Homework 1 – DB – CSC 462 F16 (Last Modified: August 31, 2016)

Subject: Homework 1 – Contact List similar to that used in **Microsoft Outlook**
Due Date: Before class Thursday, September 15 submit an:

- Electronic eclipse project archive soft copy as a moodle assignment file upload
- Hand in a hard copy at start of class**.** Note: insure yours archive has:
    - the SQL script to create your MySQL database along with data to insert into the database
    - a file, UML.doc, containing your UML class diagram.
- Insure your **ComputingSciencesAtElon/<team name>** GitHub repository has a **tag** with the name **homework1** with contains the identical source code to submitted hard copy and electronic eclipse project archive.

General: This assignment will give you a chance to refresh your java programming skills learned in CSC 130, CSC 230 and CSC 330 while using the **eclipse 4.6 Neon** programming environment, the MySQL database and the MySQL Workbench. If your skills are rusty then you will want to get an early start. ***Collaboration is only allowed with your assigned team mates(s)***. The only exception to this rule is that you may ask Professor Powell for assistance. The assignment is to implement a Contact List similar to that maintained with Microsoft Outlook but much simpler. All of the requirements are enumerated below.

Objectives: Review/refresh the following Java Skills from CSC 130, CSC 230 and CSC 330 to demonstrate the

- Ability to design and implement a solution to a set of customer requirements.
- Ability to create an **event driven interface** in Swing.
- Ability to use **JDBC** to access and manipulate a local or remote SQL database.
- Ability to use SQL for **CRUD** operations.
- Ability to use **JUnit** to test the methods that perform CRUD operation on the database.
- Ability to use **UML** to represent the class diagram for submitted project.
- Ability to use multiple classes with composition.
- Ability to use a **java.util collection** of ArrayList or LinkedList with Generics.
- Ability to apply Google standard Java coding conventions and standard Oracle Javadoc documentation conventions.
- Ability to quickly learn and extend concepts in undergraduate CS courses to develop a problem solution with a deadline to meet requirements. Specifically, you will apply newly learned **Git** and **GitHub** code versioning along with database creation.

Grading: I am expecting a high quality piece of work with close attention to exactly meeting all requirements. Each requirement is discussed below. If you are unsure about a requirement then please see me for interpretation. All functional requirements must be completed and working to receive a passing grade. I expect Google Java Style Guide Conventions and Oracle/Sun Javadoc conventions to be followed exactly. You must have a **copyright statement on each source file with each team member name on it to indicate that your team originally designed and wrote each line of code with no assistance.** No late homeworks will be accepted.

Deliverables:

1. Your team is required to submit your eclipse exported, archived Project. If my project folder was **Contact** then I would have an exported, archived zip file called **Contact.zip**. Your zip file must be submitted in moodle as an assignment file upload by the due date and time. No late submissions are accepted. You need to insure that I can import an existing project into an eclipse workspace with your archive file. I will navigate to the file within the Project folder at **src/edu/elon/contact/ContactApplication.java**, and select *Run As* to start your application.
2. You are required to submit a Google Java Style Guide formatted hard copy of all of the Java files submitted with your eclipse project archive.
3. You are required to submit your git repository workspace to your GitHub Private Organization Repository, **ComputingSciencesAtElon/<team name>** with a git tag of **homework1**. For example, if my GitHub team name was **alamance** then I would push my workspace to **ComputingSciencesAtElon/alamance**.

Requirements:
1. You are required to use the package **edu.elon.contact** or packages under **edu.elon.contact** for all of your files. This package should be in the Contact Project of the workspace repository that you clone from **https://github.com/ComputingSciencesAtElon/csc462f16dpowell2**
2. Set up eclipse so the source code goes under a folder called **src** and the compiled classes in a folder called **classes**. (Note: my intent here is to insure you understand the Eclipse IDE and can control it instead of it controlling you.)
3. You must implement a Swing GUI. When the application starts, the GUI should match as close as possible the one shown in Figure 1. All of the Menu Items for the File and Menu should be enabled **after** connection to the database. However, until connection to the database is made, only **File – Connect** and **File - Exit** should be enabled.
4. The class that contains the main method to start your application should be called **ContactApplication**. (Note: In my case, I used a project name of Contact so Eclipse stored my **ContactApplication.java** file in the folder, **Contact/src/edu/elon/contact**, and the **ContactApplication.class** file in the folder, **Contact/classes/edu/elon/contact**). You must have a similar structure.

