DISTRIBUTED SYSTEM DESIGN

COMP 6231
Assignment 2

Submitted By:

Zakiya Jafrin (40021416)

How to run the Project:

1. Start ORB.
2. Start the servers for all three cities
3. Launch Login.java

Description:

In this assignment, we aim to implement a distributed event management system (DEMS) for a leading corporate event management company. A distributed system used by an event manager who manages the information about the events and customers who can book or cancel an event across the company's different branches.

There are three branches in different cities, thus three different servers representing and holding informations from those cities.

1. Toronto (TOR)
2. Montreal (MTL)
3. Ottawa (OTW)

The users of the system are event managers and event booking customers. Event Managers and customers are identified by a unique managerID and customerID respectively, which is constructed from the acronym of their branch's city and a 4 -digit number (e.g. TORM2345 for an event manager and TORC2345 for a customer) . Whenever the user performs an operation, the system must identify the server that the user belongs to by looking at the ID prefix (i.e. TORM or TORC) and perform the operation on that server.

Manager can create slots of events corresponding to event type as per the availability. There are three event types for which event slots can be created,

1. Conferences
2. Trade shows
3. Seminars

There are three time slots available for each event type on a day- Morning(M), Afternoon(A) and Evening(E). An event id is a combination of city, time slot and event date (e.g. TORM100519 for a morning event on 10th May 2019 in Toronto, OTWA100519 for an afternoon event on 10th May 2019 in Ottawa and MTLE100519 for an evening event on 10th May 2019 in Montreal).

The roles of a Manager and customer is different. Manager can perform all kind of operation. But customer is offered only with selective type of tasks.
The event manager could perform:

1. Add a New Event
2. Remove a new Event
3. List all the available Events depending the given event type
4. Book an Event

5. Cancel an Event
6. Get a Booing ecedule of a customer
7. Swap Event

Customer only limited to Following kinds of work:

1. Book an Event
2. Cancel an Event
3. Get a Booing ecedule of a customer

A customer or manager can book an event in any branch of the company, for any event type, if the event is available for booking (if the events slots corresponding to an event on a particular date is not yet full). A server (which receives the request) maintains a booking-count for every customer. If the availability of an event is full, more customers cannot book the event. Also, a customer can book as many events in his/her own city, but only at most 3 events from other cities overall in a month.

Requirements:

The application has the following requirements:

- Use IDL Interface definition for the City Servers with specified operations
- Use ORB tools.
- Implement the Servers
- Design and implement a CustomerClient, which invokes the server system to test the correct operation of the DEMS invoking multiple servers and multiple customers.
- Design and implement a EventManagerClient, which invokes the server system to test correct operation of the DEMS invoking multiple servers and multiple event managers.

Protocols Used:

1. **IDL Compiler:** To register remote objects using.
2. **ORB :** Using CORBA naming service register the objects remotely. Then Access the objects by clients. Clients and servers are running on different threads
3. **Java Programming Language**
4. **UDP/IP Socket :** Inter Server Communication is done using UDP Protocol. As we try to access any distributed information. The servers need to communicate with each other.

Architecture:

1. Client:

In this package we aggregated classes that are used to identify the type of user and the city the client resides. Thus we can invoke necessary measures. While taking the clients we are validating the input and the city he/she belongs to

   a. CustomerCliet
   b. ManagerClient
   c. User
   d. Login: this is where our Project Starts running

2. Logging:
   This Package is used to save the logging informations. All types of clients that is Manager and Customer saves every action they did. This is to keep track of the request response of the DEMS system. Also it helps keeping track which method resulted in wich output. Classes include:
   a. MyFormatter
   b. MyLogger

3. RemoteObject:
   This Is the package for remote object Registry. We registered the server using IDL interface in this package. The IDL compiler tools will generate necessary files to incorporate with Java(The language we used for our server side program). The remote methods registers themselves to invoked when called.
   The ORB tools comes handy to let customer use the remote objects. Server Registers itself using port.
   a. EnrollmentImpl

