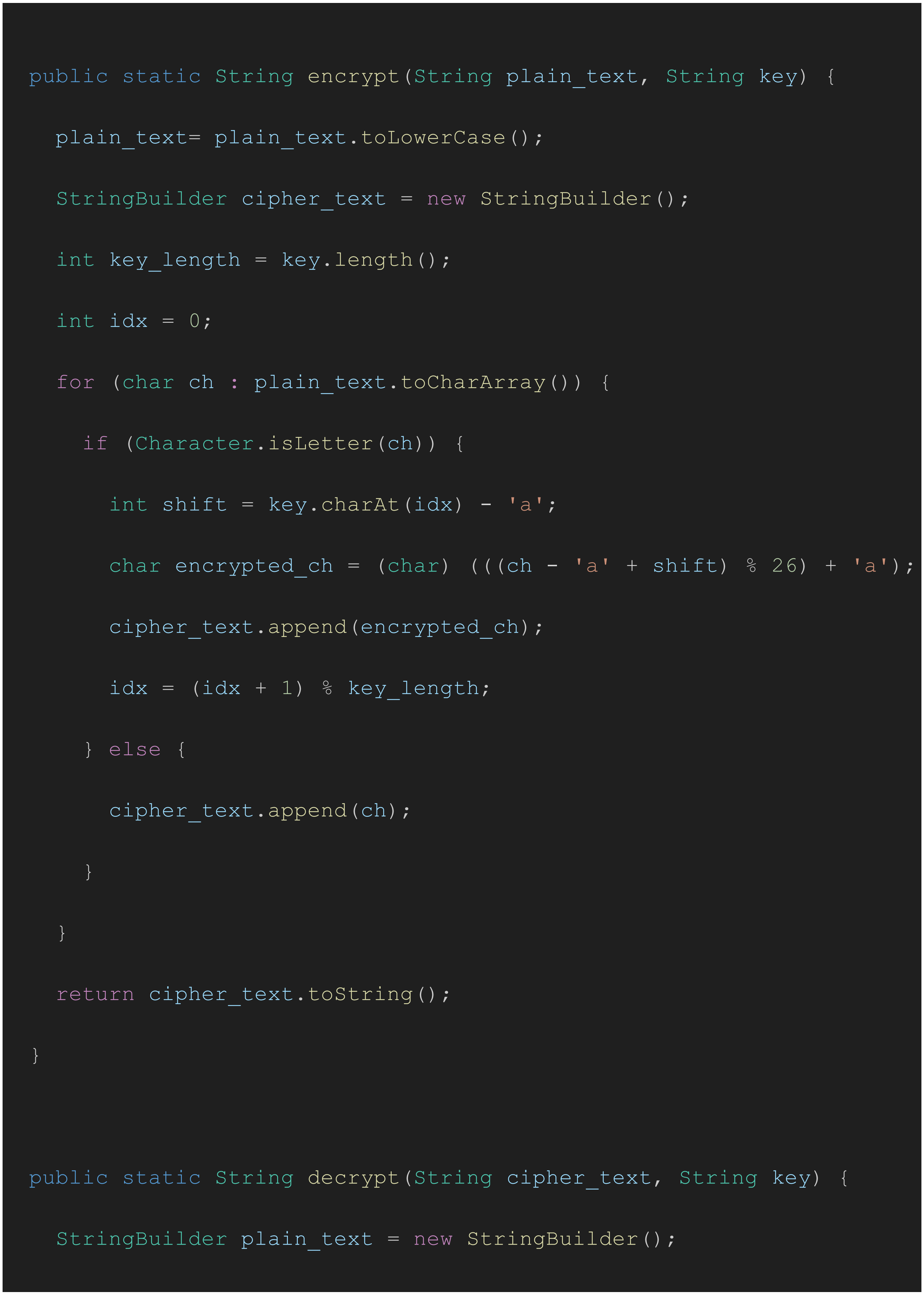
ISC ASSIGNMENT - 06

ROLL NO: U21CS052
NAME : PANCHAL GUNGUN PARESH

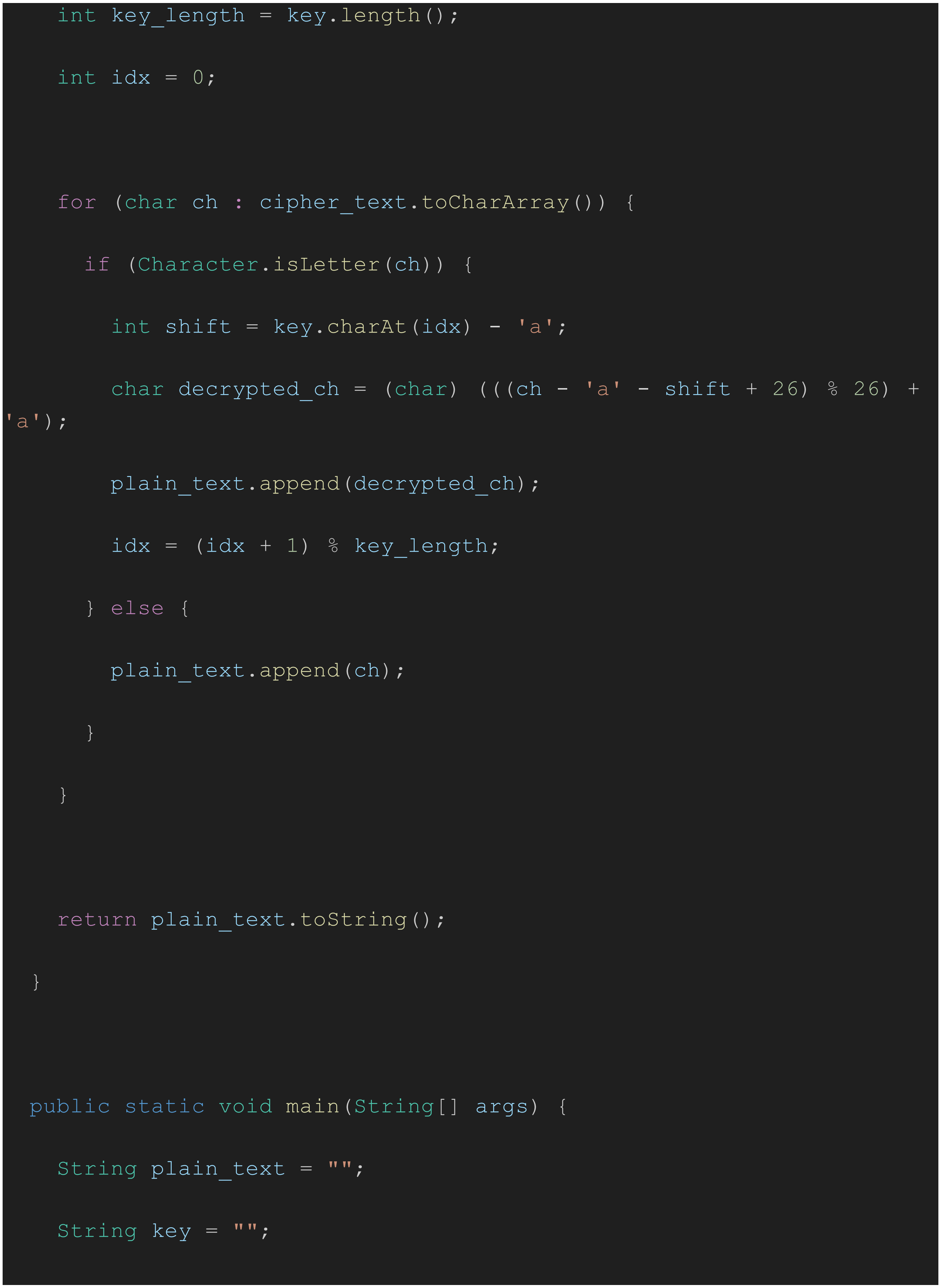
Vigenere Cipher.

1) Design and implement a program to perform encryption and decryption using the Vigenère Cipher with repeating keywords. Consider the following
inputs for the program.
For Encryption:
a) Input1 – Plaintext and Key
b) Output1 – Ciphertext
For Decryption
Input2- Ciphertext
Output2- Plaintext

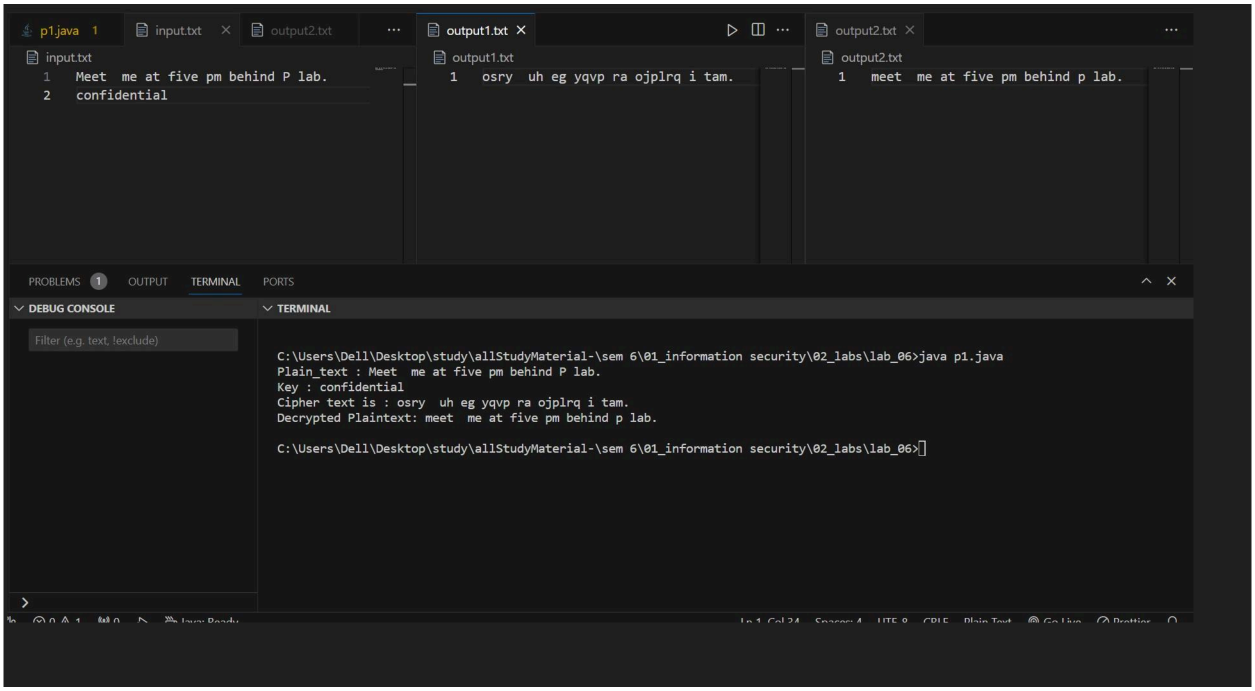
```
import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.*;



public class p1 {
```



```
public static String encrypt(String plain_text, String key) {
  plain_text= plain_text.toLowerCase();
  StringBuilder cipher_text = new StringBuilder();
  int key_length = key.length();
  int idx = 0;
  for (char ch : plain_text.toCharArray()) {
    if (Character.isLetter(ch)) {
      int shift = key.charAt(idx) - 'a';
      char encrypted_ch = (char) (((ch - 'a' + shift) % 26) + 'a');
      cipher_text.append(encrypted_ch);
      idx = (idx + 1) % key_length;
    } else {
      cipher_text.append(ch);
    }
