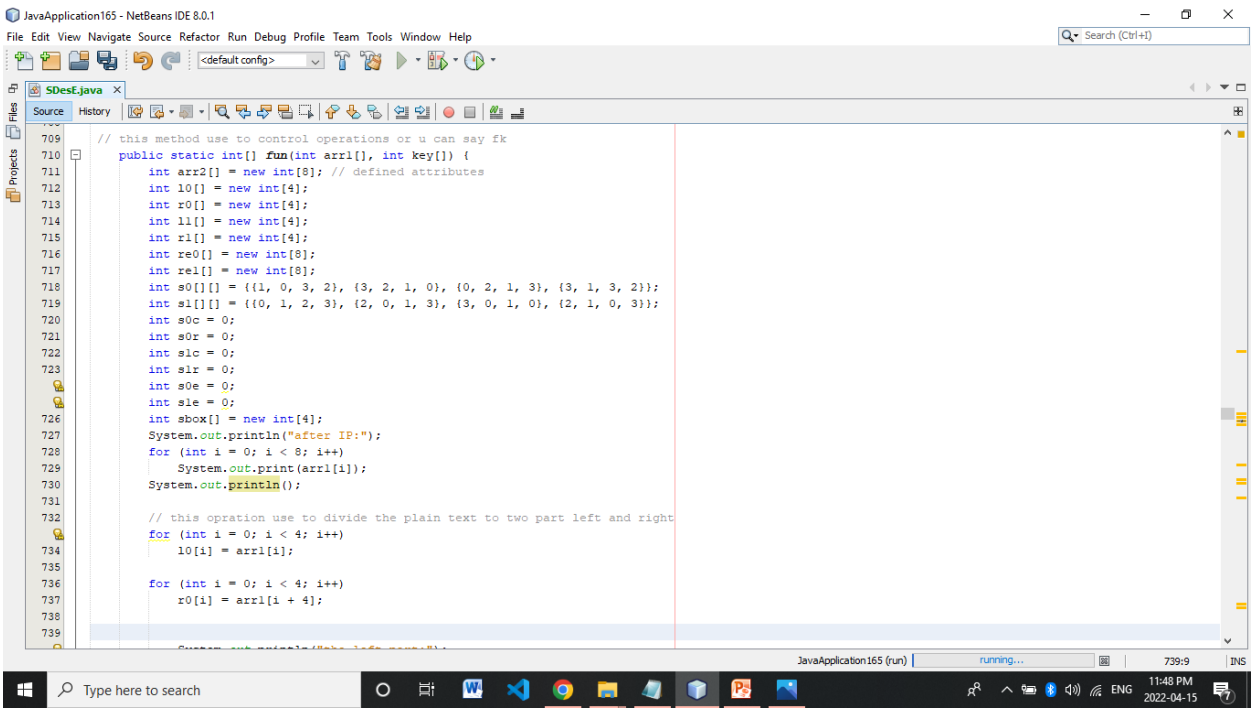
Now let's continue with our encryption process :

The 8-bit plain text enters the stage then it enters –IP- which is the initial permutation so let's see its code :



```
    }
    //this method to make IP to plain text
    public static int IP(int arr[], int i) {
        if (i == 0) {
            return (arr[1]);                    /*          transposition       */
        }
        if (i == 1) {
            return (arr[5]);
        }
        if (i == 2) {
            return (arr[2]);
        }
        if (i == 3) {
            return (arr[0]);
        }
        if (i == 4) {
            return (arr[3]);
        }
        if (i == 5) {
            return (arr[7]);
        }
        if (i == 6) {
            return (arr[4]);
        } else {
            return (arr[6]);
        }

    }

    // this method make expanding to right part after the plain text exit IP method
```

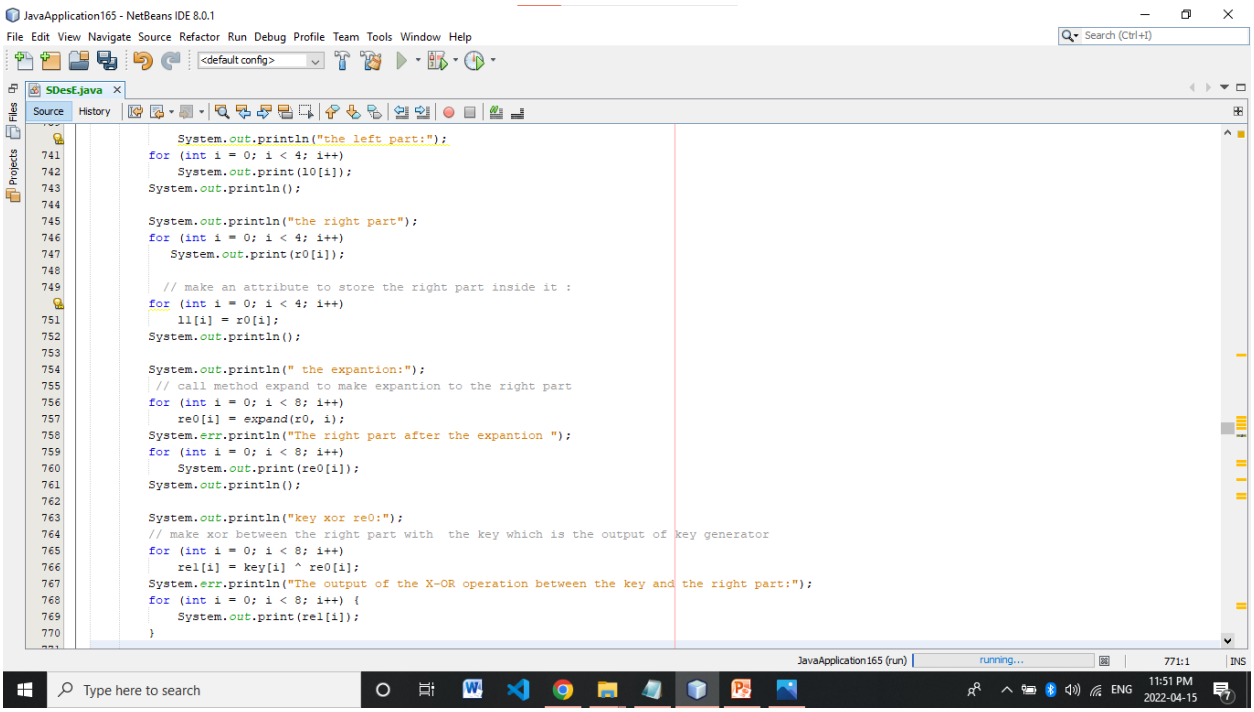
The output from the IP goes to fk which is kindda function do some complex processes we will see from the code :



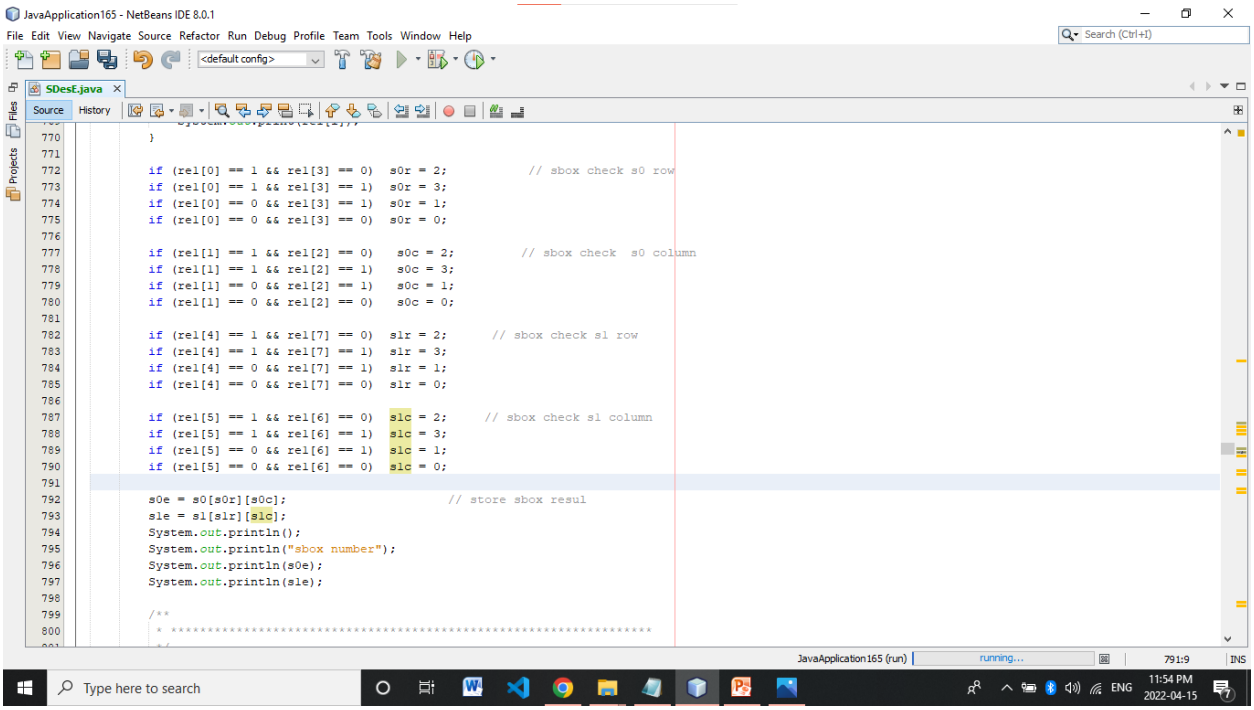
This function receive key1 and the output from the IP, it divides it to 2 parts with 4bits for each , left and right as in the code above.

