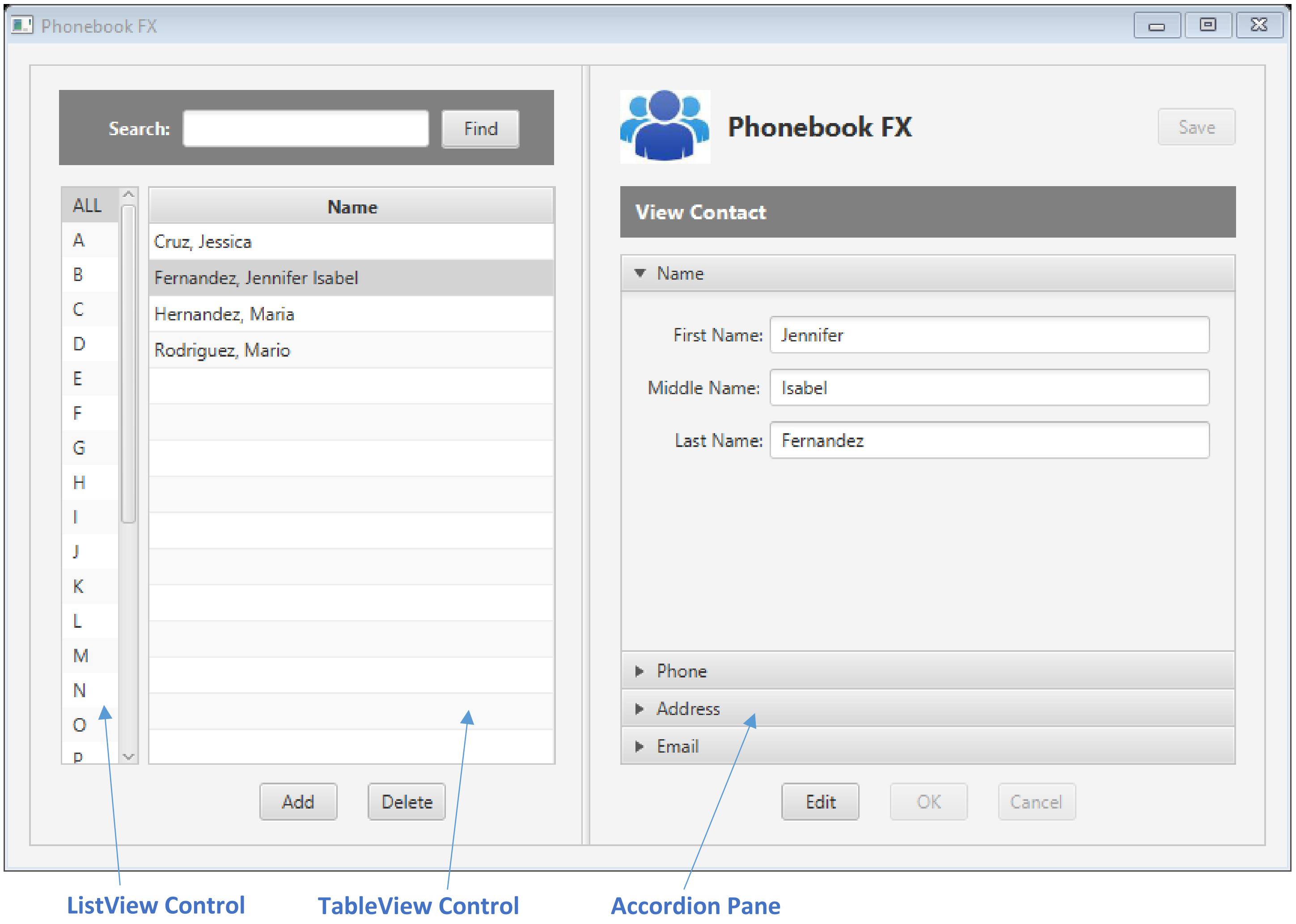
Final Project – Phonebook FX

You will create a Phonebook app using JavaFX. You will use **JavaFX Scene Builder 2.0** from Oracle to create your user interface. **JavaFX Scene Builder 2.0** can be downloaded from http://www.oracle.com/technetwork/java/javafxscenebuilder-1x-archive-2199384.html

The user interface **must be the same** as in the diagram below.



