**CS 5523 OPERATING SYSTEMS**

**PROJECT REPORT**

**MYBOOKS.COM- The World's smallest Bookstore**

Varun Kavalipurapu

@01519517 (rot 893)

**Overview**

For this project I am implementing an online book store that will use Java RMI to allow members to check out bookstore resources. The goal of this project is for you to better understand distributed object computing using Java RMI.

**Status:** Completed

**Setup and Run the Project**

For project setup, download Client and Server projects. Client folder contains three java files (Client.java, ClientMain.java, and RMI.java) and the server folder contains four files (Book.java, RMIImpl.java, RMI.java, and Server.java). Follow the following steps to run the project.

* Run Server.java from Server project folder. Right click on the file-> Run as-> Java Application
* When you see server success message, run ClientMain.java from Client Project folder. Right click on the file -> Run as -> Java Application
* Enter “localhost” to connect to the server or enter IP address of the computer you want to connect

**Implementation:**

To maintain the book information in the file , I have created a text file which holds all the details like book number, book name , price, no of copies available and which topic it is (Distributed Systems or Graduate school) . It is scalable. We can have a method to check the inventory. If the value under no of items is 0, it can tell us that we will have to update the inventory. We can set the inventory size.

We have the following three files in Client folder.

**Client.java:**  Client .java will request a connection to the server and establish a connection to the server. It will display a menu where the user can traverse through it and perform number of tasks.

**ClientMain.java:** ClientMain .java is the main class in Client. It creates a new client thread for every request it sends.

**RMI.java:** This file is the common interface in both server and the client. This contains all the methods that are defined in the server and are available to the client.

We have four files in Server folder .

**RMI. Java :** RMI.java is an interface where all the methods that are implemented in the server that are available to the client are listed .

**RMIImpl.java :** This file is the implementation of RMI.java interface . All the methods are implemented in this class. All these methods in this class will call another methods in Book.java. To achieve more security , I have written another file for the implementations .

**Server.java :**  This class will establish connection will every client which tries to connect to the server . This has the server connection configuration.

**Book.java:** Book.java will have all the method implementations. Every client request will call these methods .

**Performance :**

To check the performance of each operation, I have calculated the execution time for each method. This will allow us to analyze how much time the method is taking to perform a task. When the inventory size is increase, we can have a better chance of analyzing the program.

**Synchronization:**

Threads communicate primarily by sharing access to fields and the objects reference fields refer to. This can be achieved using Synchronization. To achieve synchronization, I have defined every method in the server synchronized and the class implements serializable.

**Thread Pool:**

Thread Pools are useful when you need to limit the number of threads running in your application at the same time. There is a performance overhead associated with starting a new thread, and each thread is also allocated some memory for its stack etc.

Using worker threads minimizes the overhead due to thread creation. Thread objects use a significant amount of memory, and in a large-scale application, allocating and de-allocating many thread objects creates a significant memory management overhead.

**Database:**

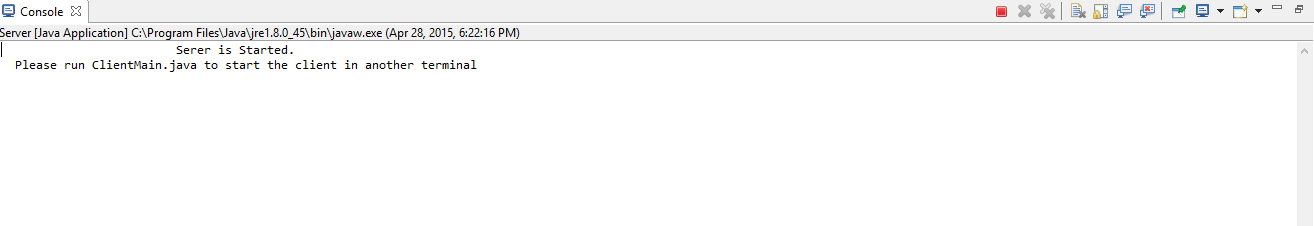
I used Mysql database to connect to the server as earlier we have used a text file. Once a client requests for the application, the data is fetched, modified and updated on to the database from the server with client’s request. I used Mysql because it is reliable and achieves high performance and security

Some of the test cases that I have designed some test cases to achieve the goals of the project. It is as shown below