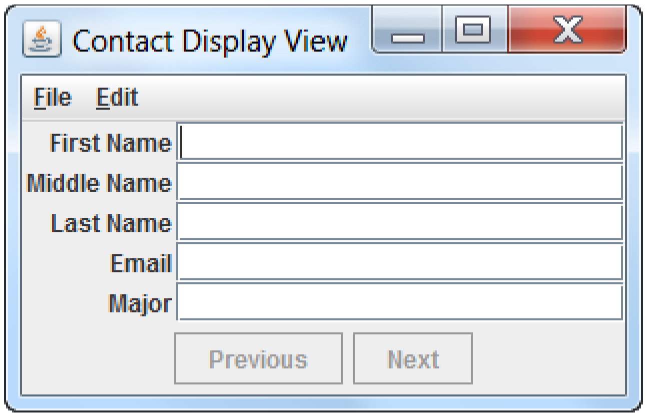


Figure 1: Initial display of Contact Application (Note: The labels are right aligned.)

5. You must have a main menu bar with two Menus on it for **File** and **Edit**. Both of these Menus should support Mnemonics. For example, if I simultaneously hit the keys Alt and F then the pull down menu for **File** will appear as shown in Figure 2. Note: Since the user had not yet connected to the database, the File Menu Item, **Clear DB**, is disabled. Alternatively, I could have clicked on **File** to bring up the pull down menu. You must implement all of the functionality shown on the pull down menu in Figure 2. Each of the menu items will be discussed separately below.

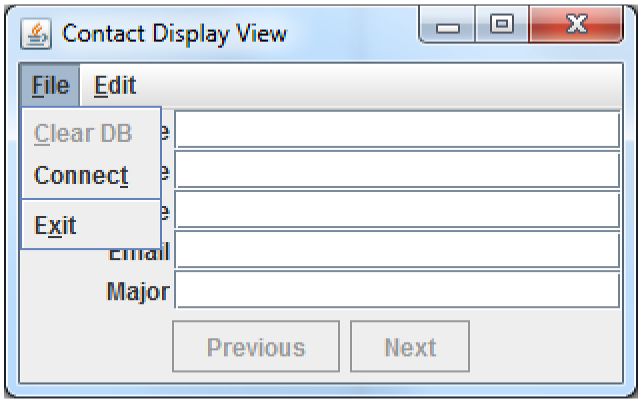
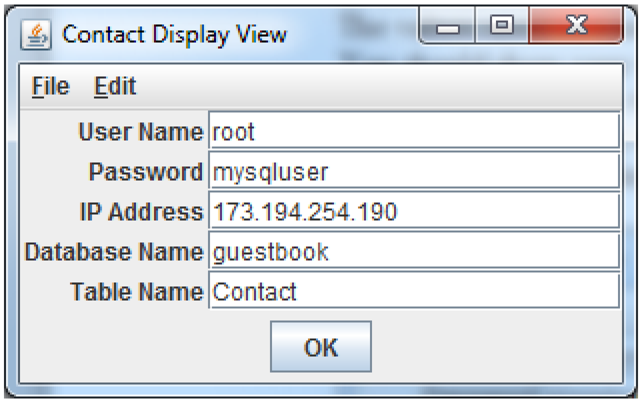


Figure 2: File pull down menu

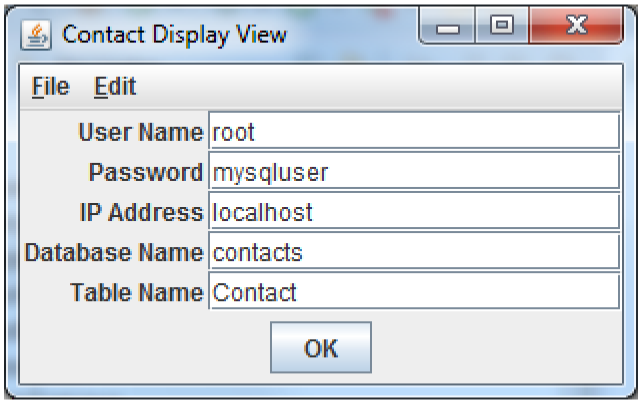
6. You must implement a **Connect** menu item. When **Connect** is selected either with the mouse or the Alt-t Mnemonic then the Contact Display View should be dynamically updated as shown in Figure 3 to allow the user to enter Database information that will be used when **OK** is clicked to connect to the specified Table Name (**Contact)** in the specified Database Name (**guestbook)** at the specified IP Address (**173.194.254.190**) with the given User Name (**root**) and Password (**mysqluser**). Note: The values used here are mine for a Google Appengine MySQL database that I have. You should show your values and not mine in Figure 3. When the display is shown for the first time, each text field should have your default values displayed for editing. This will make testing a lot easier. If this is not the first display then the last entered values should be displayed.



