Lab Exercise #6

Menus, Toolbars and Dialogs

Computer Science 2334

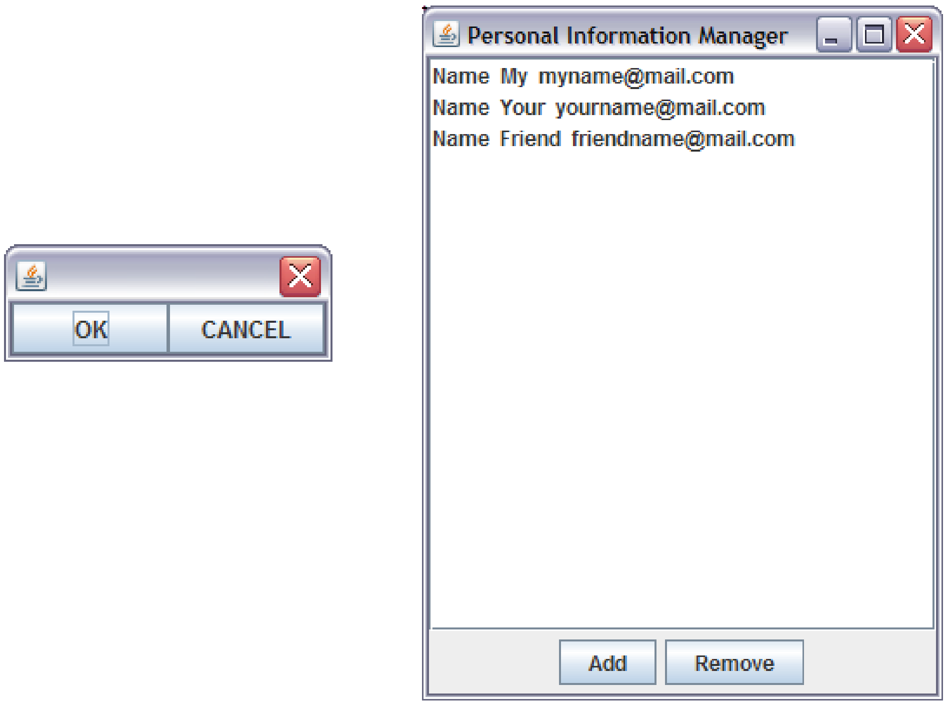
<u>Due by: Friday, March 27, 2009, 2:00 pm</u>

Members: ______________________________
____
____
____
____

Learning Objectives:

- Understand how to create a dialog box based on the **`JDialog`** class.
- Understand how a data model is used to transfer information between the dialog and the main program.
- Understand how to create a menu system using **`JMenuItem`**, **`JMenu`**, and **`JMenuBar`.**
- Understand the interfaces provided by the GUI classes for menus, toolbars, and dialogs that allow classes using these GUI classes to know what operations a user performed.

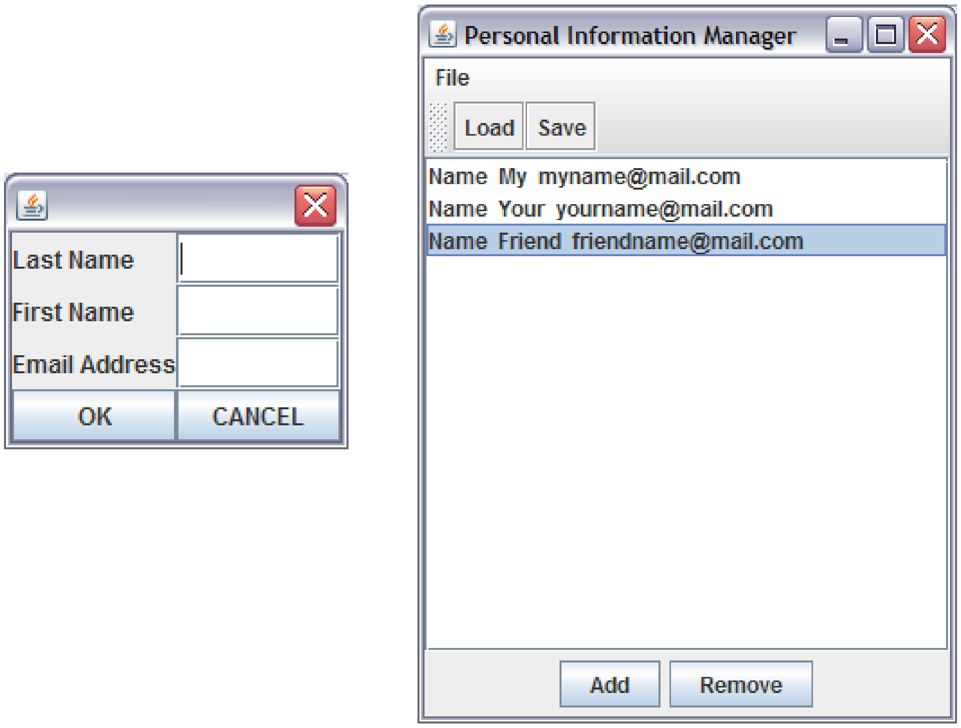
In this Lab you will implement a simple graphical Personal Information Manager (PIM). The PIM allows a user to add and remove contacts from a collection of contacts. For this exercise, you will complete the functionality of adding a new contact to the collection, by completing the implementation of the dialog shown below in Figure 1.



```
Name My myname@mail.com
Name Your yourname@mail.com
Name Friend friendname@mail.com
```

Figure 1.

