

COMP 6231- Distributed System Design

Instructor: R. Jayakumar

Distributed Movie Ticket Booking System Using Java RMI -
Assignment 1

Winter 2023

Submitted by: Antas Jain

ID: 40233532

CONTENTS

Sr. No.	Title	Page no.
1	Overview	3
2	Class Diagram	5
4	Data Structure	7
5	Test Cases	8

1. Overview

Distributed Movie Ticket Booking Systems have 2 types of users:

- Admin - Actions:
 - Create Movie Slots
 - Update Movie Slots
 - Delete Movie Slots
 - Do Actions on behalf of customer
 - Get a list of movie slots for a particular movie.
- Customer - Actions:
 - Book a movie Slot
 - Cancel a movie slot (Completely/Partially)
 - Get their booking details.

There are 3 servers for movie theatres:

- ATW- Atwater
- VER- Verdun
- OUT- Outremont

There are 3 timing slots for a movie:

- A- Afternoon
- M- Morning
- E- Evening

A Client ID Consist of SERVER ID+A(for Admin)/C(for Customer)+4 unique digits.

A movie ID Consists of SERVER ID+Timing Slot(A/M/E)+Date (in DDMMYY format).

The task was to create an RMI-enabled Distributed System for movie booking.

- Implementations:

MovieModel.java: POJO for Movie Servers (getters, setters and other methods).

ClieModel.java: POJO for Client Servers (getters, setters and other methods)..

Server.java: Creating Server instances.

Client.java: Interacting with User, displaying all options for a client, sending details that forward a UDP request to the server for actions.

ServerInst.java: Interface for Server Actions.

MovieManager.java: Middleware between Server and Client, sends UDP requests.

Logger.java: Logs actions to a text file.

Data is stored in nested map structures as shown in (3).

- UDP Server Ports Used:

ATWATER: 7878
VERDUN: 8989
OUTREMONT: 9090

- RMI Server Ports Used:

ATWATER: 2964
VERDUN: 2965
OUTREMONT: 2966

- Logs: Logs are saved for every client and every server.

Path for SERVER Logs: /src/Logs/Server/ServerName.txt
Path for CLIENT Logs: /src/Logs/Client/ClientId.txt