The right part enters to an expanding process to make it bits which we will show its code,

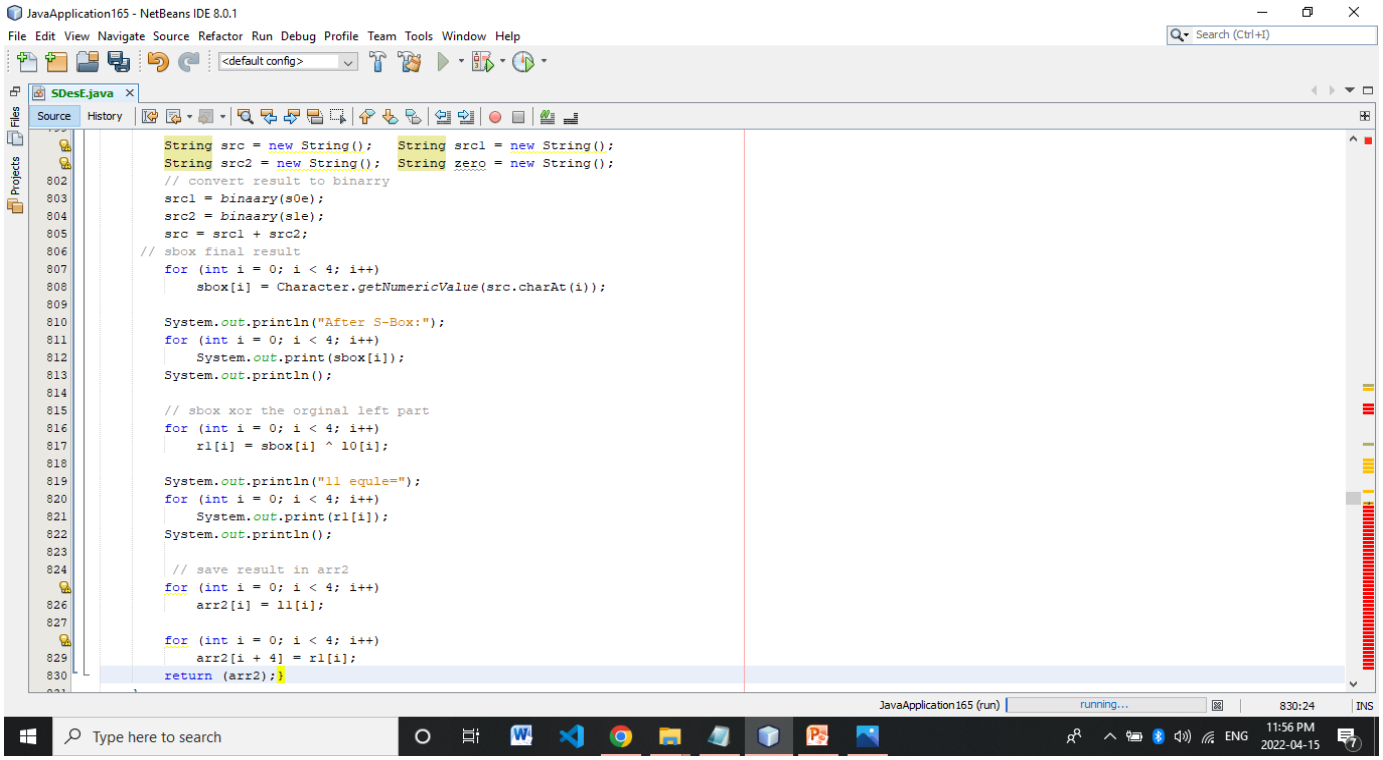
then the output from the expanding will enter with the key1 in an X-OR process as in the code ,

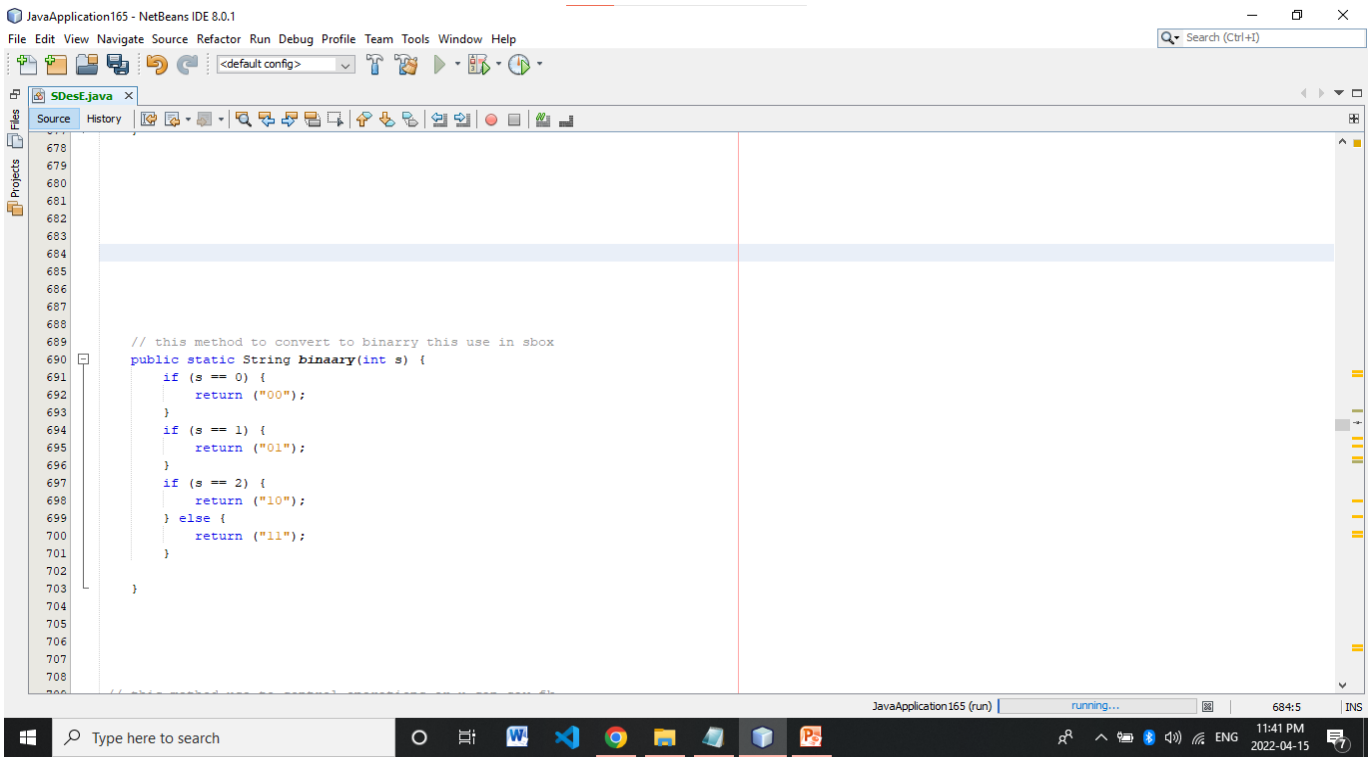


Each part from the output enter to a process called S-Box to do a specific operation as in the code but the ouptput from the S-Box will be 2-bits instead of 4-bits as the input .



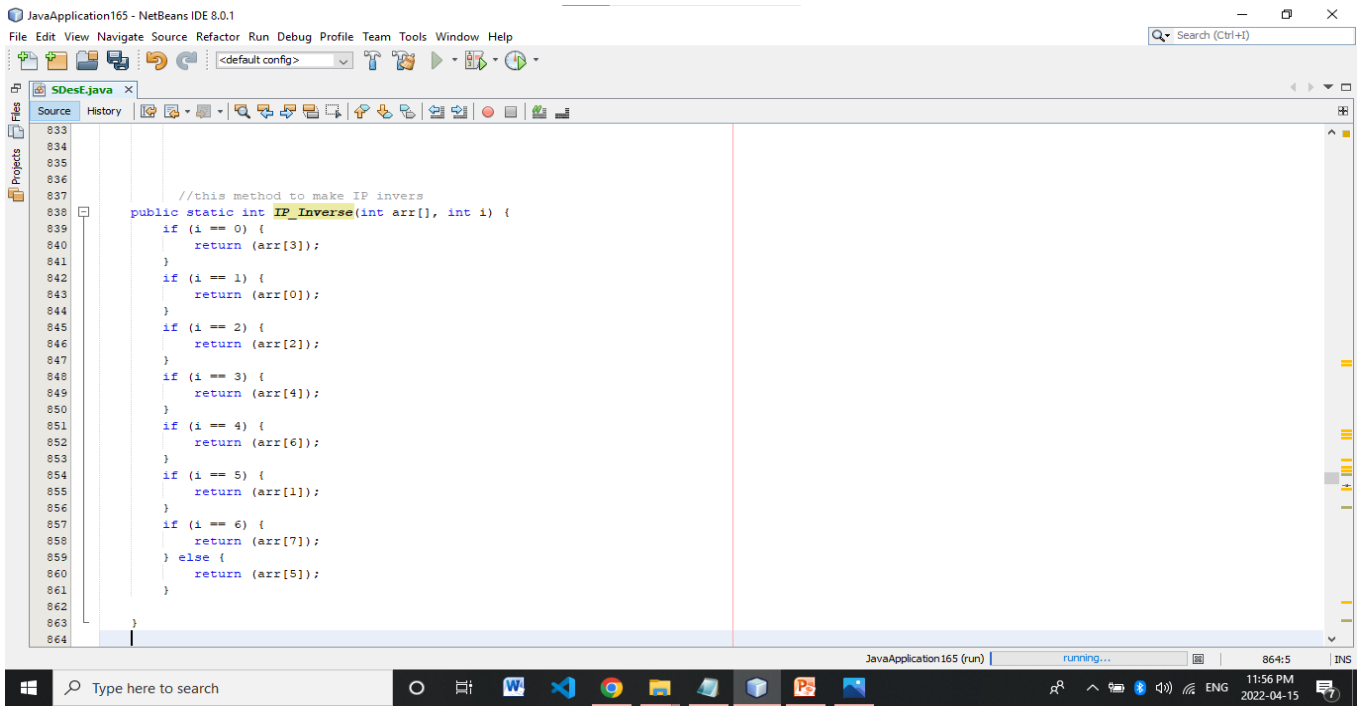
Then what will happen is X-Or between the first left part with the output of P4 which the input of it Is the output from the S-Box and the result will be with a swap .

