
# Distributed Event Management System using Java RMI

## Assignment 1 Report

Submitted By:

Gursimran Singh -40080981
Natheepan Ganeshamoorthy- 29335838

## Table of Contents

# INTRODUCTION

This assignment is focused on building a distributed event management system which uses Java's RMI(Remote Method Invocation). This distributed system can be used by event managers to preform task such as remove/add events and also by customers to book/cancel or list events. The events and related data is stored on the servers and the clients request the servers to get appropriate data using Java's RMI API.

# TECHNIQUES USED

## (a) RMI(Remote Method Invocation):

RMI is used to implement the connection between the servers and the clients. The Java Remote Method Invocation (RMI) system allows an object running in one Java virtual machine to invoke methods on an object running in another Java virtual machine. RMI provides for remote communication between programs written in the Java programming language [1].

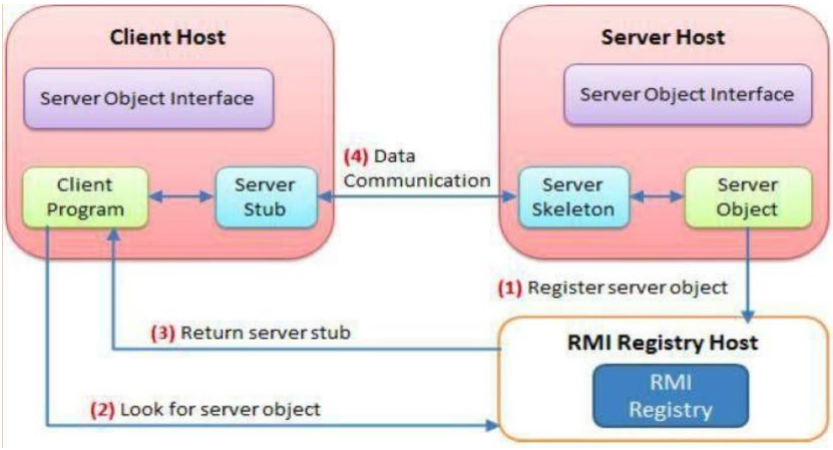



*Figure 1 Java RMI [2]*

It allows to invoke an object on other running JVM. RMI uses stub and skeleton object for communication with the remote object. The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. The skeleton is an object, acts as a gateway for the server-side object. All the incoming requests are routed through it.

The registry is the naming server that relates object with unique names.

## (b) UDP(Unigram Data Protocol):

UDP (User Datagram Protocol) is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss-tolerating connections between applications on the internet [3]. The service provided by UDP is an unreliable service that provides no guarantees for delivery and no protection from duplication UDP provides a minimal, unreliable, best-effort, message-passing transport to applications and upper-layer protocols.

We used UDP in java using Datagram Socket API and it was used for all communication requests between the servers.

### (c) Multithreading:

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process [4]. Multithreading is the ability of a central processing unit (CPU) to execute multiple processes or threads concurrently, appropriately supported by the operating system.

The 3 servers in our assignment receive UDP requests from other servers using threads which helped to aid concurrency in the operations.

## ARCHITECTURE

### (a) DATA STRUCTURE

HashMap: Java HashMap is a hash table based implementation of Java's Map interface. It is a part of Java's collection since Java 1.2. It provides the basic implementation of Map interface of Java. It stores the data in (Key, Value) pairs. It maps keys to values.

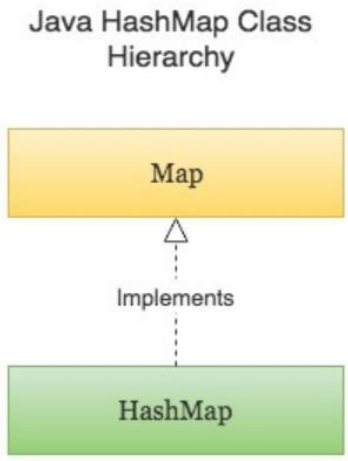



*Figure 2 HashMaps In Java*

In assignment 1, We have used nested HashMap to store the key value pairs like:

```
private static HashMap<String, HashMap< String, String>> databaseOttawa = new HashMap<>();
private static HashMap<String, HashMap<String, HashMap< String, Integer>>> customerEventsMapping = new HashMap<>();
```

*Figure 3 HashMap Declaration in Java*

The hash maps are imported from Java's Collection Framework to store the relevant data in them.