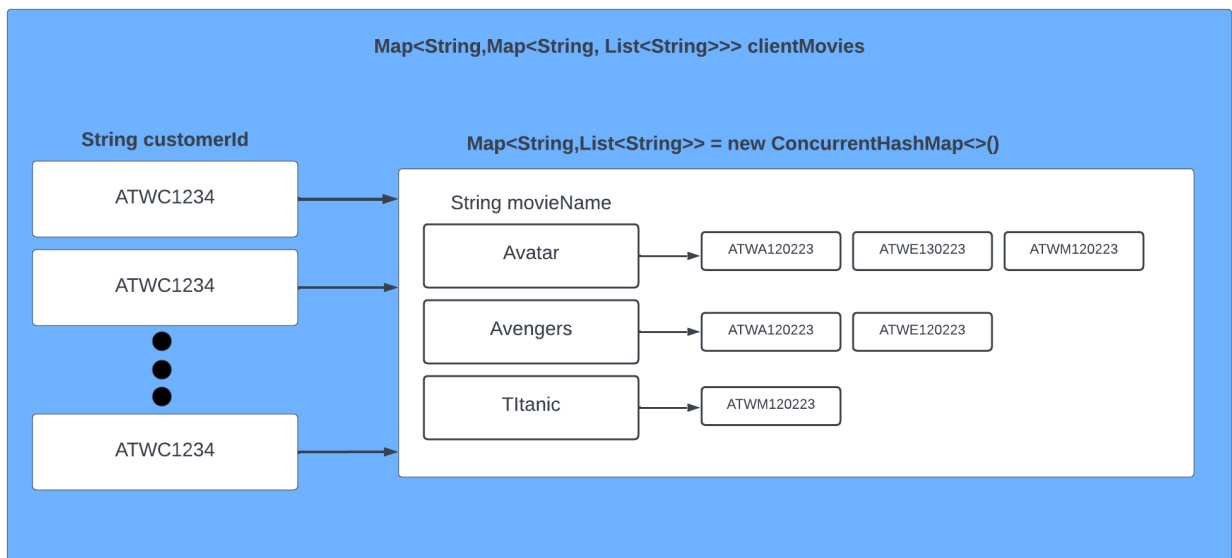
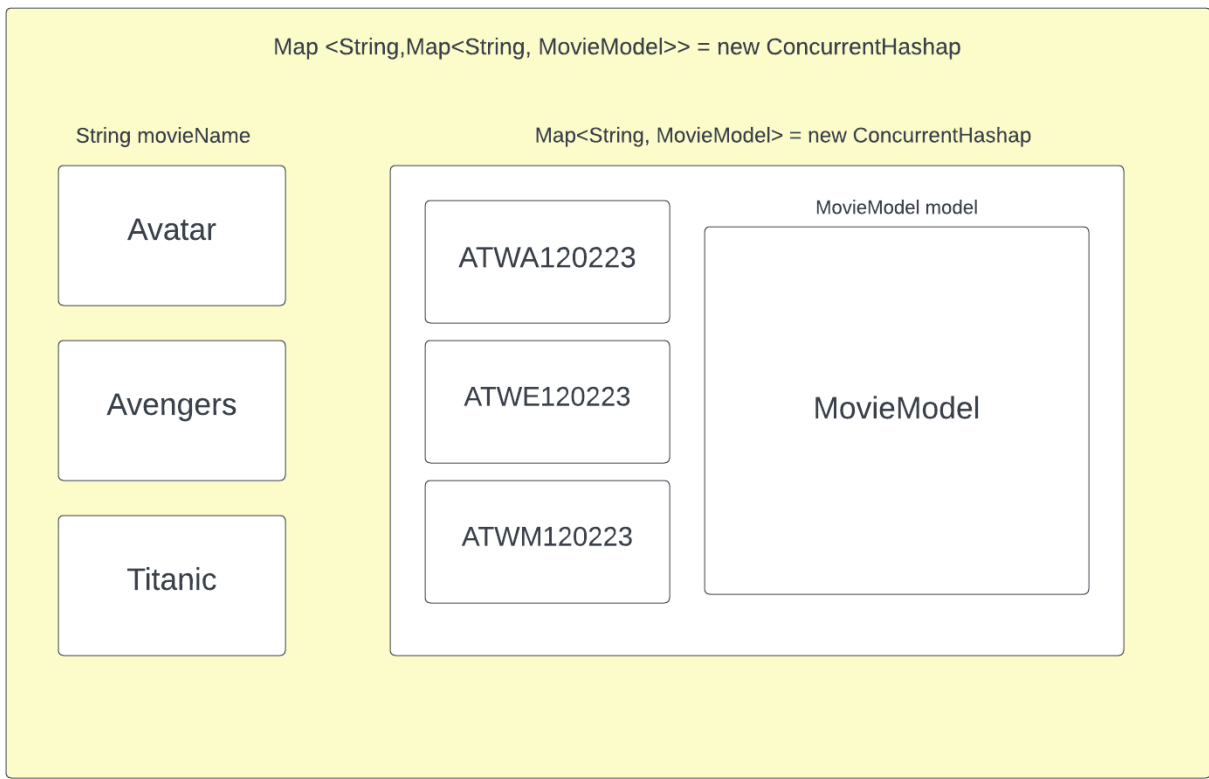
- Concurrent hashmaps were used to ensure maximum efficiency.
- Hardest Part of the assignment was to implement Book Movies and Cancel movies while keeping the track of the number of movie tickets the user need to book, for that I used a Concurrent hashmap with a unique id that consists of customerId+movieid+movieName. This was also the most important part according to my understanding.

