COMP30220 Distributed Systems Practical

Lab 2: RMI-based Distribution

Work individually. Submit your code on csmoodle.ucd.ie by the deadline given on moodle.

Scenario:

You have just been asked to take over a project by your manager. The project is to build an insurance quotation system for a broker. A previous developer was initially working on the project and they have built a small prototype. The reason you have been asked to take over the project is that the previous developer did not know how to distribute the solution (they built only a standard Java solution). The basic system is organised around two main services:

- **Quotation Services**: these services generate quotations based on supplied client information. The client information is passed as an instance of the `ClientInfo` interface.
- **Broker Services**: these services are used to access the quotation services. An instance of a `ClientInfo` object is passed to the service, and the broker must then contact ALL the available Quotation Services gathering as many quotations as possible. Quotations are returned as instances of the `Quotation` interface.

Currently, Quotation Services have been built for the following companies:

- **Girl Power Insurance**: This company provides discounts for female drivers
- **Auldfellas Insurance**: This company provides discounts for men over the age of 60.
- **Dodgy Drivers Inc**.: This company provides discounts for drivers with penalty points.

To test the system, some test classes have been created in the `test` package. Specifically, the `test.Main` class provides a test run of the system where the three Quotation Services are instantiated; a Broker Service is instantiated; and 3 test clients are submitted to the broker.

In the prototype implementation, the developer attempted to enforce a clear separation of concerns by implementing a `ServiceRegistry` class. Quotations Services register with this class, and the Broker Service uses the class to locate all available Quotation Services.

Problem 1: Distributing the Solution **Total: 50%**

The immediate task you are assigned by your manager is to distribute the code. Given it is all written in Java, you decide to use an RMI based approach. To do this, you must replace the `ServiceRegistry` class with an RMI-based Implementation.

a) Modify the Main class to register the quotation services as Remote Objects. This may involve modification to some of the other Java code. **10%**

b) Create a new implementation of the `BrokerService` interface called `RMIBrokerService`. This class should perform the same behaviour as the `LocalBrokerService` class, but using RMI instead of the `ServiceRegistry` class. **25%**

c) Modify your code so that each Quotation Service and each Broker Service can be run in a separate JVM. 5% of the marks will be given if you can find a way to allow the Quotation Services to be started in (i) ANY ORDER and (ii) using only Java (it is built into each program). Write a short text file called README.txt (it should be in the root folder of your project) explaining how to run your distributed solution FROM THE COMMAND LINE. **15%**

Problem 2: Adding a Vetting Service **Total: 40%**

The second problem you are assigned is to implement a Vetting Service. This additional service also takes as input an instance of the `ClientInfo` interface. It contains an in memory database of the number of penalty points officially attributed to each driver. Drivers are represented by their driving licence number. The service simply confirms that the number of penalty points submitted by the client matches that actual number of penalty points the driver has. To help you get started, the previous developer had already defined the interface: `core.VettingService`. You should implement this service, make it available as a remote service through RMI and modify the broker code so that only clients that pass the vetting receive quotations.

a) Implement the Vetting Service by creating a class called `RMIVettingService` that implements the `core.VettingService` interface. Test your service locally (you should use the `XXXServiceTest` classes as an example of how to do this). **20%**

b) Now distribute the service using RMI and modify the Broker Service to call this new service before the Quotation Services are called. Revise your README.txt file to include details of how to deploy this additional service **20%**

IMPORTANT: As with last week, <u>10% of your final mark</u> will be allocated based on non-functional requirements (quality of code, commenting, indentation, and the quality of the README.txt file).

NOTES:

- I recommend that you download and run the Calculator example from the last lecture before you start writing your own solution.

- When you are faced with a problem, please try to (a) google the problem and (b) read your notes before asking for help. For most problems you face, there is already an answer on the web. This is what I do when I need additional information and it is good practice for you to get into the habit of doing this.