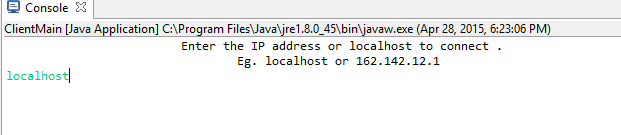
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Unit to test** | **Test data** | **Steps to be executed** | **Expected result** | **Actual result** | **Pass/Fail** |
| 001 | Connection Establish with server | Connect to server | -Run Server.java  -Run ClinetMain.java  -Enter localhost | Connection should be established with no errors. | Connection established | Pass |
| 0002 | Show menu | Should display menu | Connect to the server and it should display a menu | Should display a menu with tasks that can be performed | Menu is displayed | pass |
| 0003 | List Books | Perform get book from | Select option 1 . | Should display all the books in the store | Books list are displayed | pass |
| 0004 | Search for a book | Search the store with keyword | Select option 2 in the menu | Search should be successful | Search is done . Book name is displayed | pass |
| 0005 | Search for a book that is not available | Search for thebook that is not there in the store | Select option 2. Enter invalid keyword | Should display “Book not found” | “Book not found” is displayed | pass |
| 0006 | Search for abook by ID | Search the store with ID | Select option 3 in the menu | Search should be successful | Search is done based on ID . | pass |
| 0007 | Search for a book with ID that is not available | Search for thebook that is not there in the store | Select option 3. Enter invalid ID | Should display “Book not found” | “Book not found” is displayed | pass |
| 0008 | Purchase a book | Order a book | Select option 4. From the menu | Should allow to enter book ID and quantity. Should display item price and inventory | Takes the input and display the book , cost | pass |
| 0009 | Exit from the application |  | Select option 5 | Should end the thread | Exited | pass |
| 0010 | Display execution time | Display execution time for each task performed | Run either search or list option | Should display time taken to perform the task | Time is displayed | pass |
| 0011 | Invalid input in menu | Provide the invalid input | Select 10 from the menu | Should display “Invalid Input” | “Invalid Input” is displayed | Pass |
| 0012 | Update database with stock | Update database | Select 4 from the menu  Enter no of items  Enter itemno | Database should be updated with the stock | Database is updated | pass |