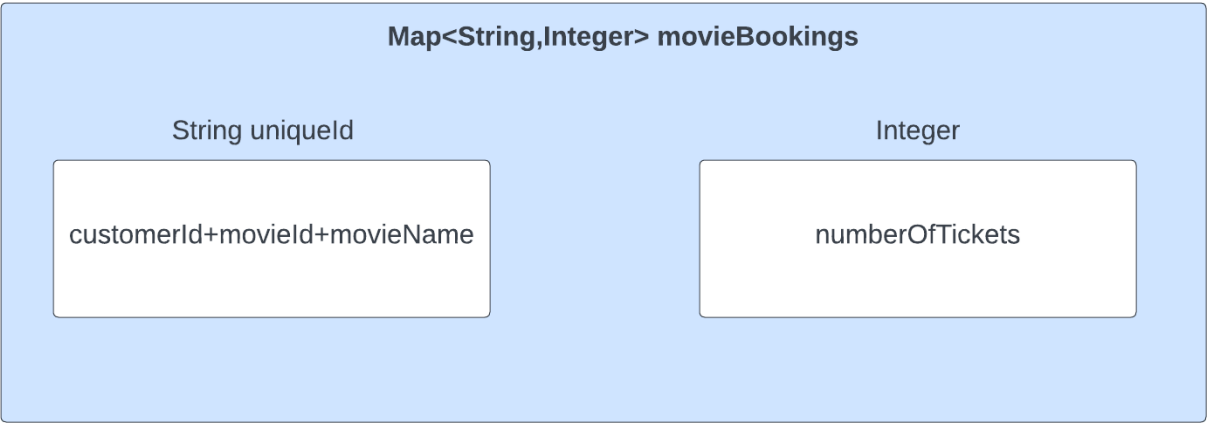
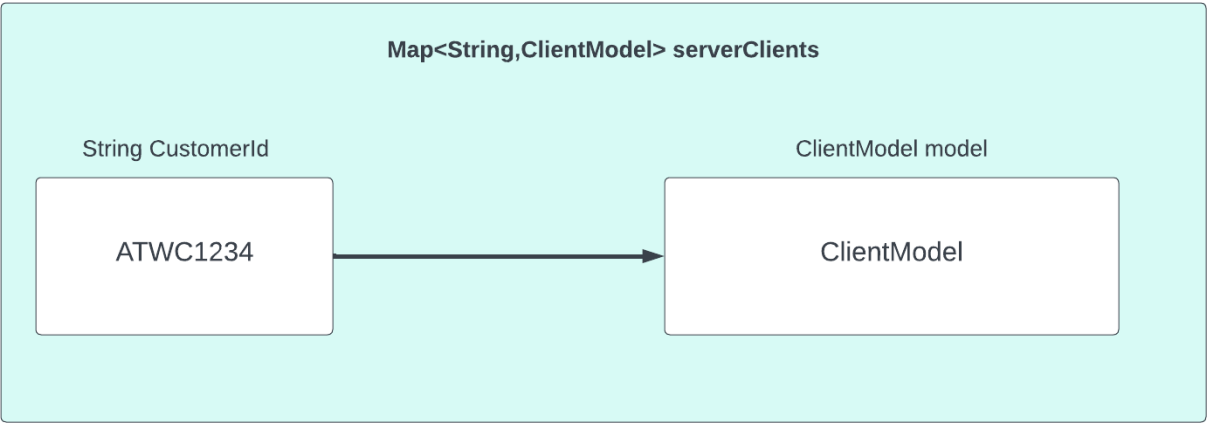
```

classDiagram
    class Serializable
    class Remote
    class RemoteObject {
        RemoteObject(RemoteRef)
        RemoteObject()
        RemoteRef ref
        hashCode() int
        readObject(ObjectInputStream) void
        toStub(Remote) Remote
        toString() String
        equals(Object) boolean
        writeObject(ObjectOutputStream) void
    }
    class RemoteServer {
        RemoteServer()
        RemoteServer(RemoteRef)
        clientHost String
        log PrintStream
    }
    class UnicastRemoteObject {
        UnicastRemoteObject(int, RMIClientSocketFactory, RMIServerSocketFactory)
        UnicastRemoteObject()
        UnicastRemoteObject(int)
        clone() Object
        exportObject(Remote, int, ObjectInputFilter) Remote
        readObject(ObjectInputStream) void
        exportObject(Remote, int, RMIClientSocketFactory, RMIServerSocketFactory) Remote
        exportObject(Remote, int, RMIClientSocketFactory, RMIServerSocketFactory) Remote
        unexportObject(Remote, boolean) boolean
        exportObject(Remote, UnicastServerRef) Remote
        exportObject(Remote) RemoteStub
        exportObject(Remote, int) Remote
        reexport() void
    }
    class MovieManager {
        MovieManager(String, String)
        addMovieSlots(String, String, int) String
        listMovieAvailabilityUDP(String) String
        getServerPort(String) int
        listMovieShowsAvailability(String) String
        removeMovieSlots(String, String) String
        addNewCustomerToClients(String) void
        bookMoviesTickets(String, String, String, int) String
        cancelMovieTickets(String, String, String, int) String
        getBookingSchedule(String) String
        getNextSameEvent(Set<String>, String, String) String
        removeMovieUDP(String, String, String) String
        addCustomersToNextMovieSlot(String, String, List<String>) void
        exceedWeeklyLimit(String, String) boolean
        sendUDPMessage(int, String, String, String, String, Integer) String
    }
    class MovieModel {
        MovieModel(String, String, int)
        movieServer String
        movieDate String
        movieName String
        movieCapacity int
        movieTime String
        movieId String
        findMovieTiming() String
        addRegisteredClientId(String) int
        toString() String
        findMovieDate() String
        removeRegisteredClientId(String) boolean
        findMovieServer(String) String?
        movieName String
        remainingCapacity int
        movieId String
        houseful boolean
        registeredClients List<String>
        movieDate String
        movieTime String
        movieServer String
        movieCapacity int
    }
    class Clients {
        Clients()
        admin(String, int) void
        main(String[]) void
        promptForMovieID() String
        getServerPort(String) int
        init() void
        promptForCapacity() void
        customer(String, int) void
        checkUserType(String) int
        promptForNumberOfTickets() void
        printMenu(int) void
        promptForMovieType() String
        askForCustomerIdFromAdmin(String) String
    }
    class Logger {
        Logger()
        clientLog(String, String, String, String) void
        serverLog(String, String) void
        serverLog(String, String, String, String, String) void
        deleteLogFile(String) void
        getFileName(String, int) String
        clientLog(String, String) void
        formattedDate String
    }
    class ClientsModel {
        ClientsModel(String)
        clientId String
        findClientServer() String
        toString() String
        findClientType() String
        clientId String
    }
    class ServerInst {
        ServerInst(String)
        listenForRequest(MovieManager, int, String, String) void
    }
    class Server {
        Server()
        main(String[]) void
    }
    class StringAssets {
        StringAssets()
    }

    Serializable <|-- Remote
    Remote <|-- RemoteObject
    RemoteObject <|-- RemoteServer
    RemoteServer <|-- UnicastRemoteObject
    UnicastRemoteObject <|-- MovieManager
    MovieManager <|-- MovieModel
    MovieManager <|-- Clients
    MovieManager <|-- Logger
    MovieManager <|-- ClientsModel
    MovieManager <|-- ServerInst
    MovieManager <|-- Server
    MovieManager <|-- StringAssets
  