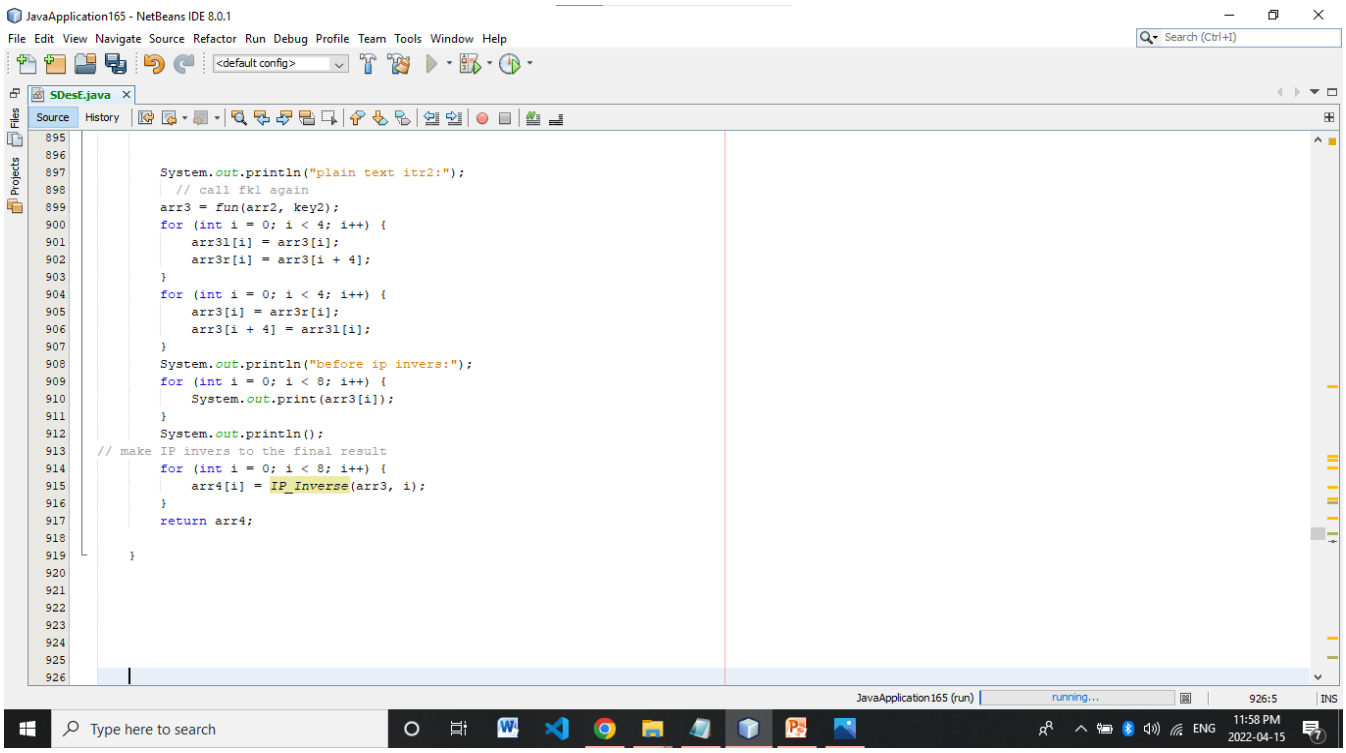




Now we finished the function processes , let's continue with the encryption mode :

The output from the first iteration will enter the function again but this time with the key 2 and the same processes will happen again but this time the last X-Or will be between key2 and the first left part, then the output from the letter step will enter IP^{-1} which we will see its code :

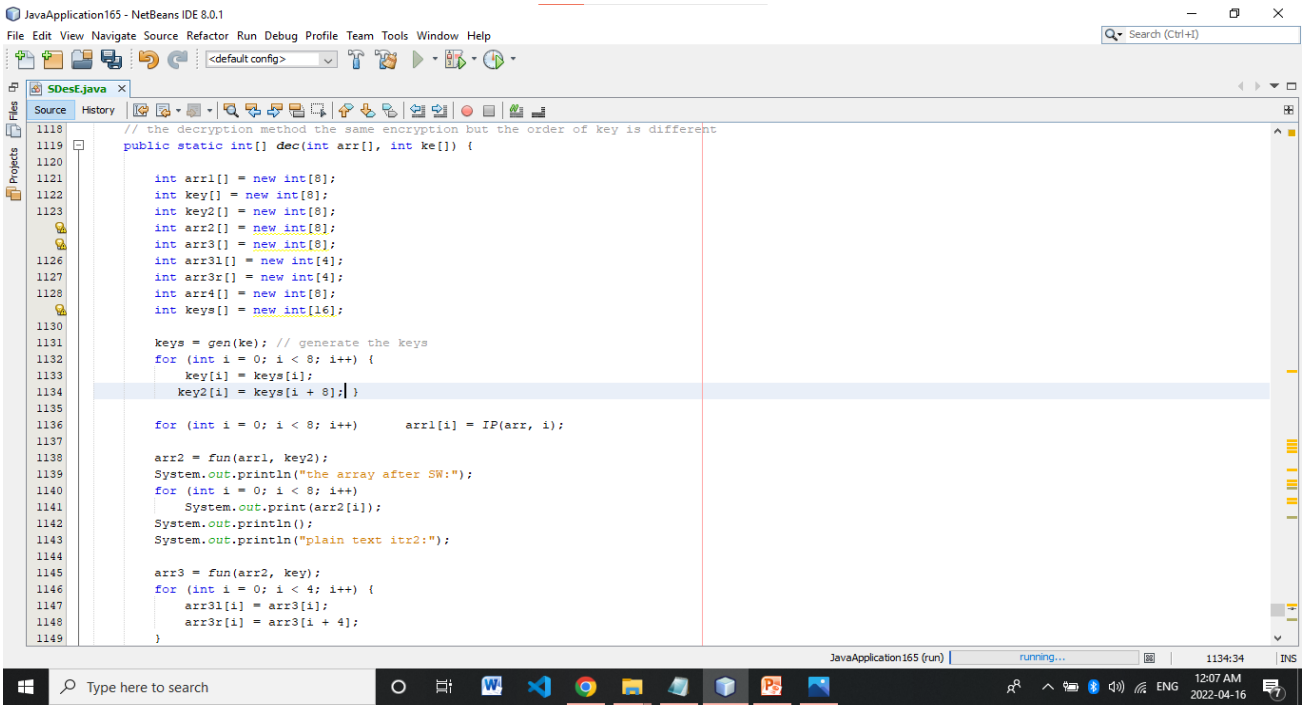


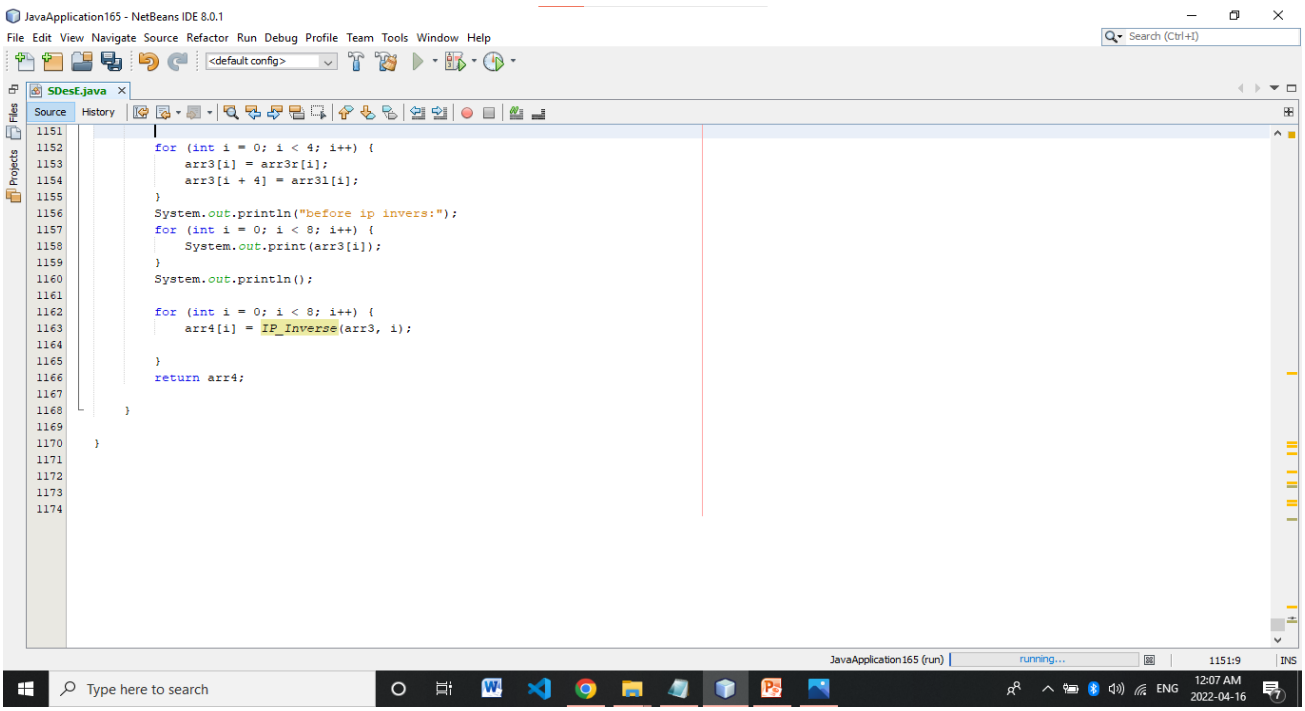


In this case the first encryption happened.

