



SOFE 4790U: Distributed Systems (Fall 2021)

Instructor: Dr. Q. Mahmoud/H. Singh

Individual Programming Assignment #2

[Monday, Nov. 1 – Wednesday, Nov. 10]

**Java RMI Application:
Book Store Register System**

Name: Esam Uddin

Student Number: 100711116

Date: 11/10/2021

GitHub Repo Link: <https://github.com/esam191/Book-Store-App-Java-RMI>

Assignment GitHub Repo:

<https://github.com/UOITEngineering2/assignment2fall2020-esam191>

Application Idea

I designed and developed a book store register system that works as a distributed object-based application using Java RMI. The application idea for this assignment is based on and extended from what I created in Assignment 1 of this course. Clients can order books by simply selecting the book store they want to order books from and start by typing in the name and quantity of the book they would like to purchase. Clients can also ask the server to recommend a top selling book if they can't choose on their own.

Once connected, the client can see menu options for the bookstore. The client has 6 options to choose from, the first option is to download a book store file, the second option lets the client order a book, the third option displays the current total cost, the fourth option recommends a best selling book from the server, the fifth option displays a receipt including the total cost plus tax (HST) and a date stamp, and finally the last option lets the client exit from the server by manually terminating the request (by typing -1).

This distributed application was created using Java RMI and the Java RMI remote interface provides 5 unique remote methods (services) to clients. The application accomplishes something useful and has several novel/interesting features. Each method has been created to sufficiently modify and/or extend to functionalities/services from the previous Assignment to best fulfill the purpose of creating a distributed application using Java RMI for this assignment.

Novel Features

The five main novel/interesting features that the server provides for the client to use have been implemented so that the JAVA RMI remote interface provides 5 unique remote methods to the clients.

Remote Method #1:

- **Download Book File:** When the client first connects to the server, they can pick the first option to download any of the 2 book store files available from which they can order their books.

Remote Method #2:

- **Calculate Cost:** The first main function of the server is to allow clients to take new book orders and compute the cost by searching through the text file to find the book and its cost.

Remote Method #3:

- Display Total Cost: Once the client gives a book order, they have a choice to add in another order so the server computes the total cost (including tax) and displays it to the user.

Remote Method #4:

- Recommends Best Seller Books: The server stores a list of top selling books from both book stores and upon request from the client, picks a random top selling book based on the book store selected and recommends it to the client.

Remote Method #5:

- Displays Receipt + HST tax + Date Stamp: Clients can view their total spending of the day by receiving the total cost that includes the HST tax amount and the date stamp.

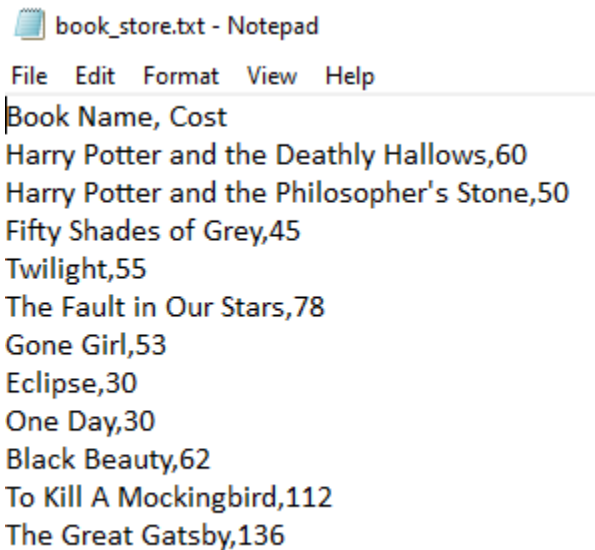
Challenges and Solutions

Throughout working on this assignment, I've faced many challenges but the one that troubled me the most was ensuring that the client was able to access methods remotely through the BookStoreInterface. I had to make sure the Interface object created in the Client side was called in the same method it was initialized, and this was getting tricky after I added a couple RMI methods in the interface. So, in order to make it easier for myself when trying to implement this, I decided not to add any more RMI methods and other high level functionality into my program until a base java rmi application was established and working. Once I was able to get that running, I went on and added all other necessary RMI methods for this application.

Some other challenges that I faced were calculating the total cost for the client because I had to make sure that the cost gets updated each time the client takes a new order and recommending a book to the client because I had to display the book to the client and take quantity input. I faced the same problem in the previous assignment, so I knew exactly how to deal with this particular problem. But since I added the selecting book store functionality for this assignment, I ran into more trouble when passing the cost values across different stores. I had to pass in the specific bookstore name in both the calcTotal, and recommendBook methods to fix the issue. In the end, I was able to work through these problems by carefully analyzing the state and action of the bugs that I faced.