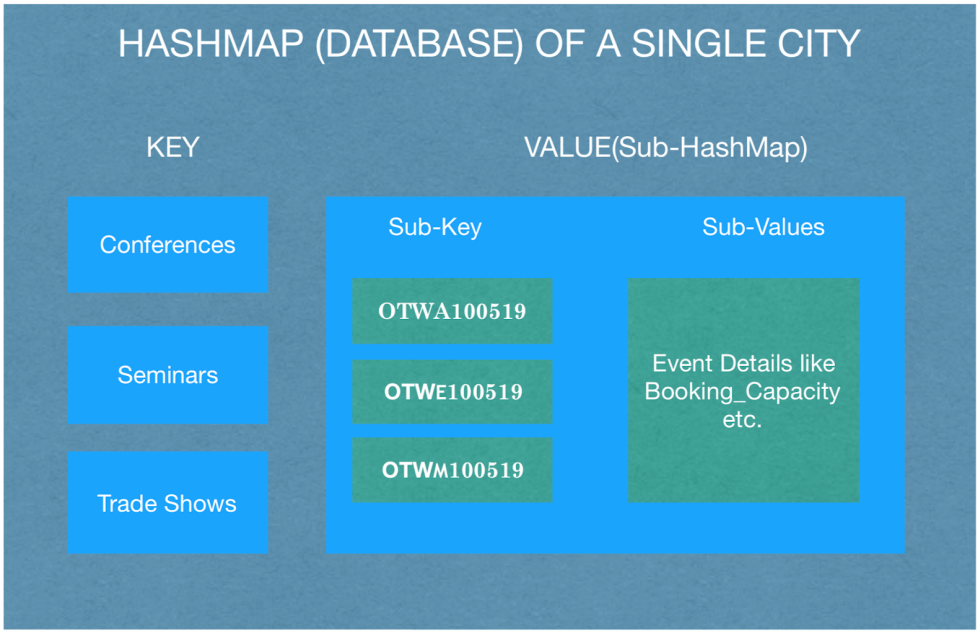
4. Server:
   This is the Package where all the servers of different city is defined. The servers communicate among themselves using UDP connection. Whenever client requires info that is distributed among the servers it does so.
   a. MTL_Server
   b. TOR_Server
   c. OTW_Server

5. Util:
   This is a Simple helper package we used to aggregate all the constants, frequently used strings, enums. This package is used in whole project.
   a. City
   b. Constants
   c. EventType
   d. Role
   e. Utils

Data Structure:

As per the requirements we stored all the data is hashmap and sub hashmap.
private HashMap<String, HashMap<String, HashMap<String, Object>>> cityDatabase

The parent hashMap has a ‘key’ that is the even type the client used- that is a String
The value of this is sub hashmap
The then the sub HashMap has a ‘key’ the eventID which is pointing at another sub hashmap
This contains a key value pair of the eventDetails. Where the key is the name and the value is the information.



We strictly followed the requirements regarding the data structure for storing the informations of DEMS.

Features:

Concurrency: Every server is running on its own thread and for every mutation in EnrollmentImpl we use synchronized functions.
For Example:

```
synchronized (this) {
        events.remove(eventId);
}
```

Test Cases:

1. **Login**

	Operation	**Input**	**Output**	**Result**
1	Validate City	HALM1234	HAL isnt recognized	Pass
2	Validate User Type	TORX1234	Role X isnt recognized	Pass
3	Case insensitivity	torm1234	Login successful	Pass
4	Invalid length	TORM12	Should be 8 digits long	Pass

2. Manager Add an Event

	Operation	**Input**	**Output**	**Result**
1	Adds an non-existing event	MONA020519	Event added	Pass
2	Adds an existing event	MONA020519	Event not added	Pass
3	Wrong eventID	MONA111111	Event does not exist	Pass

3. Manager Remove an Event

	Operation	**Input**	**Output**	**Result**
1	Removes an existing event	MONA020519	Event successfully removed	Pass
2	Removes an non-existing event	MONE020519	eventType does not have any event with thiseventID	pass

4. Manager List Event Availability

	Operation	Input	Output	Result
1	List all the events available specifying wrong eventType	Movie	EventType not found	Pass

5. Manager/Customer Book an Event

	Operation	Input	Output	Result
1	Book an Event that is not already booked	MONE020519	SuccessFully booked an event	Pass
2.	Book an Event that is already booked	MONE020519	The event is already booked with MONE020519. Can not book an event	Pass

6. Manager/Customer Cancel an Event

	Operation	Input	Output	Result
1	cancel an Event that is not already booked	MONE020519	SuccessFully cancelled an event	Pass
2.	Customer trying to cancel wrong event	MONE020519	Sorry this event is not booked by customerID.	Pass

6. Customer/Manager getting the booking Schedule:

	Operation	Input	Output	Result
1	Customer accessing booking schedule		Gets the booking schedule by the customer	Pass
2.	Manager accessing booking schedule		Gets all the event those are booked across the servers	Pass

6. Swap Event:

	Operation	Input	Output	Result
1	Customer accessing booking schedule		Gets the booking schedule by the customer	Pass
2.	Manager accessing booking schedule		Gets all the event those are booked across the servers	Pass