Homework Assignment 2
Java Sockets
ECE 155B – Spring 2009
Due April 27

In the first assignment you explored Java's "write-once-run-anywhere" feature by implementing your Party Planner Service as a Java Applet running in a browser. A secure program for a PartyPlanner Service is unlikely to run as an Applet in a browser. Therefore, in this assignment, you will develop a PartyPlanner Service that has almost the same user interface as the first assignment but, instead of an Applet, it will be a Java Application.

Requirements

In this assignment you will revise and extend your PartyPlanner Service to use a Java Client Application (instead of a Java Applet) communicating with a Java Server Application using Java Sockets. In addition to the Java Server Application, i.e., PartyPlanner Service, you will introduce a Client for this project. In general, there can be multiple multiple Clients of a PartyPlanner (i.e., use multithreading for the PartyPlanner). The demo for this assignment will have one PartyPlanner and two Clients, as shown in Figure 1.

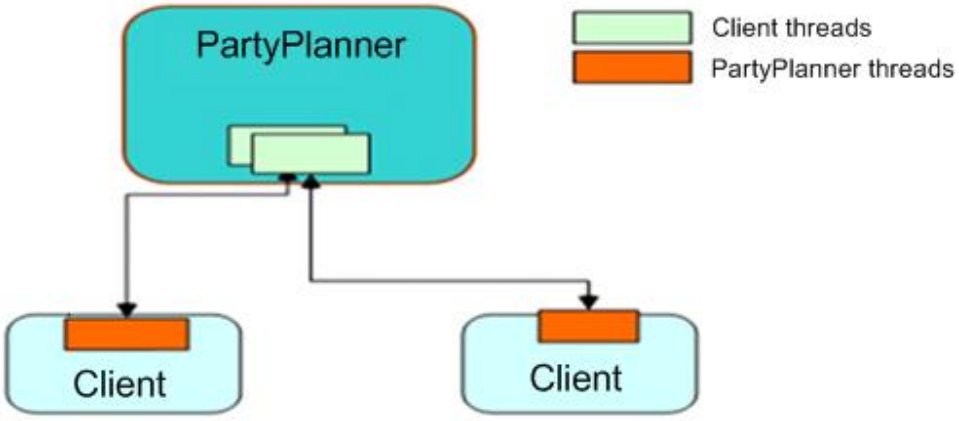

Figure 1. Demo Scenario.

The PartyPlanner Application must have a Graphical User Interface (GUI), and provide the same functionality as your Java Applet. You might consider using a tab in *JTabbedPane* in the GUI for the PartyPlanner to display client-specific information, and another tab to display information specific to the PartyPlanner, in order to have a simple GUI.

The PartyPlanner Application has a calendar that keeps track of future parties. A Client connects to the PartyPlanner Application and requests to schedule a party for a particular date/time. While the PartyPlanner is scheduling a party for one Client, it cannot also be scheduling a party for another Client. You will be asked to demonstrate this feature during the demo.

In addition, when arranging a party for a Client, the PartyPlanner Application will associate notes with the specific party for the Client. For future reference, the PartyPlanner saves the following information in a PartyPlanner XML file, as shown on the next page.

PartyPlanner Information
PartyPlanner first name, last name, etc. - **Clients** (for each client) • Information about client o First name, last name o Phone number o Interests o Address, etc. - **Facilities** (for each facility) • Information about facility o Name o Address o Phone number o Amenities, etc. – **Parties** (for each party) • Information about party o Date/time of party o Notes associated with the party ▪ Host ▪ Guest list ▪ Occasion, etc.

A Client sends a request to schedule a party to the PartyPlanner Application for a particular date/time. This communication will be done using Java Sockets. For future reference, the Client saves the following party information in a Client XML file.

Client Information
Client • PartyPlanner List o Name of party planner, URL:port information • Parties o Date/time of party o Facility name o Guest list

Socket communication requires, in addition to TCP/IP, an application-layer protocol between a communicating client and server. You will need to design the communication methods before you start coding. The client needs to know how to ask for information, and the server needs to know how to interpret a client's request and respond in an appropriate way.

Design Approach

In a project like this, it is better first to design a system where a single client connects to the server and the server greets the client or replies with the same message (i.e., an echo server). Then, using multiple threads, you can allow more clients to connect to the server and to send multiple messages to the server over their respective open connections.

Design Considerations

When transferring data between the Java Client Application and the Java Server Application, you will pass XML-formatted messages between the client and the server. You might need a common Message object and special XML parsers to do this. You will need *three different XML parsers:*

1 Messages exchanged between the client and server
2 Server (to keep server-specific information)
3 Client (to keep client-specific information).

For this project, the client needs to send well-formatted messages that the server will be able to understand. Consider the following:

1. *Authenticate Client*
   The Client sends name, password.
2. *Authenticate Client Reply*
   The PartyPlanner positively acknowledges the Client if there is such a Client record; otherwise, the communication ends with a *terminate message*.
3. *Request Facility of PartyPlanner*
   The Client requests a facility of the PartyPlanner for a specific date/time.
4. *Request Facility Reply*
   The PartyPlanner replies with the facilities that are available for the specific date/time, if any. If the PartyPlanner has no facility available, then the PartyPlanner replies with "No Facility Available" and the Client must reissue its request for another date and time.
5. *Request Party*
   The Client requests a party for a specific date/time and facility, with a list of guest names and the occasion.
6. *Request Party Reply*
   The PartyPlanner sends an acknowledgement to the Client that it has scheduled the party at the particular facility for the particular date/time.
7. *More as you wish.*

You will define your own protocol, i.e., how the client asks for information and how the server understands and responds to the client. It is not meaningful for the client to send messages that the server does not understand, and vice versa. Therefore, before implementing your protocol, make sure you have thought about every little detail.

Report (10%)

In addition to the programming portion of this assignment, you will write a small report. The report should be clear, well organized and concise. It needs to include the following:

1) An overview of how your application operates
2) Details of your design
3) A class diagram sketch outlining the hierarchy of your class structure
4) Sample input and output, e.g., the XML content before and after an operation, such as add or update.

Demonstration (10%)

You will be performing project demonstrations for this assignment and all others. You will be responsible for scheduling a 5-15 minute timeslot for your demonstration with the TA. Because time is limited, you should run your application on the PCs on which you will do your demo before the demo. In particular, make sure that those PCs are set up correctly. You will be required to demonstrate your application's functionality, and answer several questions covering the technology and your particular implementation. Demonstrations for the assignments will be scheduled through email, or during office hours/discussion.

Submission (10%)

You will submit a compressed file *(tar)*. You will need to type "*ant submission*" in order to create your submission file. If there is anything missing in the submission file, edit the *build.xml* file to include the missing information. Make sure you modify the *User* property in the *build.xml* file. The *ant* call will create submission file with the username you provide.

The compressed file should include a folder with all project content necessary to build/run the program (.java, README, Report). The projects are to be sent to ece155b.ta@gmail.com with subject line "Project 02: FirstName LastName".