## (b) Project Structure

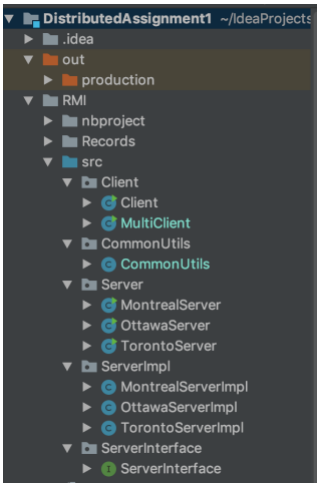



*Figure 4 Project Structure*

- **Client**: The client contains the classes for the client class which authenticate the users and display the respective lists of operations that they may perform by invocating functions on the servers. It also validates the user inputs.
- **CommonUtils:** This class is used for storing all the contains in the project such as server port numbers, server names etc.
- **Server:** This package contains the classes for all of the servers. The server classes create the registry and when client looks for the server methods, they can bind to the appropriate server. Servers communicate with each other using UDP.
- **ServerImpl:** This has the implementation of RMI interface. It consists of three classes; and the data on the servers is maintained using their individual HashMaps. They contain operations which allow managers to add, remove and list events in the system. Also, it allows clients to book/cancel/list events.

Below are the UML diagrams for all the packages in the project

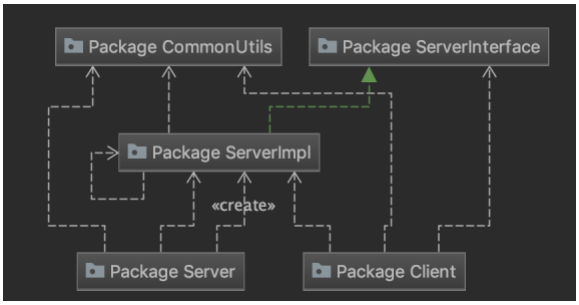
- 


*Figure 5 UML Project*

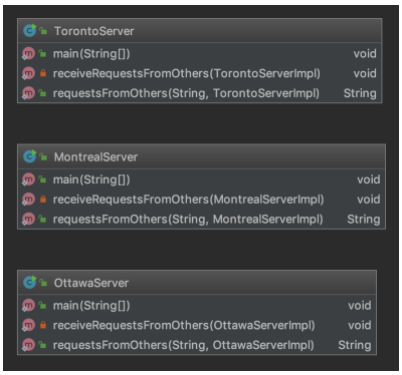



*Figure 6 Server UML*

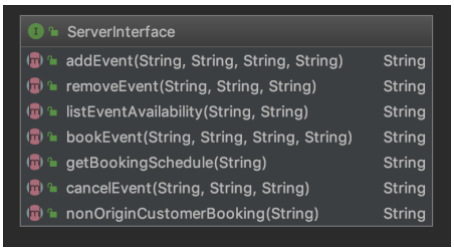



*Figure 7 Server Interface Class UML*

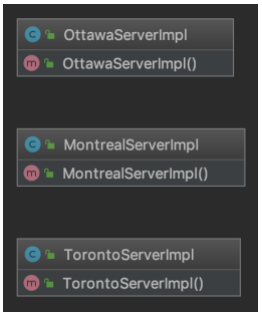



*Figure 8 Server Implementation Classes UML*

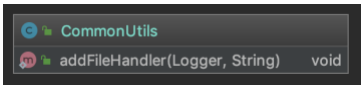



*Figure 9 CommonUtils Class UML*

# WORKING

## (a) Customer

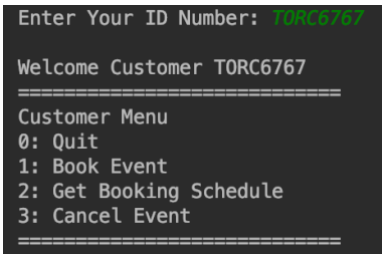



*Figure 10 Enter Valid UserID*

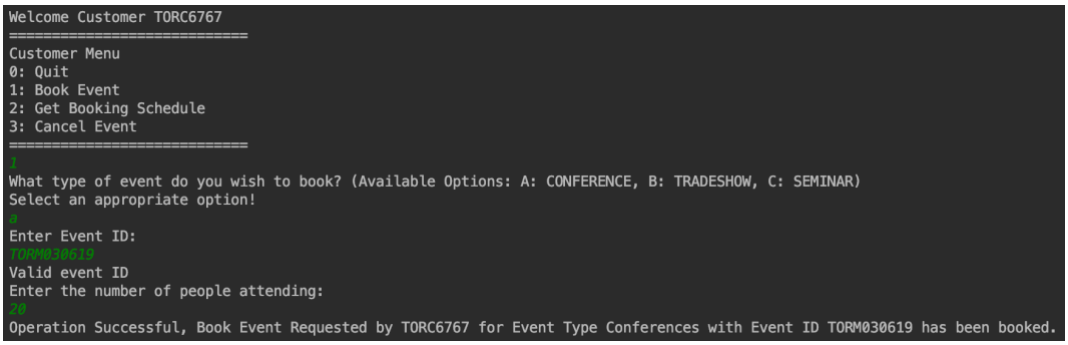



*Figure 11 Booking an Event*

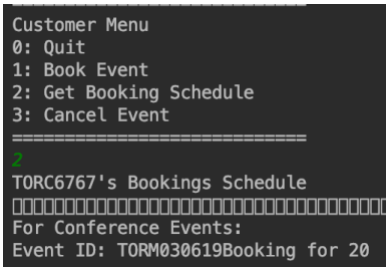



*Figure 12 Getting Booking Schedule*

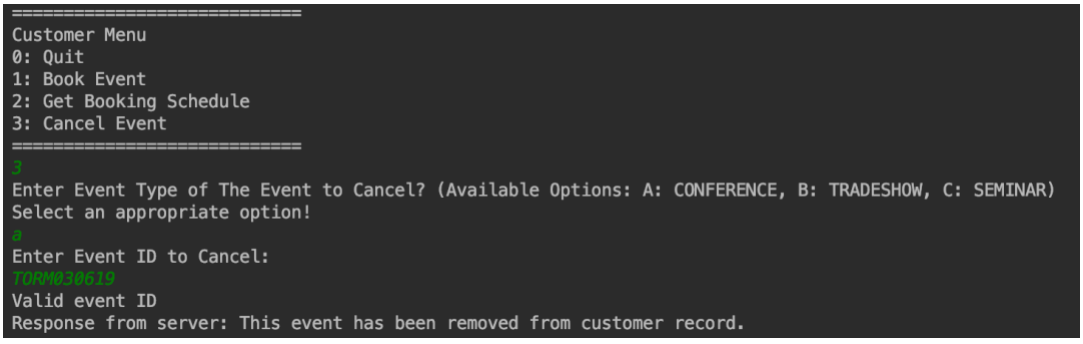



*Figure 13 Canceling an event*

