CIS656 – Distributed Systems – Fall 2022
Programming Assignment 4 – Java RMI and Callback
Maximum Points: 100 (3% of the final grade)
Due: 8:30PM on Wednesday, November 2, 2022
(Please finish this programming assignment individually, or as group of two or three people)

Objective

The objective of this programming assignment is for students to understand how Java RMI and Callback work in distributed systems. Upon completion, students will be able to use Java RMI and Callback to create a prototype for communication between the server and clients.

Description

Please download "RMI.zip" from BlackBoard. It provides a Java RMI example to perform a simple addition method initiated by the client.

First, please study the sample code and understand how Java RMI works. You can follow the presentation "Java-RMI-Implementation-Tutorial and Callback.pptx" to compile and run the programs.

Next, modify "MyClient.java", "Method.java" and "MethodRemote.java" to get the result as follows.

- Run the server first. The server should print a statement "The server is running." and waits for the clients' connections.
- The client asks the user to enter a string.
  - If a string "time" is entered, the client will get current date and time from server by remotely invocating the method "action()" in server machine.
  - If another string other than "time" is entered, then the client will get the capitalized version of the string the user just entered by remotely invocating the method "action()" in server machine.
  - Your client program should keep asking the user's input for a string unless an empty string is entered (i.e., just press ENTER key without anything else). When an empty string is entered, this client will quit.
- While a client is being connected to the server, the server should be able to receive new clients' connections for current date/time or capitalized strings. In other words, the server should concurrently handle multiple clients' connections and requests well.
- After a client quits, the server should keep running and be able to receive other clients' connections, unless you manually shut down the server program.

Then, you need to further modify the programs to implement Callback. The "Callback" will let the server remotely call a method on a client. In this assignment, the server will use Callback to pass a string as the argument to return the message (a client asks for) to the client. Please follow the idea in the presentation "Java-RMI-Implementation-Tutorial and Callback.pptx" to finish that. The detail is as follows.

A client still asks for a string from the user. Same as the previous step, based on whether it is "time" or any other string, the server will return current date/time or capitalized version of the string to the client. Here, since Callback is implemented, a client actually has two ways to get the message from the server, RMI and Callback.

Hence, in addition to RMI, the Callback will be there for the server to send the message back to a client. The method of the Callback is on the client, so that the argument of this callback method is the exact message the server will return to the client. By taking advantage of argument passing, the server can successfully send the message back to the client.

In that case, the following example works as follows. We assume the server runs on the "eos02" machine, and a client runs on the "eos03" machine. On the client's terminal, it shows

Enter a string to send to the server (empty to quit):
Distributed systems
Using Callback (eos03): ***DISTRIBUTED SYSTEMS***
Using RMI from eos02: ***DISTRIBUTED SYSTEMS***

Enter a string to send to the server (empty to quit):
time
Using Callback (eos03): ***Mon Oct 11 00:18:27 EDT 2021***
Using RMI from eos02: ***Mon Oct 11 00:18:27 EDT 2021***

Here, the client program keeps asking the user's input. We can see that the client gets two identical messages (date/time or capitalized string in bold) from the server (one via RMI and the other via Callback).

- For RMI, we can see that the client remotely invokes a method on the server's machine, so that the message is from "eos02", where the server program resides.
- For the Callback, since the server remotely invokes the callback method on a client, the message is only printed on a client's machine ("eos03" in this case). In other words, the message should not be printed on server's machine, so that the server should only have one print statement "The server is running." on its terminal.

Note:

- **For RMI implementation, the client must use RMI to get a message from the server. In other words, instead of locally getting the string on the client itself, the client should remotely invoke a method in the server to acquire the message.**
- **For Callback implementation, you must make sure the server indeed remotely calls a method on a client, so that the message is printed on the**

client side, rather than on the server side, as shown in the previous example.

- **Same as the previous step, your client program should keep asking the user's input for a string unless an empty string is entered. When an empty string is entered, this client will quit. The server should concurrently handle multiple clients' connections and requests well. After a client quits, the server should keep running and be able to receive other clients' connections, unless you manually shut down the server program.**

Deliverables: For full credit, **please demo your program and show the result to the instructor. Please keep your source files for future programming assignments.**