DS – Lab 3
Asynchronous Communication

RMI with remote callback

The given program emulates a temperature sensor, which would notify the remote clients that are listening, whenever there is a change in the temperature.

1. For all the methods given in the source files, write detailed comments explaining the purpose of each and every function and class.

2. Insert the code to register the TemperatureListener object (client remote interface) in the server object. The location is marked with a “To do” in the TemperatureMonitor (client) class.

3. Insert the code in the server (TemperatureSensorServer) to notify the Listeners (TemperatureListener objects in the client), whenever there’s a change in the temperature. This is marked with a “To do” in the TemperatureSensorServer class.

4. Compile the modified source files.

5. Run the rmiregistry using command line (command: start rmiregistry)

6. Run the Server

7. Run the client and see whether the asynchronous callback function (temperatureChanged) on the client (TemperatureMonitor) is called by the server. You may run multiple clients.

Asynchronous messaging with Message Queues

For this, you will need to download and run Apache Active MQ in your system. If you are using your own pc, you can download it from here. Extract the downloaded zip file to a directory in your pc.

https://activemq.apache.org/components/classic/download/

1. Run the Apache Active MQ message queue. In order to run Apache MQ, you may have to go inside the relevant bin directory. For example, *<<Apache MQ install directory>>/bin/win64/activemq.bat* for a 64 bit system.

2. Go to the administrative console by opening the url http://localhost:8161/

   Type the username and password 'admin' 'admin' to login. View the available options.

3. Open the given eclipse project.

4. Run the Receiver class inside Java Resources as a java application (right click-> Run as-> java application).

5. Run the Sender as a java application.

6. Go to the admin console and view Queues to view the message status.

7. Currently the sender sends messages synchronously. Do the changes suggested in the TO DO comments in both the Sender and Receiver to make it send async messages. Run the Receiver and Sender to see what happens.

8. Observer the message queue console to see how the message status is updated. Debug the code and try to understand how the synchronous and the asynchronous messaging works.

Submission:

Upload the modified code (excluding the project) of both exercises as a single zip file to the courseweb link. The name of the file should be your registration number