Computer Science Department

Mount Holyoke College

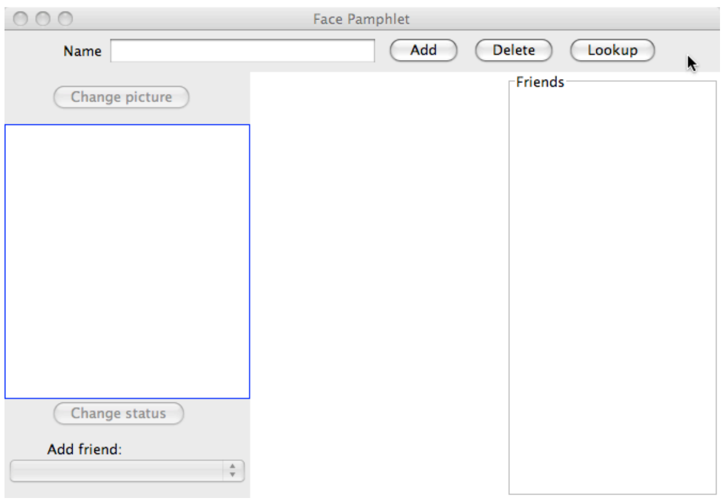
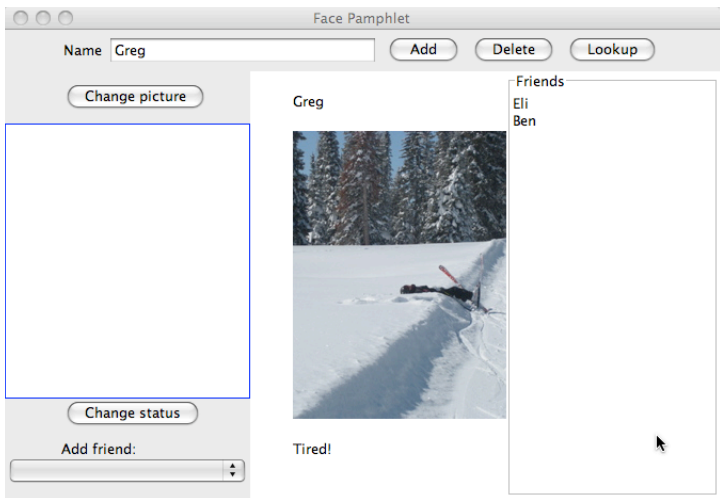
CS 102
Advanced Object-Oriented Programming
Lab 5 - Face Pamphlet
Due: March 29/30, 11:30 PM

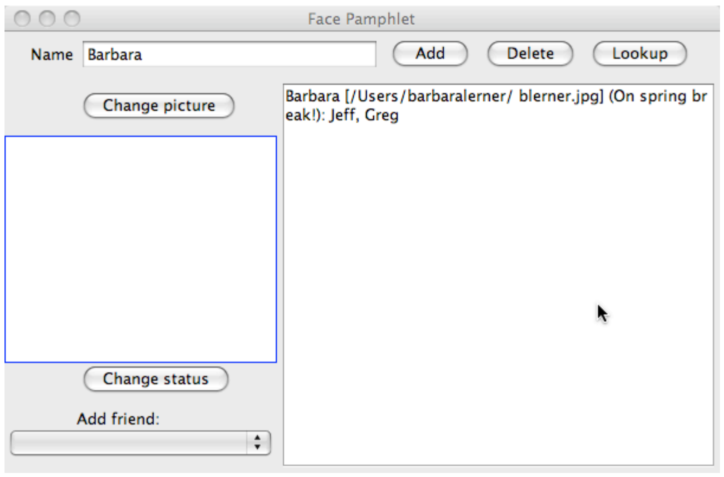
Introduction to the Assignment

In this lab, you will create a program that provides a small part of the functionality that Facebook provides. Users will be able to create profiles consisting of names, photos, status messages and friend lists.

The objectives of this assignment are to become familiar with the ArrayList data structure and also to practice using assert statements.

