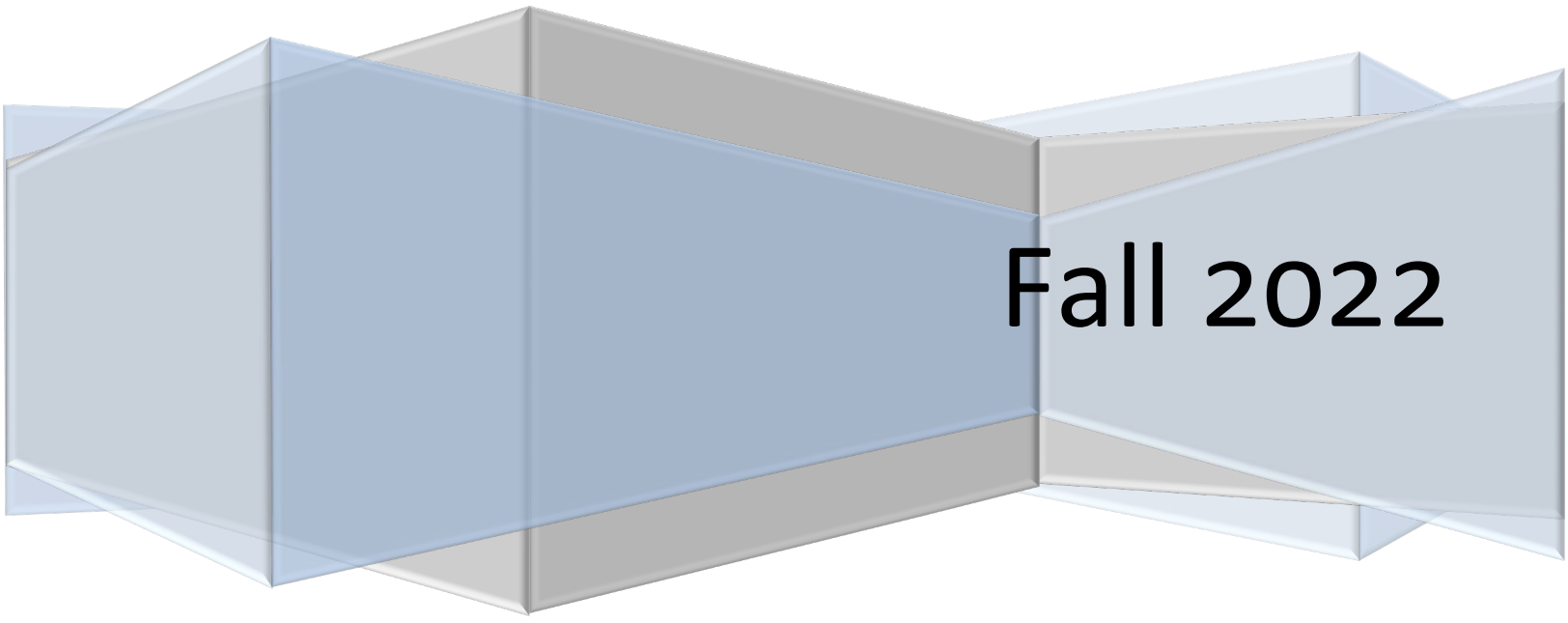
UTSA

CS 5523 Operating Systems

Bonus Programming Projects

Name: Shobnom Roksana **UTSA ID: nok057**

Date: 12/07/2022



Remote Method Invocation (RMI) facilitates object function calls between Java Virtual Machines (JVMs). JVMs can be located on separate computers - yet one JVM can invoke methods belonging to an object stored in another JVM. Methods can even pass objects that a foreign virtual machine has never encountered before, allowing dynamic loading of new classes as required.

Consider the follow scenario:

- Developer A writes a service that performs some useful function. He regularly updates this service, adding new features and improving existing ones.
- Developer B wishes to use the service provided by Developer A. However, it's inconvenient for A to supply B with an update every time.

Java RMI provides a very easy solution. Since RMI can dynamically load new classes, Developer B can let RMI handle updates automatically for him. Developer A places the new classes in a web directory, where RMI can fetch the new updates as they are required.

