

**Summer 2005: CS 438/631/731 Distributed Computing
Homework-1**

200 points. Individual Work Only. Due June 23, 2005 before class.

1. Read the online tutorial on Java RMI at:
<http://java.sun.com/developer/onlineTraining/rmi/RMI.html>. Download, compile, and test the Calculator program described in the tutorial following the instructions given. Also complete the Simple Banking System exercise. Note that the solutions are also provided in the tutorial.
[25 points]

2. Design and implement a client-server chat program in Java using Java RMI for communication between the client and server programs. The client and server programs must provide the following interfaces and methods: **[150 points]**
 - a. Server Program (ChatServer):
 - i. `public void connect(String name, ChatClient c)`
 - ii. `public void disconnect(String name)`
 - iii. `public void broadcast(String name, String message)`
 - iv. `public ChatClient lookup(String name)`
 - b. Client Program (ChatClient):
 - i. `public void update(String name, String message)`
 - ii. `public void gossip(String name, String message)`

Implement these interfaces using the following guidelines:

- The server methods will be invoked remotely by the client program to register a connection, send the chat message, and disconnect from the chat session.
- When the connect method is called the server adds the user name and the chat client reference to the list of chatters and then sends out a message to all clients (except the new user) that a new user has entered the chat session.
- Similarly when a client decides to leave the session the disconnect method is invoked and the server sends a message to all clients (except the one disconnecting) to indicate that this user has left the session.
- The client program must call the broadcast method of the server to send a message to all other users currently connected to the server. The server sends this message to all clients (except the sender) using the list of chatters that is maintained by the server.
- The server program must invoke the update method of the client program to communicate with the client programs whenever a user send a chat message or a new user enters the chat session or a user leaves the chat session.
- The client program can also obtain reference to a specific chat client using the lookup method of the chat server program and then invoke the gossip method of the corresponding client program to talk directly to another client without requiring the server to forward the message. The gossip method enables direct communication between two client programs and this message is not sent to other clients.
- Note that not only the client programs can invoke methods on the server but also the server can invoke methods of the client programs. Also one client can invoke the remote method of another instance of the client program (this implies that the client program must also extend `java.rmi.Remote`).

**Summer 2005: CS 438/631/731 Distributed Computing
Homework-1**

The main method in the server program will create an instance of the server and bind it to a URL and wait for keyboard input. The client programs will create an instance of the client, obtain a reference to the server using naming lookup, and connect to the server. Then the client program will read keyboard input in a loop until the string entered equals "quit." After reading a line the client program will check if the line starts with the character "@"." If the line starts with "@" then extract the user name present between two "@" characters, lookup the user using the server methods, and then call the gossip method of the corresponding client program. If the line does not start with "@" then call the server broadcast method.

Here is a simple chat session, assuming that rmiregistry and server program are started first followed by client programs for User1, User2, and User3. User entered text is shown in bold and italics characters.

\$ java ChatServerImpl

Server started, waiting for connections:

User1 Connected

User 2 Connected

User 1 Disconnected

.....

\$ java ChatClientImpl User1

(Server) Welcome User1, there are 1 user(s): User1

Hello any one out there?

(Server) User 2 joining now

(User2) Hello User1, How are you?

Hello User2, I am glad to hear from you.

(Server) User 3 joining now

(User3) Hello All, How are you?

[(User3) Hey hope you have not told anything to User2]

@User1@ No, don't worry

.....

exit

\$

\$ java ChatClientImpl User2

(Server) Welcome User2, there are 2 user(s): User1, User2

Hello User1, How are you?

(User1) Hello User2, I am glad to hear from you.

(Server) User 3 joining now

(User3) Hello All, How are you?

.....

(Server) User1 Disconnected, there are 2 user(s): User2, User3

.....

\$ java ChatClientImpl User3

(Server) Welcome User3, there are 3 user(s): User1, User2, User3

**Summer 2005: CS 438/631/731 Distributed Computing
Homework-1**

Hello All, How are you?

@User1@ Hey hope you have not told anything to User2

[(User1) No, don't worry]

.....

(Server) User1 Disconnected, there are 2 user(s): User2, User3

.....

Test your programs by starting the server on one machine and starting the clients on different machines. Instead of typing user inputs for the client program you can include the inputs into a file and redirect the input to the client program (e.g., java ChatClientImpl userx < inputfile).

3. How are parameters passed between the calling program and a function/method in Java on a single machine? Is there a difference between passing primitive parameters vs. passing objects as parameters? **[5 points]**
4. What happens when an object is cloned in Java on a single machine? **[5 points]**
5. What are remote objects in Java? How is remote object parameter passing handled in Java? **[5 points]**
6. What happens when a remote object is cloned in Java? How does the client program interact with the cloned remote object? **[10 points]**

For Graduate Students (Bonus question for undergraduate students):

If multiple clients programs are started at the same time will there be any synchronization issues that will arise? How do we address such synchronization issues? Identify the potential areas where these problems could arise and explain how these issues will be addressed. There is no need to implement these changes, just provide a description. **[20 points]**

Feedback Questions (answer to these questions has no impact on your grade):

- Was this homework too difficult, or too easy?
- Was the assignment fun or challenging?
- Was there something that was unclear?
- Was the project too long for the given amount of time?
- What did you learn from this homework?

Submission Instructions:

Make two separate directories one for the server and one for the client program:

Following files must be included in the Server directory:

1. Server Service Interface Definitions
2. Server Service Implementations
3. Skeletons for the Server Implementations

**Summer 2005: CS 438/631/731 Distributed Computing
Homework-1**

4. Stubs for the Server Implementations
5. Stubs for the Client Implementations

Following files must be included in the Client directory:

1. Client Service Interface Definitions
2. Client Service Implementations
3. Skeletons for the Client Implementations
4. Stubs for the Client Implementations
5. Stubs for the Server Implementations

List ALL the references you used in this homework as well as test cases used to test your programs. This includes any classes that you used that you did not write and any help you received from any other sources. Use appropriate class name and include comments to indicate various operations performed by the program. Your program must have the following header information within comments:

/*

Name:
BlazerId:
Homework #

*/

Create a tar/zip file using the following filename format: blazerId-cs438-hw1.tar or blazerId-cs438-hw1.zip. The tar/zip file must include the source code and class files, txt/word/pdf/ps document for the typed solution to questions 3-6 and the feedback questions, instructions for executing the programs and test data used to test the programs. Email the tar/zip file with the subject header "CS 438/631/731 Homework1" to puri@cis.uab.edu. If you do not email with the correct message header your submission will not be processed in a timely manner, so please make sure you include the header. Also please do not make multiple submissions. There is no need to turn in any printed solutions for this homework, email submission is sufficient.

Late Submissions:

Submissions must be made on the due date before the beginning of the class. Late submissions will lose 10% for every 24-hour period, up to a maximum of 50% (weekends and holidays count as one 24-hour period). Any submissions made after one-week will receive a score of 0 for this homework.