## (b) Manager:

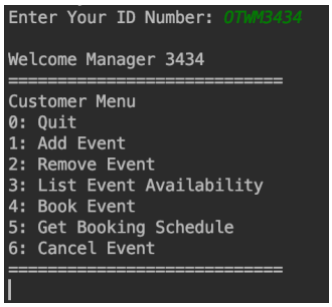



*Figure 14 Enter Valid Mnager ID*

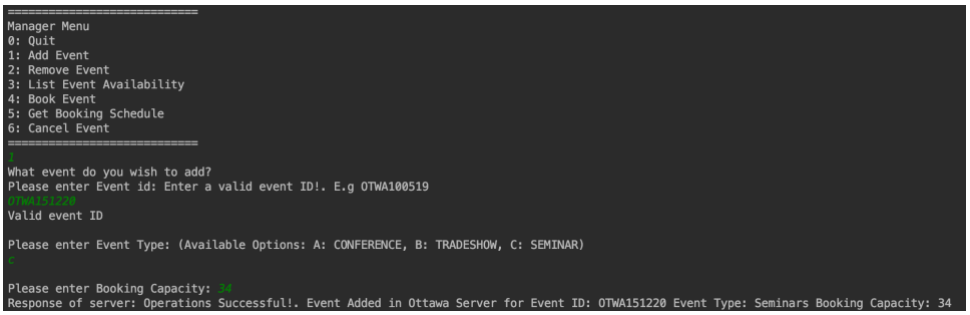



*Figure 15 Add an Event*


*Figure 16 Remove an Event*

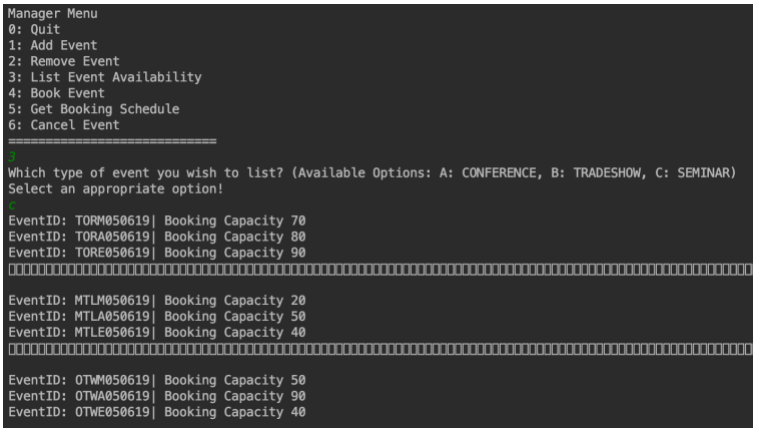



*Figure 17 List Event from All the servers*

# TEST SCENARIOS

| S. No | Test Cases | Set Up | Expected Result | Actual Result |
|---|---|---|---|---|
| **1** | Add an Event | -Enter manager id<br>-Manager id is valid, list of operations to be perform display<br>-Select 1 to add event<br>-Enter EventID, event Type and Booking Capacity | -Logs are generated<br>-Event added | -Same as expected result |
| **2** | Remove an Event<br>a) Event Remove<br>b) Event Remove if already booked by a customer for the same eventType. | -Enter manager id<br>-Manager id is valid, list of operations to be perform display<br>-Select 2 to remove item<br>-Enter EventID<br>a) Remove event. | -Logs are generated<br>-Event removed | -Same as expected result |
| **3** | List Event Availabilty | -Enter manager id<br>-Manager id is valid, list of operations to be perform display<br>-Select 3 to list events<br>-Select A/B/C to select event type | -Logs are generated<br>-List of all available events in an event Type | -Same as expected result |
| **4** | Book an Event | -Enter user id<br>-User id is valid, list of operations to be perform display<br>-Select 4 to book an event<br>-Enter customerID<br>- Select A/B/C to select event type<br>-Enter Event ID<br>-Enter number of guests. | -Logs are generated<br>-Event Booked, if event exits | -Same as expected result |
| **5** | Getting Booking Schedule | -Enter user id<br>-User id is valid, list of operations to be perform display<br>-Select 5 to get booking schedule of a customer<br>-Enter customer id | -Logs are generated<br>-List of bookings done by the customer according to their event types. | -Same as expected result |