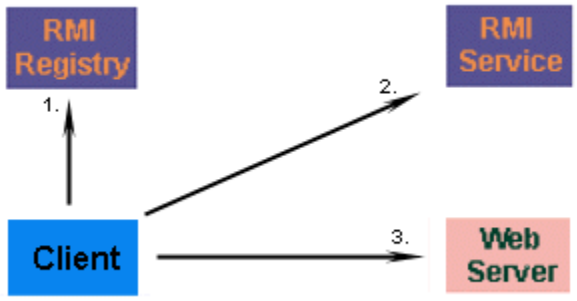


Figure 1 - Connections made when client uses RMI

Figure 1 shows the connections made by the client when using RMI. Firstly, the client must contact an RMI registry, and request the name of the service. Developer B won't know the exact location of the RMI service, but he knows enough to contact Developer A's registry. This will point him in the direction of the service he wants to call.

Developer A's service changes regularly, so Developer B doesn't have a copy of the class. Not to worry, because the client automatically fetches the new subclass from a webserver where the two developers share classes. The new class is loaded into memory, and the client is ready to use the new class. This happens transparently for Developer B - no extra code needs to be written to fetch the class.

Status

The status of my project is complete. We have implemented *MATH Server, Client. Math server* create a remote Math object to provide *magicAdd, magicSubtract, magicFindMin, magicFindMax* methods.

Remote Math object have dedicated counters to record the number of different math operations it has performed, and the corresponding methods is implemented to retrieve these numbers.

Client generates 1000 RMI requests, where each request randomly chooses one of the 4 operations as well as the corresponding required parameters. At the end, the client retrieves the number of operations performed by the server using corresponding methods.

Program supports running more than one client concurrently. Results provided with 2 clients running concurrently but more than 2 clients can run concurrently, and server can handle requests successfully when more than 2 clients running concurrently. To handle concurrency, 4 methods provided by remote Math object is synchronized.

The server and client programs run successfully on separate machines. On client implementation just need to provide IP address (Default localhost) of the machine where server is running for successful communication between server and clients.

Design

CalculatorInterface - A remote interface provides the description of all the methods of a particular remote object. The client communicates with this remote interface. It extends the predefined interface Remote

CalculatorRmi – Implements the *CalculatorInterface* class and provide implementation to all the abstract methods of the remote interface. All the methods are synchronized to handle multiple clients running concurrently.

CalculatorServer – Server program to extend *CalculatorRmi* class, create the remote math object and bind it to the **RMIregistry**.

Client – program to fetch the remote Math object and invoke the required method using this object. Multithreaded so that 2 or more client can run concurrently to invoke methods on the same remote object.

Figure of Results

Figure 1 shows that Server started running and is ready for client connections.

