PROJECT WORK

ON

JAVA Program for maintaining a Phonebook

BY

IKEDINOBI ONYEKA

1300512

AND

ISAAC NKWAZEMA

1304793

February 2022

DECLARATION

This is to certify that this project work was done by **Ikedinobi Onyeka** and **Isaac Nkwazema** and to declare that this project represents our original work which has not been previously submitted by anyone.

ACKNOWLEDGEMENTS

I will like to acknowledge all those who have given support and helped make this project a success I also wish to express gratitude to my faculty for his valuable guidance and support for the completion of this project I finally would like to thank my colleague for the valuable roles they played to make the completion of this project a success.

ABSTRACT

In computer programming, create, read, update, and delete (CRUD) are the four basic functions of persistent storage. Alternate words are sometimes used when defining the four basic functions of CRUD, such as retrieve instead of read, modify instead of update, or destroy instead of delete. CRUD is also sometimes used to describe user interface conventions that facilitate viewing, searching, and changing information. In this project work we developed a console based java application for phonebook. Our implemented system can create, read, update and delete from the text file. We used Java because it is an object oriented programming language and because of its ability to write once and runs everywhere.

TABLE OF CONTENT

CHAPTER ONE

INTRODUCTION

1.1) Background to study

A database is generally used for storing related, structured data, with well defined data formats, in an efficient manner for insert, update and/or retrieval (depending on application).

On the other hand, a file system is a more unstructured data store for storing arbitrary, probably unrelated data. The file system is more general, and databases are built on top of the general data storage services provided by file systems.

The aim of the project is to develop an application to take in contacts details and store them in a text file.

1.2) Problem of definition

Your Program should allow user to "ADD", "DELETE", "MODIFY", "SEARCH", "Display All Records" and "COUNT"

records.

All data should be stored in text file and should be available whenever the program is executed.

1.3) Aim & Objectivies

We will be using the File system since it's more efficient than database. For simple operations, read, write and delete file operations are faster and simple with the file system

Objectives

- To add contact details
- To delete contact details
- To modify contact details
- To search for contact details using an inputted term
- To display all contact details
- To count number of contacts in the text file

1.4) Scope of study

Using a text file with the Java programming language

CHAPTER TWO

ANALYSIS AND DESIGN

Analysis emphasizes an investigation of the problem and requirements, rather than a solution. For example, if a new computerized library information system is desired, how will it be used?

"Analysis" is a broad term, best qualified, as in requirements analysis (an investigation of the requirements) or object analysis (an investigation of the domain objects).

Design emphasizes a conceptual solution that fulfills the requirements, rather than its implementation. For example, a description of a database schema and software objects. Ultimately, designs can be implemented.

As with analysis, the term is best qualified, as in object design or database design.

2.1) Analysis

File should have following data:

First Name

Last Name

Address

City

Phone Number

Search should be available on Last Name, City and Phone Number or combination of either two. Search should

display complete record.

Records should be properly displayed either GRID WISE [Page Scroll should be maintained] or One Record at a

time with option to move to next record or quit.

Proper validations should be maintained on all data fields.

Key Field should be Phone Number, hence no two records can have same phone number

2.1) Design

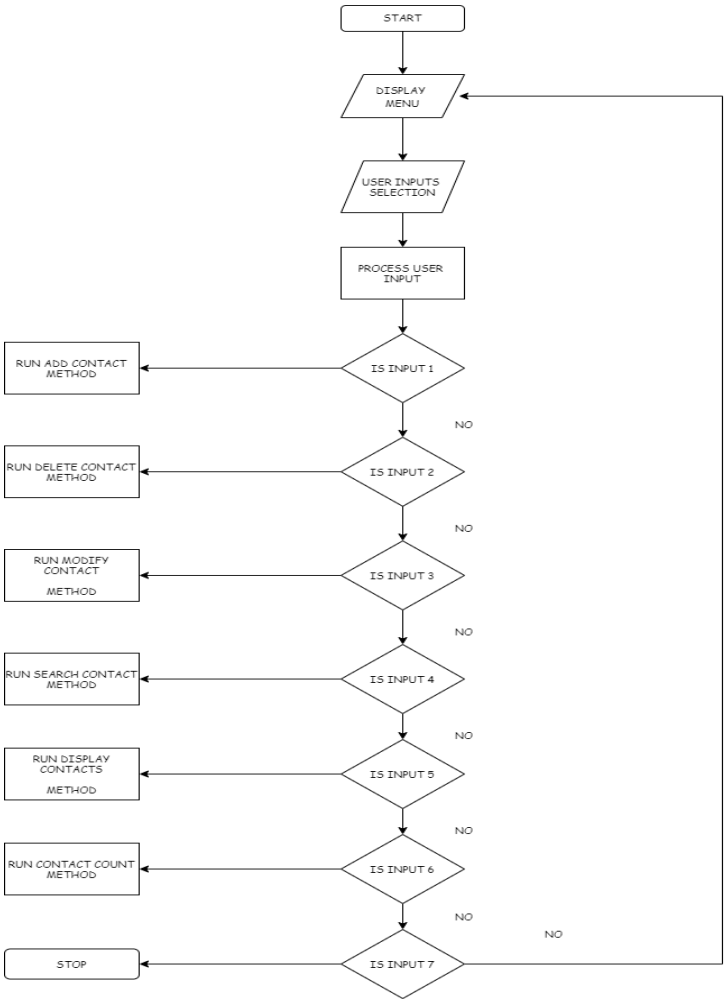
Flowchart for the Menu:

Step 1 : Display Menu

Step 2 : Ask User for Input

Step 3 : Process User's Input

Step 4 : If input is 1, run add contact method , else if input is 2, run delete contact method, else if input is 3, run modify contact method, else if input is 4, run search contact method, else if input is 5, run display contact method, else if input 6, run contact count method, else if input is 7, exit program, else go back to Step 1



Flowchart for Add Contact:

Step 1 : Specify Filename

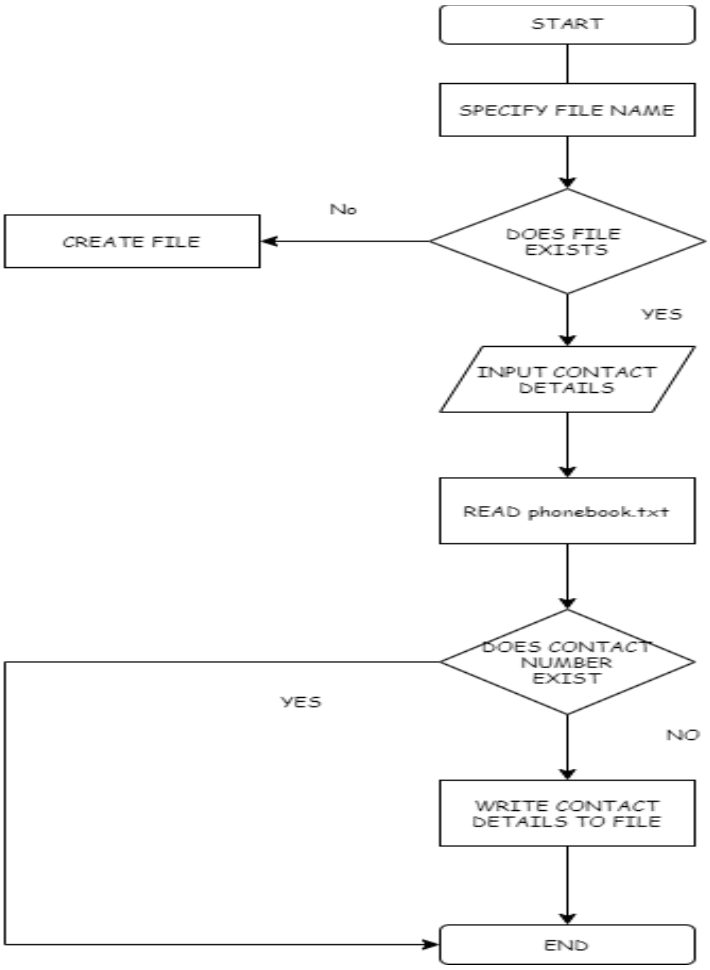
Step 2 : If file exists go to Step 3, else create file and go to Step 3

Step 3 : Prompt User for Contact details

Step 4 : Read phone.txt

Step 5 : If contact number exists break out, else move to Step 6

Step 6 : Write contact details to file



Flowchart for Delete Contact:

Step 1 : Specify Filename

Step 2 : Specify new Filename

Step 3 : Prompt User for contact number

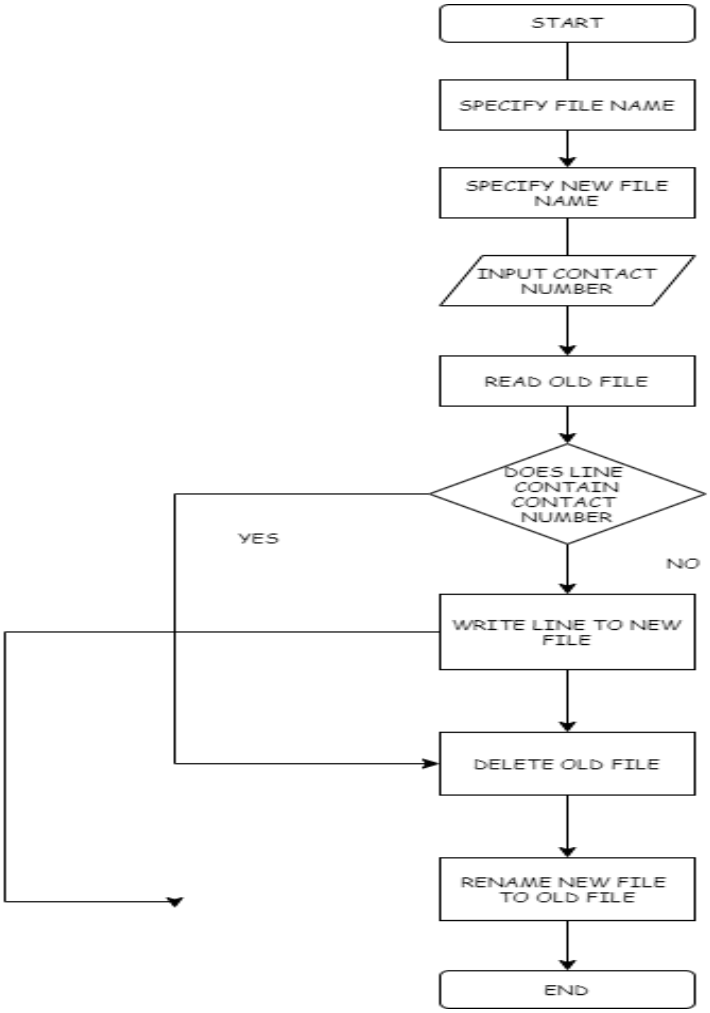
Step 4 : Read old file

Step 5 : If old contact line contain inputted number move to step 7, else move to Step 6

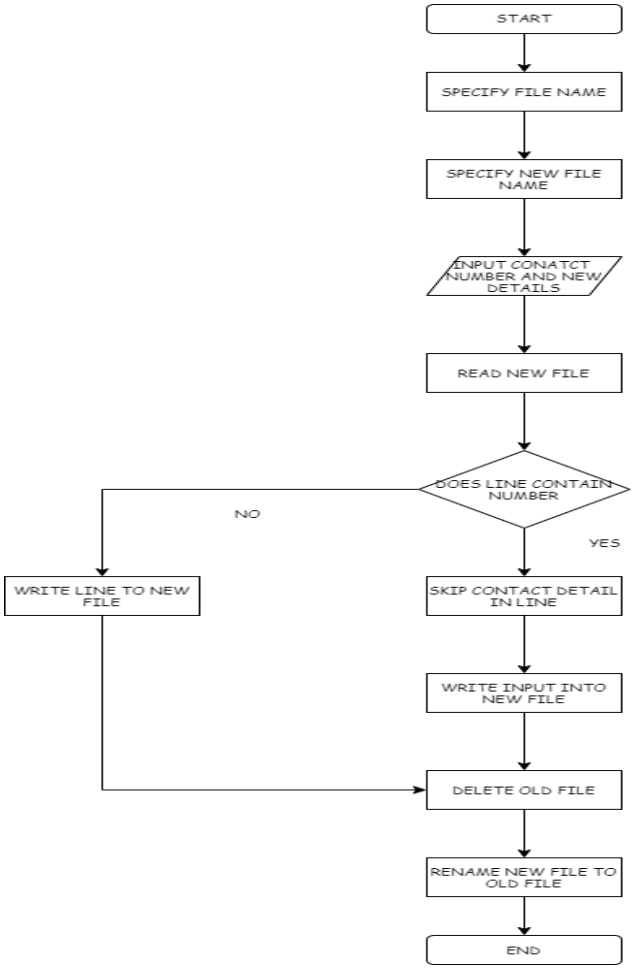
Step 6 : Write line to new file

Step 7 : Delete old file

Step 8 : Rename new file to old file



Flowchart for Modify Contact:



Flowchart for Search Contact:

Step 1 : Specify Filename

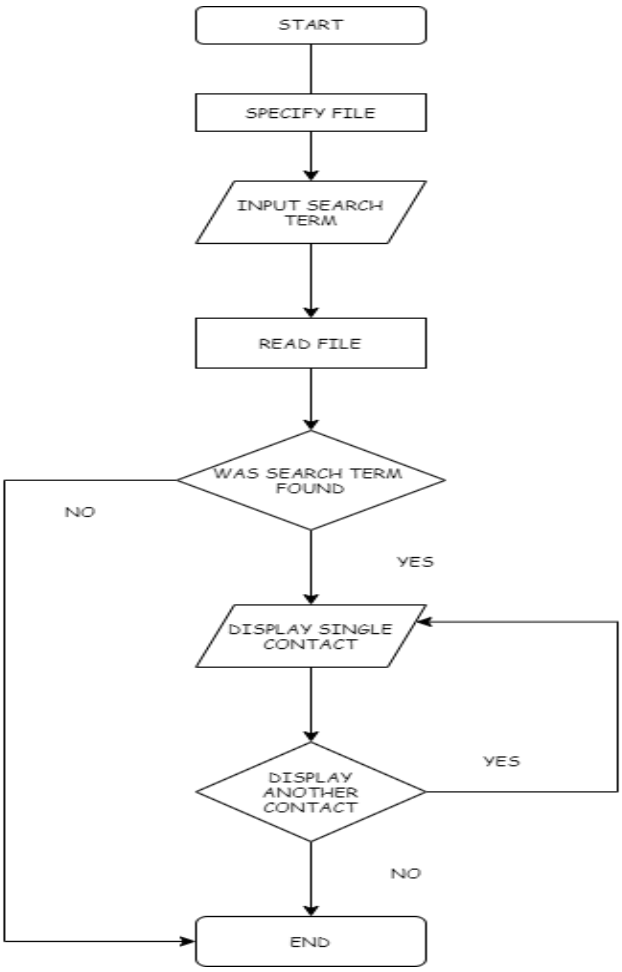
Step 2 : Prompt User for Search term

Step 3 : Read file

Step 4 : If search term was found, move to Step 5, else break out

Step 5 : Display single contact

Step 6 : if user wants to display another contact, move to Step 5, else end



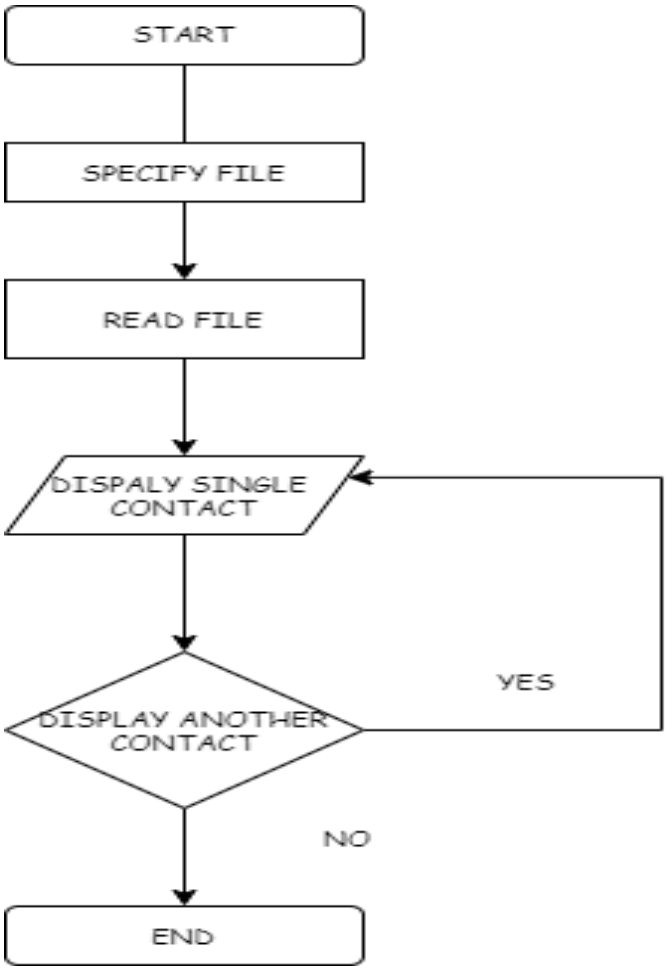
Flowchart for Display Contact:

Step 1 : Specify Filename

Step 2 : Read File

Step 3 : Display single contact

Step 4 : if user wants to display another contact, move to Step 3, else end



CHAPTER THREE

IMPLEMENTATION

3.1 Language Justification

Java has significant advantages over other languages that makes it suitable for just about any programming task

The following are advantages of using Java

- Java is object-oriented which allows us to create reusable codes
- Java is platform-independent, The ability to run the same program on many different systems

3.2 System Requirement

The requirements needed for the program to run are:

3.2.1 Hardware Requirements

- **Dual Core Processor**: For best performance, we recommend a dual core processor like core i3, core i5 as minimum requirement.
- **Ethernet connection (LAN) or a Wireless Adapter (Wi-Fi):** The system needs internet connection to work.
- **Memory (RAM):**2GB RAM capacity and above will enable the system run smoothly
- **Processor speed**: 2.0 GHz and above
- **Operating System**: 32-bit or 64bits
- **Hard disk:** Size of 60GB and above

- **File system:** NTFS file system is required for the hard disk.

3.2.2 Software Requirements

- JRE installed
- Notepad or any Java enabled IDE
- Operating System (OS): Microsoft Windows 7 or higher, Mac, ubuntu and so on

3.3 User Manual

1) Open up the phoneBook Java program with the required IDE

2) Locate the main.java class and run

3) An Interactive console interface menu will open up for user, Enter 1 to add new contact, 2 to delete contact, 3 to modify contact, 4 to search for contact, 5 to display contact, 6 to get number of contacts and 7 to exit the program

