**PROJECT REPORT**

**Project Status:**

Successfully completed the project.

**Instructions** to run the project **without Makefile** :

Open a terminal and execute the below commands on server side.

→ javac Server.java RemoteInterface.java BookIds.java -d .

→ rmiregistry &

→ java myapp.Server

Once you see the ‘server ready’ statement on console, open another terminal for client and run the following commands.

→ javac Client.java RemoteInterface.java BookIds.java -d .

→ java myapp.Client

Upon executing above commands follow the instructions that appear on the console.

**With Makefile :**

Server side terminal

→ make clean

→ make server

Open another terminal for client

→ make client

For suppose if you face any error as mentioned below while running the project on server side,

-----------------------------------------------------------------------------

java.rmi.server.ExportException: Port already in use: 1099

-----------------------------------------------------------------------------

Use the below commands to make that port free.

→ lsof -i :<port number> For the above error the port is 1099 so use the command as below

lsof -i :1099

→ After running above command you can find a process id. Kill that port and re-run server.

kill -9 <PID\_number> For example: kill -9 33748

For Test File:

→ make test1

The above tests for one client.

→ make test2

The above tests for more than one client.

**How to maintain these books information in server? Whether it is scalable or not?**

* There is a seperate file named RemoteInterface.java where I have declared all the services provided by the server. And the Server class implements all the services declared in the remote interface.
* I have implemented a seperate file BookIds.java which sets properties of a book like title, topic, cost. And a function which sets up all the books and related information in the bookIds through a parameterized constructor.
* I have limited the stock quantity to 20 books for each type of book. Once if anyone tries to order after exceeding maximum number of orders then the application will show you a message that book is out of stock.

**How to handle a request from the client? If you are using thread-pool, how to do that?**

* If it is only one client and a server then there is no problem, we can implement directly without the concept of distributed systems.
* If there are multiple clients and a server which accepts multiple client requests then the concept of RMI in distributed systems can be used with the help of multi-threading.
* They are synchronized with the help of synchronized blocks as explained in detail in the below section.

**How to use synchronization in server?**

* Java provides a way of creating threads and synchronizing their task by using synchronized blocks.
* I had placed a synchronized keyword for each of the services provided by server.
* Synchronized blocks synchronized on the same object can only have one thread executing inside them at a time. All other threads attempting to enter the synchronized block are blocked until the thread inside the synchronized block exits the block.
* **synchronized** public String order(int itemNum)

**How to measure the performance of serving a request?**

* Divide total time in micro seconds taken to serve a request from client by total number of requests for a particular service gives the performance of a service.
* **totalSearchTime/ searchCount**  gives the performance measure in  **ms/request.**

**Discuss the challenges and difficulties that you have encountered when you design this project and how you overcome them?**

* Firstly, the concept RMI itself is a hard concept to understand, somehow I managed to understand by seeing few basic distributed system applications like addition and see how the process is going on.
* Then I started to understand the project and designed my own way on how to implement the project and started with writing Remote Interface where I have declared all the methods needed for the project.
* When I tried to run the project it is throwing an exception that port is in use. For few hours I wasted time on that issue and after googling I am able to found a way to get rid of that error as mentioned in the instructions to run the project.
* Another challenge is when the client is another machine, I faced difficulty in specifying the host in the server code.
* I tested various ways whether the application is running as expected manually. It succeeded all the services correctly. But, when it comes to checking with deadlock situation at the server end, i faced difficulty in triggering deadlock situation using multiple threads.

**How you designed some test cases to verify whether your program works correctly or not?**

* The basic test case which i have written verifies the basic functionality using a single Client. It checks whether it can call all the remote methods in the Server, and the maintenance of books records by the Server, the main goal is to check whether the stock quantities are updated and service reports are correct. The test results show correct functionality.
* The other test case verifies for multiple clients by creating 2 threads and making them both order the same book at a time. This process is repeated 20 times in a loop. Test2\_main.java makes uses class file Test2.java, which consists of all the services requested by the client, which implements Java's Runnable interface. This indicates that we can serve as many clients as number of threads created.
* After executing the test program in which there is a loop statement which creates threads for 20 times and make sure that no deadlock occurs in the distributed application.

**References**:

<https://stackoverflow.com/questions/8337215/remote-method-invocation-port-in-use/22341408>

<https://www.javatpoint.com/RMI>