Here are some snapshots from my implementation of FacePamphlet:

 Face Pamphlet Name Add Delete Lookup Change picture Friends Change status Add friend:	This is the starting screen. The row of GUI controls at the top are used to add, delete, and lookup profiles associated with the name entered in the text field. The column to the left allows the user to change the picture or status message or add a friend to the profile being displayed. In the starting screen, no profile is displayed.
 Face Pamphlet Name Add Delete Lookup Change picture Friends Change status Add friend: Greg Greg Eli Ben Tired!	This shows the window when a profile is being displayed. Notice that the GUI controls in the left column are only enabled when there is a profile showing.

 Face Pamphlet Name Barbara Add Delete Lookup Change picture Change status Add friend: Barbara [/Users/barbaralerner/ blerner.jpg] (On spring break!): Jeff, Greg	This shows an alternative display of profile information. Here, the display is textual, showing the user name followed by the name of the file containing the picture in square brackets, then the status message in parentheses, with the list of friends following a colon.

The Program Design

This project has 5 classes and 2 interfaces organized into 3 packages. I will provide the interface definitions as well as implementation of the user interface display. The classes and interfaces are these:

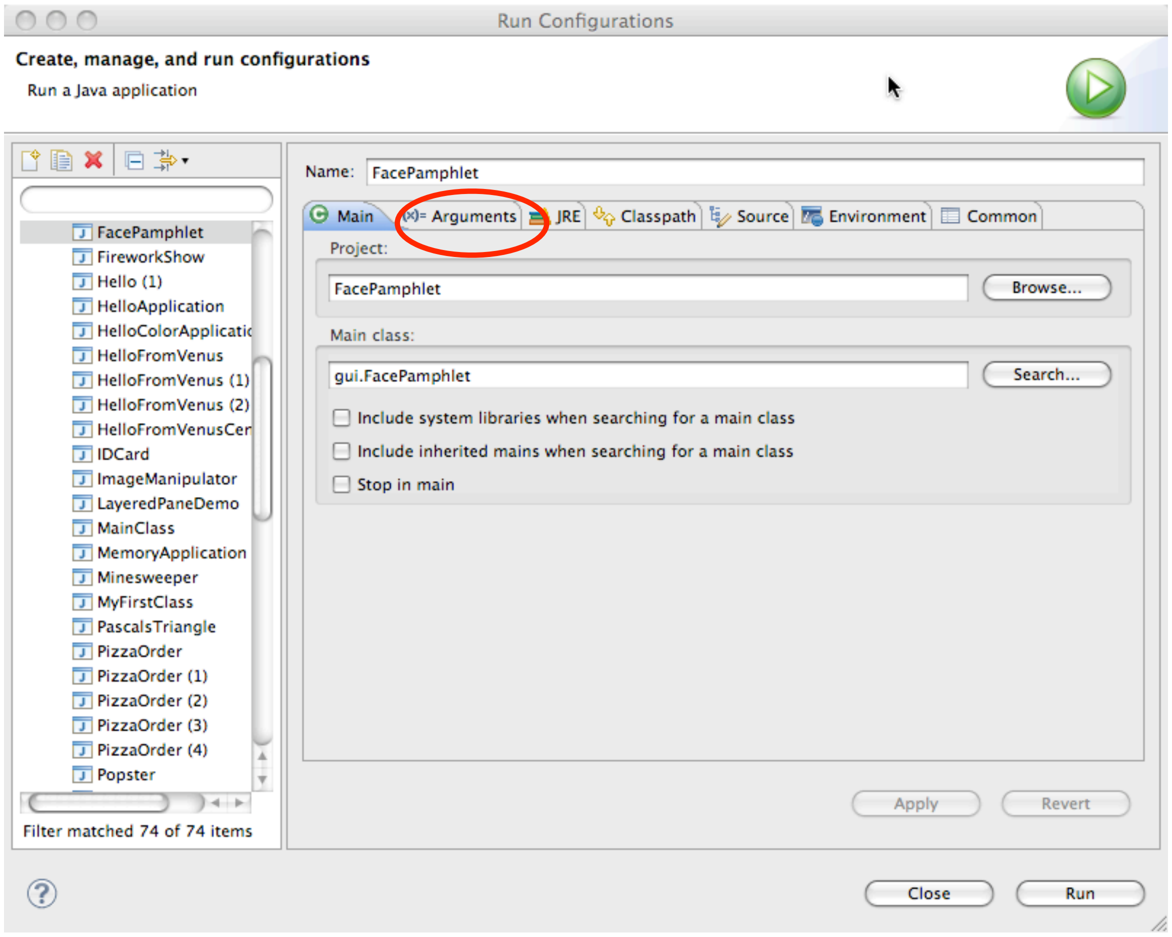
- gui.FacePamphlet - this contains the main method as well as the GUI code for the window layout and all the buttons, text fields and the friend menu. I will provide the code that does the display portions. You will need to add to this the listeners for the buttons and menu.
- gui.Canvas - this is an interface for the portion of the program in which the profile is displayed. I am providing this interface along with 2 implementations:
  - gui.TextCanvas - this displays the contents of the profile as text as shown in the 3rd screenshot above
  - gui.VisualCanvas - this displays the contents of the profile visually as shown in the 2nd screenshot above. This class is incomplete. You can get extra credit by completing it.
- model.Profile - this class holds the information about an individual person's profile. You will need to define this class.
- database.Database - this interface defines the functions required to store a collection of profiles. I will provide this interface.
- database.LocalDatabase - This is an implementation of the Database interface that manages the collection of profiles in an ArrayList in memory. You will need to define this class.