  }
  return cipher_text.toString();
}

public static String decrypt(String cipher_text, String key) {
  StringBuilder plain_text = new StringBuilder();
```



```
    int key_length = key.length();
    int idx = 0;

    for (char ch : cipher_text.toCharArray()) {
      if (Character.isLetter(ch)) {
        int shift = key.charAt(idx) - 'a';
        char decrypted_ch = (char) (((ch - 'a' - shift + 26) % 26) + 'a');
        plain_text.append(decrypted_ch);
        idx = (idx + 1) % key_length;
      } else {
        plain_text.append(ch);
      }
    }

    return plain_text.toString();
  }

  public static void main(String[] args) {
    String plain_text = "";
    String key = "";
    try {
      BufferedReader br = new BufferedReader(new
FileReader("input.txt"));



      plain_text = br.readLine();
      key = br.readLine();
      System.out.println("Plain_text : "+plain_text);
      System.out.println("Key : "+key);
      br.close();
    } catch (IOException e) {
      e.printStackTrace();
    }

    /*ENCRYPT THE PLAIN TEXT USING THE KEY. */
    String cipher_text = encrypt(plain_text, key);
    System.out.println("Cipher text is : " + cipher_text);

    /*WRITE THE CIPHER TEXT TO FILE. */
    try {
      BufferedWriter bw = new BufferedWriter(new
FileWriter("output1.txt"));

      bw.write(cipher_text);

      bw.close();

    } catch (IOException e) {

      e.printStackTrace();

    }



    /*READ INPUT CIPHER TEXT FROM THE FILE. */

    String input_cipher_text = "";



    try {

      BufferedReader br = new BufferedReader(new
FileReader("output1.txt"));

      input_cipher_text = br.readLine();

      br.close();

    } catch (IOException e) {

      e.printStackTrace();

    }



    /* DECRYPT CIPHERTEXT USING KEY. */

    String decrypted_plain_text = decrypt(input_cipher_text, key);
    // System.out.println("Decrypted Plain text: ",
decrypted_plain_text);

    System.out.print("Decrypted Plaintext: ");

    System.out.println(decrypted_plain_text);

    try {

      BufferedWriter bw = new BufferedWriter(new
FileWriter("output2.txt"));

      bw.write(decrypted_plain_text);

      bw.close();

    } catch (IOException e) {

      e.printStackTrace();

    }

  }}
```



