SUMMER 2019

ASSIGNMENT 2

DISTRIBUTED EVENT MANAGEMENT SYSTEM (DEMS)

CORBA USING JAVA IDL

DESIGN DOCUMENTATION

Submitted by:

Tushar Verma 40089254

Siddhant Arora 40085538

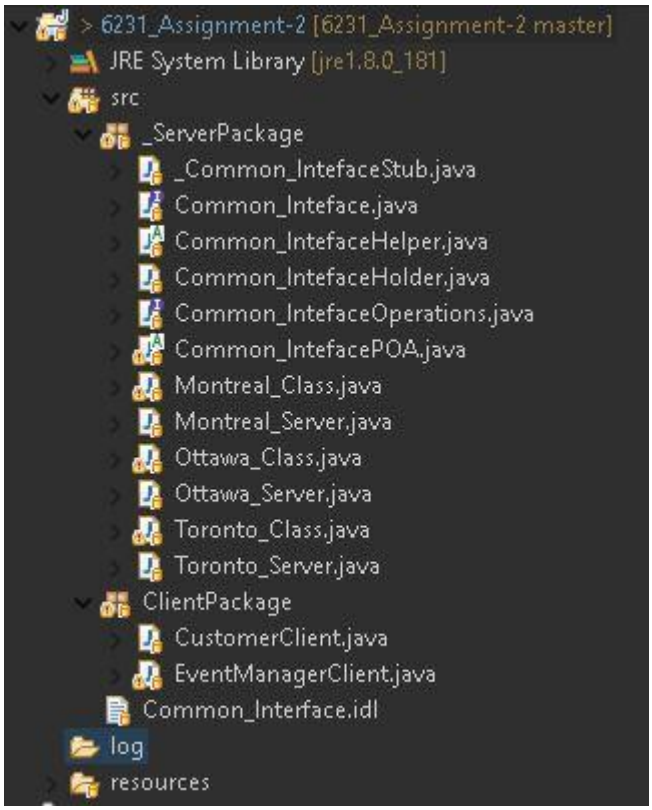
Table of contents

Introduction

The aim for this assignment was to design and build a distributed event management system, for a leading corporate event management company: a system which distributed through different servers in which event manager who manages the information about the events and customer who is also book and cancel an event across the company's different branches.

Architecture

The architecture used is clean and easily divided into clients and servers, all the client files are placed in a single package client and three different server packages are made each having implementation, interface and server classes.



Packages

1. Client- There is of two types either CustomerClient or an EventManaegrClient.
   CustomerClient- The operations that can be performed by a customer are the following:
   - Booking of a course.
   - Display their booking schedule.
   - Cancel a booked event.

EventManagerClient- A manager has responsibilities of a Customer as well as some additional responsibilities like adding an event or removing one. The operations that can be performed by an event manager are the following: -:

• Add a new event for their city.
• Remove an event.
• Check the event available for the given event type.

2. Server - Each server package(Montreal_Server/Ottawa_Server/Tororno_Server) contains the same classes defined below-:

• server class- Each class creates a remote object registry that accepts calls on a specific port. A thread is also started which handle the socket connection on a particular port using UDP as protocol. It accepts all the UDP messages from different servers and routes them to the implementation class for the correct response.
• Interface class- Contains all the methods performed by the client. It is used by the client to make calls to the servers
• Implementation class- Implements the interface methods. All the business logic is written here.

