COMP 6231 Distributed System Design

Assignment 1 Report

Distributed Movie Ticket Booking System using Java RMI

Submitted to: Professor Rajagolapan Jayakumar

By

Namrata Brahmbhatt - 40233323

Overview

- I used **Java RMI (Remote Method Invocation)** to implement the communication between Manager/Clients and servers (ATW, VER, and OUT).
- I used **UDP** to implement the communication between servers to allow admin and client operations across all servers.
- I used **HashMap** to store the records key pair values of movieID and MovieName. We also used this data structure to store BookingSchedule and MovieShowsAvailability.
- I used a multithreading technique so that multiple clients can act simultaneously.

Implementation

Project implementation consists of carrying out all the activities that we need to do. In addition, this helps us to aim towards the perfect execution of the code.

The project implementation starts by defining interfaces which includes the definition and parameters of all methods we need. To be able to **add movie slots**, **book movie tickets**, and **get a booking schedule** are some of the methods we implemented.

After this we implemented three server.java (ATWServer, VERServer, OUTServer) files. Each server is running on a different port. The assignment also includes a **client** that asks for the ID.

Finally, based on this ID, the Admin or Customer functions are invoked which allows them to perform all required functions. As per project implementation document, we also needed to implement a separate log file for each server which will keep the track record of all the operations. **Log.java** contains the code specific to the project and the **Logs** folder contains all the server's log files (ATW.txt, VER.txt, OUT.txt).

To start please follow the following steps:

1. Firstly, execute the file **ServerInstance.java**. This file contains all server instances that allow the initialization of the ports.
2. Secondly, run the file **Client.java**. This file then binds the connection to all those ports which were initialized in the first step.
3. Then the program asks for a **user ID**. It then asks for an input to ask the user to perform ID specific functions. For instance, customers can book movie tickets and perform some more functions which are only available for customers.
4. All servers are up and running at the moment and as per the user ID, that specific client server connection is invoked. This allows successful completion of **Admin** and **Customer** specific operations.
5. As we perform these functions, we have implemented and imported **Log.java** in **Client.java**. This import allows all three servers to create and maintain logs of all the operations that Admin or Customer might perform at a given time.

Architecture

Interfaces

If we have a class implementing several methods, we use Interface to define these methods. An interface only specifies the method signatures. This allows multiple classes to implement the same interface. This allows all classes to have their own unique implementation for its methods.

I have implemented the following methods in **InterfaceOperations.java** file:

- addMovieSlots
- removeMovieSlots
- listmMovieShowsAvailability
- bookMovieTickets
- getBookingSchedule
- cancelMovieTickets

Client

This file contains client specific code. Here choices are provided to the user (admin, customer) for which operation is to be performed. Validation of inputs is provided in these files.

This file is the starting point of the application. This file asks for the UserID whose prefix (first 3 characters) are used to run the server. If the UserID is ATWA1234, then ATW Server will be up and running and so on for the other two servers. Here, validation of UserID takes place, if it is not in the format ATW/VER/OUT followed by A (for Admin) and C (for Customer) and four digits, then it won't work.

Servers (ATWServer / VERServer / OUTServer)

These files are responsible for getting the server up and running.

Server name, number and port are assigned here. The specific <server>Class's object is created here and binding of registry takes place.

I have assigned specific port numbers to all the servers. All these are unique and have their server instance in **ServerInstance.java** file. The port number for ATWServer is 8001, for VERServer is 8002, and for OUTServer is 8003.

Log

This file is used to create log files and add information of all activities taking place in the application.

Server logs are named as ATW.txt, VER.txt, OUT.txt and are saved in the root folder (src). The **Log.java** code is implemented in such a way that it does not create a new log every time a user logs into the particular server. It checks if the server log file is available and makes new entries to that existing file itself.

Note- Please make sure to change the log file path before starting the application in **MovieBookingSystem.java**.

Data structures

We use Hashmap as data structure to store key-value pairs for the movie details and bookings.It consists of an id (string type) as the key and values are Hashmaps of movieslots.