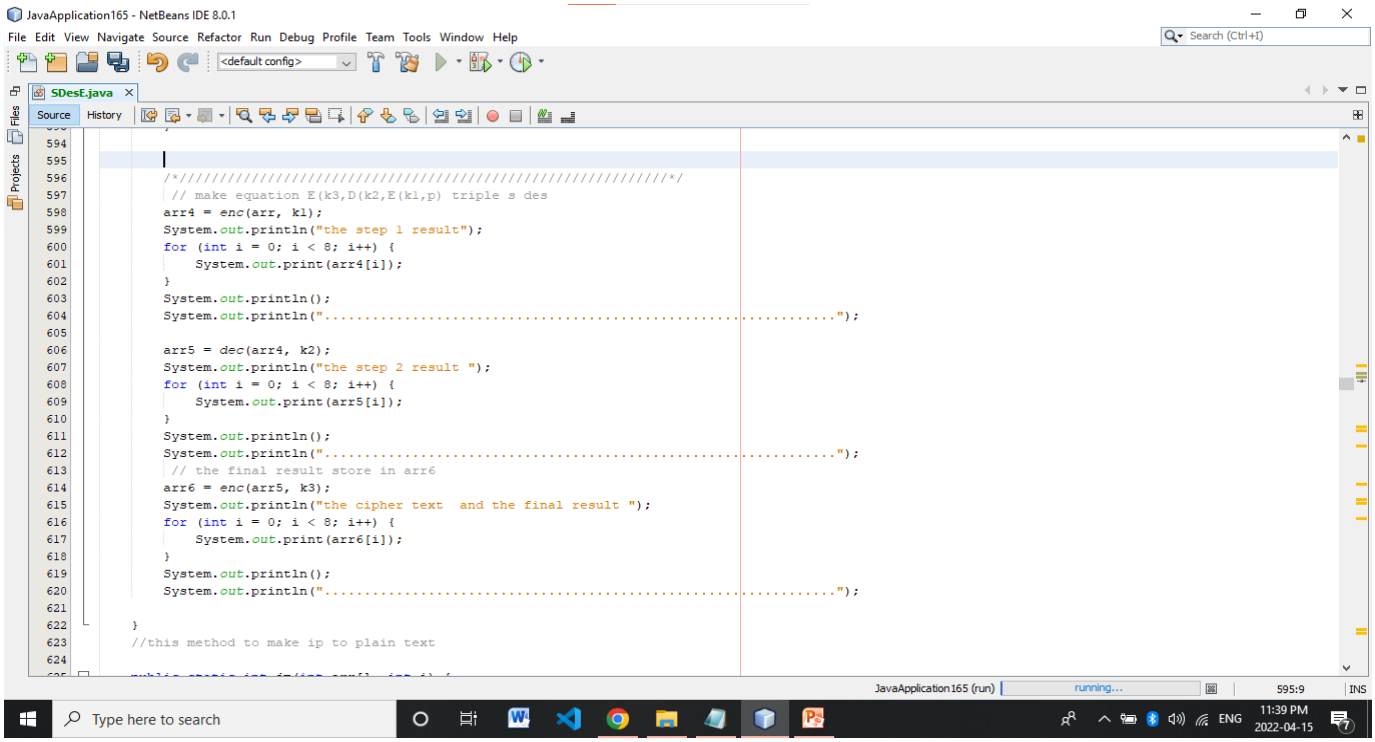
Now the second step is to do a decryption between the key2 and the output from the first step .

The decryption process is similar to the encryption process but all what we need is to replace the 2 keys , I mean we put key1 in the place of key2 , and the key2 in the place of key 1 as in the code bellow.

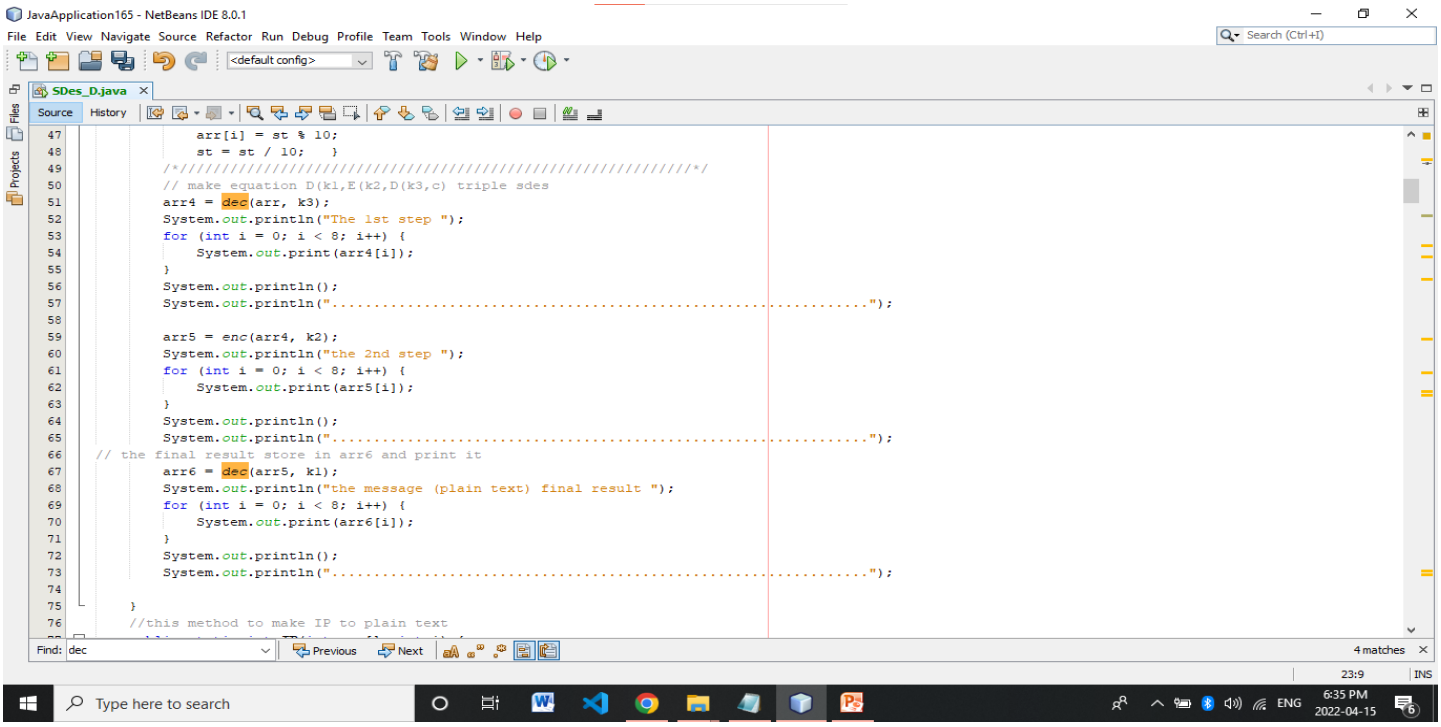




Now the last step is to do an encryption process between the output of the second step with key 3 , we talked about the encryption process before and how it works. Then the output from this step will be the cipher text that we need as bellow.



Now let’s talk in a nutshell about the decryption of the triple S-DES because it’s the same as the encryption the with this equation : D(k3,E(k2,D(k1,p) instead of E(k3,D(k2,E(k1,p) for the encryption with this code for the step :

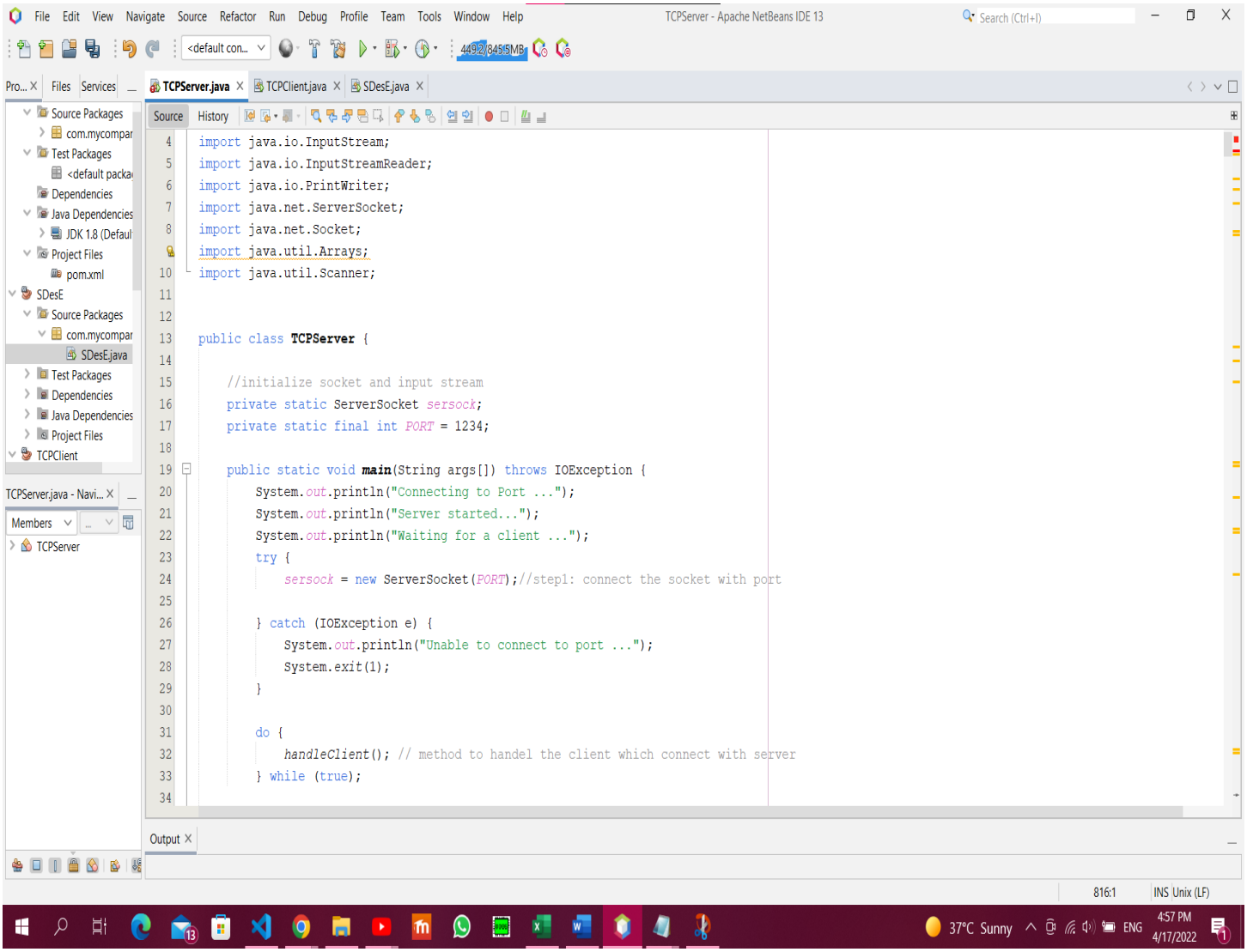


From the code above we can see that we have to start with the decryption process with the cipher text this time and then to do an encryption process using key2 and finally to do the decryption again with using the key 3 with exactly the same code and operations as those that we used in the encryption process .