User Name	root
Password	mysqluser
IP Address	173.194.254.190
Database Name	guestbook
Table Name	Contact

Figure 3: Database information needed to connect to my table Contact on my Google App Engine MySQL database

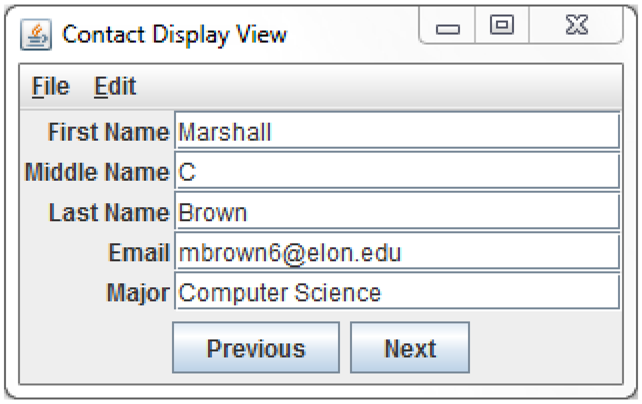
7. Figure 4 shows the database properties specified to connect to the same local machine that java is executing on at IP Address, **localhost** (at the default port of 3306), to a table called **Contact** in a database called **contacts** with a MySQL user name of **root** and a password of **mysqluser**.



User Name: root
Password: mysqluser
IP Address: localhost
Database Name: contacts
Table Name: Contact

Figure 4: MySQL user, root, is about to access table Contact in database contacts on the local machine called localhost on the default port of 3306

8. After clicking OK in Figure 4, if a successful connection is made then all of the Contacts in the database should be returned with the first Contact displayed in an updated Contact Display View. In the current database used for this example and the ones that you should create, I have initialized it with the first five students on the class roster: Marshall C Brown, Haris Cesko, Maddie C Chili, Keith A Davis and Zareh H Deirmendjian. Since Marshall is the first contact, he is shown in Figure 5. The user should be able to continually select **Previous** and **Next** to sequentially move through the Contact List. After the successful connection, all Menu Items should be enabled.

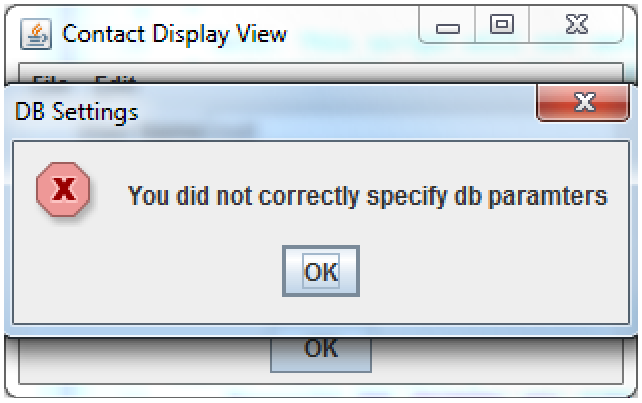


First Name: Marshall
Middle Name: C
Last Name: Brown
Email: mbrown6@elon.edu
Major: Computer Science

Figure 5: Contact List retrieved from database with first Contact Marshall displayed

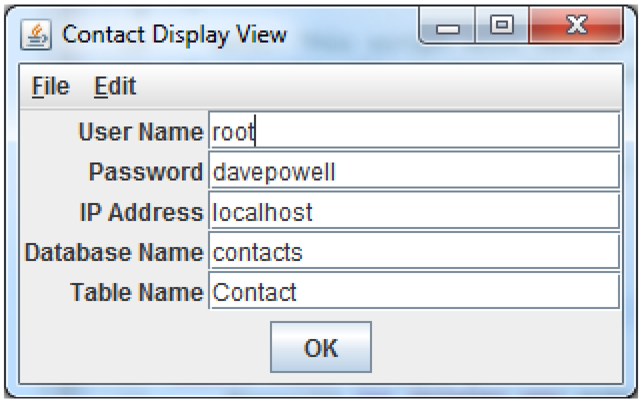
9. If the information provided in Figure 3 or Figure 4 was not correct when **OK** was clicked then you should display a modal **JOptionPane** as shown in Figure 6 to indicate invalid connection properties. After selecting **OK** in the Figure 6 **JOptionPane**, redisplay the database properties as shown in Figure 7 for the user to correct. For example, if in Figure 4, I entered an invalid password of davepowell, Figure 7 redisplays all of the invalid information to make it simpler to fix the errors