input.txt

```
Meet  me at five pm behind P lab.
confidential
```

output1.txt

```
osry  uh eg yqvp ra ojplrq i tam.
```

output2.txt

```
meet  me at five pm behind p lab.
```

TERMINAL

```
C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_06>java p1.java
Plain_text : Meet  me at five pm behind P lab.
Key : confidential
Cipher text is : osry  uh eg yqvp ra ojplrq i tam.
Decrypted Plaintext: meet  me at five pm behind p lab.

C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_06>
```

2) Design and implement a program to perform encryption and decryption using the Vigenère Cipher with a running key (key is as long as plaintext). Consider the following inputs for the program.

For Encryption:

a) Input3 – Plaintext and Key

b) Output3 – Ciphertext

For Decryption

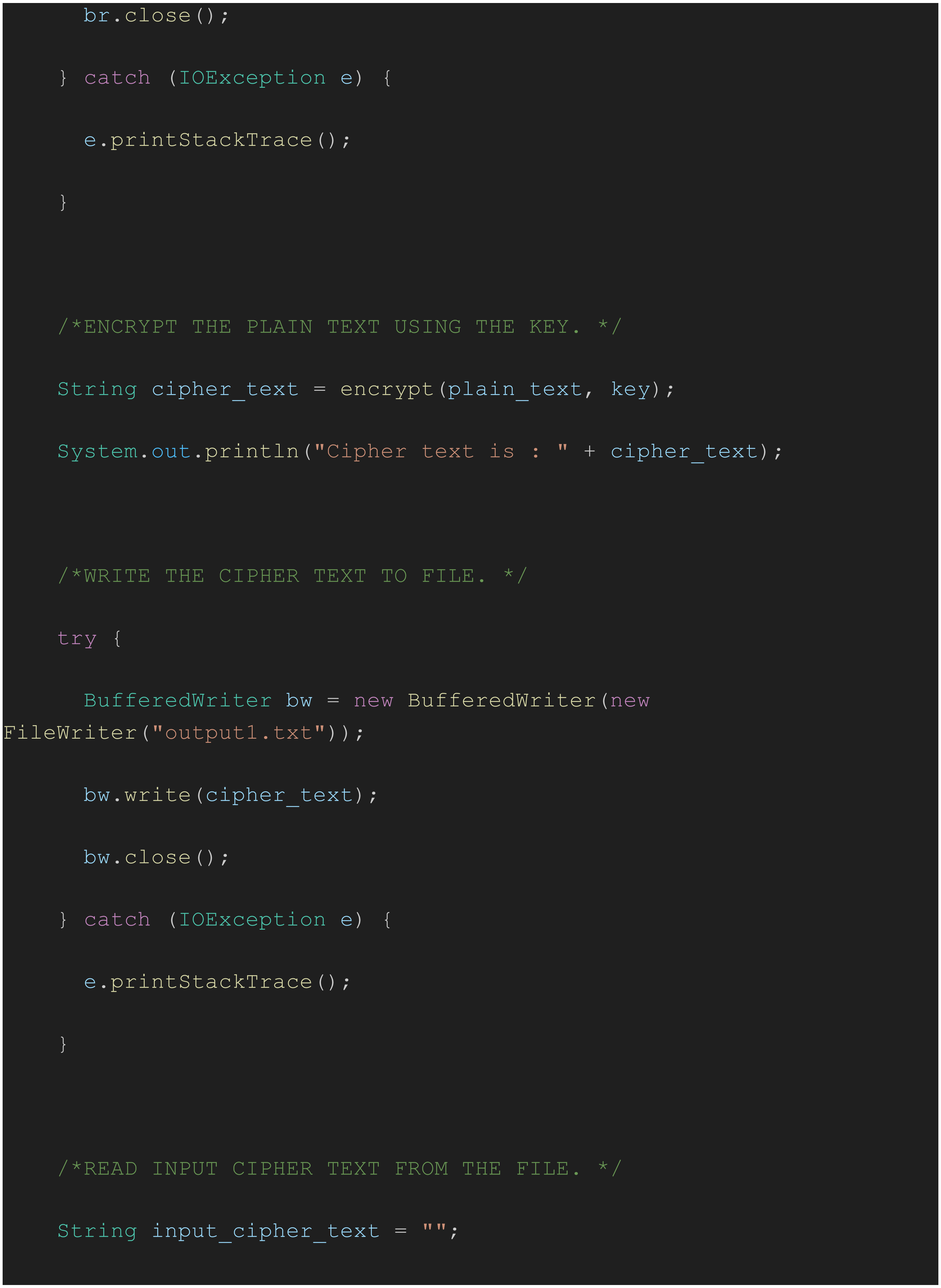
a) Input4- Ciphertext

b) Output4- Plaintext

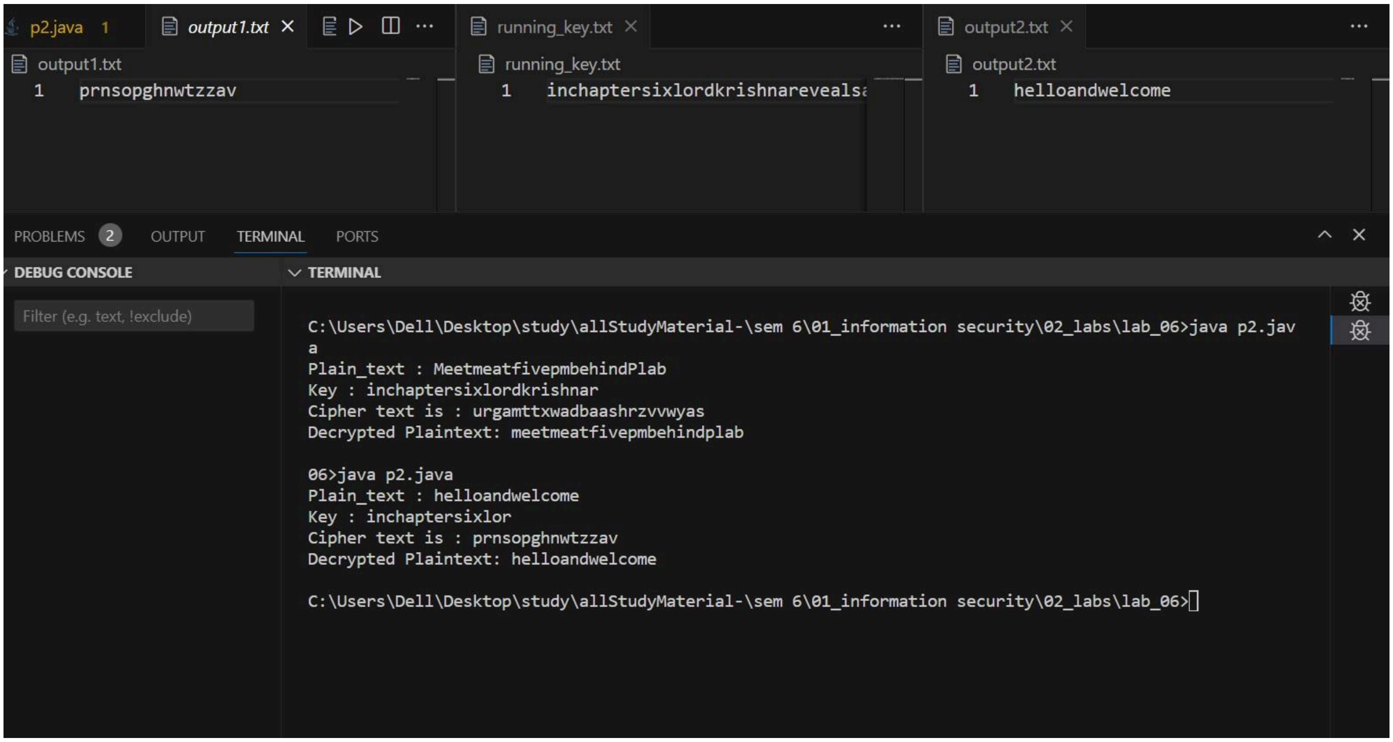
```
import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.*;



public class p2 {



  public static String encrypt(String plain_text, String key) {
```

```
  plain_text = plain_text.toLowerCase();

  StringBuilder cipher_text = new StringBuilder();

  int key_length = key.length();

  int idx = 0;

  for (char ch : plain_text.toCharArray()) {

    if (Character.isLetter(ch)) {

      int shift = key.charAt(idx) - 'a';

      char encrypted_ch = (char) (((ch - 'a' + shift) % 26) + 'a');

      cipher_text.append(encrypted_ch);

      idx = (idx + 1) % key_length;

    } else {

      cipher_text.append(ch);

    }

  }

  return cipher_text.toString();

}



public static String decrypt(String cipher_text, String key) {

  StringBuilder plain_text = new StringBuilder();

  int key_length = key.length();

  int idx = 0;
```

```
    for (char ch : cipher_text.toCharArray()) {

      if (Character.isLetter(ch)) {

        int shift = key.charAt(idx) - 'a';

        char decrypted_ch = (char) (((ch - 'a' - shift + 26) % 26) +
'a');

        plain_text.append(decrypted_ch);

        idx = (idx + 1) % key_length;

      } else {

        plain_text.append(ch);

      }

    }

    return plain_text.toString();

  }



  public static String extractKey(String filename, int key_length) {

    StringBuilder key = new StringBuilder();

    try {

      BufferedReader br = new BufferedReader(new
FileReader(filename));

      String line;