Book Store #1:

The image to the right shows the first bookstore based on which the client makes the book orders. Each line in the text file contains the name of the book and its cost separated by a comma.



book_store.txt - Notepad

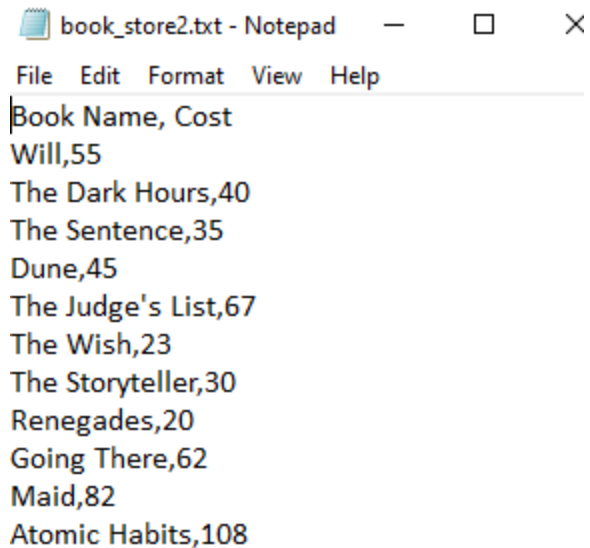
File Edit Format View Help

Book Name, Cost

Harry Potter and the Deathly Hallows,	60
Harry Potter and the Philosopher's Stone,	50
Fifty Shades of Grey,	45
Twilight,	55
The Fault in Our Stars,	78
Gone Girl,	53
Eclipse,	30
One Day,	30
Black Beauty,	62
To Kill A Mockingbird,	112
The Great Gatsby,	136

Book Store #2:

The image to the right shows the second Bookstore that was added based on which the client makes all the new book orders. Similarly, each line in the text file here contains the name of the book and its cost separated by a comma.



book_store2.txt - Notepad

File Edit Format View Help

Book Name, Cost

Will,	55
The Dark Hours,	40
The Sentence,	35
Dune,	45
The Judge's List,	67
The Wish,	23
The Storyteller,	30
Renegades,	20
Going There,	62
Maid,	82
Atomic Habits,	108

The image below shows an example test of client 1 connected to the server, and you can see that the client can see all menu options as soon as it gets connected. Each time the client selects an option and once their request is completed, the menu options reappear so the client can choose what to do next. The client first downloads a bookstore file from available ones by typing 1, and then orders the book by typing 2, then they get to enter the bookstore, book name and quantity. Now if the client wishes to see the total cost, client can type 3 as shown below and view the total cost, and 4 to get a book recommendation, 5 to see the receipt for the day, and finally, -1 to exit the application. Each time the client selects an option, it is accessing the appropriate methods remotely through the Interface object. ex. obj.recommendBook()

Client 1:

```
D:\Projects\Personal\DS Assignment 2\Client>java BookStoreClient localhost
Client Ready - remote stub active...
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
1
Client Selected Option 1
Pick a Book File to download: book_store.txt , book_store2.txt
book_store.txt
Successfully Received Book File From Server!
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
2
Client Selected Option 2
Enter the name of the store:
book_store.txt
Enter the name of the book:
Gone Girl
Enter the quantity:
2
New Order Added Successfully!
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
3
Client Selected Option 3
Server Sent: Your total cost is $106.0 plus tax is: 114.48
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
4
Client Selected Option 4
Enter the name of the store:
book_store.txt
Server Recommended Book: To Kill A Mockingbird
Enter the quantity:
1
New Order Added Successfully!
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
3
Client Selected Option 3
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
5
Client Selected Option 5
Server Sent: Receipt: Your total order for today including HST (ON - 8%): 235.44 - Today's Date: 2021-11-10:18:50-23
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
-1
Client Selected Option -1
D:\Projects\Personal\DS Assignment 2\Client>
```

The following screenshot is a sample test run with a second client while the server is still running. The client picks the same options as shown in the image above as well as accessing the same methods, but this time using the second book store to show that both bookstores are working properly.

Client 2:

```
D:\Projects\Personal\DS Assignment 2\Client>java BookStoreClient localhost
Client Ready - remote stub active...
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
1
Client Selected Option 1
Pick a Book File to download: book_store.txt , book_store2.txt
book_store2.txt
Successfully Received Book File From Server!
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
2
Client Selected Option 2
Enter the name of the store:
book_store2.txt
Enter the name of the book:
Will
Enter the quantity:
2
New Order Added Successfully!
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
3
Client Selected Option 3
Server Sent: Your total cost is $110.0 plus tax is: 118.8

Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
4
Client Selected Option 4
Enter the name of the store:
book_store2.txt
Server Recommended Book: The Wish
Enter the quantity:
3
New Order Added Successfully!
```

```
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
3
Client Selected Option 3
Server Sent: Your total cost is $179.0 plus tax is: 193.32

Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
5
Client Selected Option 5
Server Sent: Receipt: Your total order for today including HST (ON - 8%): 193.32 - Today's Date: 2021-11-10:18:58-59
Please select one of the options: 1 - Get Book File, 2 - Give New Book Order, 3 - View Total Cost, 4 - Best Seller Recommendation, 5 - View Receipt, -1 - exit
-1
Client Selected Option -1
D:\Projects\Personal\DS Assignment 2\Client>
```

The images below show the rmiregistry and the server side. First I started the rmiregistry and then ran the server. Each time a new client gets connected, the server prints a message on the terminal according to the mmi methods accessed by the client.

Starting rmiregistry:

```
D:\Projects\Personal\DS Assignment 2\Server>start rmiregistry
```



Server:

```
D:\Projects\Personal\DS Assignment 2\Server>java BookStoreServer
Server listening...
Calculating Total Cost...
Displaying Cost...
Displaying Receipt...
Sending Book File...
Calculating Total Cost...
Displaying Cost...
Recommending Top Seller...
Calculating Total Cost...
Displaying Cost...
Sending Book File...
Calculating Total Cost...
Displaying Cost...
Recommending Top Seller...
Calculating Total Cost...
Displaying Cost...
Displaying Receipt...
```