and click OK. The cycling between Figures 6 and 7 will continue until either valid database properties are provided or the user selects **File – Exit**.



You did not correctly specify db paramters

Figure 6: Database connection could not be made



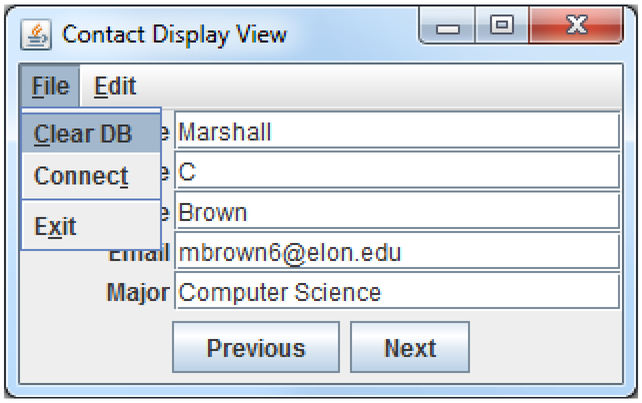
User Name root
Password davepowell
IP Address localhost
Database Name contacts
Table Name Contact

Figure 7: User prompted with recent information for correction for resubmission. In this case, the password is incorrect.

10. If you are on your own machine then you create your own username, password and database name for your MySQL installation. If you are running on localhost in the lab then the MySQL username is **root**, the password is **mysqluser** and you can create your own database name. When running on localhost, to make it simpler to create the database, to create the table and to insert five names into the table, you should create a folder under the **src** folder in eclipse called **sqlscripts** with a sql file logically named containing a series of sql commands that can be executed from MySQL workbench to drop the database if it exists, to create the database, to create needed table(s) and to insert five name entries. (Note: I will use this file to test your java code by first running your sql file from MySQL Workbench to create the database, create your table(s) and insert five students into the table(s)).

11. When I am done with the current contact list, I might want to clear the database of all entries. In this case, I would select **File – Clear DB** as shown in Figure 8. This menu item will completely clear out the contact list in memory and in the database. It will

display the empty contact list as shown in Figure 9.



Marshall
C
Brown
mbrown6@elon.edu
Computer Science

Figure 8: Clear DB selected to clean out all contacts

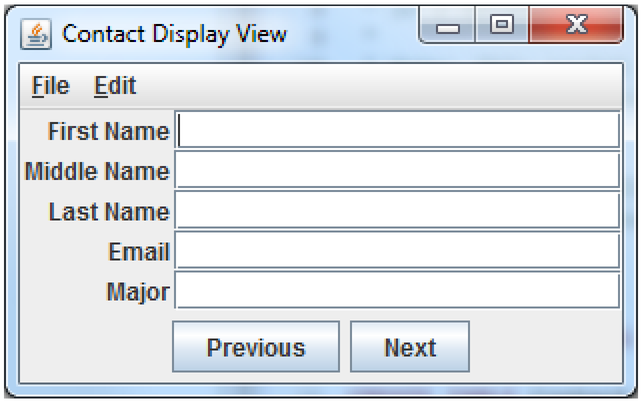


Figure 9: Empty list of Contacts

12. Figure 10 shows the pull down menu items for **Edit**. You must support Mnemonics for the Menu **Edit** and implement all of its menu items. Note: until the user has successfully connected to a database, the **Edit** menu items are disabled. Each of the menu items will be discussed separately below. Figure 11 shows the active menu items after the successful connection to the database in Figure 5.