Technologies Used

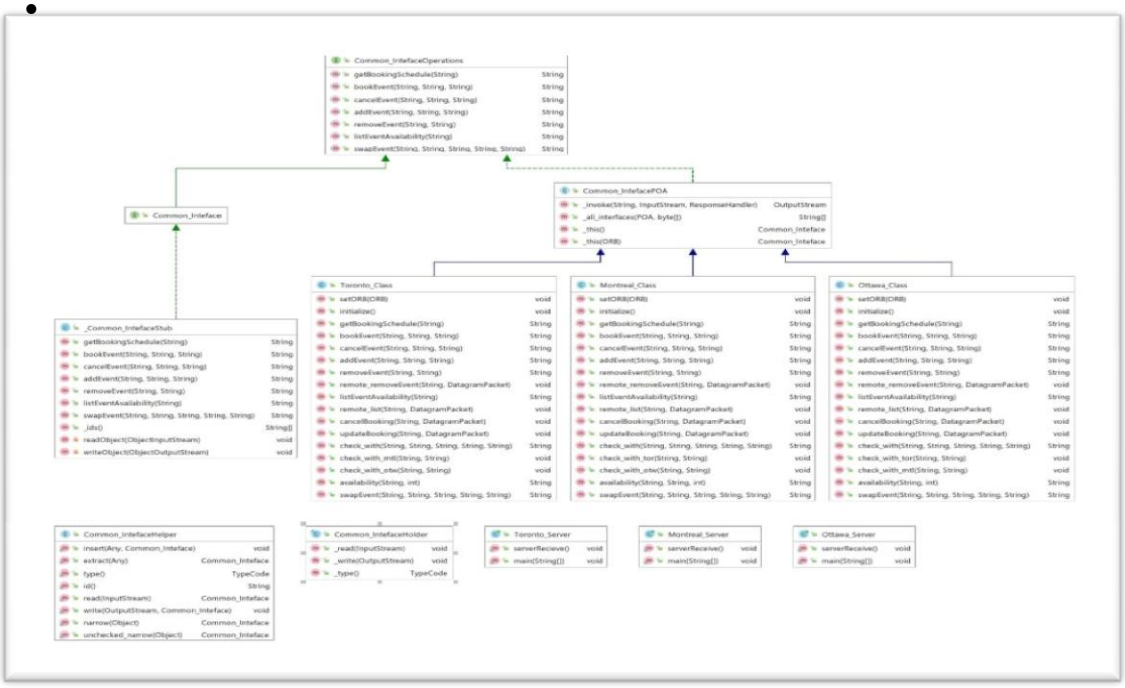
1. UDP

The UDP protocol provides a mode of network communication whereby applications send packets of data, called datagrams, to one another. A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed.

The DatagramPacket and DatagramSocket classes in the java.net package implement system-independent datagram communication using UDP. [1]

UDP has been used in the project to perform Inter Server communication i.e.

•



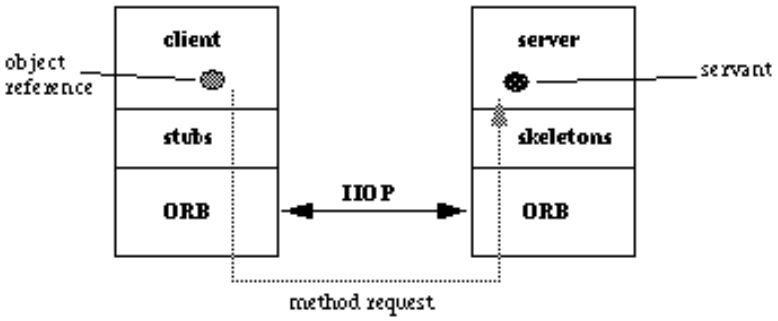
of booking for a course by a client of different cities.
• By the manager to get the course availability from different cities.

2. Multithreading

   Multithreaded execution is an essential feature of the Java platform. Every application has at least one thread — or several if you count "system" threads that do things like memory management and signal handling. But from the application programmer's point of view, you start with just one thread, called the *main thread*. [2]

3. Java IDL

   Java IDL is a technology for distributed objects – that is, objects interacting on different platforms across a network. Java IDL enables objects to interact regardless of whether they're written in the Java programming language or another language such as C, C++, COBOL, or others.
   This is possible because Java IDL is based on the Common Object Request Brokerage Architecture (CORBA), an industry-standard distributed object model. A key feature of CORBA is IDL, a language-neutral Interface Definition Language. Each language that supports CORBA has its own IDL mapping--and as its name implies, Java IDL supports the mapping for Java. To learn more about the IDL-to-Java language mapping, see IDL-to-Java Language Mapping.
   To support interaction between objects in separate programs, Java IDL provides an Object Request Broker, or ORB. The ORB is a class library that enables low-level communication between Java IDL applications and other CORBA-compliant applications. [3]

