Lab#2 – Remote Method Invocation

Objective

In this lab you will implement a service that is remotely invoked by clients using Java RMI. You will demonstrate your running programs to the TA. You will also write a short lab report containing answers to questions at the end of this document. Submit the lab report on Blackboard, the deadline is the date of Lab#2.

The *Election* Service

The interface of the *Election* service provides two remote methods:

- *vote*: This method has two parameters through which the client supplies the name of a candidate (a string) and the "voter's number" (an integer used to ensure each user votes once only). The voter's numbers are allocated sparsely from the range of integers to make them hard to guess.
- *result*: This method has two parameters through which the server supplies the client with the name of a candidate and the number of votes for that candidate.

The *Election* service must record a vote whenever a user thinks they have cast a vote. The records should remain consistent when it is accessed concurrently by multiple clients.

Tasks

Task #1:

Define the interface to the *Election* service in Java RMI.

Presentation to the TA:

Show the TA the *Election* interface.

Task #2:

Implement the *Election* service in Java RMI. Your implementation should ensure that a vote is recorded whenever a client thinks they have sent in a vote.

Write a client program and run multiple instances of the client to cast votes. A client may also query the server for the results and display them.

Presentation to the TA:

Show the TA the running programs. You should show (1) multiple clients sending in votes; (2) a client querying the server for the results and displaying them.

Task #3:

Now modify, if necessary, your implementation to ensure that the votes are recorded even when the server process crashes.

Presentation to the TA:

Show the TA the running programs as above with the additional scenario where the server process terminates unexpectedly and the votes that have already been cast are still recorded.

Task #4:

Modify your implementation, if necessary, to ensure that the records remain consistent when it is concurrently accessed by multiple clients.

Presentation to the TA:

You do not need to demonstrate this part to the TA. You should put an outline of this in your lab report (see below).

Lab Report:

Write in your lab report the answers to the following questions:

1. What did you use in your implementation to ensure that the *Election* service records a vote whenever a client thinks they have cast a vote?
2. How did you achieve the guarantee that all votes are safely stored even when the server process crashes?
3. Outline your implementation for ensuring that the records remain consistent when it is concurrently accessed by multiple clients. You should include snippets of your program to help with your explanation.

Submitting lab report :

Submit your lab report on Blackboard by the day of the lab session (11:59PM).