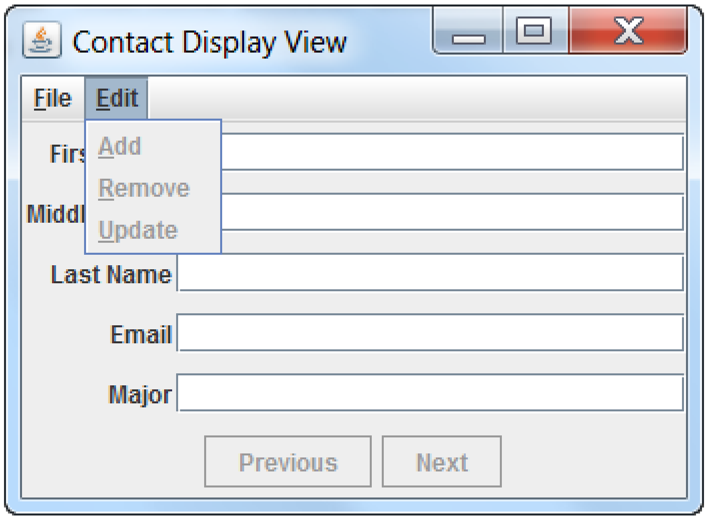
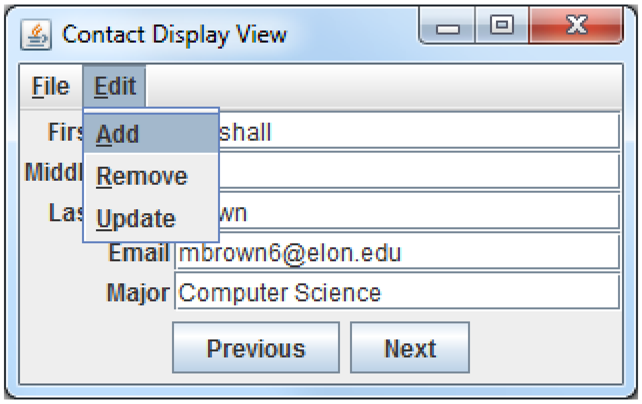


Figure 10: Edit pull down menu



shall

wn

mbrown6@elon.edu

Computer Science

Figure 11: Active Menu Items after successful database connection

13. You must implement an **Add** menu item. When **Add** is selected either with the mouse or the Alt-A Mnemonic then the Contact Display View frame should reconfigure itself with a different set of buttons on the bottom as shown in Figure 12 and all of the text fields should be empty. (Note: the same frame reconfigures itself. It does not create and display a second frame.)

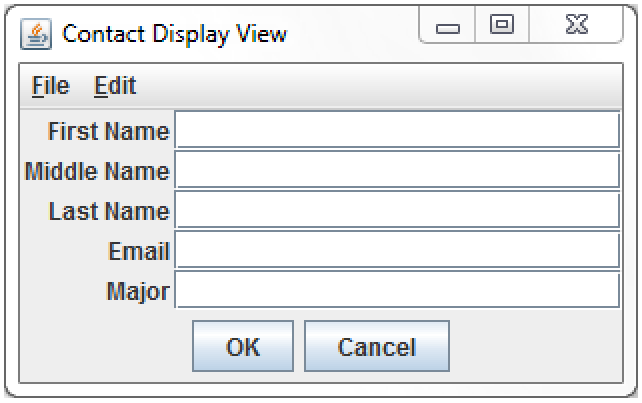
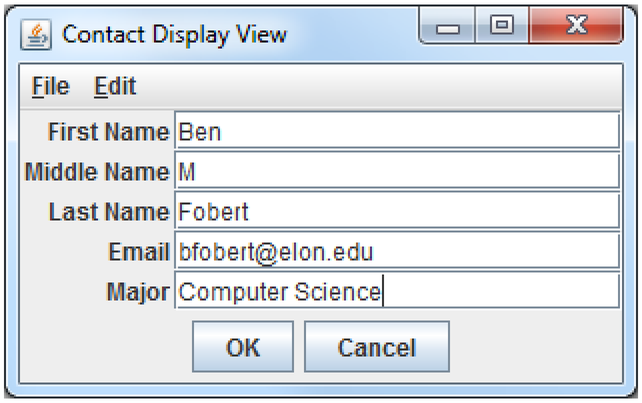


