Institute of Computer Engineering Technology

iCET Certified Developer

Coursework

Assignement	Programming Fundamentals
Batch No	iCD 109
Name	iFriend Contact Organizer
Ass. Date	24th April 2024

iFriend Contact Organizer

Case Study

Contact Organizing is done in different ways. It is difficult to organize manually and you are assigned to create a Contacts Organizer to make it easier.

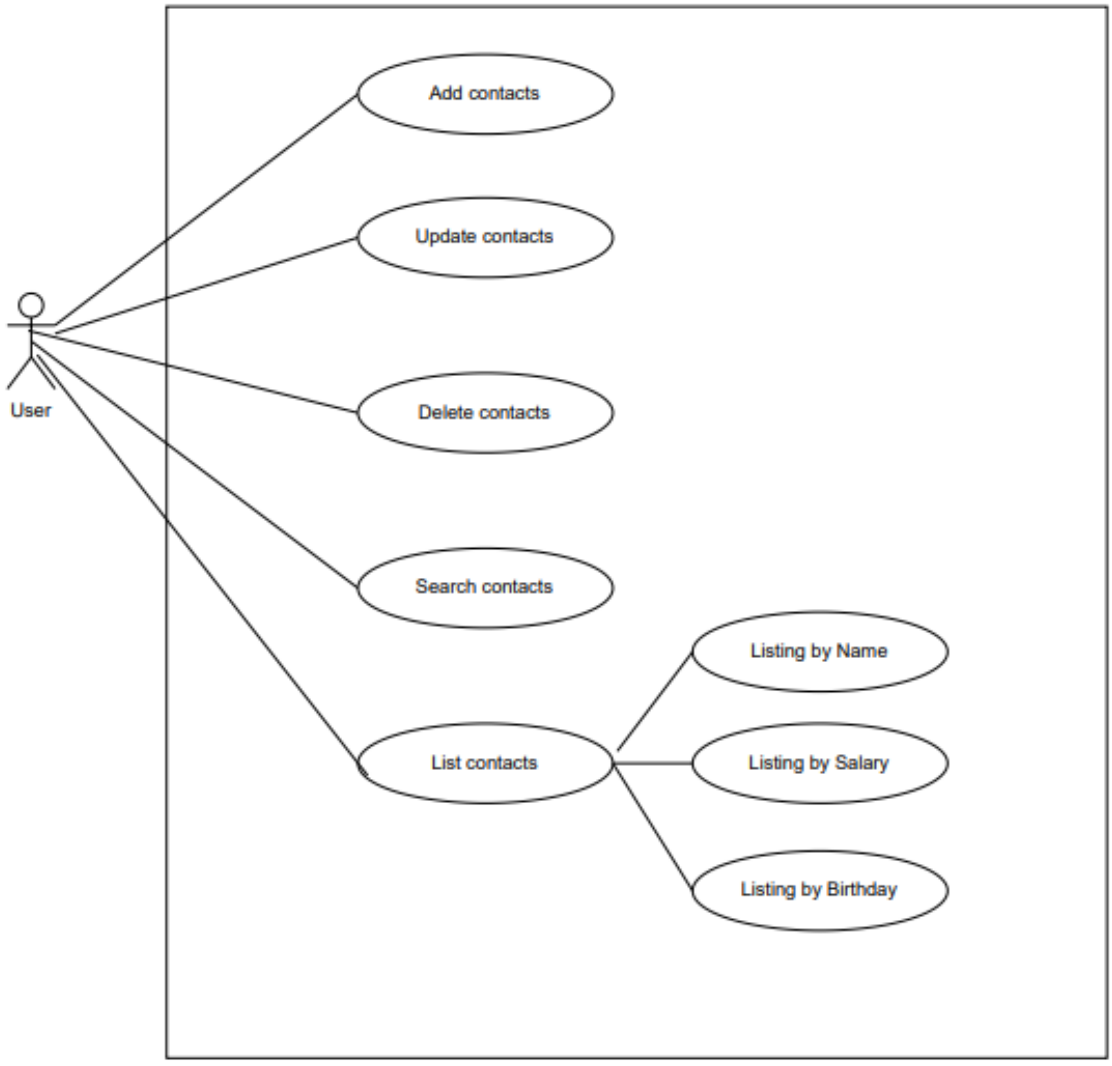

Figure 1 – Use case Diagram

Requirements

You are supposed to create a Java application to save Contacts Details. In the application, you need to implement the following use cases.

