DEPARTMENT OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6231 - DISTRIBUTED SYSTEM DESIGN
WINTER 2019

DISTRIBUTED EVENT REGISTRATION SYSTEM USING CORBA

Assignment-2

Submitted By
CH. Venkat Mani Deep - 40080924
CH. Satish - 40091187

Overview

The assignment is to develop a Distributed System for a group of events: used by event managers to manage the information about the items that are available in the servers and users are to add, cancel, view and swap events across the servers.
This document presents the designs used in implementing the project.

Terminology

Features

Add event	This allows Manager to add/update details about the events into the respective servers.
Remove event	This allows the Manager remove the event from the respective library.
View event	This feature is used by manager to look all the events of a certain type in all the servers.
Book event	This allows the user to book an event, If the booking capacity of the event is available
Cancel event	This allows the user to cancel an event, If the event is been registered
View Booked events	This feature is used by user to look all the booked events of a user in all the servers.
Swap Event	This feature is used by user to swap an old event with new event.

System Architecture:-

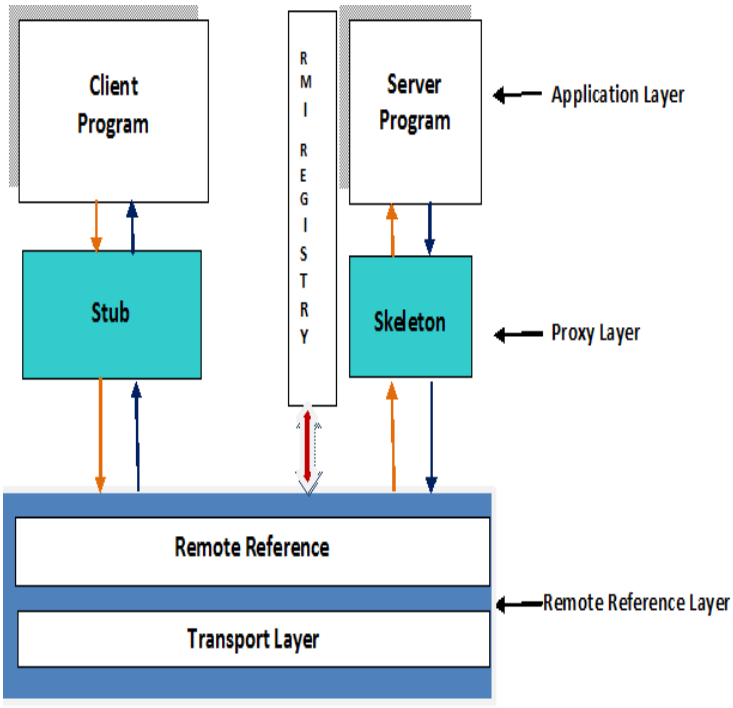


Fig: RMI Architecture

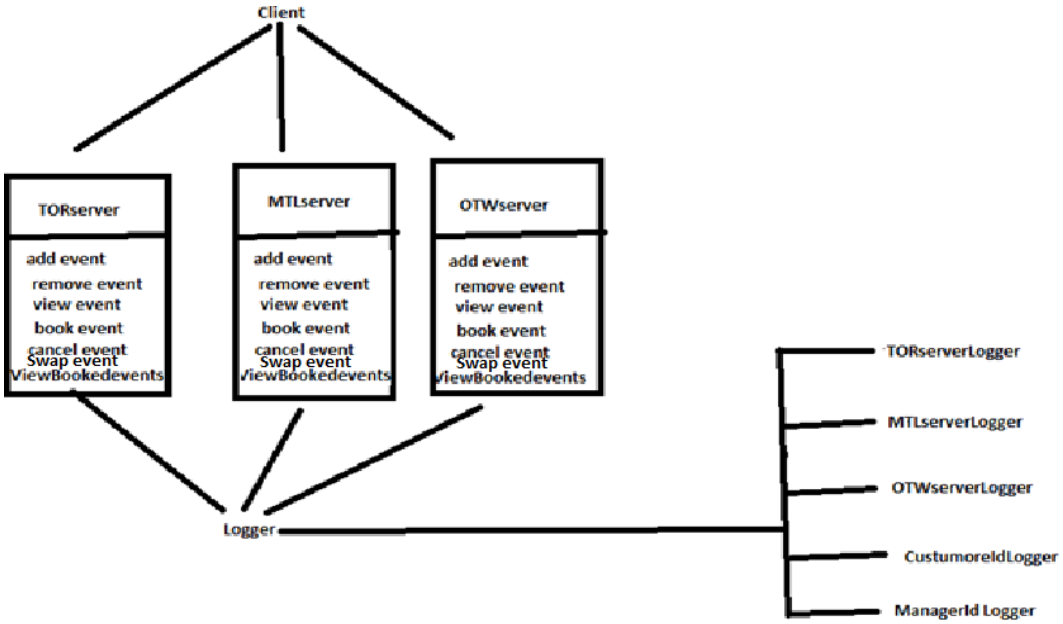
About Corba

The **Common Object Request Broker Architecture** (**CORBA**) is a standard designed to facilitate the communication of systems that are deployed on diverse platforms. CORBA enables collaboration between systems on different operating systems, programming languages, and computing hardware. CORBA uses an interface definition language (IDL) to specify the interfaces that objects present to the outer world. CORBA then specifies a *mapping* from IDL to a specific implementation language like C++ or Java

Corba Steps

1. Create an remote.idl file
2. Generate the bindings
3. Create Remote Interface
4. Implement servers b
5. Implement client