      while ((line = br.readLine()) != null && (key.length() < key_length)) {

        /*REMOVE NON-ALPHABETIC CHARACTERS AND ADD LETTERS TO THE KEY. */



        for (char ch : line.toCharArray()) {

          if (Character.isLetter(ch)) {

            key.append(ch);

            /* key.append(ch.toUpperCase(ch)) */

            if (key.length() == key_length) {

              break;

            }

          }

        }

      }

      br.close();

    } catch (IOException e) {

      e.printStackTrace();

    }



    return key.toString();

  }
  public static void main(String[] args) {

    String plain_text = "";

    String key = "";




try {

      BufferedReader br = new BufferedReader(new
FileReader("input.txt"));


      plain_text = br.readLine();

      /*WRITE THE KEY EXTRACTOR CODE HERE. */

      String filename = "running_key.txt";


      // Specify the length of the key

      int keyLength = plain_text.length();


      // Extract the key from the file

      key = extractKey(filename, keyLength);

      System.out.println("Plain_text : " + plain_text);

      System.out.println("Key : " + key);
```



```
      br.close();
    } catch (IOException e) {
      e.printStackTrace();
    }

    /*ENCRYPT THE PLAIN TEXT USING THE KEY. */
    String cipher_text = encrypt(plain_text, key);
    System.out.println("Cipher text is : " + cipher_text);

    /*WRITE THE CIPHER TEXT TO FILE. */
    try {
      BufferedWriter bw = new BufferedWriter(new
FileWriter("output1.txt"));
      bw.write(cipher_text);
      bw.close();
    } catch (IOException e) {
      e.printStackTrace();
    }

    /*READ INPUT CIPHER TEXT FROM THE FILE. */
    String input_cipher_text = "";
    try {

      BufferedReader br = new BufferedReader(new
FileReader("output1.txt"));

      input_cipher_text = br.readLine();

      br.close();

    } catch (IOException e) {

      e.printStackTrace();

    }



    /* DECRYPT CIPHERTEXT USING KEY. */

    String decrypted_plain_text = decrypt(input_cipher_text, key);

    // System.out.println("Decrypted Plain text: ",
decrypted_plain_text);

    System.out.print("Decrypted Plaintext: ");

    System.out.println(decrypted_plain_text);



    try {

      BufferedWriter bw = new BufferedWriter(new
FileWriter("output2.txt"));

      bw.write(decrypted_plain_text);

      bw.close();
    } catch (IOException e) {

      e.printStackTrace();

   }

  }}
```