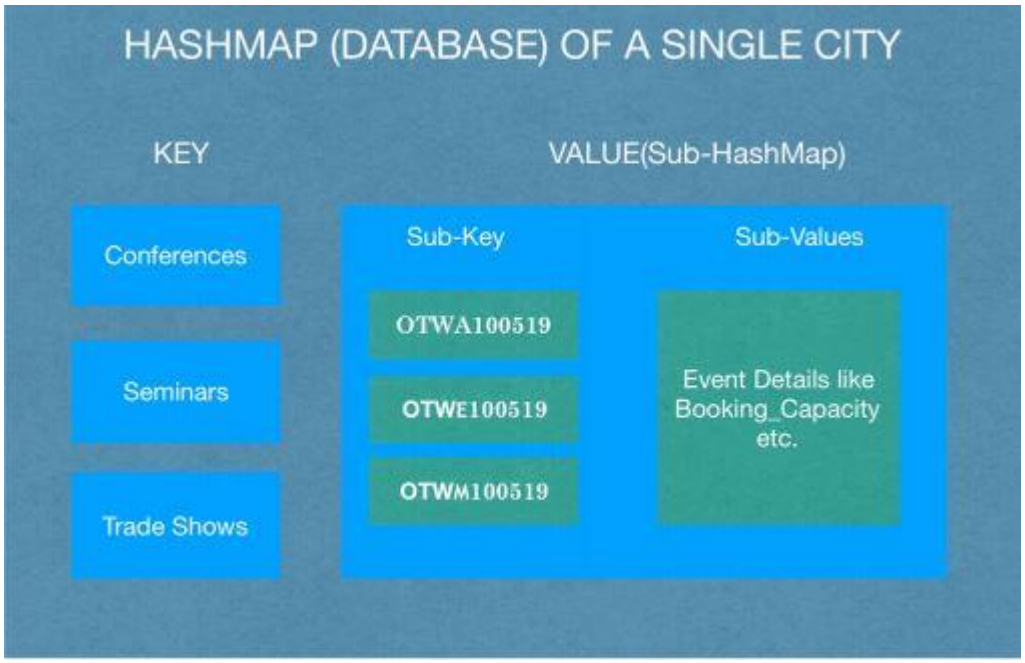


Data Structures

- HashMap “city”_hashmap: is used to maintain the list of events of each type, here semester (Seminar/Conference/TradeShow) are used as keys and the value is again a HashMap containing key as the Event ID (e.g. MTLE100519) and its value is an ArrayList of string type. This ArrayList as

the key holds the information like the capacity of the event and number of available seats remaining in that course.

- HashMap client_" city"_info is used to hold information about the client. Key is the event type and the value is an ArrayList of a string containing the number of events in a particular event type and their total number.



Logging

Each server also maintains a log file containing the history of all the operations that have been performed on that server.
These are some details that a single log file record contains:

- Date and time the request was sent.
- Request type (book an event, cancel an event, etc.).
- Request parameters (clientID, eventID, etc.).
- Request successfully completed/failed.
- Server response for the particular request.

