Department of Computer Science
COMP212 - 2020 - CA Assignment 2
Distributed Communications
Java RMI

Assessment Information

Assignment Number	2 (of 2)
Weighting	10%
Assignment Circulated	4th March 2020
Deadline	26th March 2020, 17:00 UK Time (UTC)
Submission Mode	Electronic via VITAL
Learning outcome assessed	(1) An appreciation of the main principles underlying distributed systems: processes, communication, naming, synchronisation, consistency, fault tolerance, and security.
Purpose of assessment Marking criteria	This assignment assesses the understanding of Java RMI. Marks for each question are indicated under the corresponding question.
Submission necessary in order to satisfy Module requirements?	No
Late Submission Penalty	Standard UoL Policy.

1 Overall marking scheme

The coursework for COMP212 consists of two assignments contributing altogether 20% of the final mark. The contribution of the single assignments is as follows:

Assignment 1	10%
Assignment 2	10%
TOTAL	20%

2 Objectives

This assignment requires you to write a Java program implementing client-server communication using **Remote Method Invocation (RMI)**.

> **Note that *no credit will be given for implementing any other form of client-server communication!***

3 Explanation of coursework

Imagine you are designing a distributed voting application: While the server is running, a client can connect to the server and vote for 1 out of 3 choices (by means of the method `vote`, see below). *After* voting, the client can request the voting results so far.

3.1 Basic Java RMI communication—50% of the assignment mark

As a first step, you are required to implement client-server communication only. At this stage you can ignore the program logic, but you should show some evidence that your program works and that communication between the client and server actually takes place.

For your convenience, we *suggest* the following remote object interface (however, feel free to modify it to your needs):

`castVote.java:`

```
interface castVote extends java.rmi.Remote {
   public void vote (int choice) throws java.rmi.RemoteException;
   public String getVotingResults() throws java.rmi.RemoteException;
}
```

You must implement the following classes.

3.1.1 Class castVoteImp.java

This class should implement the `castVote` interface.

3.1.2 Server side

The server should initiate and register the remote object.

3.1.3 Client side

The client should submit the user choice to the remote server and then print the current vote and voting results on screen.

Hint. Start with downloading the RMI example from VITAL under the "Assessment Submission" tab. Following the instructions, make sure it works on your computer. Then use that code as a basis for your implementation.

3.2 Program logic—50% of the assignment mark

To get further 30%, you need to implement the remote object, the counting of votes and returning voting results, and also implement the client side functionality. To get the remaining 20%, you need to extend the communication model to make sure that the server will only allow a client to see the current voting results *after* submitting its own vote.

Hint. Such a functionality can be achieved, for example, by assigning to every client a *ticket* (a randomly generated integer number). A client without a ticket will be denied all communication. A client first requests a ticket from the server and then passes this ticket as an extra parameter to all remote methods. When the server issues a ticket, it saves the ticket to a data structure (for example, a hash table) so that all remote methods (`vote`, `getVotingResults`, etc) will be able to check that the ticket is valid.

You are more than welcome to suggest your own method to control the right order of calls.

4 Deadline and Submission Instructions

- The deadline for submitting this assignment is **Thursday, 26th March 2020, 17:00 UK time (UTC)**.
- Submit
  (a) The Java source code for all your programs and
  (b) A README file (plain text) describing how to compile/run your code to produce the various results required by the assignment.

  Compress all of the above files into **a single ZIP** file (the electronic submission system won't accept any other file formats) and **specify the filename** as *Surname-Name-ID*.zip. The zip should contain all the appropriate .java and .class files of the client side and server side in *two separate directories*, one for the client code and the other for the server code. Each directory should contain all the necessary files to run the client/server program including the interface file. Please write a client and a server that run on the same host (localhost) in order to facilitate the execution of the code on our computers. It is extremely important that you adhere to the required format of submission, otherwise we may encounter difficulties in executing and assessing your program!

- Submission is via the “Assessment Submission” tab of 201920-COMP212-DISTRIBUTED SYSTEMS on VITAL.

Good luck to all!