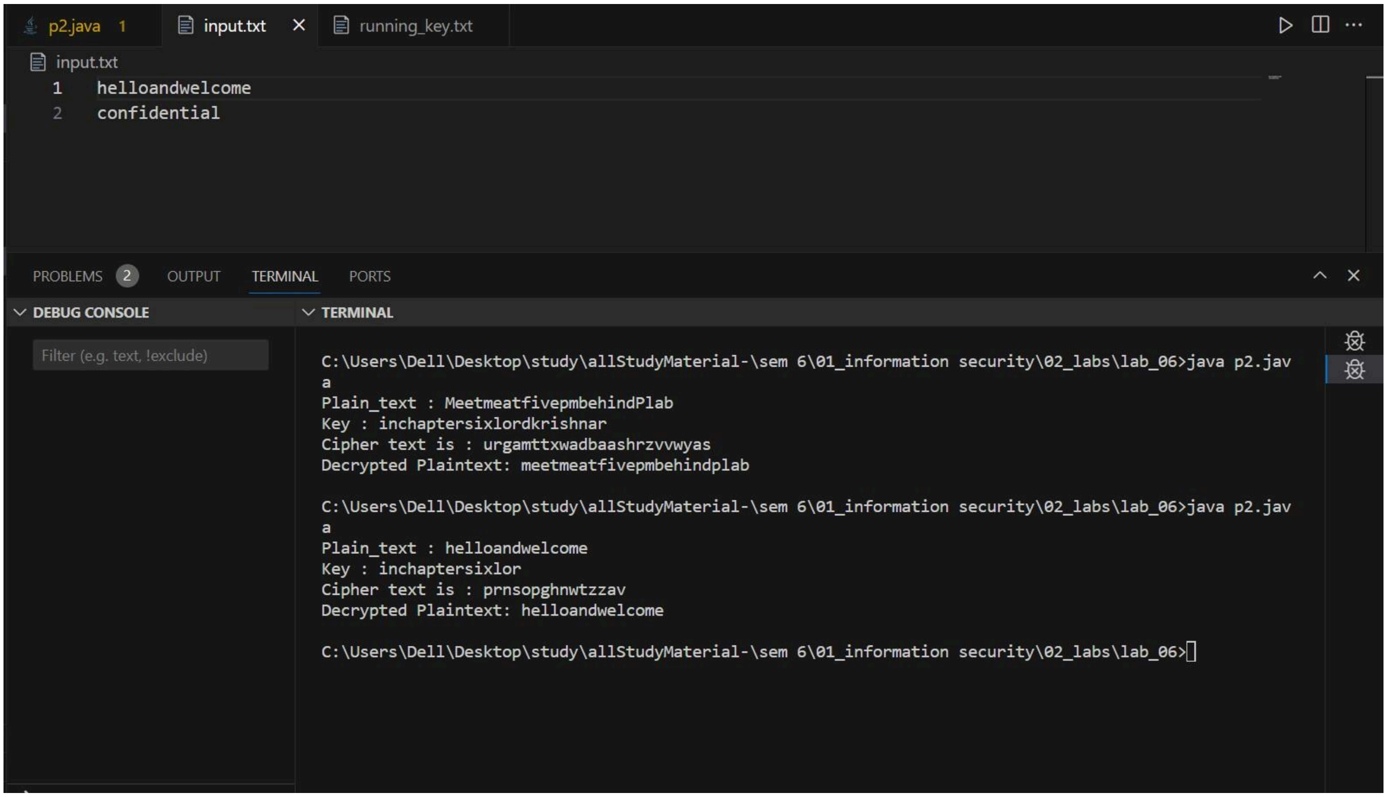


output1.txt
```
prnsopghnwtzzav
```

running_key.txt
```
inchaptersixlordkrishnareveals
```

output2.txt
```
helloandwelcome
```

```
C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_06>java p2.jav
a
Plain_text : MeetmeatfivepmbehindPlab
Key : inchaptersixlordkrishnar
Cipher text is : urgamttxwadbaashrzvvwyas
Decrypted Plaintext: meetmeatfivepmbehindplab

06>java p2.java
Plain_text : helloandwelcome
Key : inchaptersixlor
Cipher text is : prnsopghnwtzzav
Decrypted Plaintext: helloandwelcome

C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_06>
```



input.txt
```
helloandwelcome
confidential
```

```
C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_06>java p2.jav
a
Plain_text : MeetmeatfivepmbehindPlab
Key : inchaptersixlordkrishnar
Cipher text is : urgamttxwadbaashrzvvwyas
Decrypted Plaintext: meetmeatfivepmbehindplab

C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_06>java p2.jav
a
Plain_text : helloandwelcome
Key : inchaptersixlor
Cipher text is : prnsopghnwtzzav
Decrypted Plaintext: helloandwelcome

C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_06>
```

3) Design and implement a program to perform encryption and decryption using the One time pad. Consider the following inputs for the program.

For Encryption:

a) Input3 – Plaintext and Generate the using a good Random Number Generator (RNG) Function

