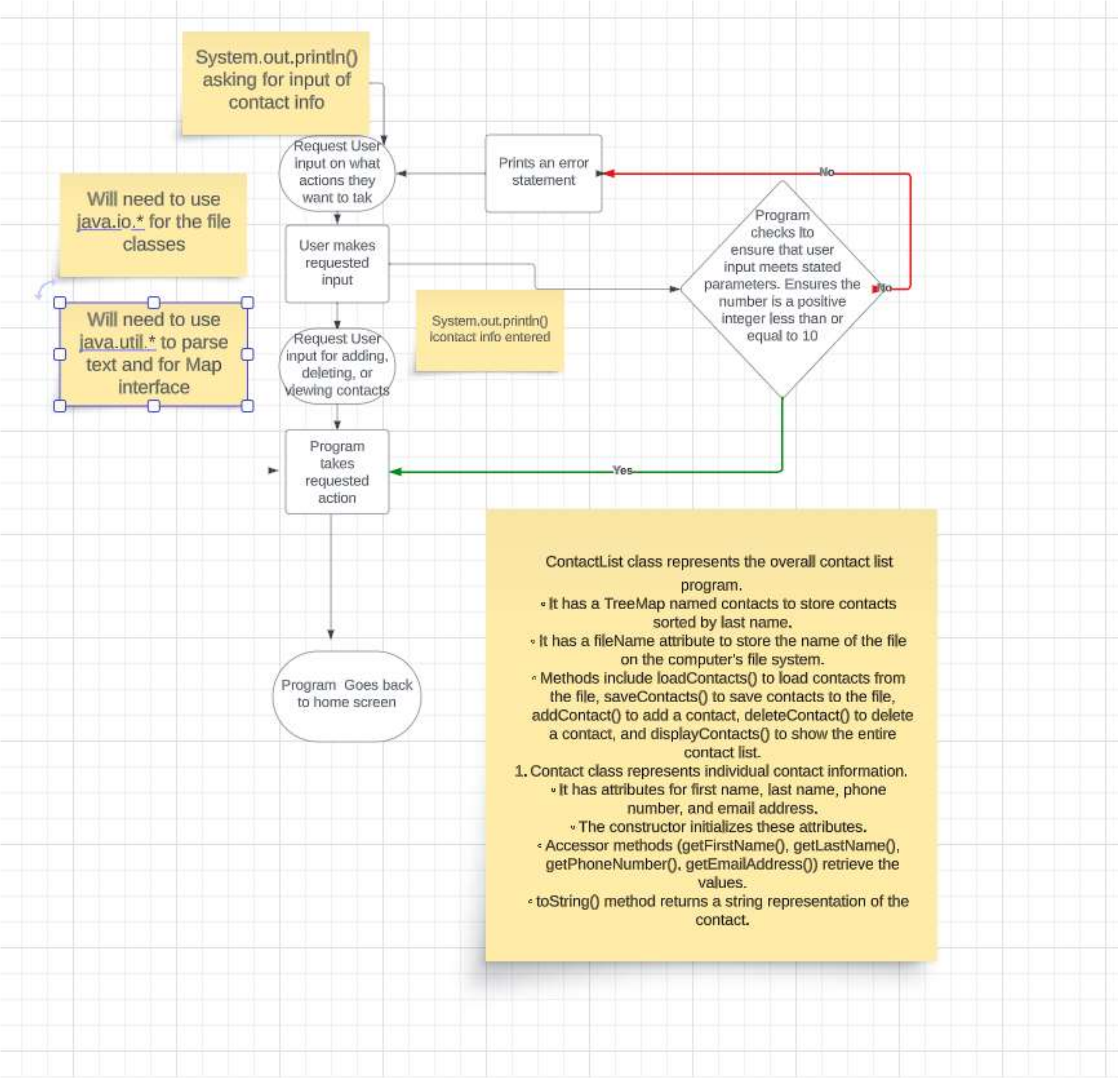
//Design


System.out.println() asking for input of contact info
Will need to use java.io.* for the file classes
Will need to use java.util.* to parse text and for Map interface
Request User input on what actions they want to tak
User makes requested input
Request User input for adding, deleting, or viewing contacts
Program takes requested action
Program Goes back to home screen
Prints an error statement
System.out.println() Icontact info entered
Program checks Ito ensure that user input meets stated parameters. Ensures the number is a positive integer less than or equal to 10
No
No
Yes

ContactList class represents the overall contact list program.

- It has a TreeMap named contacts to store contacts sorted by last name.
- It has a fileName attribute to store the name of the file on the computer's file system.
- Methods include loadContacts() to load contacts from the file, saveContacts() to save contacts to the file, addContact() to add a contact, deleteContact() to delete a contact, and displayContacts() to show the entire contact list.