System Architecture:-



Detailed Architecture

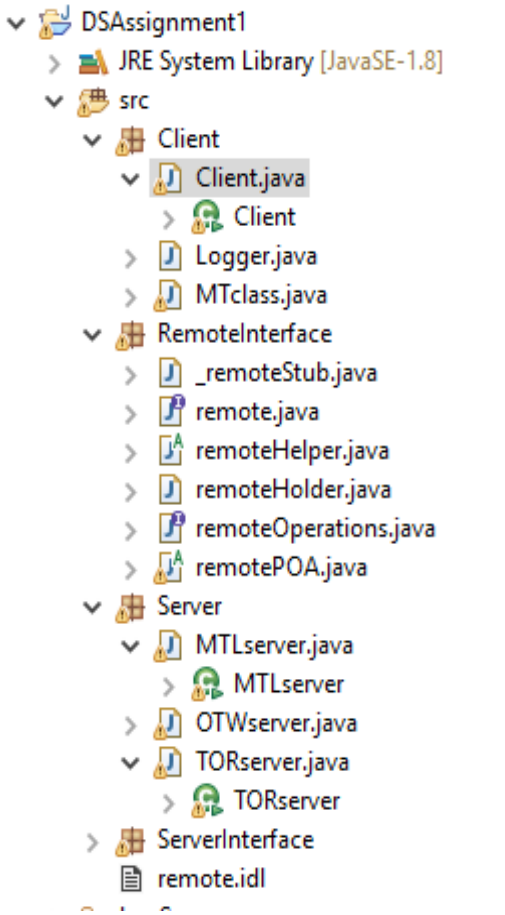
Our DEMS consists of three servers namely, MTL, TOR and OTW All the servers perform same type of operations.

The functionalities mentioned above will be defined in an interface. The implementations of the functionalities will be added in Server Classes.

We also have Manager Client and User Client to perform their respective functionalities.

Data is stored in the form of HashMap in each server.
Server to Server communication is maintained by UDP.

Workspace Overview:-



Packages

1. Client
2. RemoteInterface
3. Server
4. ServerInterface
5. remote.idl

Client

1) Evaluates the operations of manager and client.

2) Has a logger class which logs the details of all other users.

Server

It has 3 classes

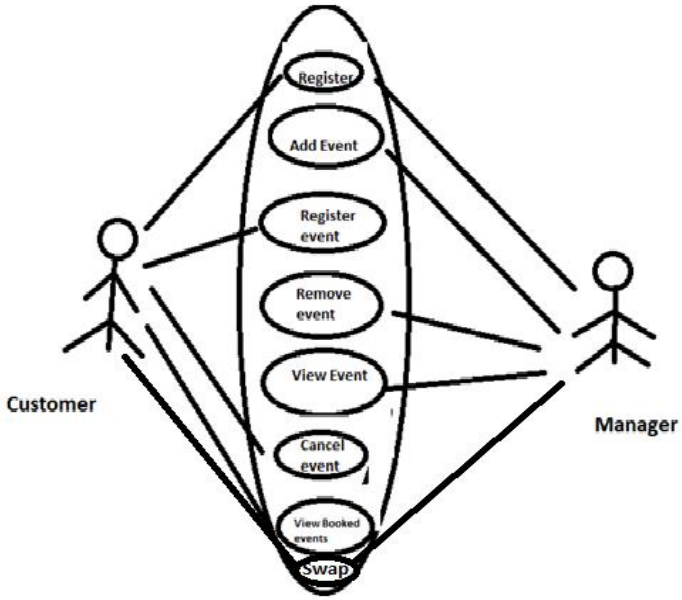
1) MTLserver

2) OTWserver

3) TORserver

1. Server class is the implementation of server interface. It contains all the definitions of the functions of the server interface.
2. It helps in proper implementation of all the methods and communication between these methods is done by datagram.

USE CASE DIAGRAM



This diagram gives a general view of the total architecture

Customer :-

1) Can register with the server data base
   Ex) Toronto customer can only register with the Toronto server with Id:TORC1000
2) Can register for any event in the database. but at max he can register for 3 events outside of the his state.
3) He can cancel the events he is been registered to
4) View booked events helps to view the ids of the all the events he is been registered into.

Manager:

1) Can register with the server data base
   Ex) Toronto manager can only register with the Toronto server with Id:TORM1000

2) Can register a customer for any event in the database. but at max he can register a customer for 3 events outside of the his state.
3) He can delete the events .
4) View booked events helps to view the ids of the all the events by event types.

Data Structures Used

Events: - Created a **HashMap** of **HashMap** where the inner map is of type String and event id where event id is the object of type event class contains Time, date, and capacity and key is the event id. The outer map key is the event type.

Event Record: - Event records are maintained in the **array list** that contains Time, date, and booking capacity.

Customer registered events - Customer registered events are maintained in **HashMap of Array List** where inner map contains key of Customer Id and value of event its.

Most Difficult Part

The most difficult part in the Assignment is Maintaining up records and validations of events (ie : A student can register up to only 3 events outside of his state) so we have to validate the events he has been registered into before registering him into any event. This part was little difficult because there was a very large amount of raw data so we had to trim the data the data before the use

The second most difficult part was to display the registered events of the customer from various servers here at our first time implementation we have had encountered a un terminating loop. We have overcome this by a very intelligent way by using two different function calls to two different methods.

It was very challenging to implement the complete system.

References

[“Common Object Request Broker Architecture.” *Wikipedia*, Wikimedia Foundation, 10 Oct. 2018, en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture.]

[Wang, Alexia. “Corba Introduction and Simple Example.” *LinkedIn SlideShare*, 3 July 2014, www.slideshare.net/lanxuezaipiao/corba-introduction-and-simple-example.]