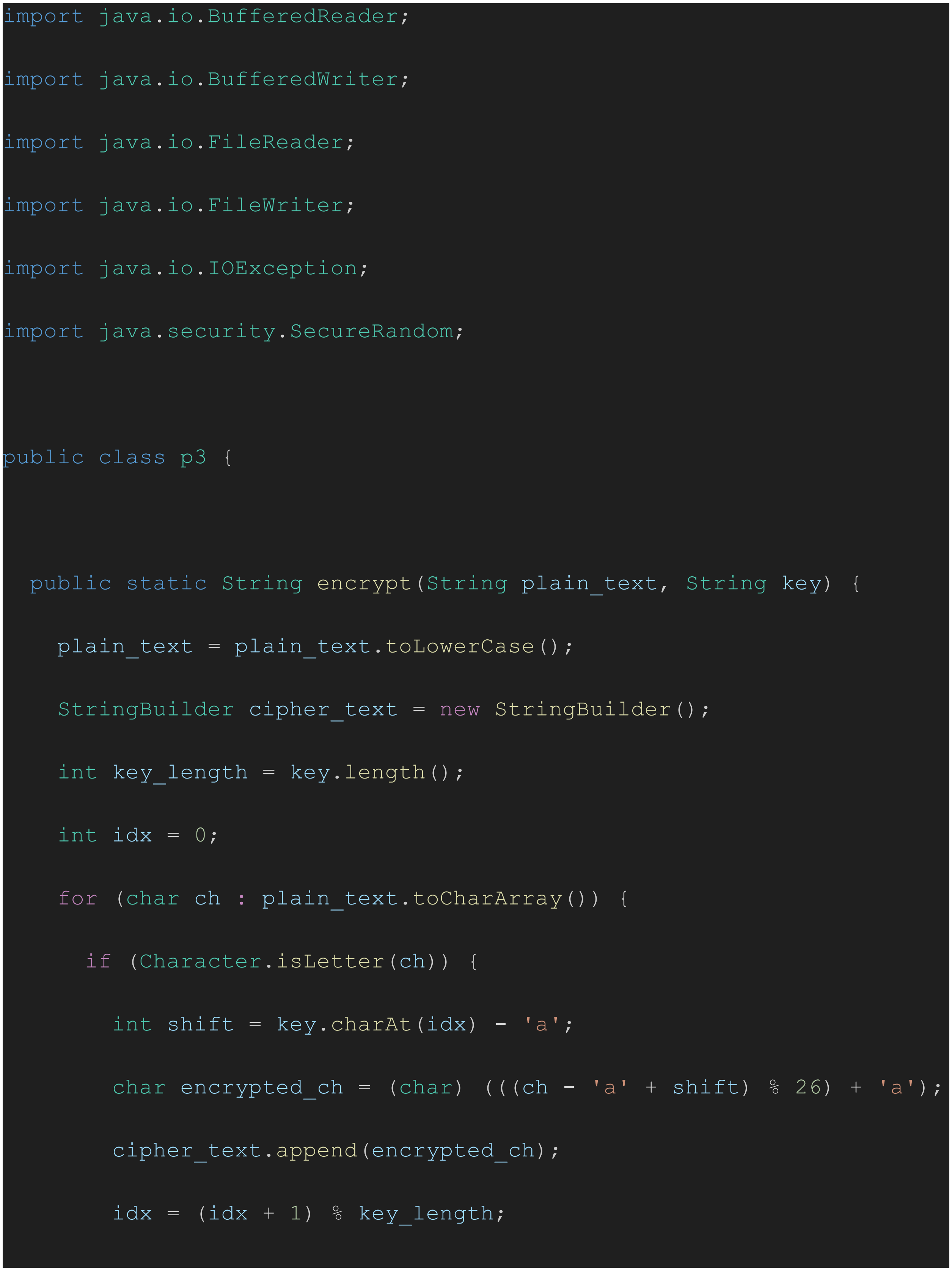
b) Output3 – Ciphertext

For Decryption

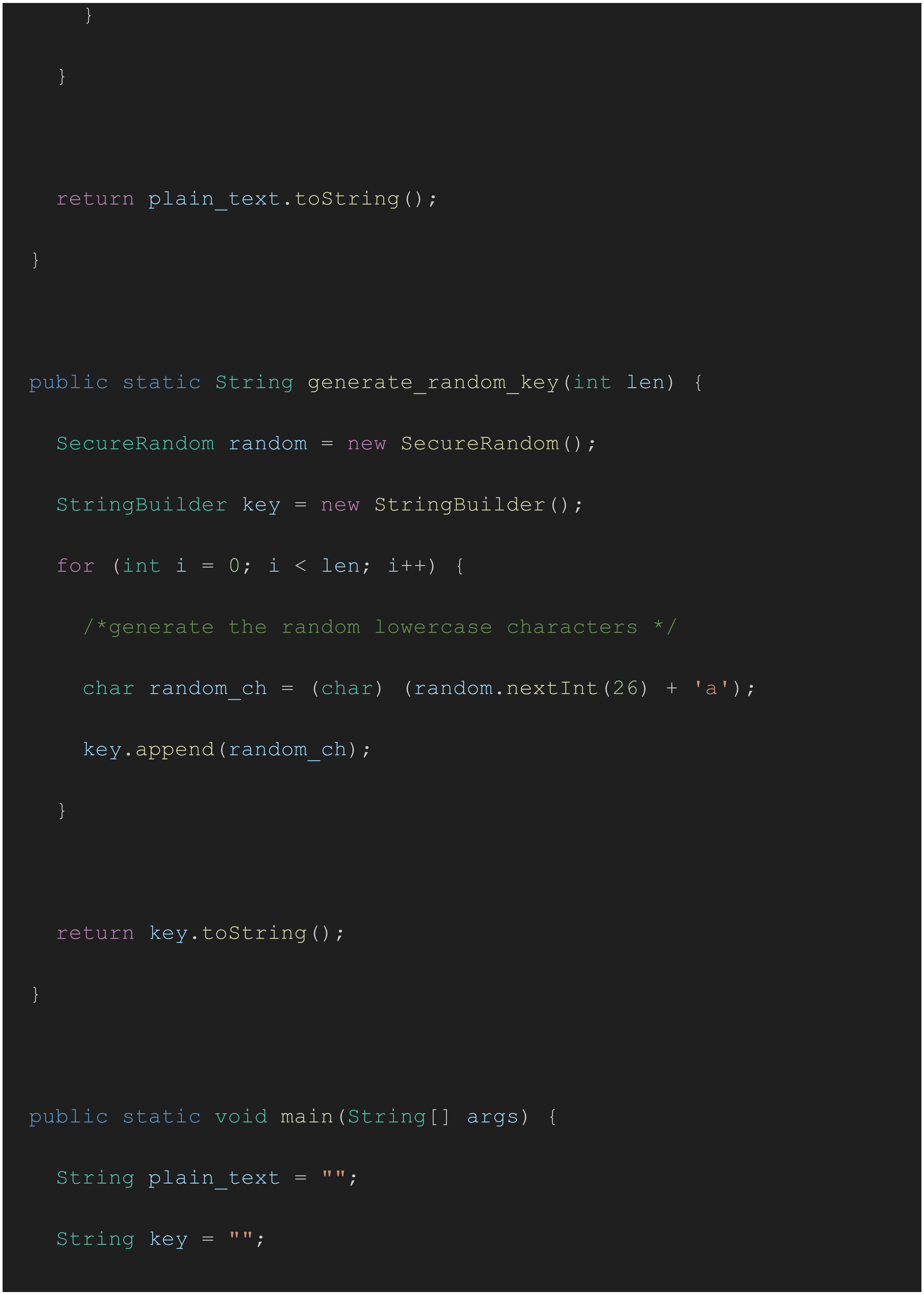
a) Input4- Ciphertext

b) Output4- Plaintext

A one-time pad (OTP) is a large non-repeating set of random key letters, written on sheets of paper, and glued together in a pad