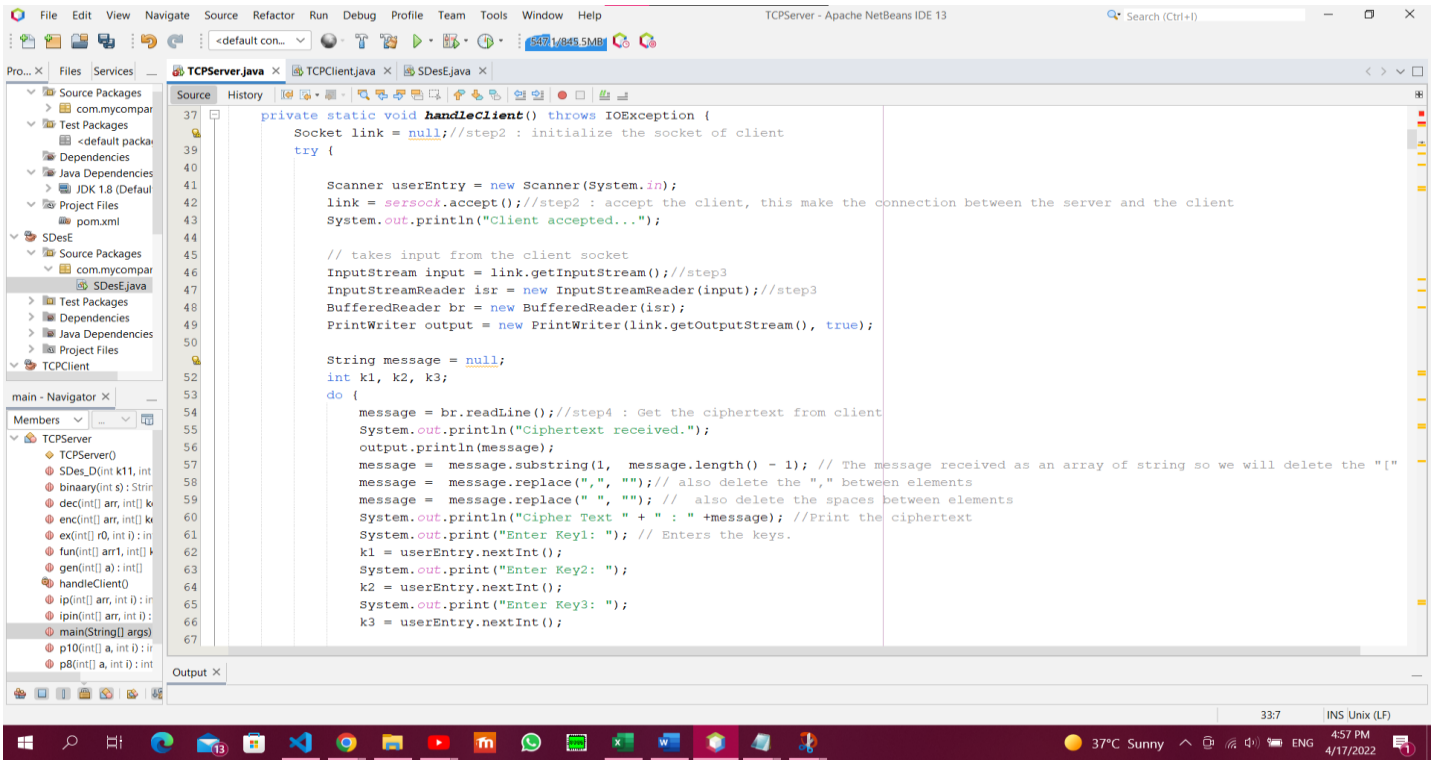
The code bellow shows how to deal with a code to connect the client with the server .

First we can see the libraries those we used and needed.

The TCP Server code [Receiver Devise]: Firstly, the device will open its socket to specific port to connect with other device.

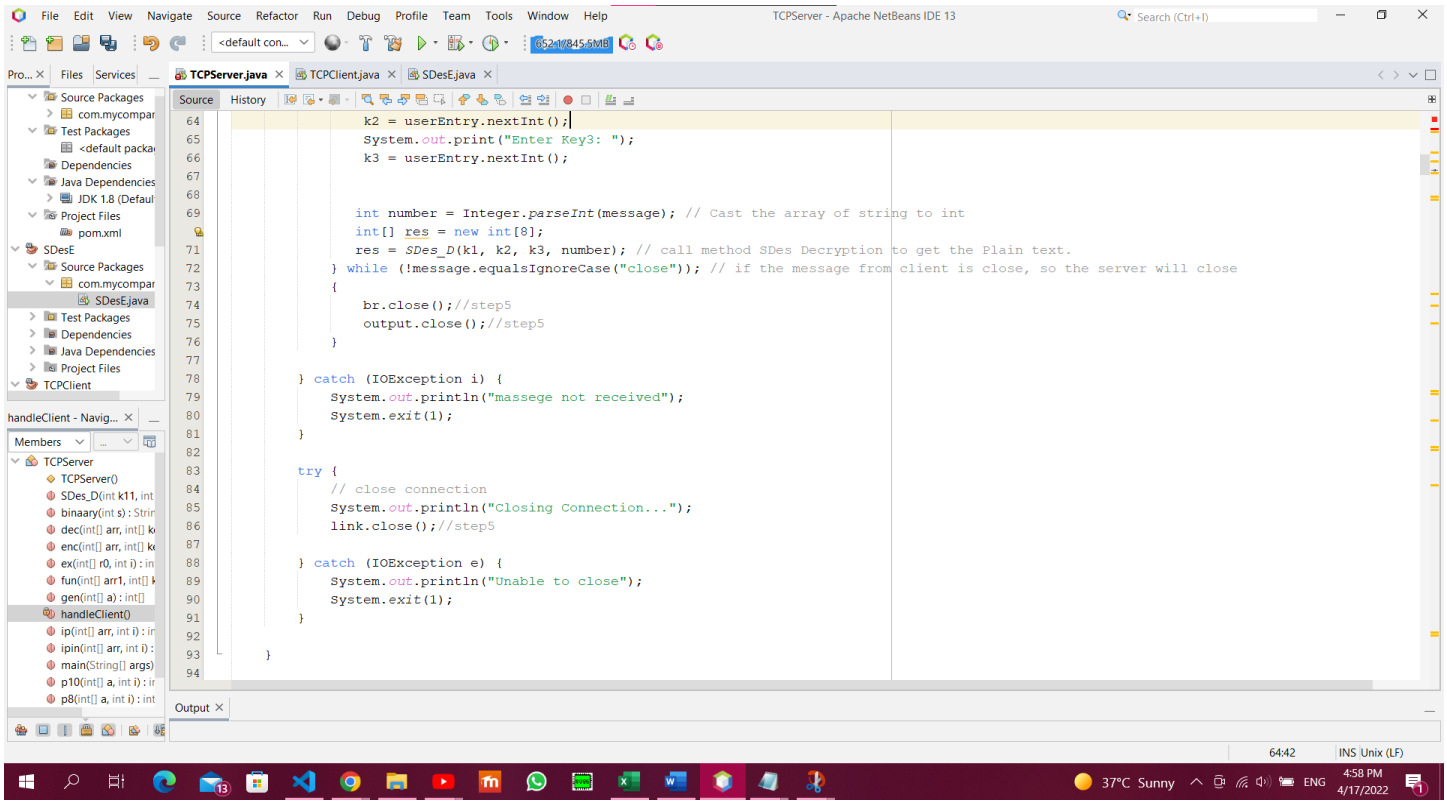


When the sender device connects with the receiver device, we call method handelClient ()



```
private static void handleClient() throws IOException {
    Socket link = null;//step2 : initialize the socket of client
    try {

        Scanner userEntry = new Scanner(System.in);
        link = sersock.accept();//step2 : accept the client, this make the connection between the server and the client
        System.out.println("Client accepted...");

        // takes input from the client socket
        InputStream input = link.getInputStream();//step3
        InputStreamReader isr = new InputStreamReader(input);//step3
        BufferedReader br = new BufferedReader(isr);
        PrintWriter output = new PrintWriter(link.getOutputStream(), true);

        String message = null;
        int k1, k2, k3;
        do {
            message = br.readLine();//step4 : Get the ciphertext from client
            System.out.println("Ciphertext received.");
            output.println(message);
            message =  message.substring(1,  message.length() - 1); // The message received as an array of string so we will delete the "["
            message =  message.replace(",", "");// also delete the "," between elements
            message =  message.replace(" ", ""); //  also delete the spaces between elements
            System.out.println("Cipher Text " + " : " +message); //Print the ciphertext
            System.out.print("Enter Key1: "); // Enters the keys.
            k1 = userEntry.nextInt();
            System.out.print("Enter Key2: ");
            k2 = userEntry.nextInt();
            System.out.print("Enter Key3: ");
            k3 = userEntry.nextInt();
```

