```java

import java.io.*;
import java.util.*;

/**
 * This program encrypts or decrypts files and prints content in different file.
 *
 * @author WahabEhsan
 */
public class Cipher {

    /**
     * Global variable for the number of letters in English alphabet.
     */
    public static final int Alphabet = 26;

    /**
     * Runs the program
     *
     * @param args not used
     */
    public static void main(String[] args) {
        userInterface();
    }

    /**
     * This method prompts user and asks if they want to encrypt, decrypt or
     * quit.
     */
    public static void userInterface() {
        Scanner input = new Scanner(System.in);
        boolean loop = true;
        boolean encrypte;
        while (loop) {
            System.out.print("Would you like to (E)ncrypte or (D)ecrypte? or (Q)uit? ");
            String in = input.nextLine();
            in = in.toLowerCase();
            if (in.equals("q")) { //if "q" entered, program quits
                loop = false;
            } else if (in.equals("d")) { //if "d" entered, sets encrypt to false and runs the procedure
                encrypte = false;
                procedure(encrypte, input);
            } else if (in.equals("e")) { //if "d" entered, sets encrypt to true and runs the procedure
                encrypte = true;
                procedure(encrypte, input);
            } else { //else says "invalid input"
                System.out.println("Invalid input");
            }

        }

    }

    /**
     * This method runs the methods depending on if encrypt or decrypt
     *
     * @param encrypte boolean value true for encrypt or false for decrypt
     * @param input The input as scanner
     */
    public static void procedure(boolean encrypte, Scanner input) {

        String key = getKey(input); //Sets input to key
        Scanner fileReader = getInputScanner(input); //makes filereader scanner
        PrintStream fileWriter = getOutputPrintStream(input); //makes filewriter printstream
        processFile(encrypte, key, fileReader, fileWriter); //processes file

    }

    /**
     * Prompts the user for and returns a valid key for encryption/decryption.
     *
     * @param console Scanner for input
     * @return Key is returned after is handled
     */
    public static String getKey(Scanner console) {
        boolean perm = true;
        String key = null;
        while (perm) { //runs while loop as long as variable perm is true
            System.out.print("Key? ");
            key = console.nextLine(); //prompts user for key
            if (key.length() == 0) { //if key has no length, prompts again
                System.out.println("Please Enter Lowercase letters.");
                continue;
            }
            perm = handlingKey(key, console); //runs handlingKey method
            if (perm == false) { //if perm returned is false, breaks loop
                break;
            }

        }

        return key;

    }

    /**
     * Handles the key entered, prints error if the input is not lowercase
     * letters
     *
     * @param key The key entered in getKey()
     * @param console Scanner input
     * @return
     */
    public static boolean handlingKey(String key, Scanner console) {
        boolean perm = false;
        for (int i = 0; i < key.length(); i++) {
            char c = key.charAt(i); //Scans every characterto make sure not uppercase or unicode
            if (!Character.isLowerCase(c) || (Character.UnicodeBlock.of(c)
                    != Character.UnicodeBlock.BASIC_LATIN)) {
                //if uppercase or unicode, breaks loop and sets perm to true
                System.out.println("Please Enter Lowercase letters.");
                perm = true;
                break;
            }
        }
        return perm;
    }

    /**
     * Returns Scanner for an input file Use a try/catch block to catch and
     * handle any FileNotFoundException's that occur
     *
     * @param console Scanner input
     * @return File is returned after valid file is found
     */
    public static Scanner getInputScanner(Scanner console) {
        Scanner file = null;
        try {
            File f; //makes file object
            do {
                System.out.print("Enter Input File: ");
                String input = console.nextLine();
```

```java
                f = new File(input);
                if (!f.exists()) { //if file not found, error message appears
                    System.out.println("File not Found");
                }

            } while (!f.exists()); //runs do-while loop until file found
            file = new Scanner(f); //passes file into scanner

        } catch (FileNotFoundException ex) { //catches FileNotFoundExceptions
            System.out.print("File not found");
        }
        return file; //returns scanner for file
    }

    /**
     * Returns PrintStream for output file Use a try/catch block to catch and
     * handle any FileNotFoundException's that occur
     *
     * @param console Scanner input
     * @return File PrintStream returned after checked with user if file already
     * exist and if they would like to overwrite
     */
    public static PrintStream getOutputPrintStream(Scanner console) {
        boolean perm = true;
        File f;
        PrintStream file = null;
        while (file == null) { //runs while loop until file no more null
            try {
                System.out.print("Enter Output file: ");
                String input = console.nextLine();
                f = new File(input);
                if (f.exists()) { //if file exists, informs user it exists
                    System.out.println("File already exists.");
                    perm = outputHandling(perm); //then runs outputhandling method
                    if (perm == false) { //if perm returned is false, re starts the loop
                        continue;
                    }
                }
                file = new PrintStream(f);//sets printstream
            } catch (FileNotFoundException ex) {//catches FilenotfoundException
                System.out.print("File not found");
            }
        }

        return file;//returns printstream

    }

    /**
     * This method asks the user if they would like to overwrite the file.