| **6** | Cancel Event | -Enter user id<br>-User id is valid, list of operations to be perform display<br>-Select 6 to cancel an event<br>- Select A/B/C to select event type<br>-Enter Event ID | -Logs are generated<br>-The event is canceled if done by the same customer who booked it.<br>-Booking Capacity increased in the main hashmap. | -Same as expected result |

*Figure 18 Test Cases*

# MOST IMPORTANT/DIFFICULT PART

- Connection between client and server (through RMI)
- Connection between server and server (through UDP)
- Implementing Synchronization: For example, if two customers trying to book the same event at the same time then only one client can access it but other users can access other events from the system.

# CONCLUSION

In assignment 1, We implemented a distributed event management system using Java RMI. Firstly, HashMaps are used to store data for each server. The architectural design of the project was created by following object-oriented principles and analysis of the requirements. The communication between the servers was achieved by UDP/IP protocol. Multithreading made application support concurrenc

# Bibliography


[1] "Getting Started Using Java™ RMI," 2019. [Online]. Available: https://docs.oracle.com/javase/7/docs/technotes/guides/rmi/hello/hello-world.html.

[2] "Course WebSlides," 2019. [Online]. Available: https://moodle.concordia.ca/moodle/pluginfile.php/3595968/mod_resource/content/1/Tutorial%202%28UDPTCP%29.pdf.

[3] "UDP (User Datagram Protocol)," 2018. [Online]. Available: https://searchnetworking.techtarget.com/definition/UDP-User-Datagram-Protocol.

[4] "Multithreading in JAVA," 2019. [Online]. Available: https://www.geeksforgeeks.org/multithreading-in-java/.