The phonebook app will be **file based**. In other words, the information of the contacts will be saved in a file. It's **suggested** that you use a **.CSV** file to store the contacts' information. When the program loads, the contacts' information will be **loaded** from the file into an array. The file will be **closed** after it has been read. The app will then **use this array** to *retrieve* contact information and store the *modifications* made to the contacts. The modifications will be saved to the file, when the user clicks the **Save** button. This operation will **overwrite** the contents of the original file.

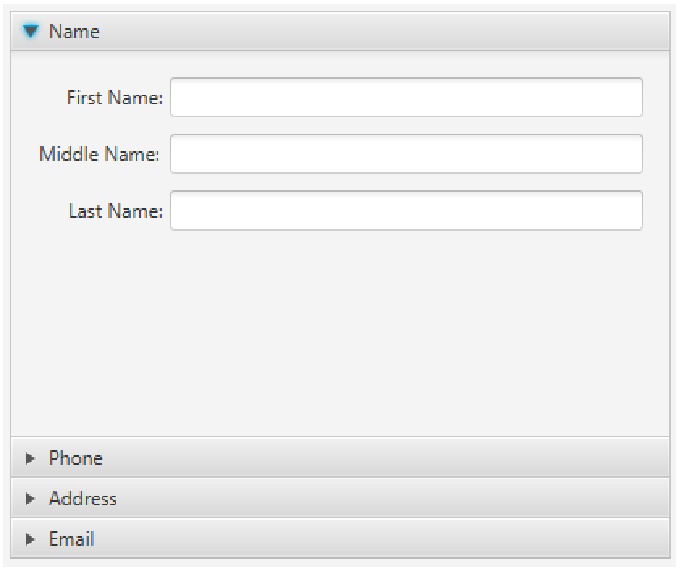
All the contacts should be displayed in the table view, after the program loads. The **format** of the contacts' names in the table view is **<Last Name>, <First Name> <Middle Name>**. *Note, if the contact doesn't have a middle name, then a space **should not** appear after the first name.*

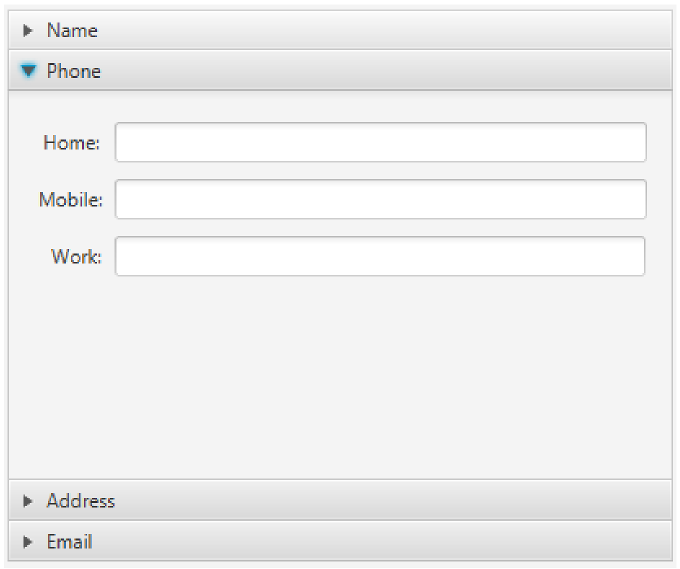
Suggestion: Maintain a **hidden column** in the table view to store the **key** of each contact.