Writing the Program

Create a new Eclipse project called FacePamphlet. Create packages named database, gui and model. Download the following files from the course Web site: Database.java, Canvas.java, TextCanvas.java, VisualCanvas.java and FacePamphlet.java. If you are one of the students who has been having trouble with Eclipse creating mysterious ._ files when you do the import, please follow these instructions to do the import. Import these into your Eclipse project by selecting your FacePamphlet/src folder, opening the File menu and selecting Import. Open the General choice and select File System. Click Next. Click the Browse button and find the directory where you downloaded the files to and select Open. Select the files to import from the lists that show up in the Import menu. Be sure that the "Into folder" field says FacePamphlet/src. Click Finish. Once the files are imported, using the Package Explorer, drag each file to the appropriate package.

In this lab, you will have much less code to write than in recent labs. I want you to pay extra attention to documenting the contracts of the classes and methods that you write and to using assert statements to check preconditions. Remember that you need to run your program with assertion checking enabled for the assert statements to be executed. To enable assertion checking, do the following:

- Start your program as usual but quit immediately.
- Right-click on your project in the Package Explorer and select Run As… and then Run Configurations. You should see a window that looks like this:


Run Configurations

Create, manage, and run configurations

Run a Java application

Name: FacePamphlet

Main | Arguments | JRE | Classpath | Source | Environment | Common

Project: FacePamphlet

Main class: gui.FacePamphlet

- [ ] Include system libraries when searching for a main class
- [ ] Include inherited mains when searching for a main class
- [ ] Stop in main

FacePamphlet
FireworkShow
Hello (1)
HelloApplication
HelloColorApplicati
HelloFromVenus
HelloFromVenus (1)
HelloFromVenus (2)
HelloFromVenusCer
IDCard
ImageManipulator
LayeredPaneDemo
MainClass
MemoryApplication
Minesweeper
MyFirstClass
PascalsTriangle
PizzaOrder
PizzaOrder (1)
PizzaOrder (2)
PizzaOrder (3)
PizzaOrder (4)
Popster

Filter matched 74 of 74 items

- Select the Arguments tab.
- In the field labeled VM arguments, enter -ea

Now, when you run your program normally, assertion checking will be enabled.

Step 1: Create a profile

When the user types a name in the name field and clicks the "Add" button, a new user profile should be created. To implement this step, you will need to add an ActionListener to the Add button and implement the Profile class.