4) The remaining steps can easily be followed

5) Open up the phone.txt text file to check records

CHAPTER FOUR

CONCLUSION AND RECOMMENDATION

4.1 Conclusion

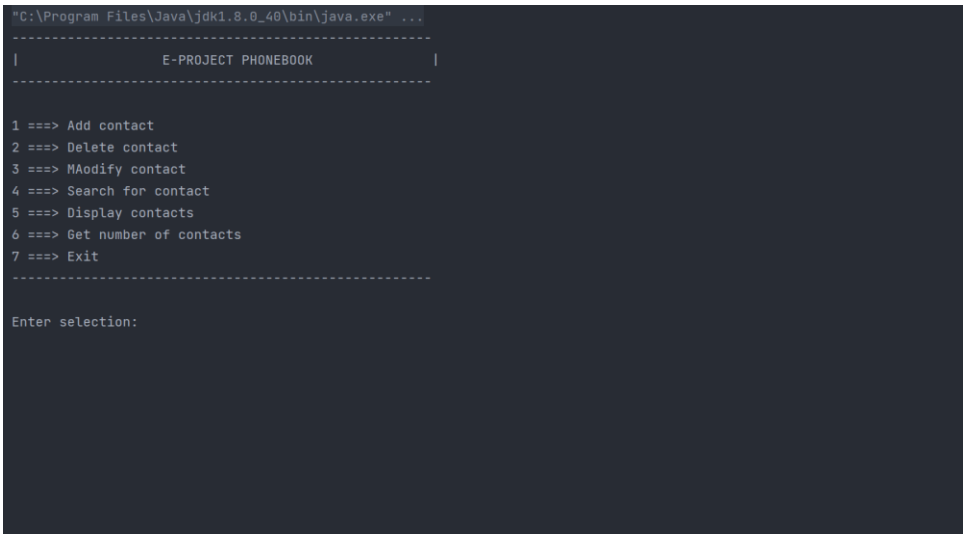
This project work was centered on developing a console phonebook application with Java programming language that executes crud. Without mincing words, we have achieved the aim and objectives of the project. We can affirmed that the project can now be deployed since all the test cases have been passed and equally met the stated objectives

4.2 Recommendation

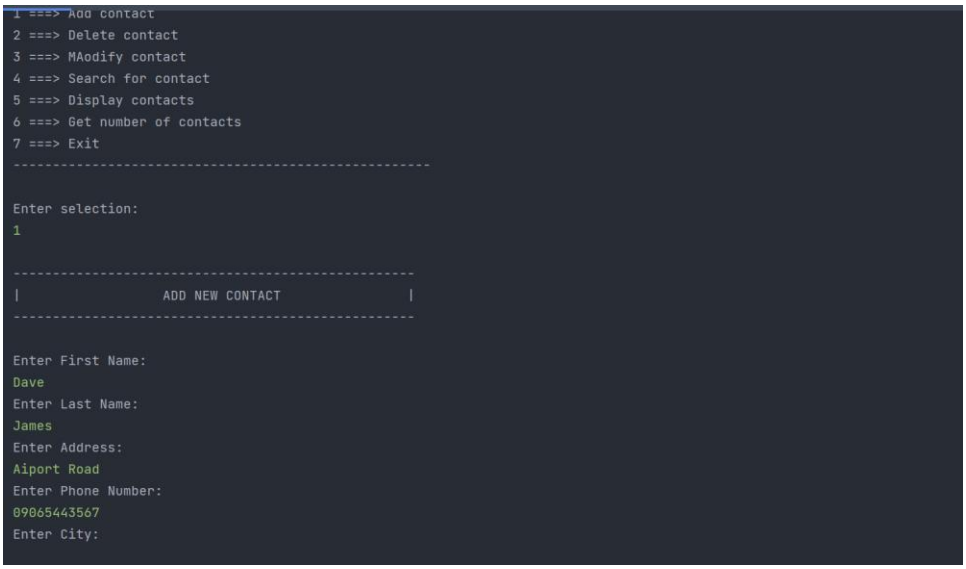
We hereby recommend that the future work should be based on graphical user interface instead of console application. Also, future works should find a way to make the crud operations with file to be more secured so that the data inside the file cannot be easily altered by an unauthorized person.

Appendix A – Screenshots

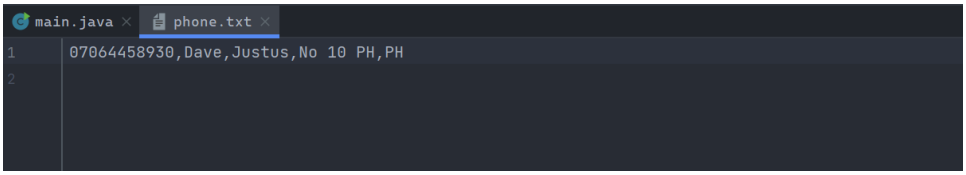
Program Console Interfaces:

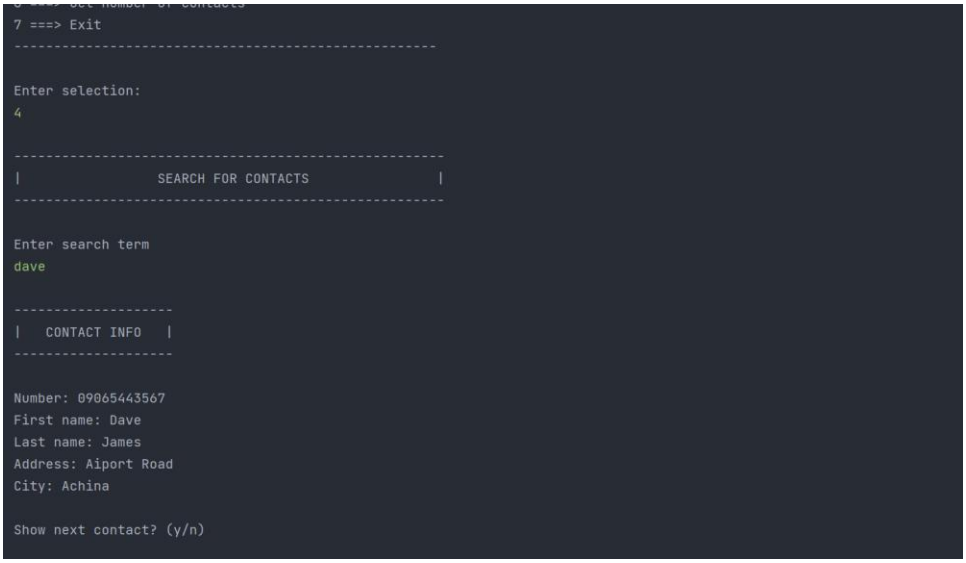


Saving Contact



Saved Contact





Appendix B– Source Code

ADD CONTACT FUNCTION

```
public static void addContact() throws IOException {

    System.out.println("-------------------------------------------------");
    System.out.println("|                 ADD NEW CONTACT                |");
    System.out.println("-------------------------------------------------\n");

    // instantiates the scanner class to be used across the method
    Scanner scan = new Scanner(System.in);
```

```
    //Specify the file name
    File file = new File("phone.txt");

    //Creates file if it doesn't exist already
    if (!file.exists()) file.createNewFile();

    // instantiates the file writer class and pass in file as argument
    FileWriter fwrite = new FileWriter(file,true);
    BufferedWriter bw = new BufferedWriter( fwrite );

    // instantiates the file reader class and pass in file as argument
    FileReader fread = new FileReader(file);
    BufferedReader br = new BufferedReader( fread );

    // declare variables to be used for the operation
    String fname, lname, address, city , number, record;

    // boolean variable to be used to check if contact exists
    boolean contactExists = false;

    // take in contact details
    System.out.println("Enter First Name: ");
    fname = scan.nextLine();

    System.out.println("Enter Last Name: ");
    lname = scan.nextLine();

    System.out.println("Enter Address: ");
    address = scan.nextLine();

    System.out.println("Enter Phone Number: ");
    number = scan.nextLine();

    System.out.println("Enter City: ");
    city = scan.nextLine();

    // loop through the file to check if contact already exists
    while( ( record = br.readLine() ) != null ) {
        // instantiate the stringtokenizer class to be used to split details from the
line been read from the file
        StringTokenizer st = new StringTokenizer(record,",");

        // convert boolean contactexists to true when a number in the file matches the
number inputted
        if (st.nextToken().equals(number)) {
            contactExists = true;
        }
    }

    // add contact to file if it doesn't exist in the file
```

```
    if (!contactExists) {
        // write to file
        bw.write(number+","+fname+","+lname+","+address+","+city);
        bw.flush();
        bw.newLine();
        // close buffered writer
        bw.close();
        System.out.println("Contact saved successfully!");
    } else {
        System.out.println("Phone Number already exists!!!");
    }

    System.out.println("");
    System.out.println("Exiting Contact Save...");

}
```

DELETE CONTACT FUNCTION

```
public static void deleteContact() throws IOException {

    System.out.println("-----------------------------------------------------");
    System.out.println("|                  DELETE CONTACT                   |");
    System.out.println("-----------------------------------------------------\n");

    // instantiates the scanner class to be used across the method
    Scanner scan = new Scanner(System.in);

    // declare variables to be used for operation
    String number, record;

    // boolean variable to be used to check if contact exists
    boolean found = false;

    // create new file for the operation
    File newFile = new File("new-phone.txt");

    // specify file name
    File file = new File("phone.txt");

    // instantiates the file reader class and pass in file as argument
    FileReader fread = new FileReader(file);
    BufferedReader br = new BufferedReader( fread );

    // instantiates the file writer class and pass in file as argument
    FileWriter fwrite = new FileWriter(newFile);
    BufferedWriter bw = new BufferedWriter( fwrite );
```

```
    System.out.println("Enter Contact Number: ");
    number = scan.nextLine();

    // loop through to search for number to be deleted
    while( ( record = br.readLine() ) != null ) {
        // instantiate the stringtokenizer class to be used to split details from the
line been read from the file
        StringTokenizer st = new StringTokenizer(record,",");

        // check if inputted number matches number of current contact that's being looped
through
        if (st.nextToken().equals(number)) {
            // convert boolean found to true when contact to be deleted is found
            found = true;

            // continue (skip the current loop) the loop when number is found, so it
won't be written to the new file
            continue;
        }

        // write records that don't match the number to be deleted to the new file
        bw.write(record);
        bw.flush();
        bw.newLine();
    }

    // close buffered reader
    br.close();
    // close buffered writer
    bw.close();

    // delete the old file
    file.delete();
    // rename new file to old file
    newFile.renameTo(file);

    // print success or error message if number was found
    if (!found) System.out.println("No contact with number "+number+" found!!");
    else System.out.println(number+" deleted successfully!");

    System.out.println("");
    System.out.println("Exiting Contact Delete...");
}
```