When you run the application, you should come up with something similar to the following Command Line Interface (CLI), where the user can enter an option number that he wants to execute. This will be the Home Page of the application that you will be developing.

Figure 2 - Home Page of the Contacts Organizer

01. ADD Contacts (Demo)

Adding a new Contacts is easy. The system keeps 6 details related to the Contacts. They are Contact ID, Name, Phone Number, Company, Salary, and Birthday.

- Contact ID – The Contact ID should be generated by the system and the Contact ID should start with 'C' and should have 4 numbers. When the user selects Add Contacts option on the home page when the Add Contact window is loaded, the Contact id should be generated by the system and the user should place the Contacts under that id. Contact Id can not generate by randomly and generate the next Contact Id accordingly to the last Contact Id. Contact Id can not be repeated.
- Name – The user should input Contact Name. It can be any name. it is okay if the name is repeated.
- Phone Number - When entering the phone number, it should be validated. Phone numbers should start with "0" and must have 10 numbers. If the user has entered an invalid phone number (as an example, start without "0" or the phone number has more than 10 numbers), the user should be kept prompted until he enters a valid phone number.
- Company - The user should input Contact Name. It can be any company name. it is okay if the company name is repeated.
- Salary - The user should input Salary. It should be a positive value.
- Birthday - The user should input Birthday. When entering the birthday, it should be validated and input birthday should be in YYYY-MM-DD format. Birthday should be a valid date, and should not be a future date.

After Adding a Contact successfully, asked from the user that "Do you want to add another Contact (Y/N): ", if the user enters "Y" the user can Add new Contact again and if user enters "N" user can go to the homepage.

