

## CS108 Assignment 5: Multiple-Window GUI

### Due Date: 14 April 2005 @ Midnight

#### Learning Objective

Learn how to build a fairly complicated graphical user interface for an application. Specifically, the GUI will employ multiple dialogs (at least 2) and should incorporate appropriate use of labels, buttons, check boxes, textboxes, lists or combo-boxes, and message boxes. The back-end implementation code will be provided to you.

#### Problem Statement

Instant Messaging is the generic name for many popular text-chat services available on the Internet and via mobile phones. An Instant Messaging system involves many computers in a hub-and-spoke topology. A centralized *server* (the hub) provides communication services to many *clients* (the spokes). Put another way, a user invokes a *client* application to connect to the server on her behalf.

In this assignment, you will implement a GUI application to be a client for an Instant Messaging (IM) system. The implementation for the IM client library will be provided to you in the updated *cs108.jar* file<sup>1</sup>. The IM server will be running on *azs.bu.edu*, port 43027.

#### Specific Requirements

Create a “main” window, which is launched at program start up:

- Display the application name in the title bar, provided intuitively named graphical widgets.
- Provide input fields to collect the server name and local user’s name.
- Provide buttons to *connect* and *disconnect* from the server, using the appropriate methods on the *IMClient* implementation class. Provide status/feedback to the user.
- Implement a “buddy list” in some manner that you choose. Track when buddies come online or go offline (by subscribing to events from the *IMMessageListenerInterface*) and update the buddy list accordingly.
- Provide a way to choose a buddy to chat with, and launch a “chat” window.

Create a “chat” window as an independent dialog. In the “chat” window:

- Provide an intuitive interface for sending and receiving messages.
- Send messages by calling methods on the *IMClient* class.
- Display received messages. Received messages are provided by subscribing to events from the *IMMessageListenerInterface*; you will receive these events in your “main” window, and you will need to post the messages to the “chat” window.

Provide error checking, using message boxes to provide feedback to the user:

- Allow connecting only when the server name and user name are specified by the user.
- Allow chatting only when connected.

**READ THE JAVADOC** to learn about how to interact with the *IMClient* library’s many classes and to learn about two sample applications that I have written. You absolutely will not be able to do this project without **reading the Javadoc** and following the code examples provided there.

---

<sup>1</sup>Get the new *cs108.jar* file from <http://people.bu.edu/azs/academics/cs108/cs108.jar>, and replace any previous version on your computer with this one. The documentation required for the implementation classes is available at <http://people.bu.edu/azs/academics/cs108/javadoc/>.

### Getting started/testing

- As always, do this project in small steps, and test as you go.
- Design the GUI first. To start, try doing everything on a single window.
- Test connect/disconnect, providing feedback to the user.
- Test subscribing for events by implementing *IMMessageListenerInterface*.
- Try sending and receiving messages. *Hint*: you can run 2 clients on your computer to test by yourself. You can also use the 2 sample clients provided in the cs108.jar file for testing.
- Once you're comfortable with connecting, and sending and receiving messages, move on to separating the "main" window from the "chat" window.

### Deliverables

- The Java code files (\*.java files).
- The NetBeans GUIs property files (\*.form files).
- A short executive summary, which will serve to introduce the application.
- Screen shots of each of your dialogs, copied and pasted into your executive summary document. You can obtain a screen shot of a dialog by invoking CTRL-PrintScreen, and then pasting (CTRL-V or whatever you usually do to paste).

### Grading Criteria

- Correct syntax, no compile errors, good code formatting, follow naming conventions, appropriate comments in the code
- GUI window is aesthetically pleasing: items lined up, font appropriate and readable, etc. User interface is intuitive, e.g. does not required instructions OR any of your friends who are not in CS108 could use it.
- Interaction with the back-end application works correctly: passing arguments to methods, retrieving return values, subscribing for events, and displaying results in the GUI window.
- Executive summary document: briefly explain the motivation for creating such an application, what the application does, how it works, why you used the programming elements you did for this type of project, as well as any lessons learned or recommendations for a revised implementation of this application.

**READ THE JAVADOC** to learn about how to interact with the *IMClient* library's many classes and to learn about two sample applications that I have written. You absolutely will not be able to do this project without **reading the Javadoc** and following the code examples provided there.

### Questions?

Bring them to class, or email me, [azs@bu.edu](mailto:azs@bu.edu).

**NOTE: I will take all questions up until 6 hours before the assignment deadline. After that time, you're on your own. Plan your time wisely and start early.**