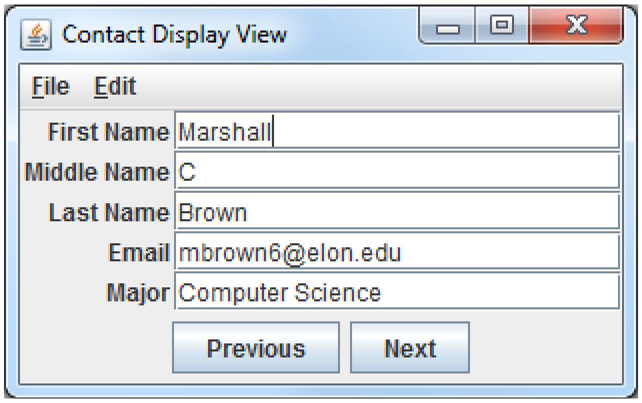
Figure 12: Frame resulting from Menu Item Add selection

14. Figure 13 shows the information added for a student in the class, Ben M Fobert, and Figure 14 shows the resulting panel after **OK** is clicked. The Contact shown in Figure 14 should be the Contact displayed prior to the **Add** button being selected or if there were no previous entries then the first item on the Contact list should be displayed. When **OK** is pressed then the item should be added to the database and added at the **end** of the Contact list. Since Marshall was the current student when **Add** was clicked in Figure 11, he should be displayed after Ben is added to the Contact List and the database. Note: in Figure 14 that the Buttons at the bottom of the screen revert to the Navigation Buttons of **Previous** and **Next** as previously seen in Figure 1.



First Name: Ben
Middle Name: M
Last Name: Fobert
Email: bfobert@elon.edu
Major: Computer Science

Figure 13: Ben Fobert information entered into Contact Form prior to OK being clicked.

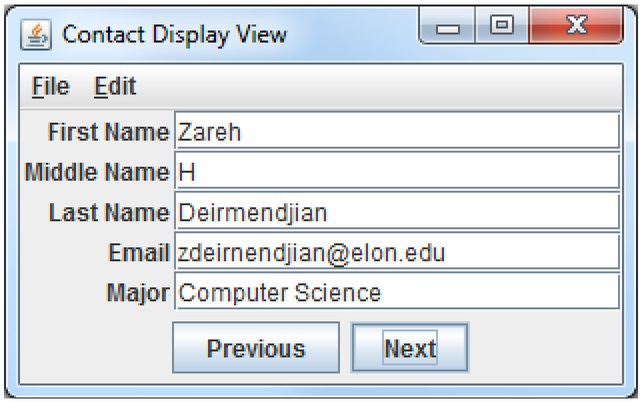


First Name: Marshall
Middle Name: C
Last Name: Brown
Email: mbrown6@elon.edu
Major: Computer Science

Figure 14: After OK is clicked in Figure 13, Ben is added to end of Contact List and Contact who was active when Add was clicked, Marshall, is displayed.

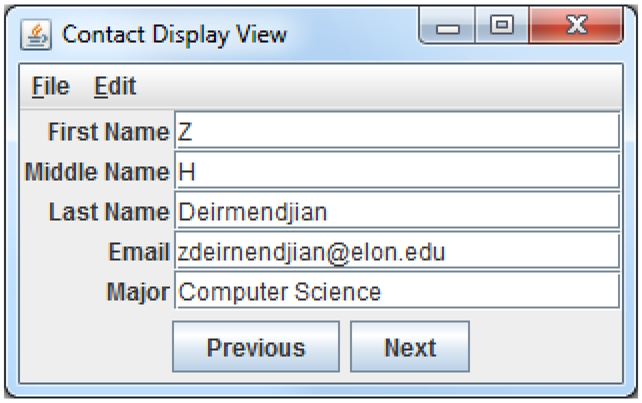
15. The **Previous** and **Next** Buttons allow the user to move the active contact of the contact list in either direction. **Previous** will move to earlier items until the beginning of the list is reached. **Next** will move to the newer added items until the end of the list is reached. When the end is reached and similarly when the beginning is reached then continued attempts to move in that direction will leave the user on the same contact.

16. The displayed contact is the active contact. The Edit Menu has Menu Items **Remove** and **Update** that work on the active item. Let us first look at Zareh Deirmendjian and assume that I want to edit his contact information to show a first name of Z instead of Zareh. I would simply navigate with the **Previous** and **Next** buttons until Zareh is displayed as shown in Figure 15. I would enter in Z for the First Name and then select the **Edit – Update** item. This will immediately update the current contact in the database with the changes shown. Figure 16 shows the entry into the First Name field to change Zareh to Z. Figure 17 shows the selection of **Edit** – **Update** and Figure 18 shows the displayed active changed contact information.