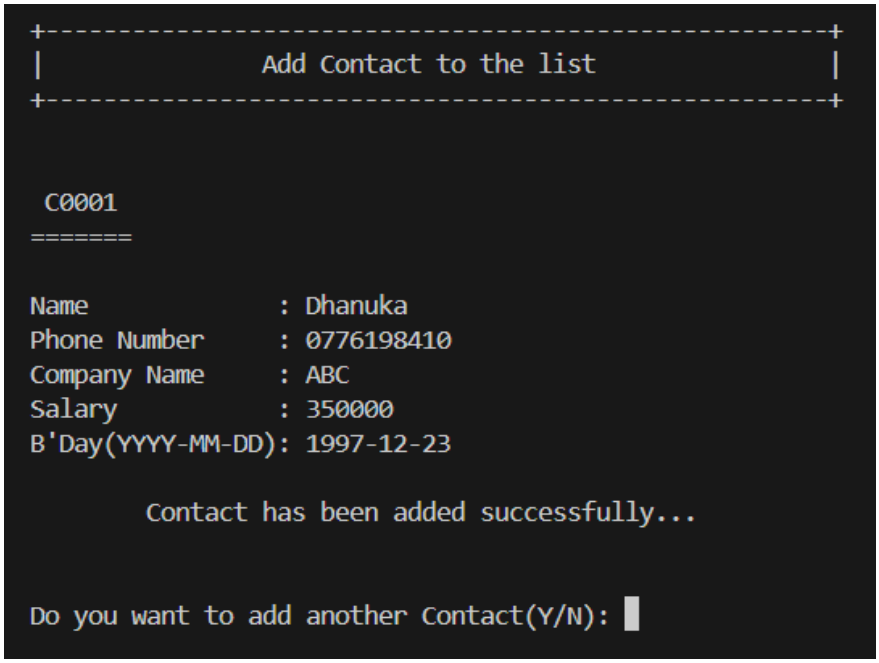

Figure 3 – Add Contact successfully

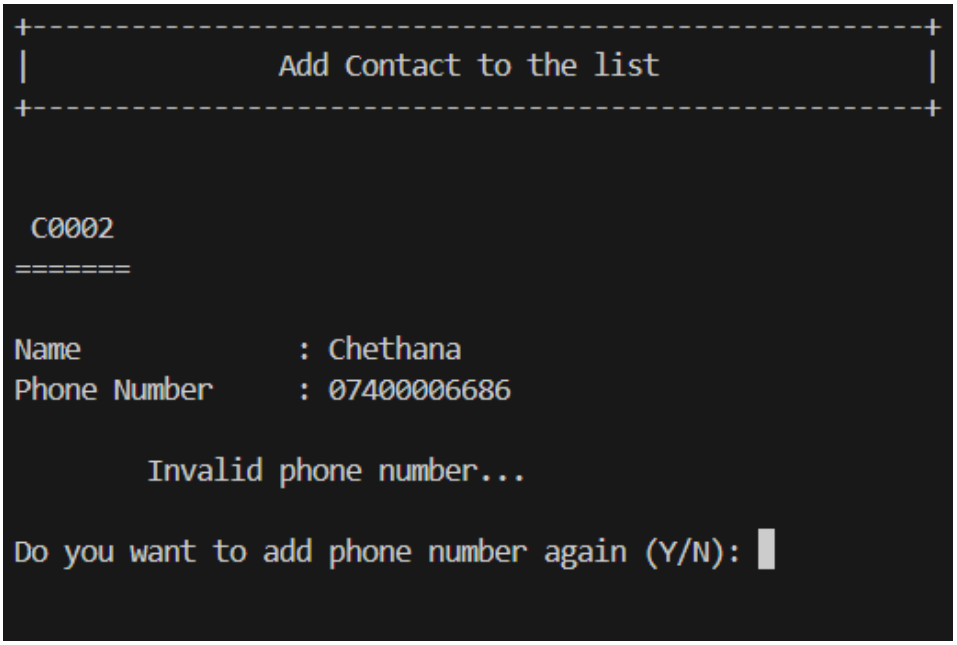

Figure 4 – Invalid phone number

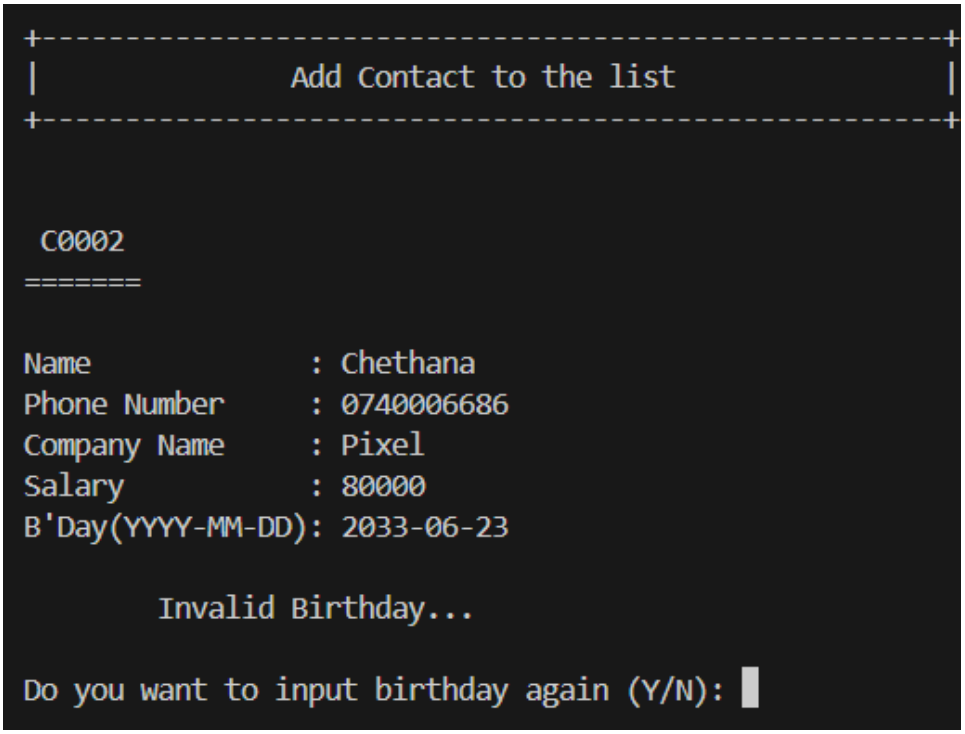

Figure 5 – Invalid birthday(future date)

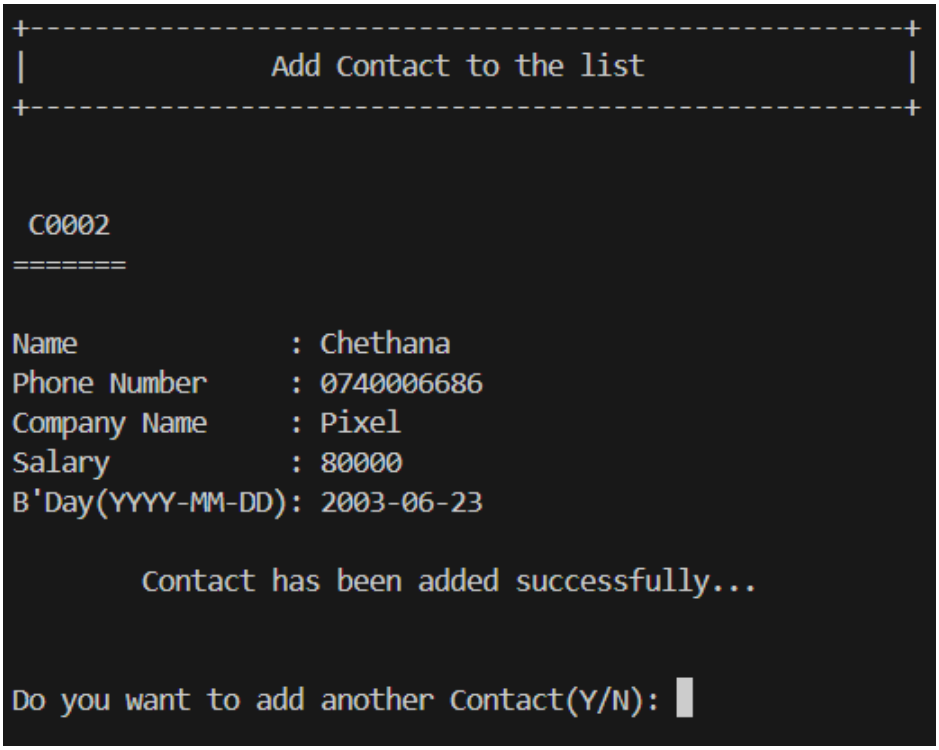

Figure 6 – Add Contact successfully

02. UPDATE Contacts (Demo)

With this, the user can update previously added Contact Details. First, the user needs to find the Contact by using Contact Name or Phone Number. After search contact, system should display all details about contact.

After that system should ask from the user what want to be update. Specially remember only Name, Phone Number, Company Name and Salary can be updated by the user. According to the user selected option, user can updated contact details and validation should be handled like previously.

Once the update has been done successfully, display update successfully massage and it should prompt whether to continue updating or go back to the main menu.