1. Contact class represents individual contact information.

- It has attributes for first name, last name, phone number, and email address.
- The constructor initializes these attributes.
- Accessor methods (getFirstName(), getLastName(), getPhoneNumber(), getEmailAddress()) retrieve the values.
- toString() method returns a string representation of the contact.

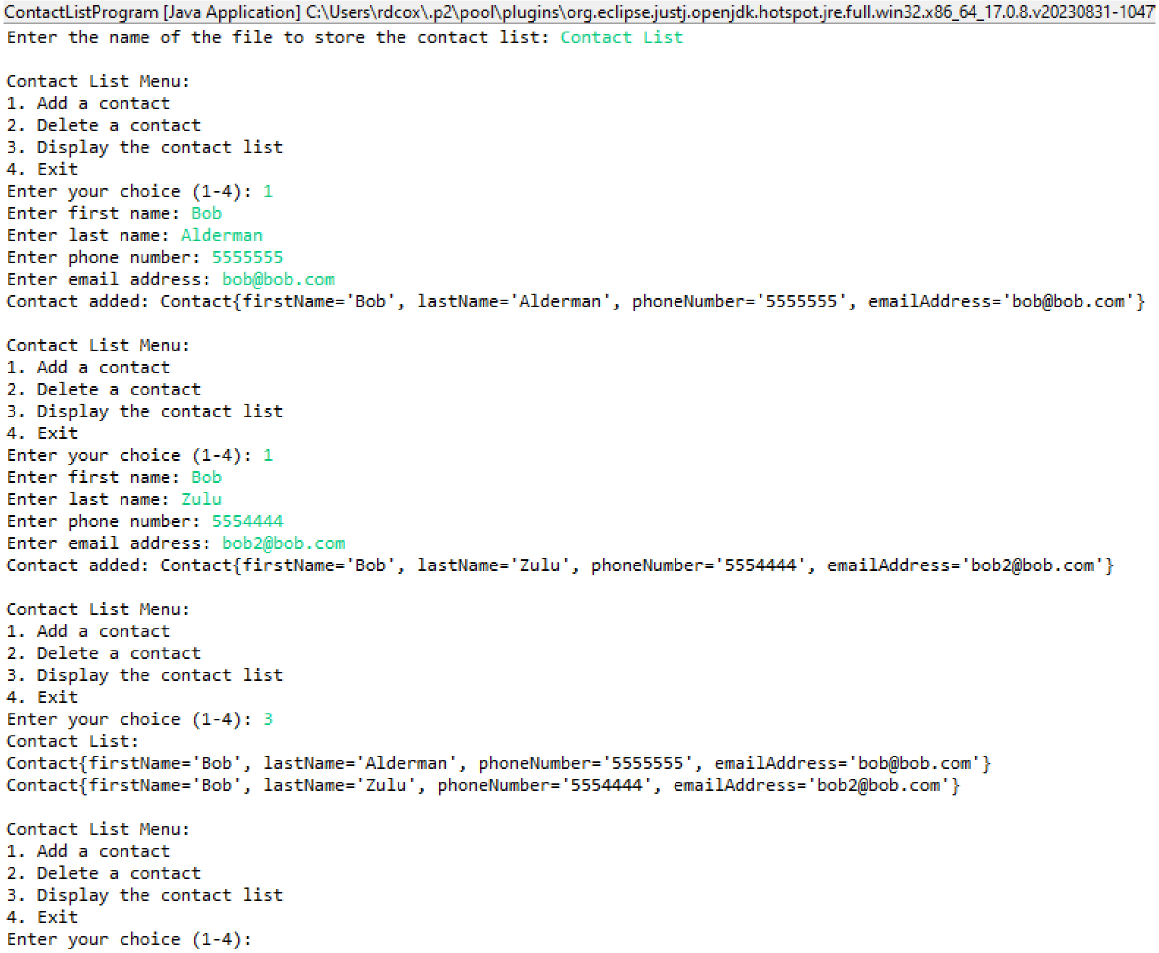
//Code

```
package contactList;

import java.io.*;

import java.util.*;


public class ContactListProgram {

    public static void main(String[] args) {

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

```
System.out.print("Enter the name of the file to store the contact list: ");
String fileName = scanner.nextLine();

TreeMap<String, Contact> contactList = new TreeMap<>();

while (true) {
    System.out.println("\nContact List Menu:");
    System.out.println("1. Add a contact");
    System.out.println("2. Delete a contact");
    System.out.println("3. Display the contact list");
    System.out.println("4. Exit");