```

3. Data Structure





4. Test Cases

Scenario No.	Requirement name	Test Scenario	Test Case
1.	Login	username	<ul style="list-style-type: none"> • Validate username for Admin and Customer • Validate Server Access
2.	Admin Actions, User Action	logout	<ul style="list-style-type: none"> • Logout current client and prompt for user id.
3.	Admin Actions	Add movie	<ul style="list-style-type: none"> • Validate movie id. • validate movie capacity. • Either adding a new movie slot or updating the capacity of existing movie slots • Movie adding only allowed for a week's range from the current date
4.	Admin Actions	Remove movie	<ul style="list-style-type: none"> • Validate movie id • Validate movie slot capacity
5.	Admin Actions	List movie shows available	<ul style="list-style-type: none"> • show the movie slots created in the current server.
6.	Admin Action, User Action	menu selection	<ul style="list-style-type: none"> • select menu options
7.	User Actions	Book movie	<ul style="list-style-type: none"> • validate movie id • book movie tickets for a movie id from the same server with a valid capacity • Constrain to 3 movie tickets from other servers. • validate booking multiple movies
8.	User Actions	Cancel movie	<ul style="list-style-type: none"> • Cancel all or a partial number of movie tickets from the server • Booking movie on next available slot
9.	User Actions	Get Booking Schedule	<ul style="list-style-type: none"> • Validate all movies booked by the customer
10.	Admin Actions	Do user actions on behalf of customer	<ul style="list-style-type: none"> • Booking movie • Cancelling movie • Get movie schedule