The Profile class is a class that encapsulates the profile information about an individual user. This consists of the person's name, the file that holds their image, their current status message and their list of friends. Eventually, you will need an instance variable for each of these pieces of information, as well as methods that allow you to find out these values for a profile and to change these values. For this step, all that you need is:

- an instance variable for the name,
- a constructor that takes a name as a parameter
- a getName method that returns the user's name, and
- a toString method that returns the user's name.

As you write these methods, also add your javadoc comments and your assert statements to check that parameters have legitimate values. For example, you should use an assert statement to ensure that the parameter passed to the constructor is not null.

Add an action listener to the add button. It should do the following:

- Check that the user has entered a name in the name field. If not, use JOptionPane to display an error message.
- Create the new profile and remember this profile in an instance variable.
- Display the new profile on the canvas by calling the following method that is already defined in the Canvas interface:
  `public void setProfile (Profile toDisplay)`
- Add the name to the people menu. The people menu is stored in the instance variable named peopleList in the FacePamphlet class. To add a name to the menu, pass the name to this method that is already defined in the JComboBox class:
  `public void addItem (Object newItem)`
- Enable the buttons to change the status and picture of the current profile and the menu to add a friend to the current profile.

To test this:

- Enter a name
- Click the Add button
- The canvas should show the new name
- The new name should appear in the menu
- The GUI components in the left column should be enabled.
- Also, check the behavior in the case that the name field is empty.

Step 2: Update the status of a profile

When the user clicks the Change status button, the text that the user has entered into the status field should be set as the status of the current profile and displayed. To do this, you need to add an action listener to the Change status button. It should:

- You will need to update your Profile class as follows:
  - Add an instance variable to hold the status associated with a profile
  - Add a method that can be passed a status value and save it in the instance variable
  - Add a method that returns the value in the status instance variable
  - Modify the toString method to add the status after the name, surrounding the status value with parentheses.
- In FacePamphlet, the action listener for the change status method should do the following:
  - Get the status from the status area.
  - Set the status in the current profile.
  - Update the display by calling the following method that is already defined for the canvas:
    `public void updateCanvas()`

    - Clear the status area.

To test this:

- Add a profile.
- Type something in the status field and click the change status button.
- The canvas should now display the name and status.
- The status field should be empty.
- Change the status and make sure the text display updates as expected.

Step 3: Add an image to the profile

When the user clicks the Change picture button, a JFileChooser should appear that allows the user to select an image file. I have already declared an instance variable named pictureChooser that you should use for this purpose. It has been created so that when it is displayed, it will show the files in the user's home directory.

To implement this step, you should modify your Profile class as follows:

- Add an instance variable whose type is File that will hold the File where the selected image can be found.
- Add a method that is passed a File parameter and saves it in the instance variable.
- Add a method that returns the value in the File instance variable.
- Modify the toString method to display the filename in square brackets following the profile name. Call the toString method on the File instance variable to get the file name to include in the string.

In FacePamphlet, you should add a listener to the change picture button that does the following:

- Display the JFileChooser by calling:
  `public int showOpenDialog (Component parent)`
  You should pass the FacePamphlet object as the parameter. This is telling the JOptionPane where it should appear on the display. It will appear near the component that is passed as the parameter.
- Compare the return value from showOpenDialog to JFileChooser.APPROVE_OPTION. This is an indicate that the user has clicked the Open button.
- If the user clicked the open button, you should find out which file she selected by calling the following method on the JFileChooser object:
  `public File getSelectedFile()`
- Set the picture associated with the current profile.
- Update the canvas.
- If the user clicked the Cancel button, you should not modify the profile or canvas.

To test this:

- Create a profile
- Click the change picture button
- Select a file and click open
- The display should update to show the filename.
- Select a different picture and make sure the display changes.
- Click the change picture button and click cancel. Make sure the display still shows the previous file.

