Object-Oriented Software Development

Project 7 — Due: Friday, May 5

Phone Directory Server and Clients

Implement a server that acts as a phone directory and a client with a graphical user interface that can connect to the server and request information from the directory.

Server Side

Use a Map to store the names and phone numbers entered by clients. Serialize the Map to a file each time a change is made to it so that the directory will be available another day. The server should continue running until someone uses *control-c* to exit the program.

When the server program is started, it tries to read from a serialized file that contains the previously created directory. If this read fails, which it will the first time the program is executed, read an initial set of pairs, name and phone number, from a file called *directoryFile*. This file contains one name and one phone number separated by a sharp sign (#) on each line. You may download this text file from the course web page.

Design a protocol for communication between a client and the server and document it carefully in the server code. Design the protocol so that requests from a client and responses from the server are each contained in a single string (a single line of text). The documentation should include the port at which the server will be listening and instructions on how to connect to the server from telnet.

The protocol should work by sending strings of text back and forth between the client and the server. Use *readLine* and *println* with BufferedReaders and PrintWriters for the transmitted data You may use a Scanner for input if you choose.

Note that the server is designed so that you can connect to it using telnet. All messages in the protocol should be text only (strings). The server should check for illegal syntax in the requests that come to it and return appropriate error messages in response to these erroneous requests.

Design the server so that it can handle more than one client at a time (use threads). Recommendation: Write the server first without threads and test it carefully. Then add the threads to handle clients.

Client Side

The client side will have a GUI designed by you. It will include:

Four text fields labeled “Server” “Port”, “Name”, and “Phone number”.

Five buttons labeled “Connect”, “Quit”, “Lookup”, “Add”, and “Remove”.

A text area labeled “Messages”

Behavior on the client side (this list may not be complete):

1. Pressing the Connect button produces one of these messages:
   Connected to <host>.cs.uiowa.edu
   Not a server address
   Server does not respond

2. Pressing the Lookup button produces one of these messages:
   Name found (and number placed in the textfield for the phone number)
   Name not found
   Name missing on client side
   No connection in place

3. Pressing the Add button produces one of these messages:
   Name and number added to directory
   Number for this name changed
   Name or number missing on client side
   No connection in place

4. Pressing the Remove button produces one of these messages:
   Name and number removed from directory
   Name not found in directory
   Name missing on client side
   No connection in place

5. Pressing the Quit button produces the message:
   Goodbye

Some of these messages are produced by the client on its own, and others depend on responses from the server.

The server and client programs will be in two different text files so they could be run on different machines.

IMPORTANT: Build this project incrementally. Solve one problem at a time, testing it carefully to ensure that it works correctly.

Pair Programming

You will complete this project with a partner using Pair Programming, which was described in lecture.

As you work on the project, *keep a diary* that records the times that you work on the project with your partner and by yourself. Give the date, the duration of the work, and a brief description of the planning or code developed.

Time Table

Thursday, April 20	Turn in a paper with two names, you and your partner.
Thursday, April 27	Turn in a report describing the meetings with your partner and the progress made in planning the project. Also turn in and submit the code for a basic Server (no threads) that can be executed from telnet. (50 points)
Friday, May 5	Final report and submission of the complete project. (100 points)

The reports should be prepared on a computer with a text editor of some sort. In the final report identify the major components of the project (say 5 to 10 components) and estimate the amount (in percentages, 30%, 50%, 60%, etc.) of work performed by you and your buddy on each part of the project. Also discuss the advantages and disadvantages of working on the project with another person.

Submit your program files as instructed by the TA and turn in printed pages for your reports.

Your project will be graded based on the reports, the partial submissions, and the correctness and clarity of the final program.