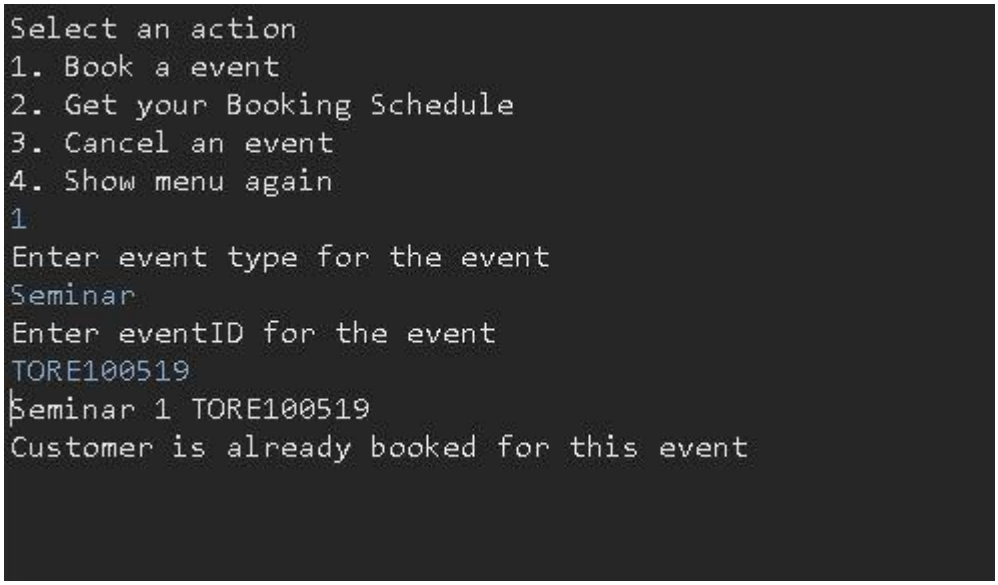
Challenges Faced during

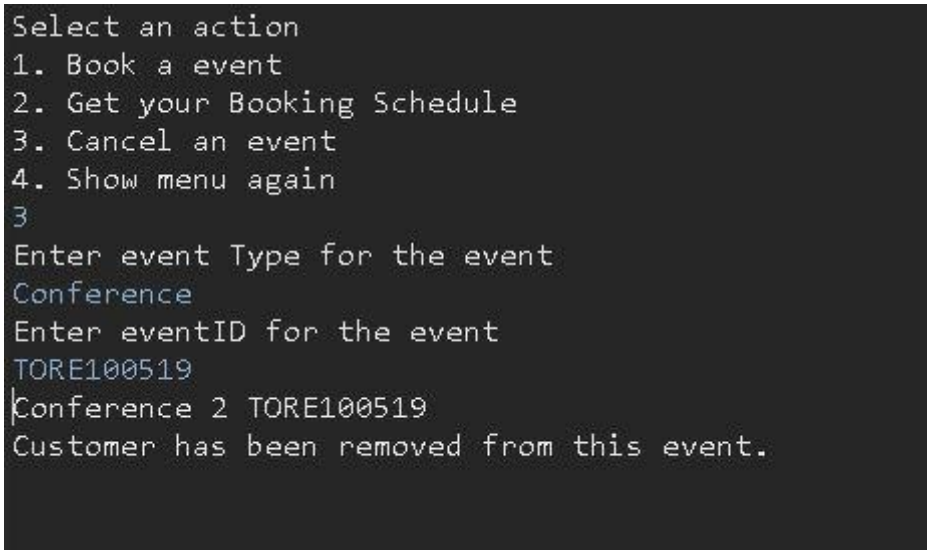
The major challenges faced were their-server communication between different cities and the back reply from them to check the occupancy and availabilities for which ExecutorService, Callable was used to create a temporary server and to managing reply to the client in a single request.
For performing the reply for a enrollment and listing courses availability ExecutorService is used , an executor thread is launched (along with callable and future) . It provides the UDP response while different other UDP calls are made.