Further, you will also implement a menu system that allows the user to load or save the contacts from/to a binary file by using either a menu or a toolbar. The final outcome after the completion of the Lab should be similar to Figure 2.



Name My myname@mail.com
Name Your yourname@mail.com
Name Friend friendname@mail.com

Figure 2.

Instructions:

This lab exercise requires a laptop with an Internet connection. Once you have completed the exercises in this document, your group will submit it for grading. All group members should legibly write their names at the top of this lab handout.

Make sure you read this handout and look at all of the source code posted on the class website for this lab exercise before you begin working.

1. Review the source code posted on the class website. Pay close attention to the **`ContactInfoGUI`** and **`ContactInfoDialog`** classes. The **`ContactInfo`** class is the data model for the dialog and you will use methods provided in this class to complete the **`ContactInfoDialog`** class.

2. Read through the source code of the **`ContactInfoDialog`** class and note the comments provided in the source code that give hints as to what needs to be done in the program.

3. Create **`JLabels`** and **`JTextFields`** for all of the information stored in the **`ContactInfo`** class. These components should be added to the **`ContactInfoDialog`** by calling the **`add()`** method of the dialog as shown in the constructor of **`ContactInfoDialog`**. Note that the code for these components will be similar to the existing code for the **OK** and **CANCEL** buttons.

Which components will be referenced by variables local to the constructor instead of by class variables and why?

4. Complete the **`actionPerformed()`** method of the **`ContactInfoDialog`** class. If the user clicks on the **`OK`** button, this method should save all of the data they entered into the data model of the dialog by using mutator methods provided by the data model. Some data has been provided for testing purposes only – you will need to use the actual data provided by the user.

   Note that if the user clicks on **`OK`** or **`CANCEL`**, the method will save the user's choice in the **`closeOption`** class variable, indicating how they closed the dialog. Finally, the method will call the **`dispose()`** method of **`JDialog`**.

5. Compile the lab assignment with your modified **`ContactInfoDialog`** class. You should not have needed to modify any class other than the **`ContactInfoDialog`** class thus far. Make sure that you can enter and remove contacts from the inventory using your code. Once you can, move on to the next step of building your menu system.

6. Create a menu item for each menu option (**`'Open'`**, **`'Save'`**, and **`'Exit'`**) to be added to the program under a **`'File'`** menu . The type of the menu item should be **`JMenuItem`**. These objects should be initialized in the constructor of the **`ContactInfoGUI`** class.

   (Note: In answering the question below consider which variables will be referenced in the **`actionPerformed()`** method.)

   Should the references to these **`JMenuItem`** objects be stored as class variables or variables local to a specific method?

To initialize the **`JMenuItem`**, the code will be similar to:

```
JMenuItem copyMenu = new JmenuItem("Copy");
```

7. In addition to the Menu and Menu items, we are going to implement a toolbar, using the **`JToolBar`** class. This toolbar will mimic some of the functionality of the menu that we have already added, but it is a nice addition. In this case, we will be adding "**`Load`**" and "**`Save`**" buttons. These buttons will do the same thing as the **`Load`** and **`Save`** menu items. In fact, your **`actionPerformed`** method should process them in the same fashion.

   To achieve this, create a **`JToolBar`** by declaring and instantiating it like you would any other GUI component. Then, create two new **`JButtons`**; one labeled "**`Load`**", and one labeled "**`Save`**". Now, add the buttons to the toolbar by calling the toolbar's **`add()`** method. Last, add the toolbar itself to the frame by calling **`frame.add()`**, just like we did with the scroll pane and button panel. We pass the thing we want to add (the toolbar) and the position we want it to have (**`BorderLayout.NORTH`** would be good) as parameters.

8. Inside the constructor for the **`ContactInfoGUI`** class you must also register an **`ActionListener`** on the **`JMenuItem`** by calling **`addActionListener()`** on each **`JMenuItem`** object. The **`ContactInfoGUI`** class should be used as the class that implements the **`ActionListener`** interface.

9. Create a **`JMenu`** object for the '**`file`**' menu. Add each menu item to the '**`file`**' menu using the **`JMenu add()`** method. Create a **`JMenuBar`** object and add the '**`file`**' menu using the **`JMenuBar add()`** method.

10. Now set the menu bar for the frame using the **`JFrame setJMenuBar()`** method.

11. Add an **`if`** statement for each menu item to the **`actionPerformed()`** method of **`ContactInfoGUI`**. The contents of the **`if`** statement should call **`System.exit(0)`**, the **`loadAction()`** method, or the **`saveAction()`** method depending on the object the event occurred on.

12. Test the program to make sure it correctly responds to menu selection events and properly reads and writes binary data files.

13. Submit the project archive following the steps given in the Submission Instructions by 2:00 pm, March, 27, 2009.

14. Turn in this lab handout to your lab instructor or one of the other instructors by 2:00 pm, Friday, March 27, 2009.