Step 4: Look up a profile

Up to this point, your program has only been able to keep track of a single profile at a time. To add the ability to manage multiple profiles, you need to implement the LocalDatabase class. This class should implement the Database interface. For this step, you will need to define the addProfile, getProfile and containsProfile methods. For now, you can give the deleteProfile method an empty body.

LocalDatabase should have an instance variable whose type is ArrayList<Profile>. This will allow you to manage an arbitrary number of profiles. Here is what the methods in LocalDatabase should do:

- addProfile should add a new profile to the array list.
- getProfile should use a for loop to walk through the array list comparing the name in each profile to the name passed to getProfile. If a match is found, it should return that profile. If no match is found, it should return null. Remember to use .equals to compare the names, rather than ==.
- containsProfile should return false if getProfile returns null. Otherwise, it should return true.

In your FacePamphlet class, you need to make these changes:

- Declare an instance variable to hold the Database and initialize it with a new LocalDatabase object.
- Change the listener to the add button to add the new profile to the database. Before creating the profile, the add listener should check that there is not already a profile with that name in the database.
- Add a listener to the lookup button. It should:
  - Get the name from the name field.
  - Display an error message if the name field is empty.
  - Lookup the profile for that name.
  - If no profile is found, it should display an error message.
  - Set the current profile to the profile looked up.
  - Set the profile displayed on the canvas by calling this method on the canvas:
    `public void setProfile (Profile toDisplay)`

To test this:

- Create 2 profiles
- Type the name of the profile not currently displayed in the name field and click the lookup button.
- The looked-up profile should now be displayed.
- Change the status and make sure the profile being displayed is updated with the new status (not the other profile!)
- Try to add a 2nd profile with the same name as one of the existing ones and check that you get an error message.
- Try to do a lookup with the name field empty. The display should not change.
- Try to look up a profile for a name that doesn't exist. There should be an error message and the display should not change.

Step 5: Add a friend

Now that you are able to retrieve profiles from a database, you will be able to add friends to a profile. To do this, you will need to modify your Profile class in the following ways:

- Add an instance variable that can hold an array list of strings. This variable will hold the names of friends.
- Define a method that can add a friend to this array list.
- Define a method to check if a particular name is already a friend.
- Modify the toString method to add a list of friend names after a colon at the end of the string currently produced.

Now, add an action listener to the peopleList menu in your FacePamphlet class. It should:

- Determine which friend to add, saving the name in a variable. You can find out which name is selected by calling this method on the people list:
  `public Object getSelectedItem()`
  Call toString on the object returned to get the name of the new friend.
- Change the menu to display the special empty item by calling the following method on the people list, passing 0 as the parameter:
  `public void setSelectedIndex (int index)`
- Check that the user did not select the empty name. If the user did select the empty name, just do nothing and return.
- Check if the selected name matches the name of the current profile. If so, display an error message.
- Check if the selected name is already a friend of the displayed profile. If so, do nothing and return.
- If all these checks have passed, it is time to add a friend. So,
  - Add the friend to the current profile
  - We want friendships to be mutual automatically, so look up the profile of the new friend. Add the current profile as a friend to the looked up profile.
  - Update the canvas so that the new friend appears in the list.

To test this:

- Create 3 profiles.
- Add 1 friend. Make sure the friend appears in the friend list on the canvas.
- Look up the friend and make sure the previous profile appears in this list.
- Look up the 3rd profile and make sure its friend list is empty.
- Go back to one of the profiles with a friend and add a 2nd friend. Make sure the display shows both friends, separated by commas.
- Now, do the error checking:
  - Try selecting the first menu entry, which looks blank. Nothing should change.
  - Try adding oneself as a friend. There should be an error message.
  - Try adding someone who is already a friend. Nothing should change.

Step 6: Delete a profile

Finally, we want to be able to delete profiles from the database. The user will accomplish this by typing a user's name in the name field and clicking the delete button.

When removing a profile, we no longer want that person to show up on any friend lists. To support this, you need to add a method to the Profile class to remove a friend.