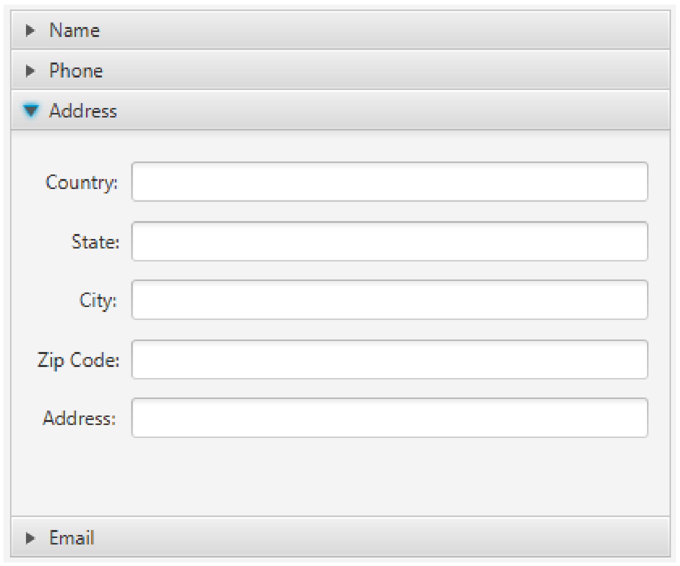
Your app must store the following information for each contact:

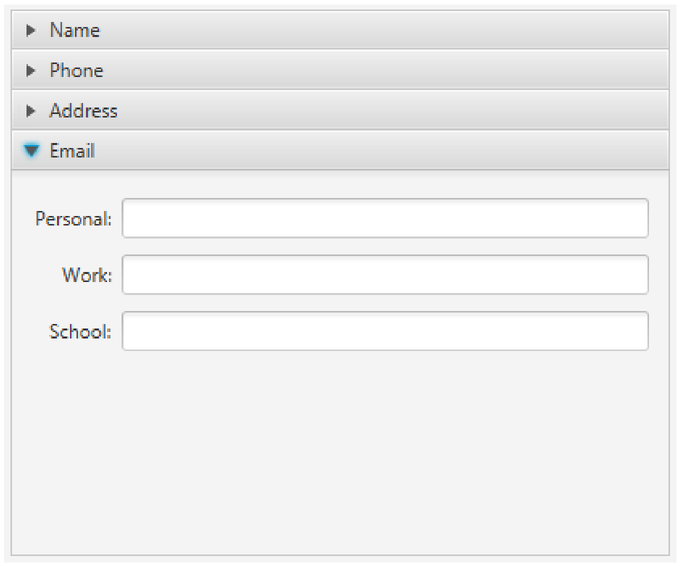
- **Name Information:**
    - First Name, Middle Name, Last Name
- **Phone Information:**
    - Home Phone, Mobile Phone, Work Phone
- **Address Information:**
    - Country, State, City, Zip Code, Street Address
- **Email Information:**
    - Personal Email, Work Email, School Email
- **Key**: A *unique* number (key) must also be stored for each contact. Each contact should have a unique key which will identify that contact. *The key information **should not be visible**.*
    - **Suggestion:** Maintain in another file —for example, settings.txt— the **next available key**. Every time a new contact is added, this number should be *incremented by one*. This number *should **never** be decremented.* For example, when a contact is deleted, the key should not be decremented! This will guarantee that each contact receives a unique key (number).
    - The app will use this key to find a specific contact in the *array of contacts*.

The contact information should be **organized** in the accordion pane as follows:









Suggestion: Add a **hidden** text field control to the accordion pane to store the **key** information of the contact.

Your app **must** implement the following features:

- **Display all the information of a contact**
  - When a contact is selected in the table view, all the information should be displayed in the accordion pane. In addition, the **Delete** and **Edit** buttons should be *enabled*. The **Delete** and **Edit** buttons should be *disabled* if *no contact* is selected in the table view.
- **Sort contacts**
  - The user should be able to sort the contacts by clicking the **Name** heading in the table view.
- **Add a new contact**
  - When the user clicks the **Add button**, *any selection* in the table view should be **cleared**, all the text field controls in the accordion pane should be **cleared** and become **editable** —except for the **key** text field control, which will *always be kept hidden*—, and the **Name** section of the accordion pane should be expanded. In addition, the next available key should be read from the settings.txt file and stored in the **hidden** key text field control. This key will be used as the key for the new contact. **Don't foget to increment the key in the file by one.** When the user clicks the **OK** button, the new contact should be **added** to the contacts array, the table view should be populated with **all** the contacts, and the *new* contact should be **selected**.
- **Delete an existing contact**
  - The **Delete** button *should only be enabled* if a contact has been selected in the table view. When **Delete** is pressed, a ***confirmatin dialog*** should be presented to the user. The contact should be deleted ***only if*** the user confirms the removal of the contact. If the removal is confirmed, the contact should be **removed** from the contacts array and the table view.
- **Edit an existing contact**
  - The **Edit** button *should only be enabled* if a contact has been selected in the table view. *Remember that the contact's information should be displayed in the accordion pane when a contact is selected in the table view*. When **Edit** is pressed, **all** the text field controls in the accordion pane should become **editable,** except for the **key** text field control, which will *always be kept hidden*. The user **shouldn't** be able to modify the key. When the user clicks the **OK** button, this contact should be **updated** in the contacts array with the information from the text field controls. In addition, the table view should be populated with **all** the contacts, and the *modified* contact should be **selected**.
- **Cancel an addition or edit in progress**
  - The user should be able to cancel the addition of a new contact or the modification of an existing contact, that's *currently in progress,* by clicking the **Cancel** button. If the addition of a new contact is canceled, all the text field controls should be **cleared** and become **uneditable**. On the other hand, if the modification of an existing contact is canceled, the *original* information for that contact should be **redisplayed** in the accordion pane, and all the text field controls should become **uneditable**.
- **Filter the contacts**
  - When the user selects a **letter** in the *list view*, the contacts in the *table view* should be filtered by the **first letter** of the **last name**. In addition, when **ALL** is selected, all the contacts should be displayed in the table view. ***Note:*** *There should be a* ***consistency*** *between the selection in list view, and the contacts displayed in the table view.*
- **Search the contacts**
  - When the user clicks the **Find** button, *only* the contacts whose names **contain** the substring entered in the **search** text field control, should be displayed in the table view. In addition, the **selection** in the list view shoud be **cleared**, to indicate that the previous filter has been removed.

- **Save the changes**
    - The **Save** button should become **enabled** whenever a new contact is added, an existing contact is edited, or an existing contact is deleted. When the **Save** button is clicked, all the contact information stored in the contacts array should be **written** to the **file**. The updated information will *overwrite* the contents of the original file. After ther information is saved, the **Save** button should become **disabled**.
- **Confirm app closing when changes haven't been saved or an addition/edit is in progress**
    - The app should present the user with a **confirmation dialog**, whenever the **exit** button is clicked and there are **pending changes**. The dialog should *inform* the user that changes will be lost, if the application is closed. If the user decides not to exit, the app should remain opened. *In addition, if there's an addition or edit in progress, the dialog should also inform the user that the current addition or edit will be lost.*

Some other requirements to keep in mind:

- Whenever the **Add** or **Edit** buttons are clicked, the user **shouldn't be allowed** to search, filter, change the selection of the table view, or click any of the following buttons: **Add**, **Delete**, **Edit**, and **Save**. However, the **OK** and **Cancel** buttons should be **enabled**, and the text field controls in the accordion pane should become **editable**.
- Whenever the **OK** or **Cancel** buttons are clicked, the user **should be able** to search, filter, change the selection of the table view, and click on the **Add** button. The **Save** button should *only* be **enabled**, if changes need to be saved. In addition, the **Edit** and **Delete** buttons should *only* be **enabled**, if there is a contact **selected** in the table view. However, the **OK** and **Cancel** buttons shold be **disabled**, and the text field controls in the accordion pane should become **uneditable**.
- Whenever possible, the program **should handle exceptions** to avoid *abnormal program termination*.
- **Do not use** JOptionPane to display dialogs. You must use JavaFX Dialogs.

Final Notes

- Include a document with a brief description of the contributions of each member.
- **Only the captain in the group will submit the project.**
- The following link contains information about **JavaFX Dialogs:** http://code.makery.ch/blog/javafx-dialogs-official/
- The **JavaFX 8 documentation** is available from the following link: http://docs.oracle.com/javase/8/javafx/api/overview-summary.html
- The **JavaFX CSS Reference Guide** is available from the following link: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html