Strathmore
UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

Bachelor of Science in Informatics and Computer Science

Bachelor of Science in Telecommunications

BTC4201 / ICS4104 – Distributed Systems

Assignment – Inter-process Communications in Distributed Environment (Worth 15%)

Assignment Deadline: **5th August 2020 @ 5.15 pm**

A. Assignment Questions

Sockets and RMI are two techniques in Java which could be used to establish two-way communication between two running programs (typically a client and a server) running on the network. But in principal, they work in quite different ways. A socket is just a way to send data (only data, not methods) on a port to a different host; it's up to you to define your own protocol.

RMI is a technology in which the methods of remote Java objects can be invoked from other Java virtual machines running on the different hosts. In this assignment, you will implement client-server communication programs based on both Java Socket and RMI techniques. In this Assignment we'll use plain sockets.

Learning about Sockets

Read the following resource carefully to learn about sockets:

http://java.sun.com/docs/books/tutorial/networking/sockets/index.html

If you never did any network programming in Java, it may be a good idea to read some more: http://java.sun.com/docs/books/tutorial/networking/index.html

You may want to have a look at other information:
http://www.javaworld.com/javaworld/jw-12-1996/jw-12-sockets.html

B. Assignment Tasks:

A Toy Merchant application example communicates with a wholesale toy-service engine on the server that keeps a toy-price table to record the different kinds of toy's prices, takes the clients' tasks, queries the toy-price table, does the calculation and returns all results.

You will need to write client and a server programs to facilitate interactions using RPC techniques for the following tasks for Client and Server respectively:

Your **client** should be able to

1. Send the toy identification details (toy code, toy name) to the server program,
2. Send the toy information (name, description, price, date of manufacture, batch number) to the server program,
3. Send the toy manufacturer details ((company name, street address, zip-code, country) to the server program,
4. Send a thank you message with a unique code (Innovate ☺) to the server program
5. Send all the above toy information in one single instruction to the server program

Your **server** should be able to

1. Ask the client program to send the toy identification details (toy code, toy name)
2. Ask the client program to send the toy information (name, description, price, date of manufacture, batch number)
3. Ask the client program to send the toy manufacturer details (company name, street address, zip-code, country).
4. Ask the client program to send a thank you message with a unique identification code.
5. Ask the client program to send all the toy information in one single instruction.
6. Server to send to client a message to indicate the communication succeeded or aborted.

Note: How to design the *server-client communication protocol* is up to you but you must ensure that you have a GUI for interactions between the Clients and Servers.

Implementation:

You have to implement in at least four (4) java classes namely:

1. *SocketClient.java*: This class connects to the server and communicates with it according to the Client Protocol
2. *ClientProtocol.java*: This class describes how your client reacts to your servers' messages (see the "Knock-Knock-Protocol in the Java Tutorial)
3. *SocketServer.java*: This class sets up a socket listening for your client to connect - and communicates with your client
4. *ServerProtocol.java*: This class describes how your server reacts to the messages of your client

Additional Information:

You are only required to test your server and client communication locally. When you create a socket, use **localhost** instead of IP address and port number. We encourage each group (**max. of 4 persons**) to run your server and client programs on different machines and do ensure port number should be dynamic noting that some ports might not be available on certain clients or servers' machines. You can use any RPC implementation or Remote Java implementation of ONC/RPC.

C. Submission Location and File

a. Submit your work on E-Learning Site against the specific Tutorial noting the deadline times appropriately to avoid disappointments

b. Use the Assignment Documentation Template to capture the relevant details for the Assignment and Group Members.

D. Reference Texts

a. Coulouris, G., Dollimore, J. & Kindburg, T.: *Distributed Systems, Concepts and Design*. Addison Wesley. ISBN 0-201-61918-0 (4th or 5th Editions).

b. Tanenbaum, A. S. & van Steen, M: *Distributed Systems: Principles and Paradigms*. Pearson Education International/Prentice-Hall International. ISBN 0-13-121786-0. (2nd Edition).

E. References Listings

For all the consulted any references for the assignment, cite/list them using the APA style. Check these URLs for guidance:

a. http://www.library.cornell.edu/resrch/citmanage/apa

b. https://apastyle.apa.org/6th-edition-resources/basics-tutorial

c. https://guides.library.ubc.ca/howtocite