     *
     * @param perm boolean value to continue or re-prompt for another file
     * @return Boolean value to continue or re-prompt for another file or
     * overwrite
     */
    public static boolean outputHandling(boolean perm) {
        boolean loop = true;
        Scanner input = new Scanner(System.in);
        while (loop) { //runs while loop until valid answer given
            System.out.println("Would you like to overwrite? (Y/N)");
            String request = input.next();
            request = request.toLowerCase(); //sets input to lowercase
            if (request.contains("y")) { //if "y" entered, it breaks the loop and sets perm true
                perm = true;
                loop = false;
            } else if (request.contains("n")) { //if "n" entered, it breaks the loop and sets perm false
```

```java
                perm = false;
                loop = false;
            } else { //if anything else entered, prints "invalid input".
                System.out.println("Invalid input");
            }
        }
        return perm; //returns the value of boolean if want to overwrite or not.
    }

    /**
     * If encrypt is true, encrypts message in input and outputs encrypted
     * message based on key, If encrypt is false, decrypts message in input and
     * outputs decrypted message based on key.
     *
     * @param encrypt Boolean value for encrypt if true or decrypt if false,
     * @param key String that is processed and checked if valid
     * @param input Scanner input for file
     * @param output Scanner output for PrintStream file
     */
    public static void processFile(boolean encrypt, String key, Scanner input, PrintStream output) {
        String content;
        while (input.hasNextLine()) { //runs while loop until input has line
            String line = input.nextLine(); //turns scanner to line
            if (encrypt == true) { //if encrpt was true, runs encrypt method
                content = encryptLine(key, line);
            } else { //if decrypt was true, runs decrypt method
                content = decryptLine(key, line);
            }
            output.println(content);//prints content on the printstream file
        }
        output.close();//closes output
        input.close();//closes input

    }

    /**
     * Returns string containing line encrypted using key
     *
     * @param key The String value the shifts the characters for the line
     * @param line The Line that processes through the input file and is shifted
     * according to the key
     * @return Sentence that are returned and added on
     */
    public static String encryptLine(String key, String line) {
        String sen = "";
        char cas = 0;

        for (int i = 0, j = 0; i < line.length(); i++) { //runs for loop for lenght of line
            char c = line.charAt(i);
            if (c == '\n' || c == '\r') { // if char c is a newline, prints content and continues
                j = 0; //sets key letter to the first letter for the new line
                sen += c;
                continue;
            }
            if (c < 'a' || c > 'z') { //if char c is not lowercase goes to this loop
                if (Character.isUpperCase(c)) { //if char c is uppercase
                    c = Character.toLowerCase(c); //sets char to c for tempraraly
                    cas = (char) ((c + key.charAt(j) - 2 * 'a') % Alphabet + 'a');//shifts the char value depending on key
                    cas = Character.toUpperCase(cas); //sets char back to uppercase
                    sen += cas;//adds char to the rest of the line
                    j = ++j % key.length();//increases to the next key character
                    continue;
                }

                sen += c;//if not a letter, just adds to the line
                continue;
            }
            sen += (char) ((c + key.charAt(j) - 2 * 'a') % Alphabet + 'a');
            //if lowercase letter shifts and adds to line
            j = ++j % key.length();
        }
        return sen; //returns sentence
    }

    /**
     * Returns string containing line decrypted using key
     *
     * @param key The String value the shifts the characters for the line
     * @param line The Line that processes through the input file and is shifted
     * according to the key
     * @return Sentence that are returned and added on
     */
    public static String decryptLine(String key, String line) {
        String sen = "";
        char cas = 0;
        for (int i = 0, j = 0; i < line.length(); i++) { //runs for loop for lenght of line
            char c = line.charAt(i);
            if (c == '\n' || c == '\r') { // if char c is a newline, prints content and continues
                j = 0;//sets key letter to the first letter for the new line
                sen += c;
                continue;
            }
            if (c < 'a' || c > 'z') { //if char c is not lowercase goes to this loop
                if (Character.isUpperCase(c)) { //if char c is uppercase
                    c = Character.toLowerCase(c); //sets char to c for tempraraly
                    cas = (char) ((c - key.charAt(j) + Alphabet) % Alphabet + 'a');
                    //shifts the char value depending on key
                    cas = Character.toUpperCase(cas); //sets char back to uppercase
                    sen += cas; //adds char to the rest of the line
                    j = ++j % key.length(); //increases to the next key character
                    continue;
                }
                sen += c;//if not a letter, just adds to the line
                continue;
            }
            sen += (char) ((c - key.charAt(j) + Alphabet) % Alphabet + 'a');
            //if lowercase letter shifts and adds to line
            j = ++j % key.length();
        }
        return sen;//returns sentence
    }

}
```