Distributed Systems Practical 01: A very basic distributed system

Introduction:

In this practical you will be introduced to Java's RMI (Remote Method Invocation) system. What RMI allows you to do is pass a computation task onto another machine and have it run the task and return to you the result. Using RMI we will experiment with a very basic distributed system. We will follow Oracle's Java RMI tutorial which is available on the web at http://docs.oracle.com/javase/tutorial/rmi/overview.html However we will make some small modifications to the RMI tutorial to get it working on our machines

NOTE: you do not need to install any software to get this working. RMI is included as a part of the standard Java package.

The tutorial defines a very simple distributed system comprised of three parts: the RMI registry, the compute server, and the client. In this system the RMI registry is a standard component that allows RMI applications to register their remote objects and the methods they expose. Thus external processes (maybe on another machine) can query and get access to these remote objects for computation tasks.

The compute server is a simple server we will define that will be able to take any computation task and run it. The compute server will have zero knowledge of the task it is running i.e. the task is defined externally and all code to compute the task will be passed to the compute engine which will run this task and pass the results back to the external process that created the task.

The last part of the setup is the client. This client will contain a task for computation in this case a computation for caluclating the value of PI (3.14159...) The client when run will send it's task to the compute server, the server will run the computation and then return the result to the client which will immediately exit.

NOTE: if you run an RMI based program an it appears to show the results of previously written code kill the rmiregistry and the compute server and restart them.

Problems:

01) Go to the tutorial listed above and go to the section marked “Compiling the Example Programs”

02) Download all 5 source files and store them in appropriate folders. Store Compute.java and Task.java in a folder called compute. Store ComputeEngine.java in a folder called engine. Store ComputePi.java and Pi.java in a folder called client.

03) Compile all the code and create a jar file out of the compute folder.

04) Go to the next section in the tutorial marked “Running the Example Programs” and start the RMI registry by using the command line as follows “start rmiregistry”.

05) Get the RMI support files zip from the moodle and extract all 4 files to your folder containing all the code. The files should have two policies for the server and client and two bat files one for the server and one for the client. For those of you using Linux or OSX you can simply convert these files to shell files by renaming from .bat to .sh and by adding the line #!/bin/sh to the top of the files.

06) run the server.bat off the command line as follows “start server.bat”. If it is working it should respond with the message “ComputeEngine bound

07) run the client.bat off the command line. If it is working it should respond with the value of PI to about 20 decimal places

08) create a pair of new files in the client folder. One called ComputeFactorial.java and the other called Factorial.java. Write a computation task similar in form to ComputePi and Pi to compute the factorial of any number passed on the command line. e.g. if you pass the number 5 you should get 120 back. if you pass the number 6 you should get 720 back etc etc.