You also need to fill in the body of the method to delete a profile from the database in your LocalDatabase class. It should:

- Look up the profile for the name given.
- The person being deleted should no longer appear on any friend lists. So, walk through the list of friends of the profile being deleted and remove the name of the profile being deleted from the list of friends of each of its friends. For example, if Catherine and Susan are friends of Tammy and you want to delete Tammy's profile, you should get Tammy's profile and walk her list of friends. There you will find Catherine and Susan. Remove Tammy from Catherine's friend list. Then remove Tammy from Susan's friend list.
- Remove itself from the array list of profiles.

You should add an action listener to the delete button that does the following:

- Get the name from the name field.
- Make sure that the name is not empty. If it is empty, display an error message.
- Look up the profile.

- If there is no profile for that name, display an error message.
- Delete the profile from the database.
- If the profile being displayed is the one just deleted:
  - Clear the display by calling the following method from the canvas class:
    `public void clear()`
  - Disable the widgets that allow the user to update a profile by calling the following method in the FacePamphlet class:
    `private void disableProfileWidgets()`
- If the profile being displayed is not the one just deleted, update the canvas. This is necessary in the event that the profile being deleted was a friend of the profile being displayed.
- Clear the name field.
- Remove the name deleted from the people list by calling the following method on the peopleList object, passing in the name to remoe:
  `public void removeItem (Object item)`

To test this:

- Create 3 profiles.
- Add a friend to one of them.
- Delete one of the friends.
- Try to lookup the deleted profile. You should get an error saying there is no profile for that name.
- Make sure the deleted profile does not appear on the friend list.
- Make sure the deleted profile does not appear in the people menu.
- Delete the currently displayed profile and make sure the display gets cleared and the GUI components in the left column are disabled.
- Delete a profile for someone that is on the friend list of the currently displayed profile and make sure the display is updated. Make sure the GUI components in the left column are still enabled.

Extra Credit Step 7: Switch to the visual canvas

The VisualCanvas class that comes with the starter code does all the layout work to display a profile. It does not actually display a method because it lacks the calls to your Profile class to extract the necessary information from the Profile. To display a profile visually, you will need to update the components in the VisualCanvas class with information from the profile. Specifically, these components exist in VisualCanvas:

- nameLabel - a label where you can display the user's name
- image - a label where you can display the image selected by the user
- statusField - a label where you can display the status
- friendsArea - a text area where you can display the list of names

In addition, VisualCanvas inherits a variable named onDisplay. This contains the profile that should be displayed.

To complete the display of the profile, you need to fill in the updateCanvas method. Currently, it only displays the image. To complete the method, it should:

- Change the name label to contain the name of the current profile
- Change the status to display the status for the profile
- Change the friends area to display a list of friends separated by newlines. You can add a newline to a string by concatenating "\n" to it.

You also need to complete the clear method, which currently only clears the image. You should change it to also clear the name, status and friend list. You can change the value displayed in a JLabel or a JTextArea by calling:

```
public void setText (String newText)
```

To use the visual canvas instead of the text canvas, go to the FacePamphlet class and change the initialization of the profileCanvas variable to be a new VisualCanvas object. That's it! Since both TextCanvas and VisualCanvas implement the same interface, you need to make no other changes to your program to switch canvas types.

Grading

10	Step 1: Create a profile
5	Step 2: Update the status of a profile
5	Step 3: Add an image to the profile
10	Step 4: Look up a profile
10	Step 5: Add a friend
10	Step 6: Delete a profile
5	Extra credit Step 7: Switch to the visual canvas
15	Comments (including javadoc comments)
20	Assert statements
15	Style

Turning in Your Work

Create a jar file, being sure to include your source code. Then, go to ella. Click on COMSC 102 in the toolbar menu across the top. Then click on Assignments in the left column. Click the submit link for this assignment. Click the Add Attachment button. Use the Browse… button to upload a local file. You should submit your jar file. Click Continue. Then click the Submit button.

Credits

This lab was inspired by a Nifty Assignment by Mehran Sahami of Stanford University.