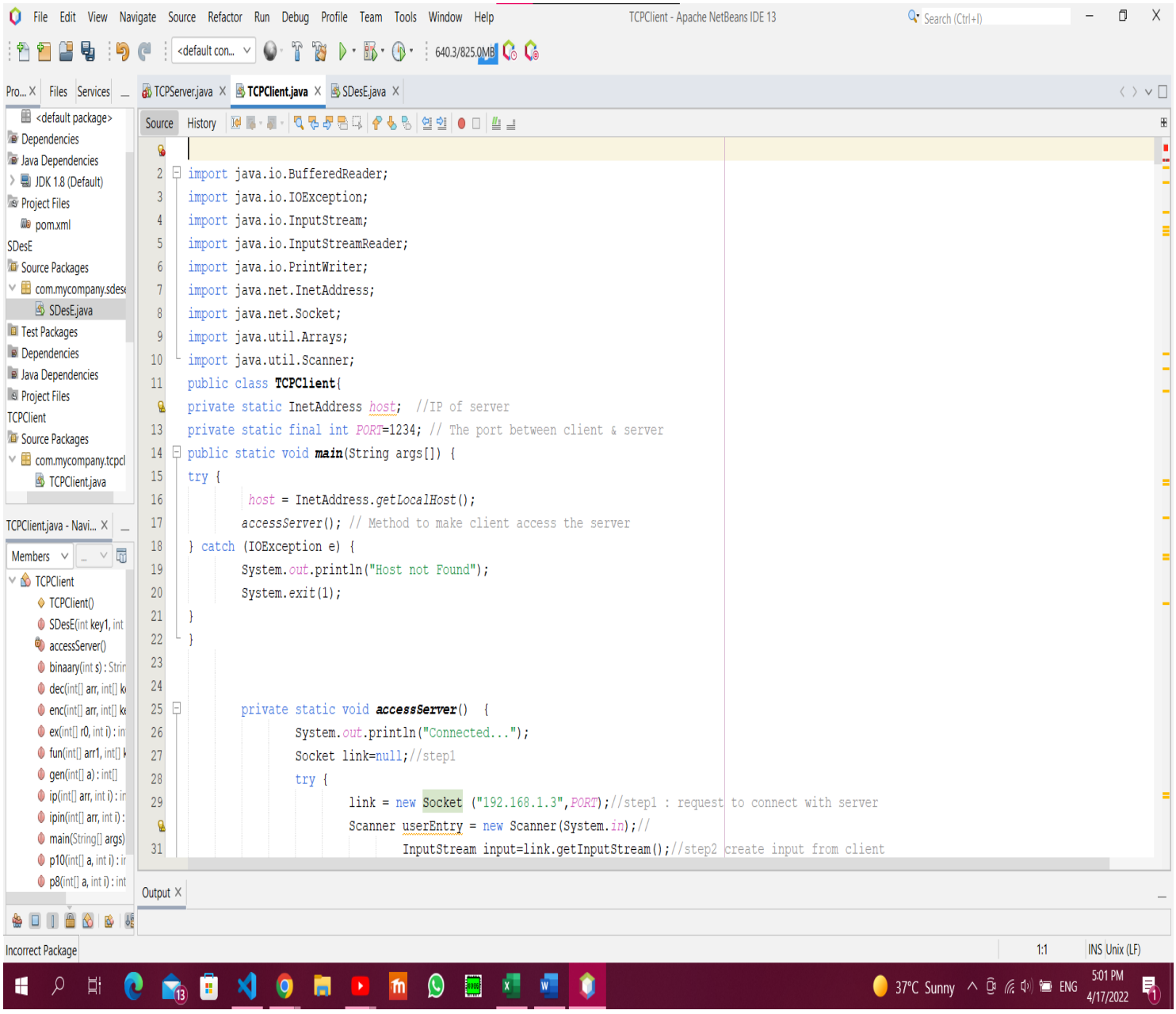
Then the sender will send the cipher text. After that the receiver will enter the three keys. Then the cipher text will be decrypted by calling the decryption method to get the plain text.



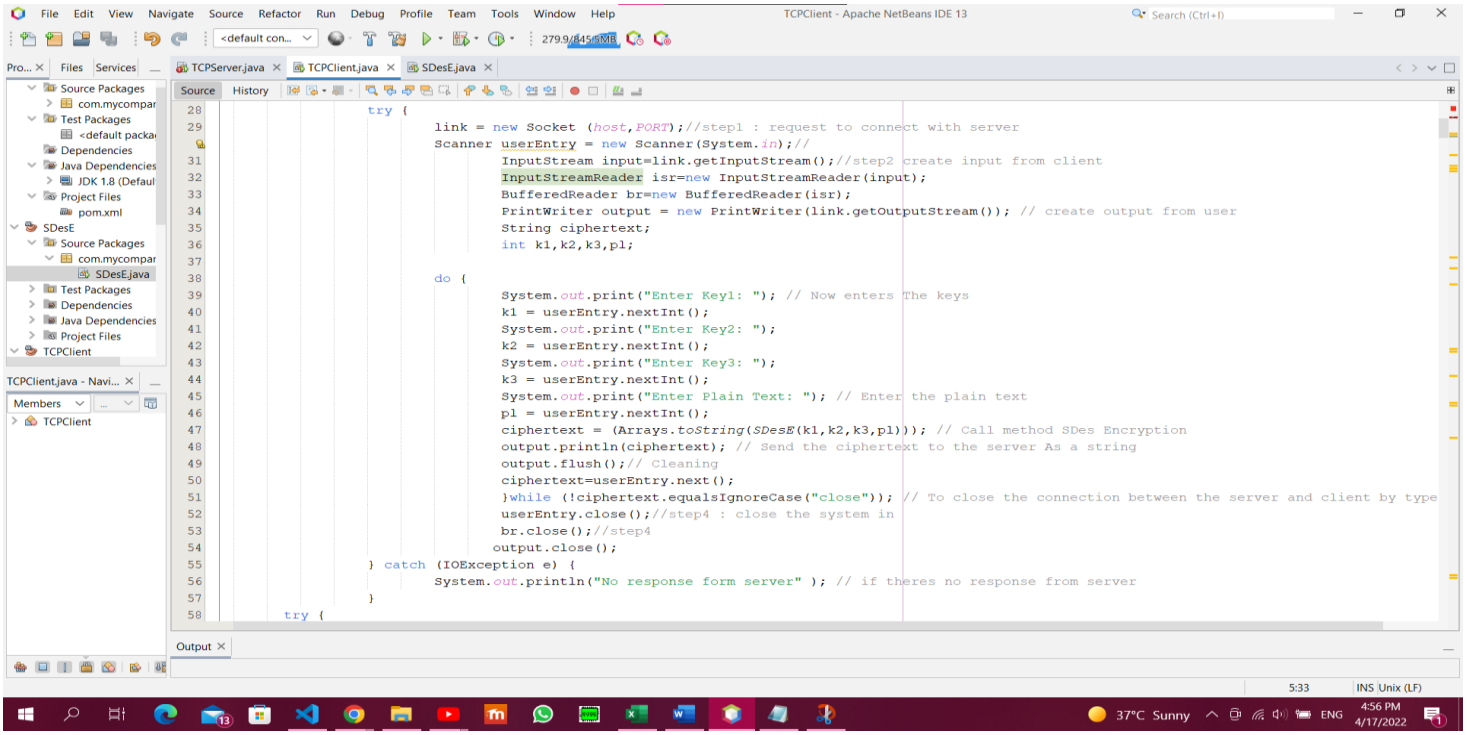
```
            k2 = userEntry.nextInt();
            System.out.print("Enter Key3: ");
            k3 = userEntry.nextInt();


           int number = Integer.parseInt(message); // Cast the array of string to int
           int[] res = new int[8];
           res = SDes_D(k1, k2, k3, number); // call method SDes Decryption to get the Plain text.
        } while (!message.equalsIgnoreCase("close")); // if the message from client is close, so the server will close
        {
            br.close();//step5
            output.close();//step5
        }

    } catch (IOException i) {
        System.out.println("massege not received");
        System.exit(1);
    }

    try {
        // close connection
        System.out.println("Closing Connection...");
        link.close();//step5

    } catch (IOException e) {
        System.out.println("Unable to close");
        System.exit(1);
    }

}
```

The Sender Device [TCP Client code]:

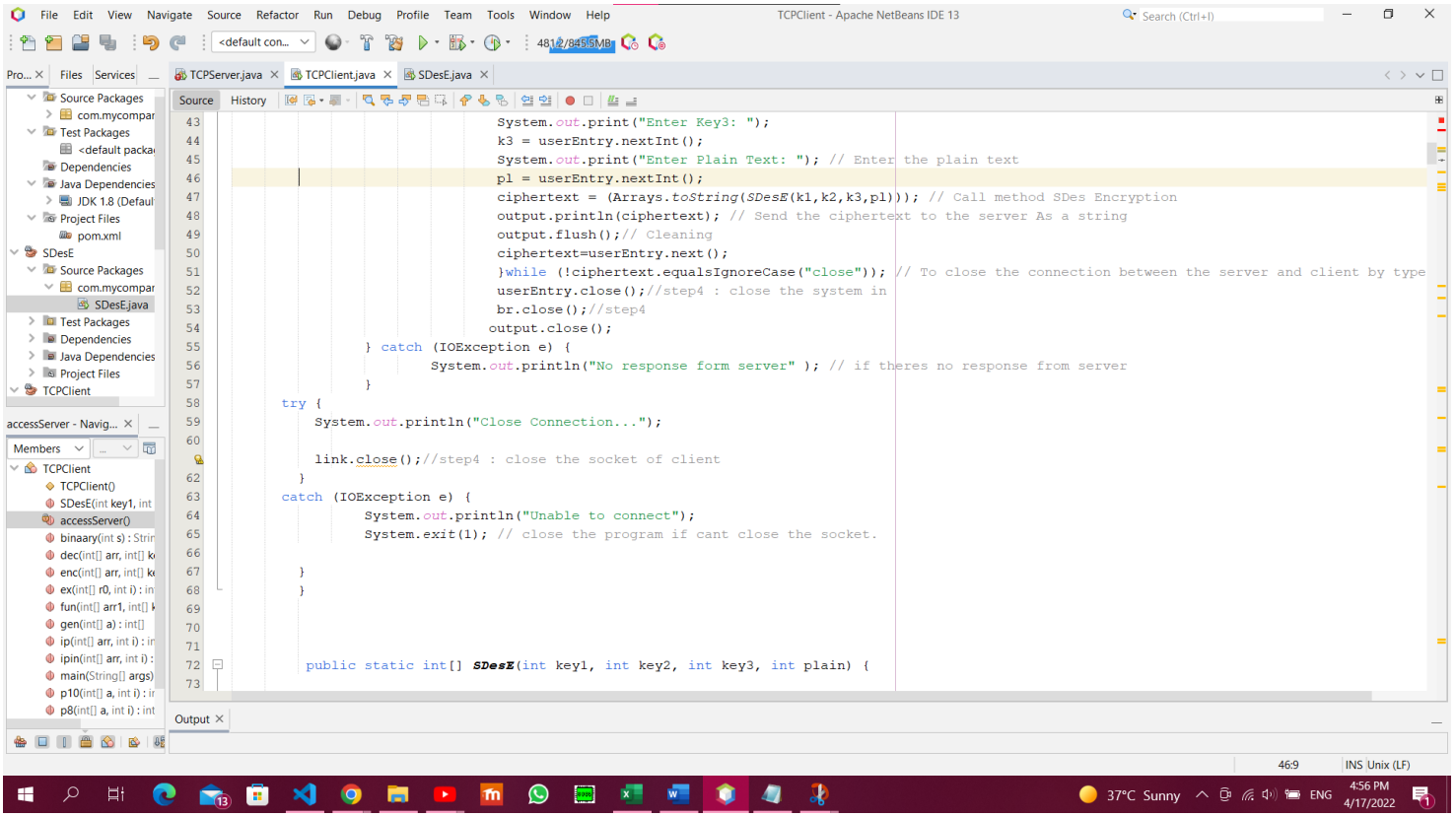
Call method access server to send the cipher text. Basically, The sender device will connect with the receiver port using IP to make connection with. And initialize the parameters that we need.



The sender will enter the three keys and the plain text. Then the plain text will be encrypted by calling encryption method to get the cipher text. Then the sender will send the cipher text to the server.



If the sender enter close, so the socket will closed and that mean the end of connection between the devices.



Verify correct work of our application:

Now let's check the results :

Network Programming in our Project

We have two devices:

1. Client (the sender).

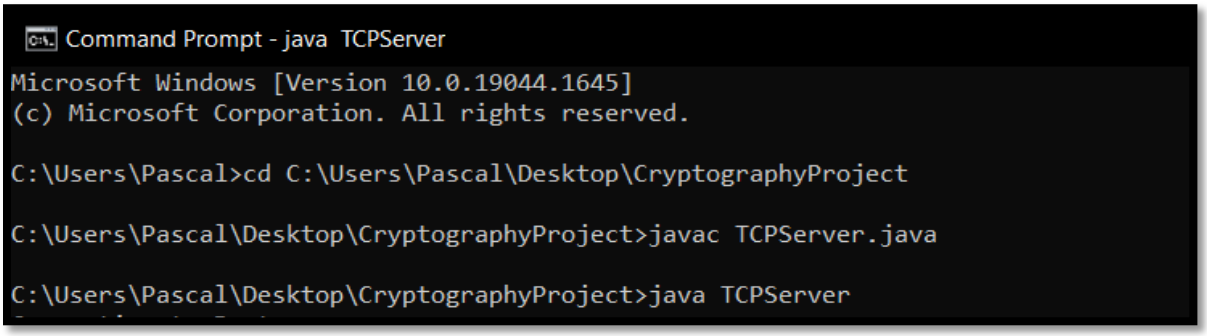
2. Server (the receiver).

We connect them with network has TCP (Transmission Control Protocol).

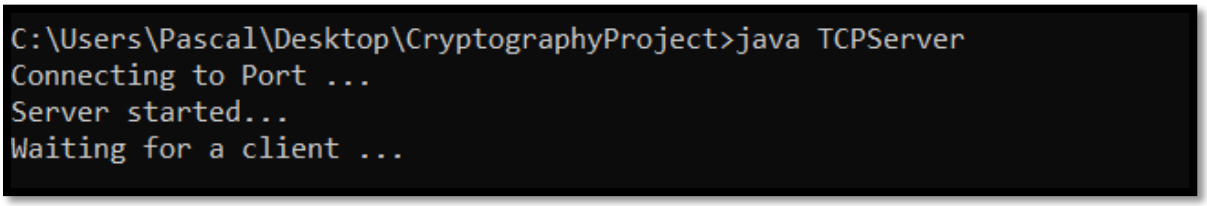
- Hardware:
  - Ethernet.
  - Two devices.
- Software:

  We used TCP code in java.

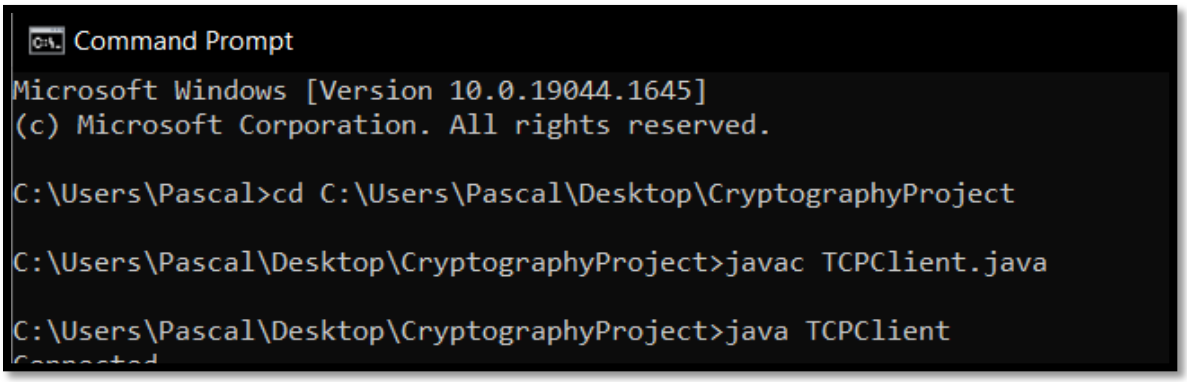
1. Run the server code from the cmd.



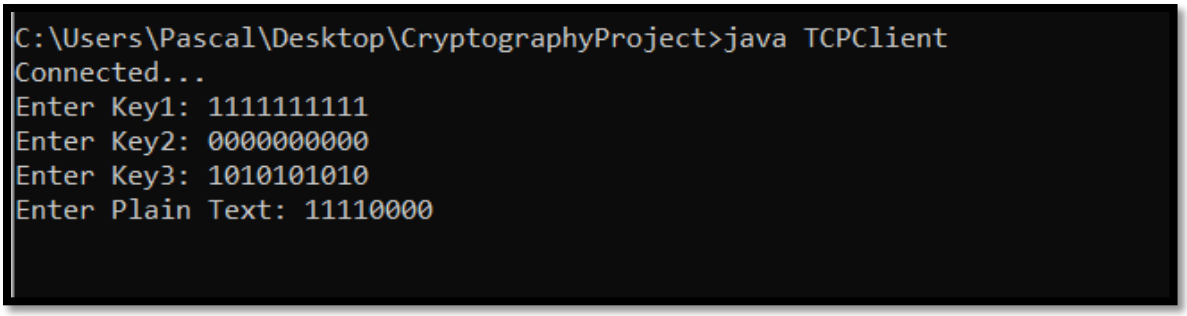
2. The server will wait for a client to connect with.