**Screenshots of the Application :**

**Server Started :**

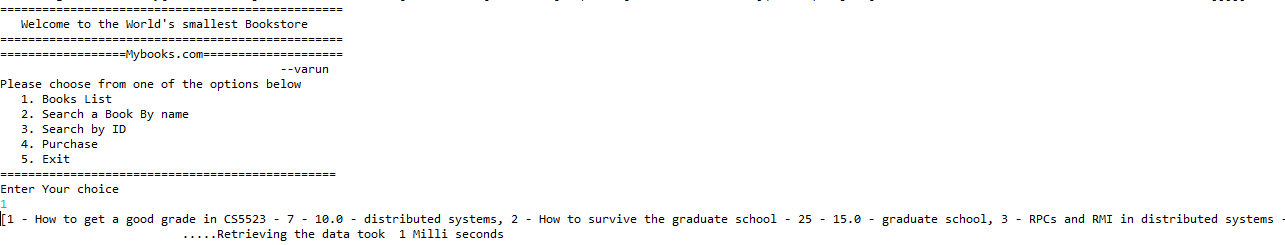


**Connect Client :**

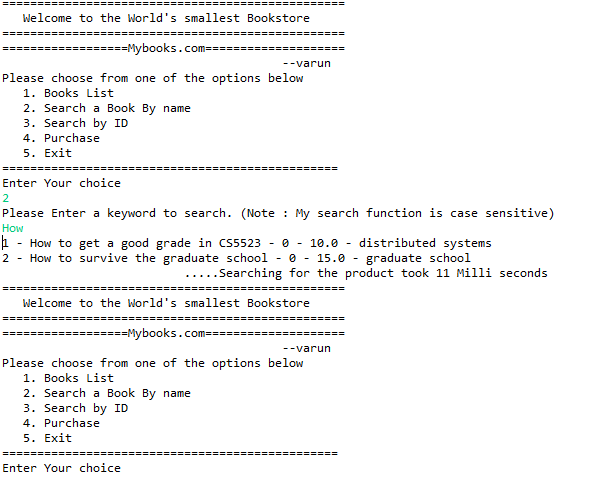


**Navigate through the menu :**

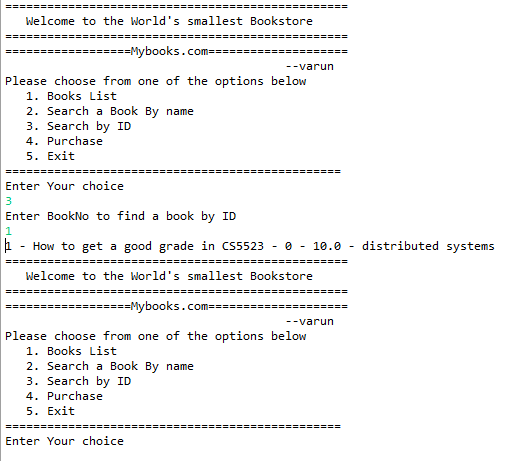
**Select option 1:**



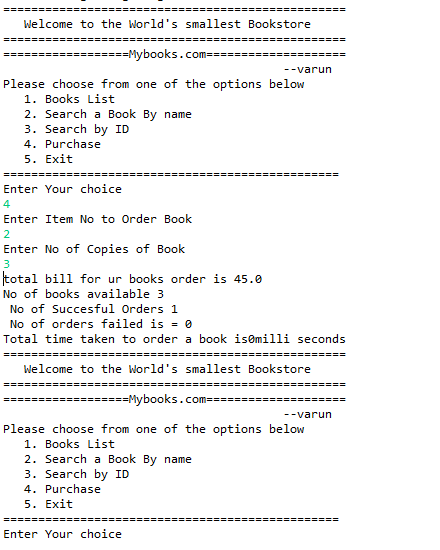
**Select option 2:**



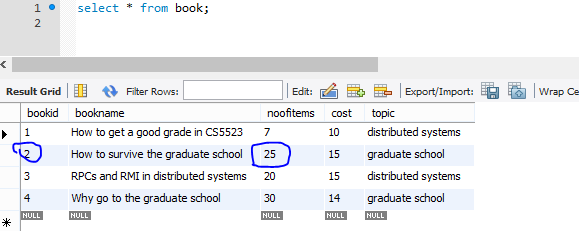
**Select option 3:**



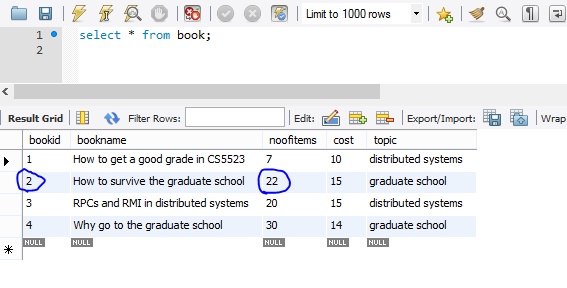
**Select Option 4:**



**Check the database before purchase :**

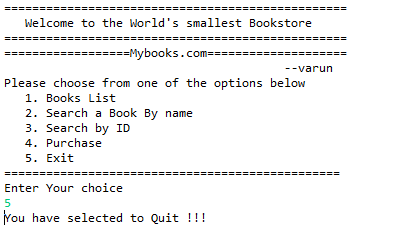


**Database after Purchase of books:**



When multiple clients access the database , on each purchase , the database is updated and if all the books are sold it shows “NOT IN STOCK” Message.

**Option 5:**



The user quits the application.

**SYNCHRONIZATION**

We can use a method as shown to maintain clock synchronizations .

void syncServerTime() {

    long t1 = System.currentTimeMillis();

    long serverTime = timeServer.getTime();

    long t2 = System.currentTimeMillis();

    if (t2 - t1 > 4000)

        log.warn("syncServerTime() could not establish time within allowable margin for error - keeping old offset");

    else

        serverTimeOffset = serverTime - (t2 - t1) / 2;

}

long getServerTime() {

    return System.currentTimeMillis() + serverTimeOffset;

}

**CONCLUSION**

Mybookstore is the smallest bookstore application designed as a part of Operating System final project. In this project we have learnt how to use RMI for building a client server application and connection to database. We can connect to multiple databases like access and sqlserver too. Building a secure applications using RMI and an external secured database can enhance the application’s security and efficiency.