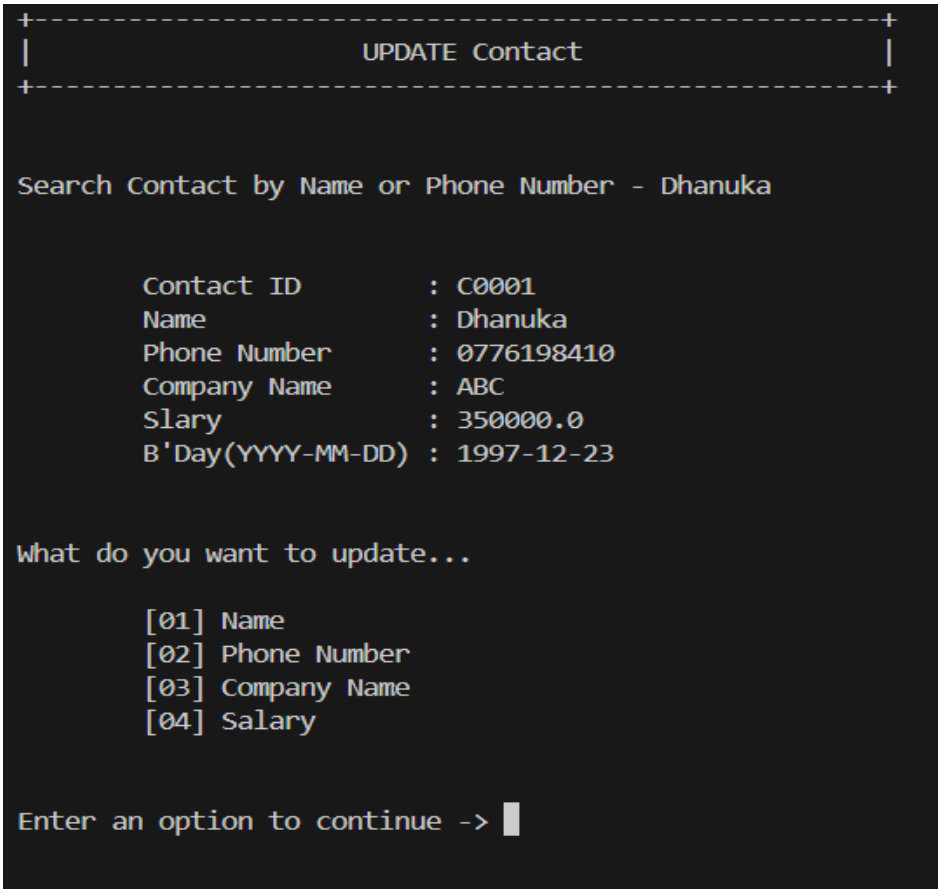

Figure 7 – Update contacts

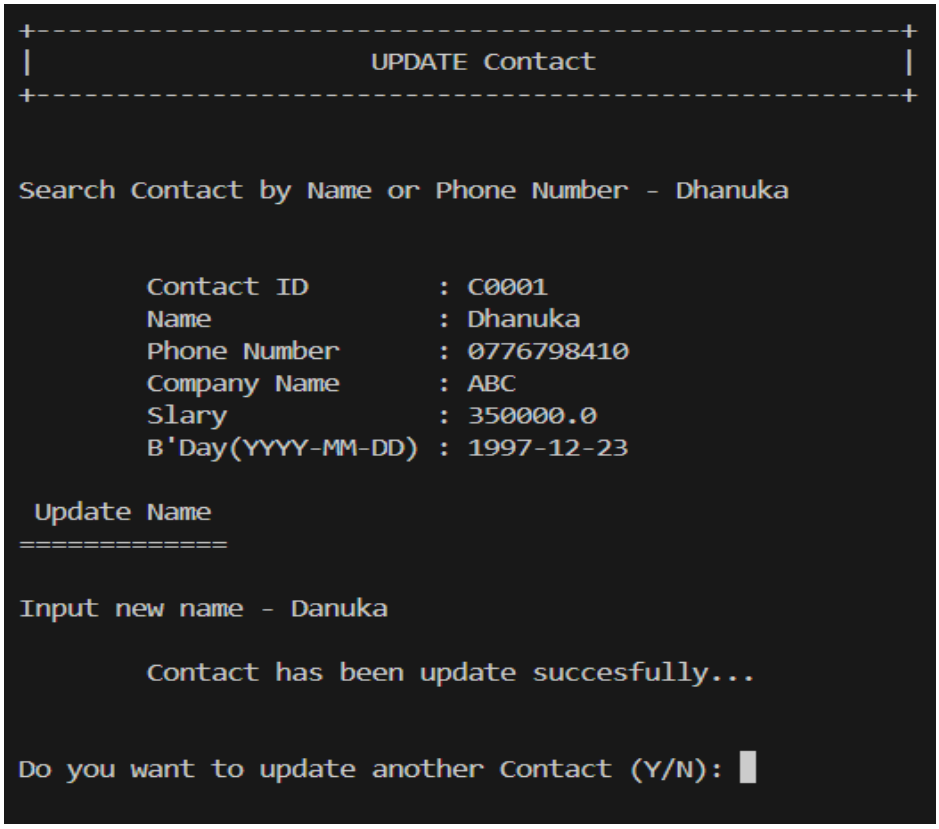

Figure 8 – Update Contact successfully

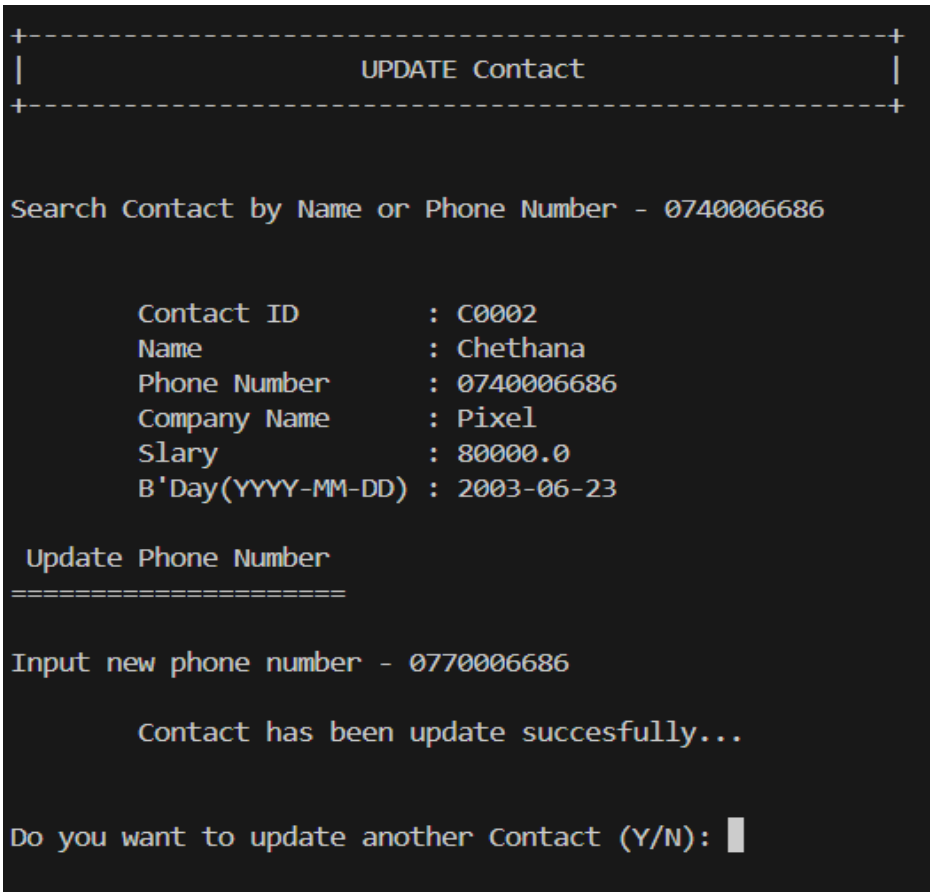

Figure 9 – Update phone number

03. DELETE Contacts (Demo)

With this option, a contact can be deleted from the system. First, the user needs to enter the Phone Number or Name. The phone Number should be validated. otherwise, it should handle like previously.

If the user has entered a valid Phone Number or the Name of a contact still hasn't been added to the system, then it should be notified as well as follows.