Before the start of application

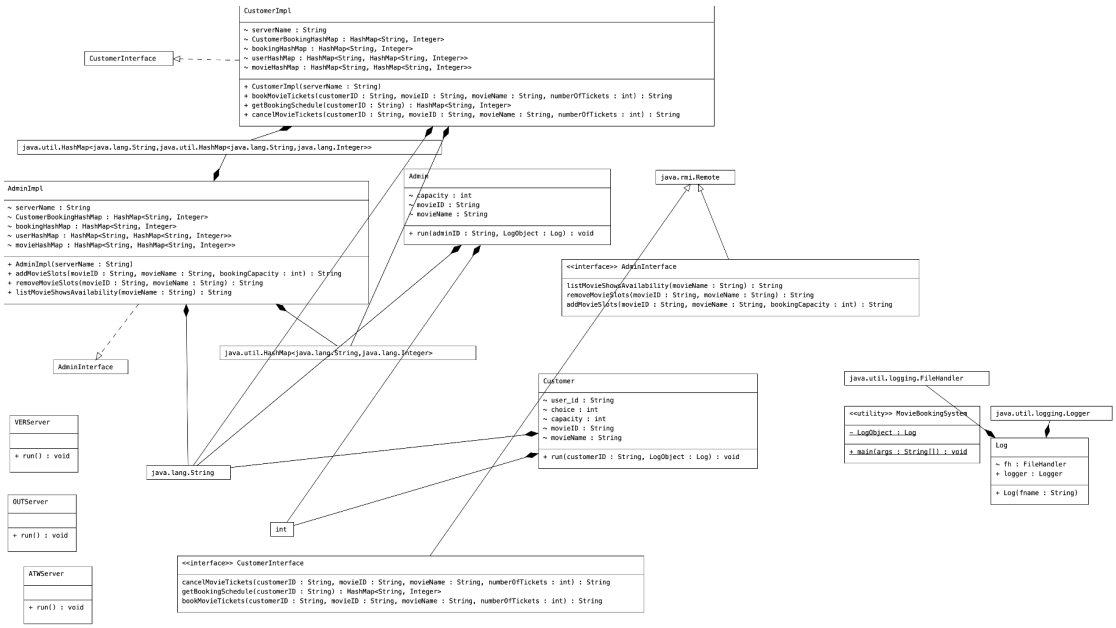
The whole application code was implemented in VS Code. But as java is a platform independent application, you can use any editor.

Now, before we start the application, open file **MovieBookingSystem.java**. This is an important step because you must change the Log folder address here. Take a look at the following image. You need to change the path in the below section.

```
LogObject = new Log(
        "/Users/namratabrahmbhatt/Desktop/DistributedMovieTicketBookingSystem/src/"
                + prefix + ".txt");

LogObject.logger.info(inputID + " has logged in the " + prefix + " Server");
```

Class Diagram



Test Cases

The project includes various implementations:

S.No.	Function Name	Function Arguments	Output	Comments
1	addMovieSlots	movieID = ATWM160523 movieName = AVATAR bookingCapacity = 2	Two tickets were booked successfully.	Can be performed by Admin
2	removeMovieSlots	movieID = ATWM160523 movieName = AVATAR	All slots were removed.	Can be performed by Admin
3	listMovieShowsAvailability	movieName = AVATAR	Movie shows were listed.	Can be performed by Admin
4	bookMovieTickets	customerID = ATWC1234 movieID = ATWM160523 movieName = AVATAR numberOfTickets = 10	Tickets were booked for the movie for the customer.	Can be performed by both Admin and Client
5	getBookingSchedule	customerID = ATWC1234	The booked movie tickets were listed for the particular customer	Can be performed by both Admin and Client
6	cancelMovieTickets	customerID = ATWC1234 movieID = ATWM160523 movieName = AVATAR numberOfTickets =	Tickets were cancelled for the particular movie for the customer	Can be performed by both Admin and Client

		10		

This next section shows all required outputs of the above test cases.