MODIFY CONTACT FUNCTION

```
public static void modifyContact () throws IOException{
```

```
        System.out.println("-----------------------------------------------------");
        System.out.println("|                   MODIFY CONTACT                  |");
        System.out.println("-----------------------------------------------------\n");

        // instantiates the scanner class to be used across the method
        Scanner scan = new Scanner(System.in);

        // declare variables to be used for operation
        String fname, lname, address, city, number, record;

        // boolean variable to be used to check if contact exists
        boolean found = false;

        // specify file name
        File file = new File("phone.txt");

        // create new file for the operation
        File newFile = new File("new_phone.txt");

        // instantiates the file reader class and pass in file as argument
        FileReader fread = new FileReader(file);
        BufferedReader br = new BufferedReader( fread );

        // instantiates the file writer class and pass in file as argument
        FileWriter fwrite = new FileWriter(newFile);
        BufferedWriter bw = new BufferedWriter( fwrite );

        // take in contact details
        System.out.println("Enter contact number to be modified: ");
        number = scan.nextLine();

        System.out.println("Enter new Firstname: ");
        fname = scan.nextLine();

        System.out.println("Enter new Lastname: ");
        lname = scan.nextLine();

        System.out.println("Enter new Address: ");
        address = scan.nextLine();

        System.out.println("Enter new City: ");
        city = scan.nextLine();

        // loop through to search for number to be modified
        while( ( record = br.readLine() ) != null ) {
            // instantiate the stringtokenizer class to be used to split details from the
line been read from the file
            StringTokenizer st = new StringTokenizer(record,",");

            // check if inputted number matches number of current contact that's being looped
through
```

```
        if (st.nextToken().equals(number)) {
            // convert boolean found to true when contact to be modified is found
            found = true;

            // write new contact details to the new file created
            bw.write(number+","+fname+","+lname+","+address+","+city);
            bw.flush();
            bw.newLine();

            // continue (skip the current loop) the loop when number is found, so it
won't be written to the new file
            continue;
        }

        // write records that don't match the number to be modified to the new file
        bw.write(record);
        bw.flush();
        bw.newLine();
    }

    // close buffered reader
    br.close();
    // close buffered writer
    bw.close();

    // delete the old file
    file.delete();
    // rename new file to old file
    newFile.renameTo(file);

    // print success or error message if number was found
    if (!found) System.out.println("No contact with number "+number+" found!!");
    else System.out.println(number+" Modified successfully!");

    System.out.println("");
    System.out.println("Exiting Contact Update...");
}
```

ADD CONTACT FUNCTION

```
public static void searchContact() throws IOException {

    System.out.println("-----------------------------------------------------");
    System.out.println("|                SEARCH FOR CONTACTS                |");
    System.out.println("-----------------------------------------------------\n");

    // instantiates the scanner class to be used across the method
    Scanner scan = new Scanner(System.in);
```

```
    // declare variables to be used for operation
    String term, record, choice;

    // boolean variable to be used to check if any contact exists
    boolean found = false;

    // instantiates the file reader class and pass in file as argument
    FileReader fread = new FileReader("phone.txt");
    BufferedReader br = new BufferedReader( fread );

    // take in search term
    System.out.println("Enter search term");
    term = scan.nextLine();
    System.out.println("");

    // loop through to searching for numbers to be displayed
    subLoop: while( ( record = br.readLine() ) != null) {

        // instantiate the stringtokenizer class to be used to split details from the
line been read from the file
        StringTokenizer st = new StringTokenizer(record,",");

        // check if record contains the search term
        if( record.toLowerCase().contains(term.toLowerCase()) ) {

            // convert boolean found to true when term searched for is found
            found = true;

            // display current contact being looped through info
            System.out.println("--------------------");
            System.out.println("|   CONTACT INFO   |");
            System.out.println("--------------------\n");

            System.out.println("Number: "+st.nextToken());
            System.out.println("First name: "+st.nextToken());
            System.out.println("Last name: "+st.nextToken());
            System.out.println("Address: "+st.nextToken());
            System.out.println("City: "+st.nextToken()+"\n");


            // option to move to next record or quit
            System.out.println("Show next contact? (y/n)");
            choice = scan.nextLine();

            switch (choice) {
                case "y" :
                case "Y" :
                    continue subLoop;
                case "n" :
                case "N" :
                    break subLoop;
```

```
                default:
                    System.out.println("Wrong Input entered!");
            }
        }

    }

    // print error message if no contact was found
    if (!found) System.out.println("No contact matches "+term);

    // close buffered reader
    br.close();

    System.out.println("");
    System.out.println("Exiting Contact Search...");
}
```