If the user enters a Phone Number or Name that is already in the system, all the related details of contact should be shown to the user. Before deleting the contact, a confirmation should be taken from the user to delete this contact. If the user confirms that this contact will be deleted, the contact should be successful delete. If the user does not confirm that this contact will be deleted, the contact should not be deleted.

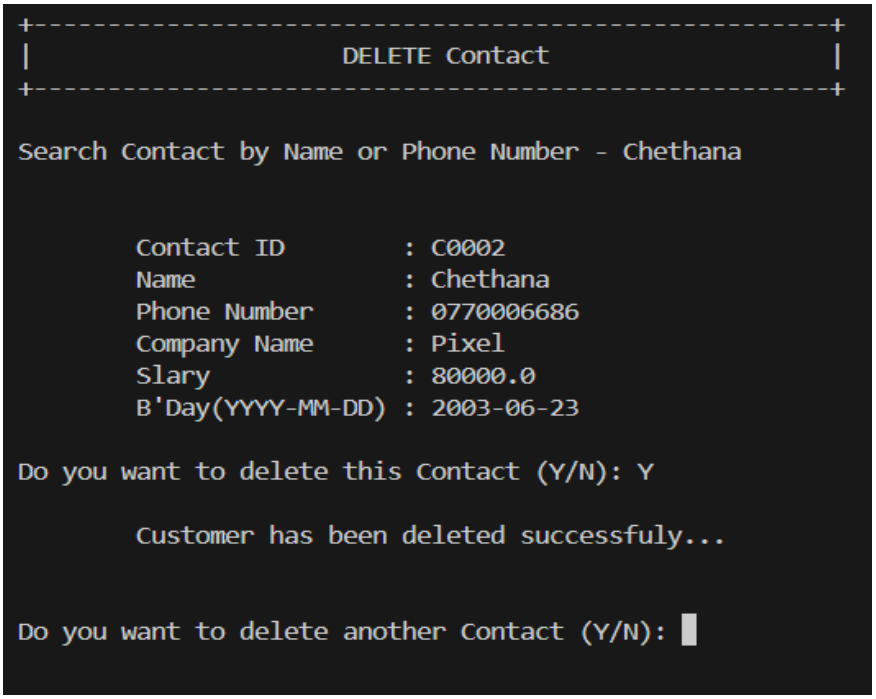

Figure 10 – Delete contact

04. SEARCH Contacts (Demo)

With this option, a contact can be searched from the system. First, the user needs to enter the Phone Number or Name. The phone number should be validated. otherwise, it should handle like previously.

If the user has entered a valid Phone Number or the Name of the contact still hasn't been added to the system, then it should be notified as well as follows.

If the user enters a Phone Number or Name that is already in the system, all the related details of contact should be shown to the user.

Figure 11 – Search contact not found

Figure 12 – Search contact successfully

05. LIST Contacts (Demo)

With this option, the user can view sorted contact details. If the user enters 1, the user can view Contact listing by Name, If the user enters 2, the user can view Contact listing by Salary, If the user enter 3, the user can view Contact listing by Birthday. Just like above, in the end, it should prompt whether the user wants to stay here or go back to the main menu.