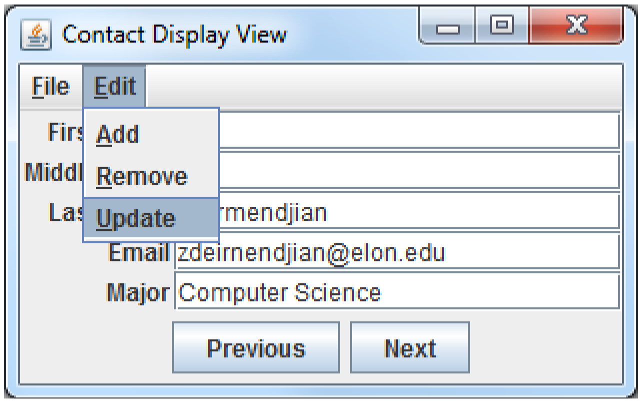
First Name: Zareh
Middle Name: H
Last Name: Deirmendjian
Email: zdeirnendjian@elon.edu
Major: Computer Science

Figure 15: Next button was clicked until Zareh Deirmendjian is the active contact.



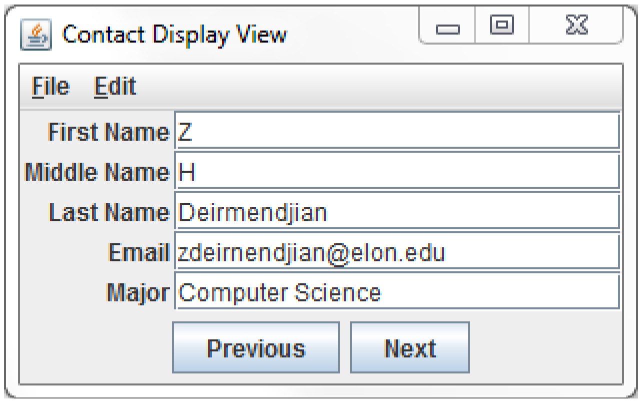
First Name: Z
Middle Name: H
Last Name: Deirmendjian
Email: zdeirnendjian@elon.edu
Major: Computer Science

Figure 16: Zareh's first name changed to Z but not yet updated.



Email: zdeirnendjian@elon.edu
Major: Computer Science

Figure 17: Update selected to officially make changes to displayed Active Contact



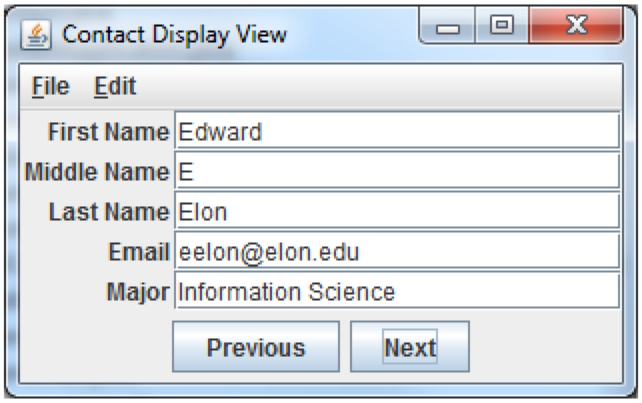
First Name: Z
Middle Name: H
Last Name: Deirmendjian
Email: zdeirnendjian@elon.edu
Major: Computer Science

Figure 18: Z's Contact change completed

17. I have gone ahead and added one additional student, Edward Elon with the **Add** method so the current contacts in the order entered are:
    - Marshall C Brown, mbrown62@elon.edu, Computer Science
    - Haris Cesko, hcesko@elon.edu, Computer Science
    - Maddie C Chili, mchili@elon.edu
    - Keith A Davis, kdavis36@elon.edu, Computer Science