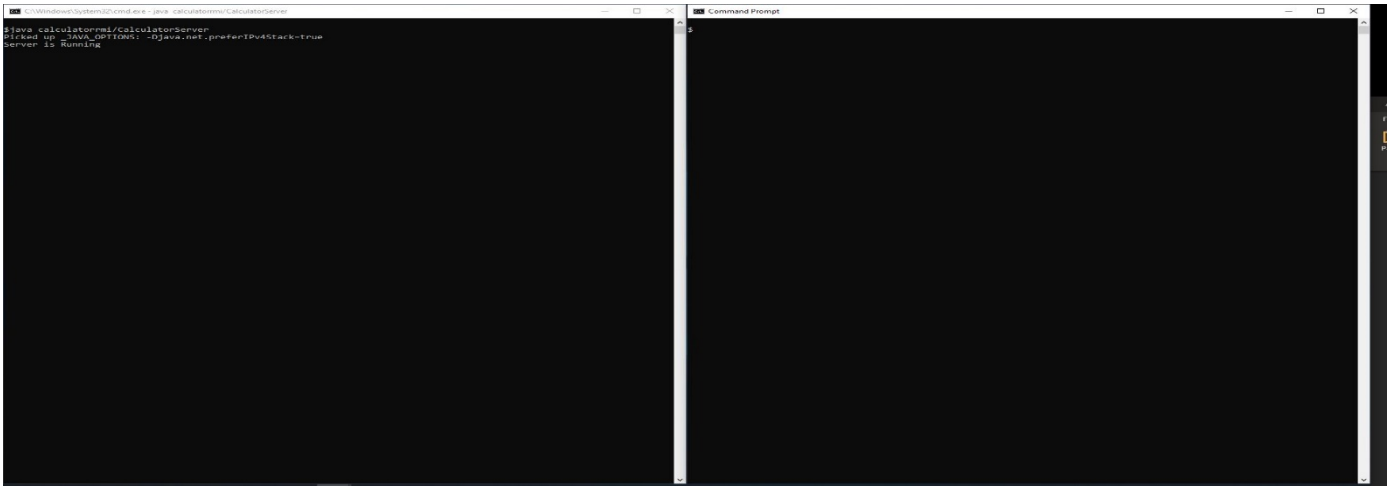


Figure 1

Figure 2 shows Client 1 and 2 stared running, connected to the server and invoking methods provide by the remote object. Server is performing corresponding method operations invoked by both Clients.

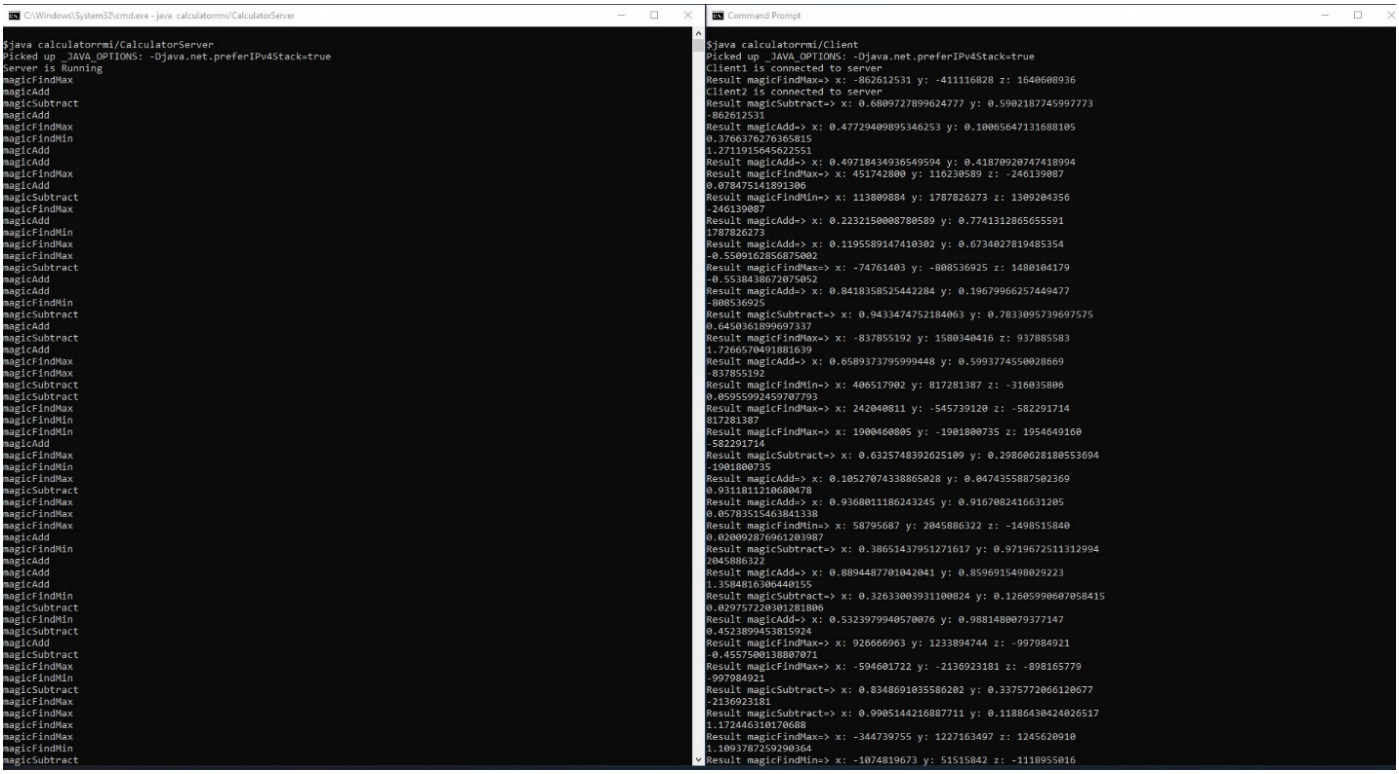


Figure 2

Figure 3 shows both client 1 and client 2 finished sending 1000 requests to the server to randomly invoke 4 math methods provided by remote math object and server performed those 4 different methods based on the requests received from both clients. Finally, both clients print out how many of the 4 different operations is performed by the server.