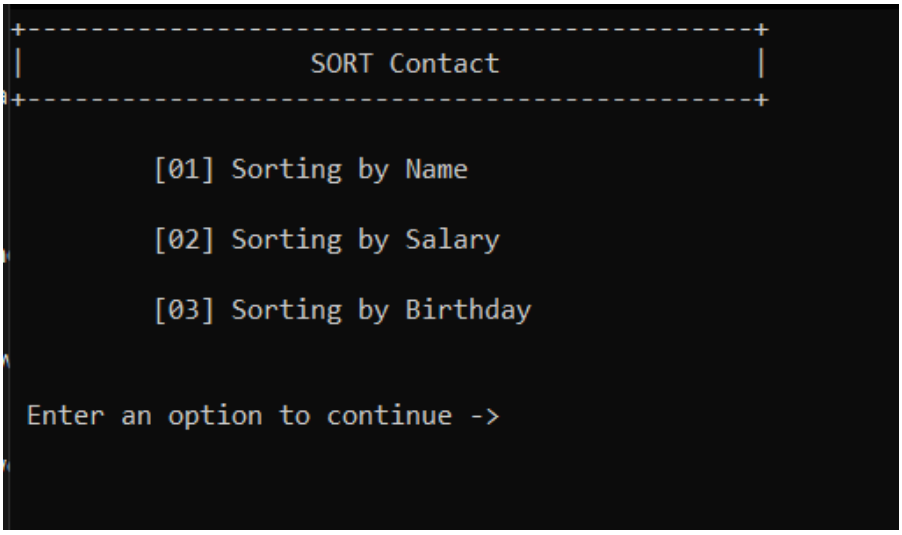

Figure 13 – Listing contact

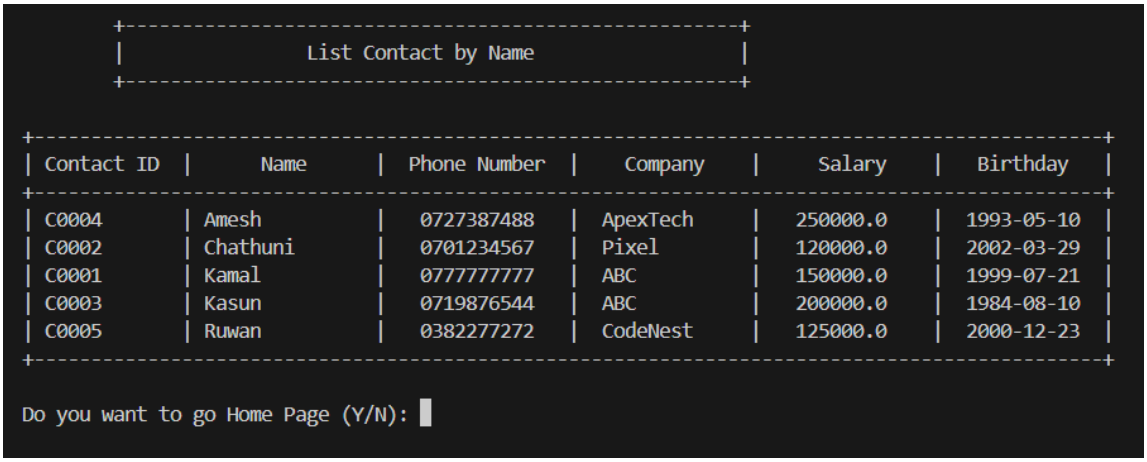

Figure 14 – Listing contact by Name

Figure 15 – Listing contact by Salary

Figure 16 – Listing contact by Birthday

Guideline

- Refer to the Coursework Guidelines at the end to understand the specific guidelines to be followed when developing the project required.
- You should use your knowledge of flow control, loops, methods, and arrays to implement this coursework.
- You can't create classes except for the class that holds the main method and you can create as many methods as you wish and can't use Arrays class method.
- Use the Scanner class to get input from the command-line interface.
- All validations that have been mentioned in this document should be implemented in your program.
- these two lines of code are used to clear the last few lines of output in the console. The first line moves the cursor up lines, and the second line clears those lines and any subsequent lines that were printed after them. You can change the highlighted number for clear lines according to your need.

```
// Move the cursor up five lines
System.out.print("\033[5A");
// Clear the lines
System.out.print("\033[0J");
```

- The code to clear the command line from inside a Java application is as follows. You can use this code when you need to clear the command line.

```
public final static void clearConsole() {
    try {
        final String os = System.getProperty("os.name");
        if (os.contains("Windows")) {
            new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor();
        } else {
            System.out.print("\033[H\033[2J");
            System.out.flush();
          }
      } catch (final Exception e) {
          e.printStackTrace();
          // Handle any exceptions.
       }
  }
```

- You can create as many methods as you wish in the only class that you have.
- Demo Videos are given at relevant places for you to understand the coursework requirement better and Demo videos may help you to clarify your doubts to some extent.
- If you still have any questions, feel free to question.