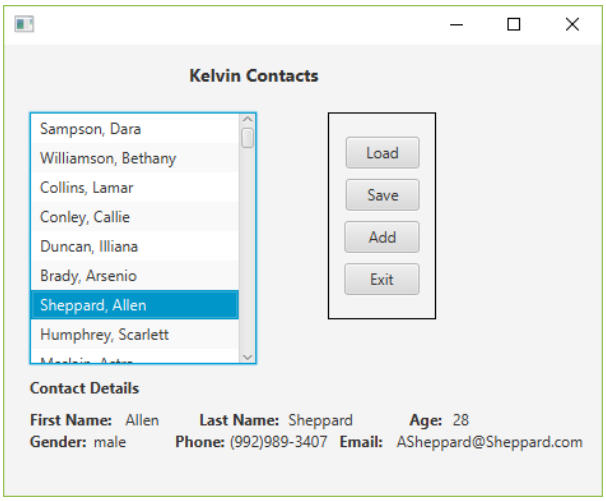
Classes and ArrayLists



You will be creating a contacts application to store information about clients for a business. I will give you a GUI project that I developed and you will to provide the backend functionality for the GUI project by developing two classes: Client and ContactList. The data for your application will come from a comma delimited file containing information about the clients. Ideally, you would use the validation methods to filter out contacts with issues. You are free to use the validation methods developed in the String Processing lab; however, it is not a requirement. You can just load all the records from the file this week. **Note:** You will have to modify the input method from the String Processing lab since I changed the input file format slightly by adding the name of the contact list as the first line of input (See the Input File Format section below).

Step 1:

Create a new project and **name it ContactsModel**. Next, create the Client class in this project under the contactsmodel package. Once you have finished adding the code for Client, test it out adding code to yur main method that declares and instantiates a Client object. Use the setter methods to modify some properties. Output the contents of the client object using the toString() and System.println() methods.

Client Class: This class will model a single client.

Client	
-firstname : String -lastname : String -gender : String -age : int -phone : String -email : String	//Typically, any needed comments about each property would go over here.
+Client() +Client(fn : String, ln : String, gndr : String, age : int, phone : String, email : String) //Getters and Setters for each property +toString() : String	Use Java default values for each property Use the String.format() from the String Processing lab so nice format output can be displayed.

Step 2:

Within the same project and package, create the ContactList class (**notice that "Contact" is not plural**). At this point, you will want to set the Working Directory to where you saved your contacts.txt input file.

For this lab, you may assume that the input file will always be named "contacts.txt". Implement the methods for the ContactList class according to the UML Class Diagram below. To test the loadContacts() method, call and return value from the toString() method of the ContactsList class. It should display each client in a nicely formatted table. You should then add some new contacts to your ContactsList object by instantiating new Client objects, populating them with data and then calling the addContact method. After adding some contacts, call the saveContacts() method and exit your program. When you restart your program using your modified input file, the added contacts should be loaded.

ContactList Class: This class will contain an ArrayList and will model the entire contact list for an individual or the entire company.

ContactList	
-listName : String -contacts : ArrayList<Client>	Name given to describe the contact list
+ContactList() +getListName() : String +setListName(listName :String) : void +toString() : String +getContact(index : int) : Client +size() : int +deleteContact(index : int) : void +addContact(Client client) : void +loadContacts(filename : String) : void +saveContacts(filename : String) : void	 Will call the toString() for each client and separate them with a System.lineSeparator character. Return the client stored at location index. Return the size of the ArrayList contacts. Delete the contact at location index.

Input File Format:

list name
$firstname_0$ $lastname_0$ $gender_0$ age_0 $phone_0$ $email_0$
$firstname_1$ $lastname_1$ $gender_1$ age_1 $phone_0$ $email_1$
….
$firstname_{n-1}$ $lastname_{n-1}$ $gender_{n-1}$ age_{n-1} $phone_{n-1}$ $email_{n-1}$

Step 3:

Integrate your code with my GUI. If you do not implement things as outlined in the UML class diagrams, then your code will probably not work with my GUI code. You need to unzip the GUI project and then use Netbeans to open the project. Just FYI, the GUI project was created using the JavaFX SceneBuilder tool so there will be a lot of stuff in there that will look foreign. You do not have to worry about understanding my code since you will not be modifying any of it. You simply need to make your classes available for the GUI. Here's how. Right click the "Libraries" folder in the GUI project (ContactsApp) and then select "Add jar/folder". Navigate to the "dist" folder of your ContactsModel folder and select

contactsmodel.jar. You should now be able to run the application. Make sure you set the ContactsApp project as your main project. Note: I did not get a chance to implement the Add functionality on the GUI so clicking the button will not initiate any action.