    System.out.print("Enter your choice (1-4): ");
    int choice = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character

    switch (choice) {
        case 1:
            addContact(scanner, contactList);
            break;
        case 2:
            deleteContact(scanner, contactList);
            break;
        case 3:
            displayContactList(contactList);
            break;
        case 4:
            writeContactListToFile(fileName, contactList);
```

```
                    System.out.println("Exiting the program. Contact list saved to " 
+ fileName);

                    System.exit(0);

                default:

                    System.out.println("Invalid choice. Please enter a number between 
1 and 4.");

            }

        }

    }


    private static TreeMap<String, Contact> readContactListFromFile(String fileName) 
{

        TreeMap<String, Contact> contactList = new TreeMap<>();


        try (ObjectInputStream ois = new ObjectInputStream(new 
FileInputStream(fileName))) {

            contactList = (TreeMap<String, Contact>) ois.readObject();

            System.out.println("Contact list loaded from " + fileName);

        } catch (FileNotFoundException e) {

            System.out.println("No existing contact list found. Creating a new 
one.");

        } catch (IOException | ClassNotFoundException e) {

            e.printStackTrace();

        }


        return contactList;

    }


    private static void writeContactListToFile(String fileName, TreeMap<String, 
Contact> contactList) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new 
FileOutputStream(fileName))) {
            oos.writeObject(contactList);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void addContact(Scanner scanner, TreeMap<String, Contact> 
contactList) {
        System.out.print("Enter first name: ");
        String firstName = scanner.nextLine();
        System.out.print("Enter last name: ");
        String lastName = scanner.nextLine();
        System.out.print("Enter phone number: ");
        String phoneNumber = scanner.nextLine();
        System.out.print("Enter email address: ");
        String emailAddress = scanner.nextLine();

        Contact newContact = new Contact(firstName, lastName, phoneNumber, 
emailAddress);
        contactList.put(lastName, newContact);

        System.out.println("Contact added: " + newContact);
    }

    private static void deleteContact(Scanner scanner, TreeMap<String, Contact> 
contactList) {
        System.out.print("Enter the last name of the contact to delete: ");
        String lastNameToDelete = scanner.nextLine();

        Contact removedContact = contactList.remove(lastNameToDelete);

        if (removedContact != null) {
            System.out.println("Contact deleted: " + removedContact);
        } else {
            System.out.println("Contact not found with last name: " + lastNameToDelete);
        }
    }

    private static void displayContactList(TreeMap<String, Contact> contactList) {
        if (contactList.isEmpty()) {
            System.out.println("Contact list is empty.");
        } else {
            System.out.println("Contact List:");
            for (Contact contact : contactList.values()) {
                System.out.println(contact);
            }
        }
    }

    // Contact class representing a person's contact information
    private static class Contact implements Serializable {
        private String firstName;
        private String lastName;
        private String phoneNumber;
        private String emailAddress;

        public Contact(String firstName, String lastName, String phoneNumber, String emailAddress) {
            this.firstName = firstName;
            this.lastName = lastName;
            this.phoneNumber = phoneNumber;
            this.emailAddress = emailAddress;
        }

        @Override
        public String toString() {
            return "Contact{" +
                    "firstName='" + firstName + '\'' +
                    ", lastName='" + lastName + '\'' +
                    ", phoneNumber='" + phoneNumber + '\'' +
                    ", emailAddress='" + emailAddress + '\'' +
                    '}';
        }
    }//end line
}//end main
```

//Output



ContactListProgram [Java Application] C:\Users\rdcox\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047

```
Enter the name of the file to store the contact list: Contact List

Contact List Menu:
1. Add a contact
2. Delete a contact
3. Display the contact list
4. Exit
Enter your choice (1-4): 1
Enter first name: Bob
Enter last name: Alderman
Enter phone number: 5555555
Enter email address: bob@bob.com
Contact added: Contact{firstName='Bob', lastName='Alderman', phoneNumber='5555555', emailAddress='bob@bob.com'}

Contact List Menu:
1. Add a contact
2. Delete a contact
3. Display the contact list
4. Exit
Enter your choice (1-4): 1
Enter first name: Bob
Enter last name: Zulu
Enter phone number: 5554444
Enter email address: bob2@bob.com
Contact added: Contact{firstName='Bob', lastName='Zulu', phoneNumber='5554444', emailAddress='bob2@bob.com'}

Contact List Menu:
1. Add a contact
2. Delete a contact
3. Display the contact list
4. Exit
Enter your choice (1-4): 3
Contact List:
Contact{firstName='Bob', lastName='Alderman', phoneNumber='5555555', emailAddress='bob@bob.com'}
Contact{firstName='Bob', lastName='Zulu', phoneNumber='5554444', emailAddress='bob2@bob.com'}

Contact List Menu:
1. Add a contact
2. Delete a contact
3. Display the contact list
4. Exit
Enter your choice (1-4):
```