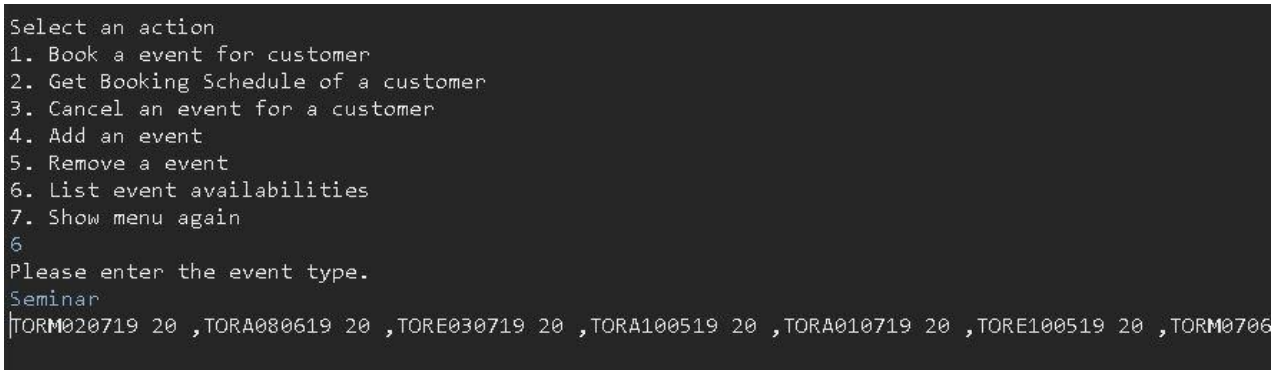
Test Scenarios

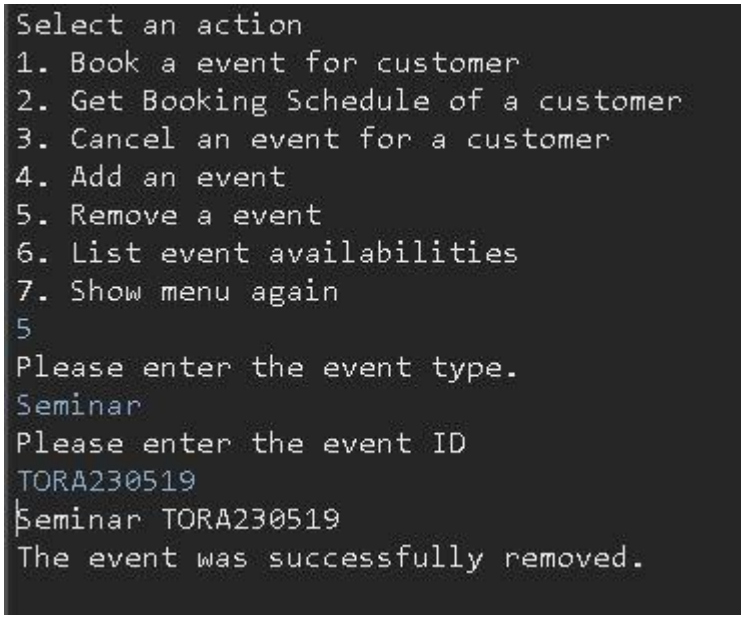
```
Entered as a Client
Enter your 8 digit ID including your city name (Montreal, Ottawa and Toronto)
TORC1111
You have entered as a client



Select an action
1. Book a event
2. Get your Booking Schedule
3. Cancel an event
4. Show menu again
```

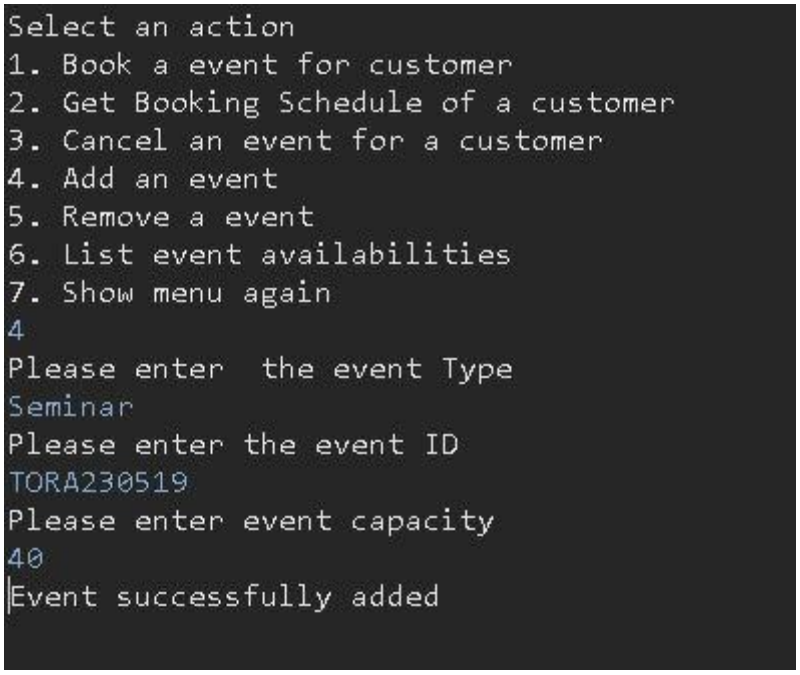
```
Select an action
1. Book a event
2. Get your Booking Schedule
3. Cancel an event
4. Show menu again
2
Following is the booking schedule for the customer
Conferences-
3
TORE100519
TORE200619
OTWE100519
Seminars-
2
TORE100519
OTWA100519
Trade Shows-
1
OTWM100519
```


Select an action
1. Book a event
2. Get your Booking Schedule
3. Cancel an event
4. Show menu again
1
Enter event type for the event
Seminar
Enter eventID for the event
TORE100519
Seminar 1 TORE100519
Customer is already booked for this event


Select an action
1. Book a event
2. Get your Booking Schedule
3. Cancel an event
4. Show menu again
3
Enter event Type for the event
Conference
Enter eventID for the event
TORE100519
Conference 2 TORE100519
Customer has been removed from this event.


Select an action
1. Book a event for customer
2. Get Booking Schedule of a customer
3. Cancel an event for a customer
4. Add an event
5. Remove a event
6. List event availabilities
7. Show menu again
6
Please enter the event type.
Seminar
TORM020719 20 ,TORA080619 20 ,TORE030719 20 ,TORA100519 20 ,TORA010719 20 ,TORE100519 20 ,TORM0706





Bibliography

[1] “Lesson: All About Datagrams (The Java™ Tutorials > Custom Networking),” Oracle, [Online]. Available: https://docs.oracle.com/javase/tutorial/networking/datagrams/index.html.

[2] “Processes and Threads (The Java™ Tutorials > Essential Classes > Concurrency),” Oracle, [Online].

Available: https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html.

[3] “Getting Started with Java IDL,” [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/idl/GShome.html.