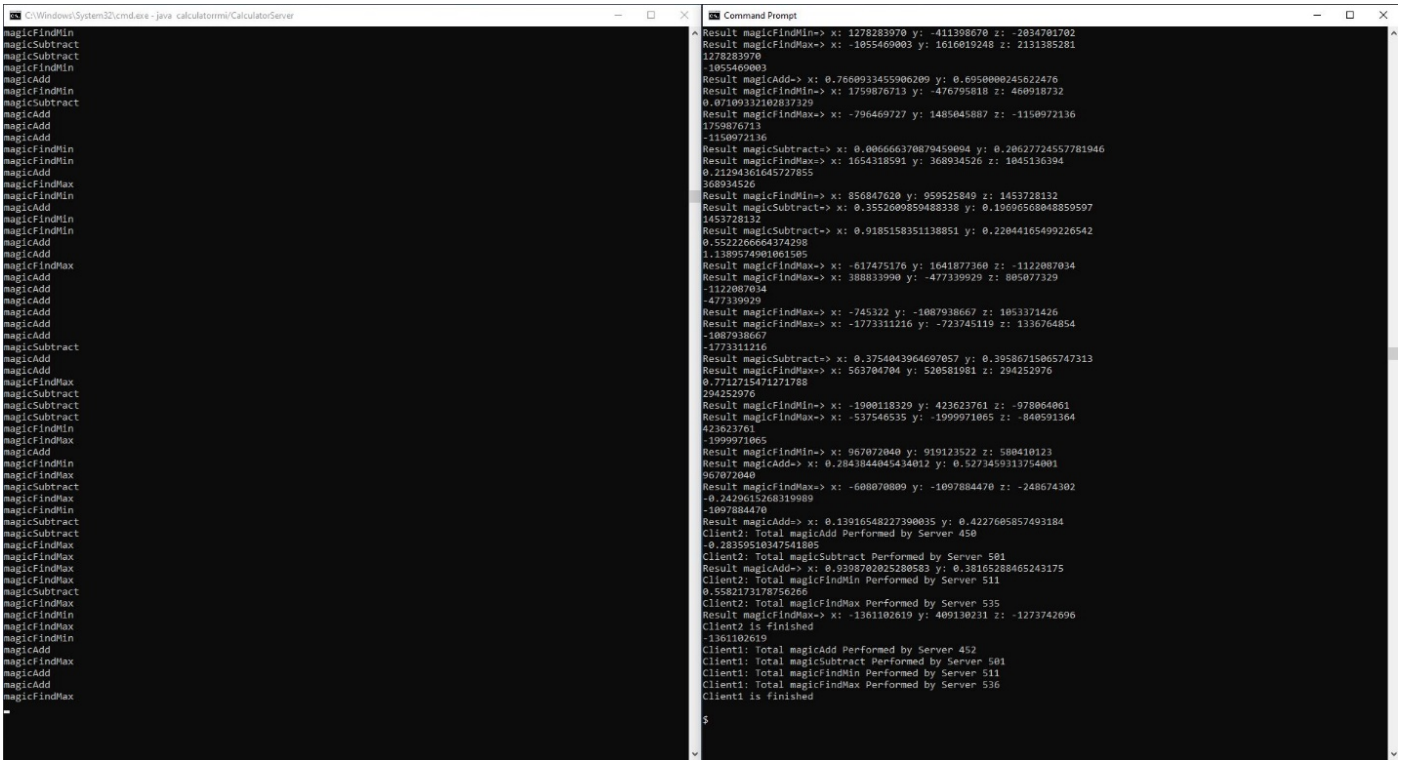


Figure 3

Comments

Java RMI is a useful mechanism for invoking methods of remote objects. Java RMI allows one Java Virtual Machine to invoke methods of another, and to share any Java object type, even if client or server has never come across that object type before. This project helps to understand the basics of RMI. We now understand how to create remote object, how to bind an object to naming sever, how to get remote object reference from remote reference module etc.