DISPLAY CONTACT FUNCTION

```
public static void displayContacts () throws IOException {

    System.out.println("-----------------------------------------------------");
    System.out.println("|                  DISPLAY CONTACTS                 |");
    System.out.println("-----------------------------------------------------\n");

    // instantiates the scanner class to be used across the method
    Scanner scan = new Scanner(System.in);

    // declare variables to be used for operation
    String record,choice;

    // boolean variable to be used to check if any contact exists
    boolean found = false;

    // instantiates the file reader class and pass in file as argument
    FileReader fread = new FileReader("phone.txt");
    BufferedReader br = new BufferedReader( fread );

    // loop through all numbers to be displayed
    subLoop: while( ( record = br.readLine() ) != null) {

        // instantiate the stringtokenizer class to be used to split details from the
line been read from the file
        StringTokenizer st = new StringTokenizer(record,",");

        // convert boolean found to true when contacts exist
        found = true;
```

```
        // display current contact being looped through info
        System.out.println("--------------------");
        System.out.println("|   CONTACT INFO   |");
        System.out.println("--------------------\n");

        System.out.println("Number: "+st.nextToken());
        System.out.println("First name: "+st.nextToken());
        System.out.println("Last name: "+st.nextToken());
        System.out.println("Address: "+st.nextToken());
        System.out.println("City: "+st.nextToken()+"\n");

        // option to move to next record or quit
        System.out.println("Show next contact? (y/n)");
        choice = scan.nextLine();

        switch (choice) {
            case "y" :
            case "Y" :
                continue subLoop;
            case "n" :
            case "N" :
                break subLoop;
            default:
                System.out.println("Wrong Input entered!");
                continue subLoop;
        }

    }

    // print error message if no contact was found
    if (!found) System.out.println("Contact List empty");

    // close buffered reader
    br.close();

    System.out.println("");
    System.out.println("Exiting Contacts Display...");
}
```

COUNT CONTACT FUNCTION

```
public static void contactCount () throws IOException {

    System.out.println("-----------------------------------------------------");
    System.out.println("|                 NUMBER OF CONTACTS                |");
    System.out.println("-----------------------------------------------------\n");

    // instantiates the scanner class to be used across the method
```

```
    Scanner scan = new Scanner(System.in);

    // declare a count variable to be increased when contacts are found
    int count = 0;

    // instantiates the file reader class and pass in file as argument
    FileReader fread = new FileReader("phone.txt");
    BufferedReader br = new BufferedReader( fread );

    // loop through all numbers for counting
    subLoop: while( br.readLine() != null) {

        // increment count whenever a number is found
        count++;

    }

    // display number of contacts found
    System.out.println(count+" Contacts exists");
    br.close();

    System.out.println("");
    System.out.println("Exiting Contact Count...");
}
```

MENU

```
Scanner scan = new Scanner(System.in);

// while loop to display option for user to pick from
mainLoop: while(true) {
    System.out.println("-------------------------------------------------------");
    System.out.println("|                  E-PROJECT PHONEBOOK                 |");
    System.out.println("-------------------------------------------------------\n");

    System.out.println("1 ===> Add contact \n" +
            "2 ===> Delete contact \n" +
            "3 ===> Modify contact \n" +
            "4 ===> Search for contact \n" +
            "5 ===> Display contacts \n" +
            "6 ===> Get number of contacts \n" +
            "7 ===> Exit");


    System.out.println("-------------------------------------------------------\n");

    // take in selected option
    System.out.println("Enter selection: ");
```

```
        int option = scan.nextInt();
        System.out.println("");

        // run methods as per user selection
        switch (option) {
            case 1:
                addContact();
                break;
            case 2:
                deleteContact();
                break;
            case 3:
                modifyContact();
                break;
            case 4:
                searchContact();
                break;
            case 5:
                displayContacts();
                break;
            case 6:
                contactCount();
                break;
            case 7:
                System.out.println("Exiting Program....");
                break mainLoop;
            default:
                System.out.println("Invalid Input");
        }

        System.out.println("");
    }
```

TASK SHEETS

1) Create a menu for user to pick action to perform

2) Create Add contact Method for adding contacts

3) Create Delete contact Method for deleting contact

4) Create Modify contact Method for editing contact

5) Create Search contact Method for searching for contact

6) Create Display contact Method for displaying contact

7) Create Contact count Method for counting number of contacts in file

8) Go through code check for errors and bugs