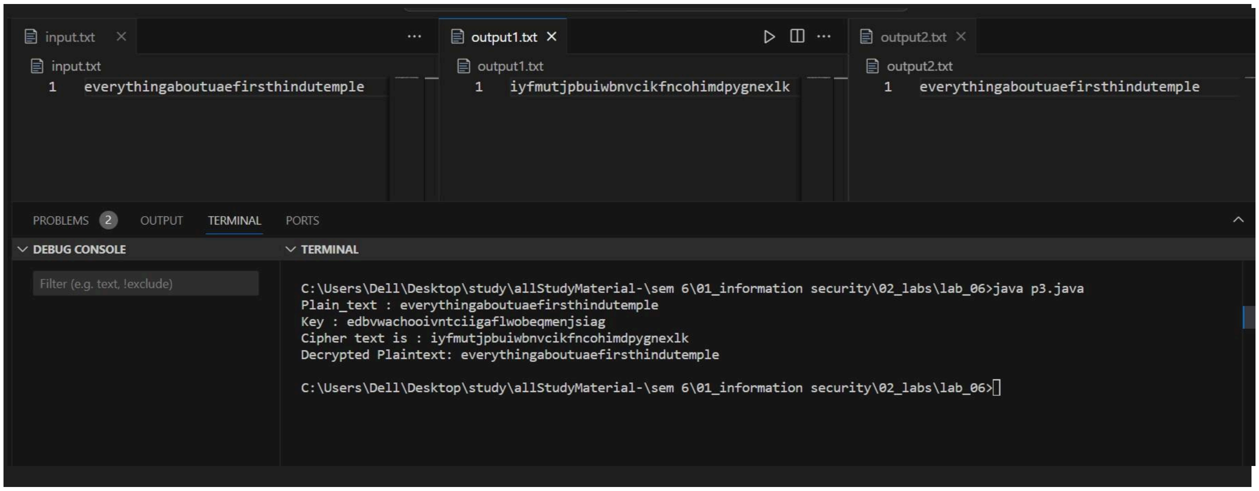


```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.security.SecureRandom;


public class p3 {


  public static String encrypt(String plain_text, String key) {
    plain_text = plain_text.toLowerCase();
    StringBuilder cipher_text = new StringBuilder();
    int key_length = key.length();
    int idx = 0;
    for (char ch : plain_text.toCharArray()) {
      if (Character.isLetter(ch)) {
        int shift = key.charAt(idx) - 'a';
        char encrypted_ch = (char) (((ch - 'a' + shift) % 26) + 'a');
        cipher_text.append(encrypted_ch);
        idx = (idx + 1) % key_length;
      } else {

        cipher_text.append(ch);

      }

    }

    return cipher_text.toString();

  }



  public static String decrypt(String cipher_text, String key) {

    StringBuilder plain_text = new StringBuilder();

    int key_length = key.length();

    int idx = 0;



    for (char ch : cipher_text.toCharArray()) {

      if (Character.isLetter(ch)) {

        int shift = key.charAt(idx) - 'a';

        char decrypted_ch = (char) (((ch - 'a' - shift + 26) % 26) + 'a');

        plain_text.append(decrypted_ch);

        idx = (idx + 1) % key_length;

      } else {

        plain_text.append(ch);
```



```java
    }
  }

  return plain_text.toString();
}

public static String generate_random_key(int len) {
  SecureRandom random = new SecureRandom();
  StringBuilder key = new StringBuilder();
  for (int i = 0; i < len; i++) {
    /*generate the random lowercase characters */
    char random_ch = (char) (random.nextInt(26) + 'a');
    key.append(random_ch);
  }

  return key.toString();
}

public static void main(String[] args) {
  String plain_text = "";
  String key = "";
```

```
    try {

      BufferedReader br = new BufferedReader(new
FileReader("input.txt"));


      plain_text = br.readLine();

      /*GENERATE THE RANDOM KEY OF THE LENGTH OF THE PLAIN TEXT  */

      key = generate_random_key(plain_text.length());

      System.out.println("Plain_text : " + plain_text);

      System.out.println("Key : " + key);

      br.close();

    } catch (IOException e) {

      e.printStackTrace();

    }


    /*ENCRYPT THE PLAIN TEXT USING THE KEY. */

    String cipher_text = encrypt(plain_text, key);

    System.out.println("Cipher text is : " + cipher_text);


    /*WRITE THE CIPHER TEXT TO FILE. */

    try {
      BufferedWriter bw = new BufferedWriter(new
FileWriter("output1.txt"));
      bw.write(cipher_text);
      bw.close();
    } catch (IOException e) {
      e.printStackTrace();
    }

    /*READ INPUT CIPHER TEXT FROM THE FILE. */
    String input_cipher_text = "";

    try {
      BufferedReader br = new BufferedReader(new
FileReader("output1.txt"));
      input_cipher_text = br.readLine();
      br.close();
    } catch (IOException e) {
      e.printStackTrace();
    }

    /* DECRYPT CIPHERTEXT USING KEY. */
    String decrypted_plain_text = decrypt(input_cipher_text, key);
    // System.out.println("Decrypted Plain text: ",
decrypted_plain_text);

    System.out.print("Decrypted Plaintext: ");

    System.out.println(decrypted_plain_text);



    try {

      BufferedWriter bw = new BufferedWriter(new
FileWriter("output2.txt"));

      bw.write(decrypted_plain_text);

      bw.close();

    } catch (IOException e) {

      e.printStackTrace();

    }

  }

}
```



input.txt

```
everythingaboutuaefirsthindutemple
```

output1.txt

```
iyfmutjpbuiwbnvcikfncohimdpygnexlk
```

output2.txt

```
everythingaboutuaefirsthindutemple
```

```
C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_06>java p3.java
Plain_text : everythingaboutuaefirsthindutemple
Key : edbvwachooivntciigaflwobeqmenjsiag
Cipher text is : iyfmutjpbuiwbnvcikfncohimdpygnexlk
Decrypted Plaintext: everythingaboutuaefirsthindutemple

C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\lab_06>
```