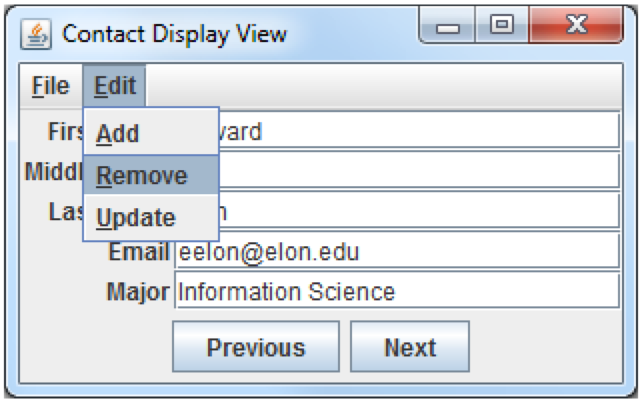
- Z H Deirmendjian, zdeirmendjian@elon.edu, Computer Science
- Ben M Fobert, bfobert@elon.edu, Computer Science
- Edward E Elon, eelon@elon.edu, Information Science

I can traverse this entire list. In doing so, I noticed that Edward E Elon is no longer at Elon so I want to remove him. To remove Edward, I traverse to his name as shown in Figure 19 and then select **Edit – Remove** as shown in Figure 20. Edward is removed from the database and the new active contact Ben Fobert is displayed as shown in Figure 21. The removal process should make the **Next** item to be displayed as the active contact. If there is no next item then the previous item should be made the active contact. This is the case with the removal of Edward Elon. Since there was no next Contact then the previous Contact Ben Fobert is made the active contact. If there was no next and no previous then the text fields should be empty.



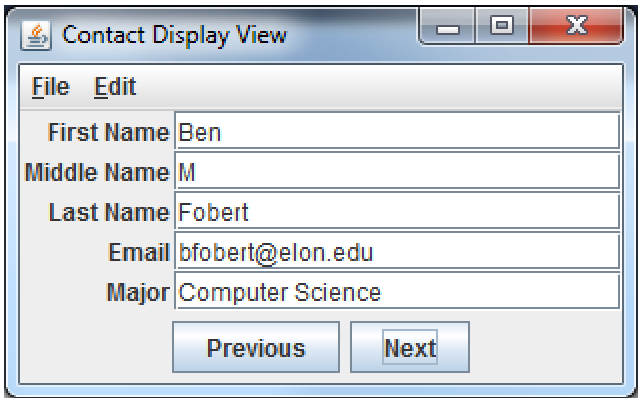
First Name: Edward
Middle Name: E
Last Name: Elon
Email: eelon@elon.edu
Major: Information Science

Figure 19: Edward Elon made current Active Contact



Email: eelon@elon.edu
Major: Information Science

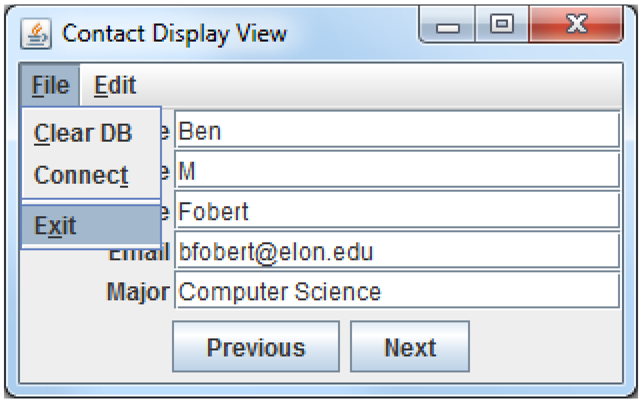
Figure 20: Remove is selected to remove Edward Elon



First Name: Ben
Middle Name: M
Last Name: Fobert
Email: bfobert@elon.edu
Major: Computer Science

Figure 21: Riley is now the Active Contact

18. The **File – Exit** MenuItem shown in Figure 22 will exit the application. However, any Contacts in the Database will be persisted and we can see these by Connecting to the Database as done in Figure 2.



Ben
M
Fobert
Email: bfobert@elon.edu
Major: Computer Science

Figure 22: File - Exit to end application

19. As a software developer, you have heard about an important best practice called Test Driven Development, where you can test the functionality of a method as you write it. For this homework 1, you are required to provide a **JUnit** test for each of the CRUD database operations that you need to support for your contacts. Specifically, you need to test your add contact, update contact, remove contact and delete all contacts to insure they are properly being put into your MySQL database. (Note: You are not testing the GUI but rather the underlying CRUD methods that interact with the DB.)

20. You will have multiple classes in your implementation. Provide a UML class diagram in a file called **UML.doc** showing your classes, class associations, class inheritance along with the key data members, constructors and methods of each class. You can hand draw this diagram and scan it in if desired. Insure you include this file in your submitted project archive.

Good Luck!