1. **Add movie Slots**

```
****** Select which Admin operation you want to perform ******
1. Add Movie Slots.
2. Remove Movie Slots.
3. List Movie Shows Availability.
4. Book your own movie tickets.
5. List your booked movie tickets.
6. Cancel your movie tickets.
7. Exit.
1
Choose a movie name you wanna add slots to::
AVENGERS         AVATAR          TITANIC
Enter the movie name::
avatar
Enter the movieID:
ATWA08022023
Enter the number of slots you want to add for the Movie: AVATAR with the MovieId: ATWA08022023
100
{AVENGERS={ATWM07022023=60}, AVATAR={OUTA07022023=60, ATWA08022023=100}, TITANIC={VERA07022023=60}}
Movie slots added for the movie.
****** Select which Admin operation you want to perform ******
1. Add Movie Slots.
2. Remove Movie Slots.
3. List Movie Shows Availability.
4. Book your own movie tickets.
5. List your booked movie tickets.
6. Cancel your movie tickets.
7. Exit.
```

Ln 69, Col 40 Tab Size: 4 UTF-8 LF {} Java

2. **Remove movie Slots**

```
2
Enter movie name for which you want to remove movie slots.
AVATAR   AVENGERS        TITANIC
avatar

Enter movieId for the movie - AVATAR
ATWA08022023

{OUTA07022023=60} after removal..!!
Movie slots for the movie AVATAR has been removed
****** Select which Admin operation you want to perform ******
1. Add Movie Slots.
2. Remove Movie Slots.
3. List Movie Shows Availability.
4. Book your own movie tickets.
5. List your booked movie tickets.
6. Cancel your movie tickets.
7. Exit.
```

3. **Display movie Shows Availability**

```
****** Select which Admin operation you want to perform ******
1. Add Movie Slots.
2. Remove Movie Slots.
3. List Movie Shows Availability.
4. Book your own movie tickets.
5. List your booked movie tickets.
6. Cancel your movie tickets.
7. Exit.
3
Enter movie name you want to see the availabilities for:
AVATAR   AVENGERS        TITANIC
titanic
{VERA07022023=60}

Here is the schedule for the movie: TITANIC
{VERA07022023=60}
****** Select which Admin operation you want to perform ******
1. Add Movie Slots.
2. Remove Movie Slots.
3. List Movie Shows Availability.
4. Book your own movie tickets.
5. List your booked movie tickets.
6. Cancel your movie tickets.
```

4. **Book movie Tickets**

```
outc1234
Outremont server started.
****** Select the operations you want to perform from the options given below:: ******
1. Book movie tickets.
2. List your booked movie tickets.
3. Cancel your movie tickets.
4. Exit.
1
Enter movie name you want to add from the option.
AVATAR   AVENGERS        TITANIC
avatar
{OUTA07022023=60}
Below are the details for the movie:: AVATAR
{OUTA07022023=60}

Enter the movieId you want to book the tickets for::
outa07022023

Enter the number of tickets you want to book for the movie: AVATAR with movieID OUTA07022023
10
10 tickets booked for the movie AVATAR with movieID OUTA07022023
****** Select the operations you want to perform from the options given below:: ******
1. Book movie tickets.
2. List your booked movie tickets.
3. Cancel your movie tickets.
4. Exit.
```

zsh
Run: Movie...

5. Booking Schedule

```
****** Select the operations you want to perform from the options given below:: ******
1. Book movie tickets.
2. List your booked movie tickets.
3. Cancel your movie tickets.
4. Exit.
2

Enter CustomerID:
outc1234
Here is your booking schedule..!!
{AVATAR-OUTA07022023=10}
****** Select the operations you want to perform from the options given below:: ******
1. Book movie tickets.
2. List your booked movie tickets.
3. Cancel your movie tickets.
4. Exit.
```

6. Cancel movie Tickets

```
2. List your booked movie tickets.
3. Cancel your movie tickets.
4. Exit.
3

Enter CustomerID:
outc1234
Enter movie name you want to cancel tickets for:.
AVATAR   AVENGERS        TITANIC
avatar

Here is the booked shows with the CustomerID - OUTC1234
Inner Key: AVATAR-OUTA07022023
OUTA07022023-10

Enter the movieID you want to cancel the tickets for.
outa07022023

Please enter number of tickets for the movie AVATAR with the movieID OUTA07022023
5
5 Movie tickets for AVATAR has been removed
****** Select the operations you want to perform from the options given below:: ******
1. Book movie tickets.
2. List your booked movie tickets.
3. Cancel your movie tickets.
4. Exit.
```

zsh
Run: Movie...