3. Now run the client code from another cmd.

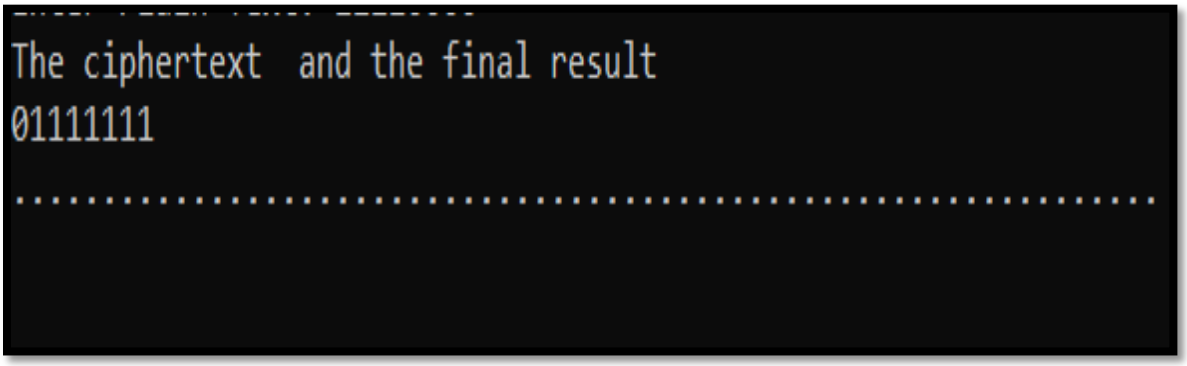


4. After run, the client will connect with server, then we want to enter the three keys and the plain text.



5. The plain text is encrypted. And now we have the cipher text, we want to send it to the server.

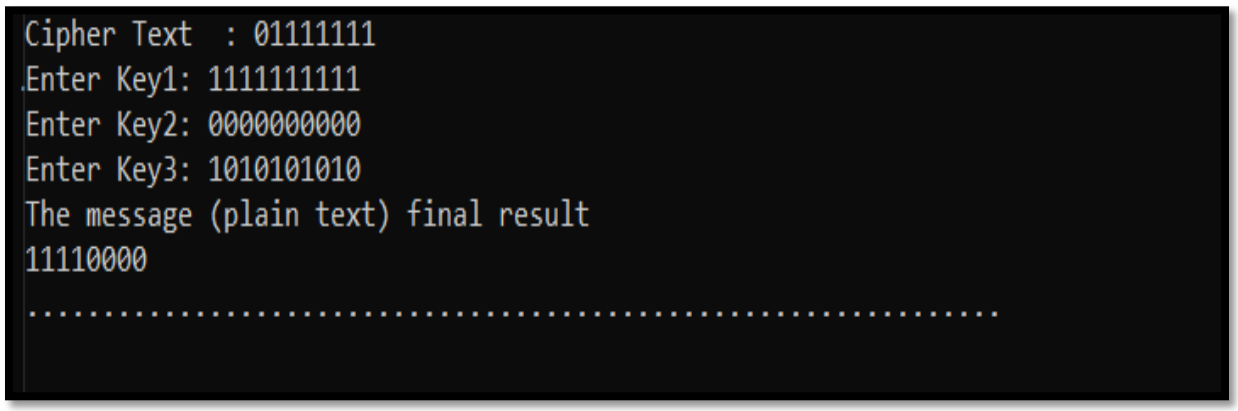
- Screen from client



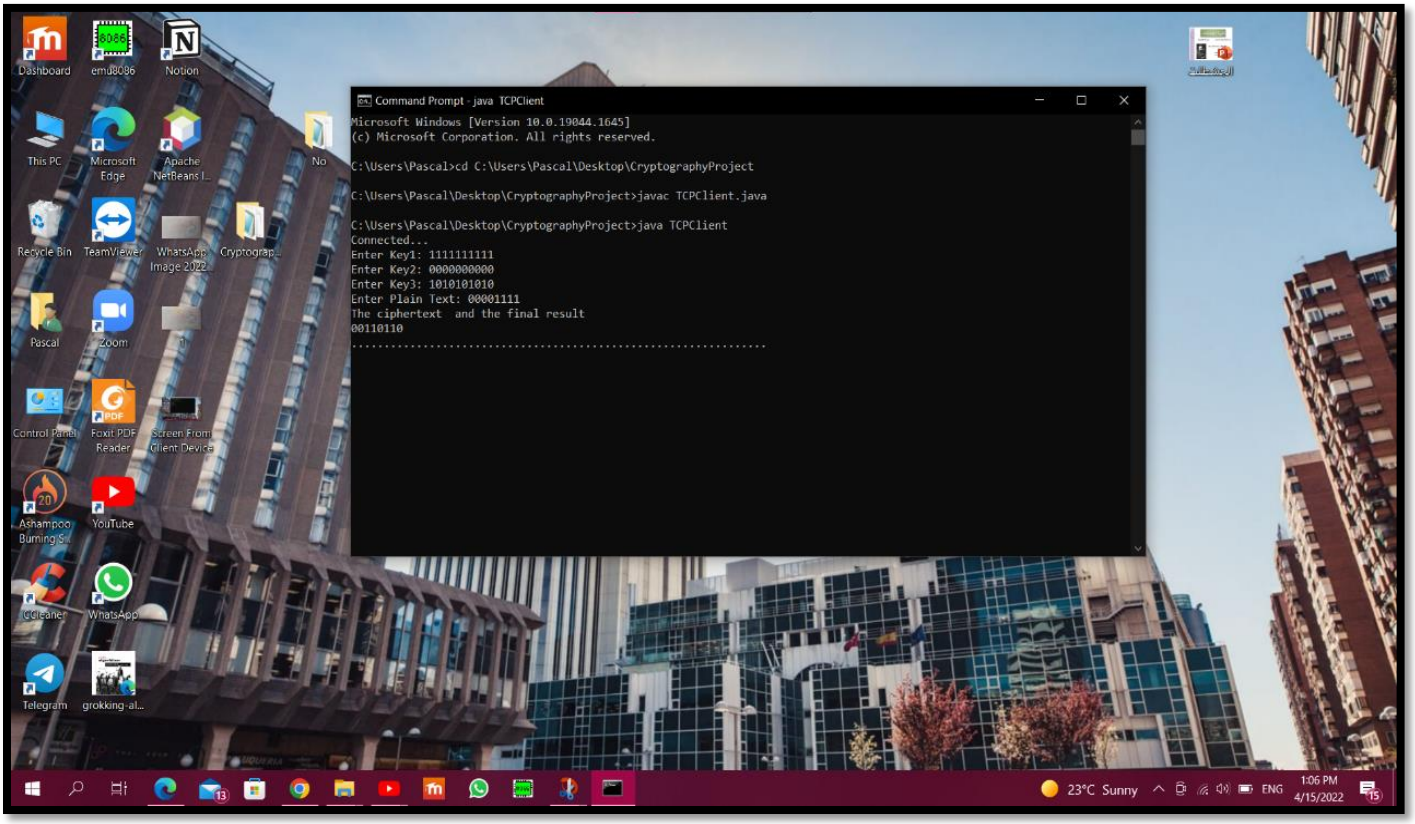
- Screen from server after received the cipher text. And we want to enter the keys.

6. Entering the keys in server.

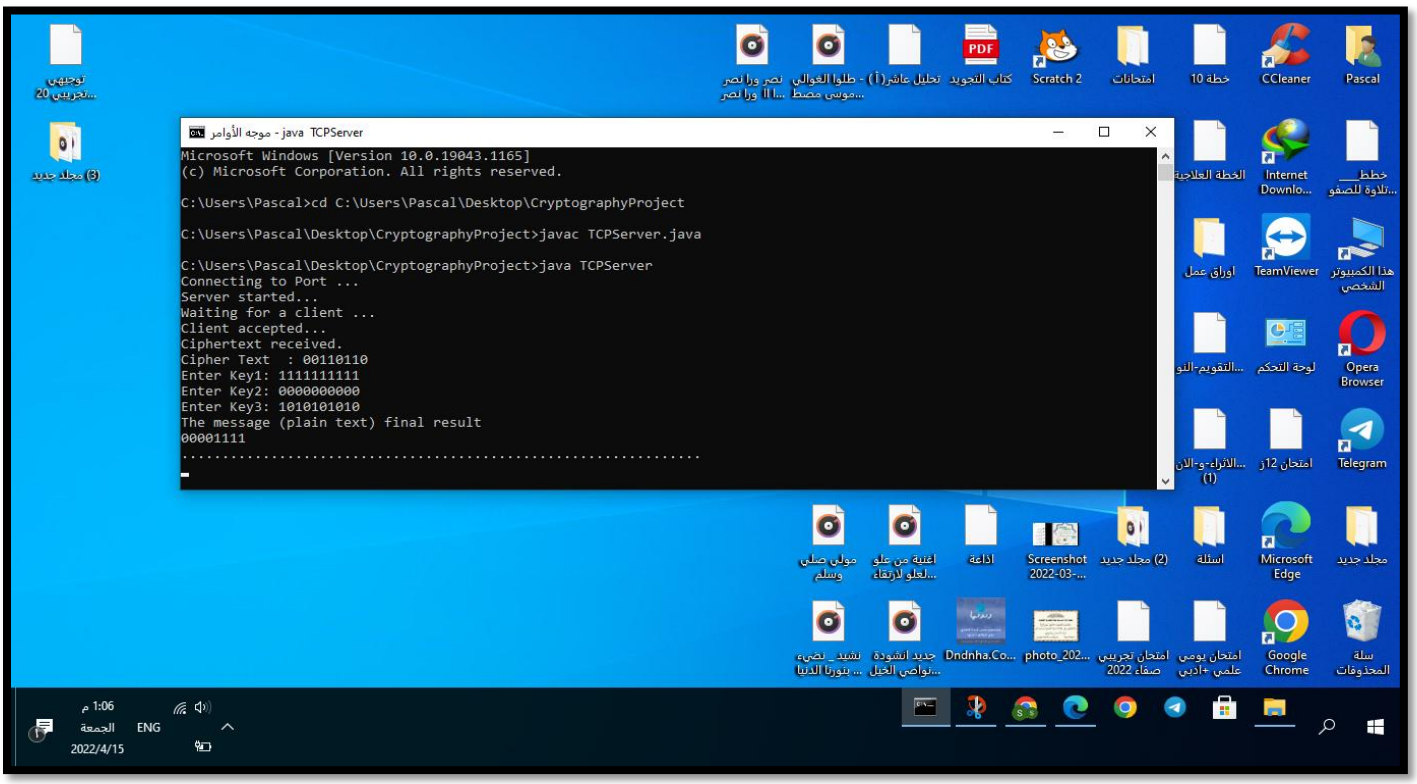
7. Now server have the encrypted message (cipher text) and the three keys, so the cipher text will be decrypted using triple S-Des to generate the plain text.



Screen from Sender Device



Screen from Receiver Device



Experiments results and discussion:

- As you can see from our codes , the language the we used to write our codes is JAVA language since it's a high level language and it's easy to deal with especially because our code was kind of long.

- About the approach – triple S-DES – as we noticed is a complicated one but we are ok with it since it increases the security to transfer a message.

- Our message was just **bits** so we deal with bits only to encrypt and decrypt them not pics or normal letters, so it was sort of easy to deal with bits only . so because of that we didn't need the internet to help us with some ready codes or anything that will benefit us to write our triple S-DES code.

Conclusion:

- The code and the whole operation work in an consistent way.

- Using a high level language as JAVA made the deal with the code so easy and satisfying .

- Triple S-DES make the sender and receiver fell comfortable since it's more secure that the normal S-DES .

- Since the triple S-DES is a developed approach of the S-DES, writing the code and understanding the algorithm is easy and clear.

References:

- ✓ The slides of the cryptanalysis material from Dr. Anas Melhem .
- ✓ Using some codes that we learnt from network programming course .
- ✓ Java network programming 4th edition Elliotte rusty harold .