

# Distributed System Design

COMP 6231: Winter 2020

Distributed Event Management System (DEMS) using Java IDL (CORBA)

Instructor: Professor R. Jayakumar

By: Seyed Pouria Zahraei Student ID: 40115175

**Introduction:**

In this assignment, we implemented a distributed event management system (DEMS) for a leading corporate event management company: a distributed system used by event managers who manage the information about the events and customers who can book or cancel an event across the company's different branches.

Managers are allowed to perform some functions such as: `addEvent`, `removeEvent`, `listEventAvailability`.

Additionally customers are allowed to perform some functions such as: `bookEvent`, `getBookingSchedule`, `cancelEvent`.

We have 3 servers, Montreal, Quebec and Sherbrook. Therefore, we have one class for implementation of servers and we instantiate each of servers, and pass the required parameters for running the servers through the constructor.

In this assignment in order to perform some of these functions we need some inter server communication. Therefore, we implemented additional functions, which you can observe bellow.

**Introduction:**

In this assignment, we implemented a distributed event management system (DEMS) for a leading corporate event management company: a distributed system used by event managers who manage the information about the events and customers who can book or cancel an event across the company's different branches.

Managers are allowed to perform some functions such as: `addEvent`, `removeEvent`, `listEventAvailability`.

Additionally customers are allowed to perform some functions such as: `bookEvent`, `getBookingSchedule`, `cancelEvent`, `swapEvent`.

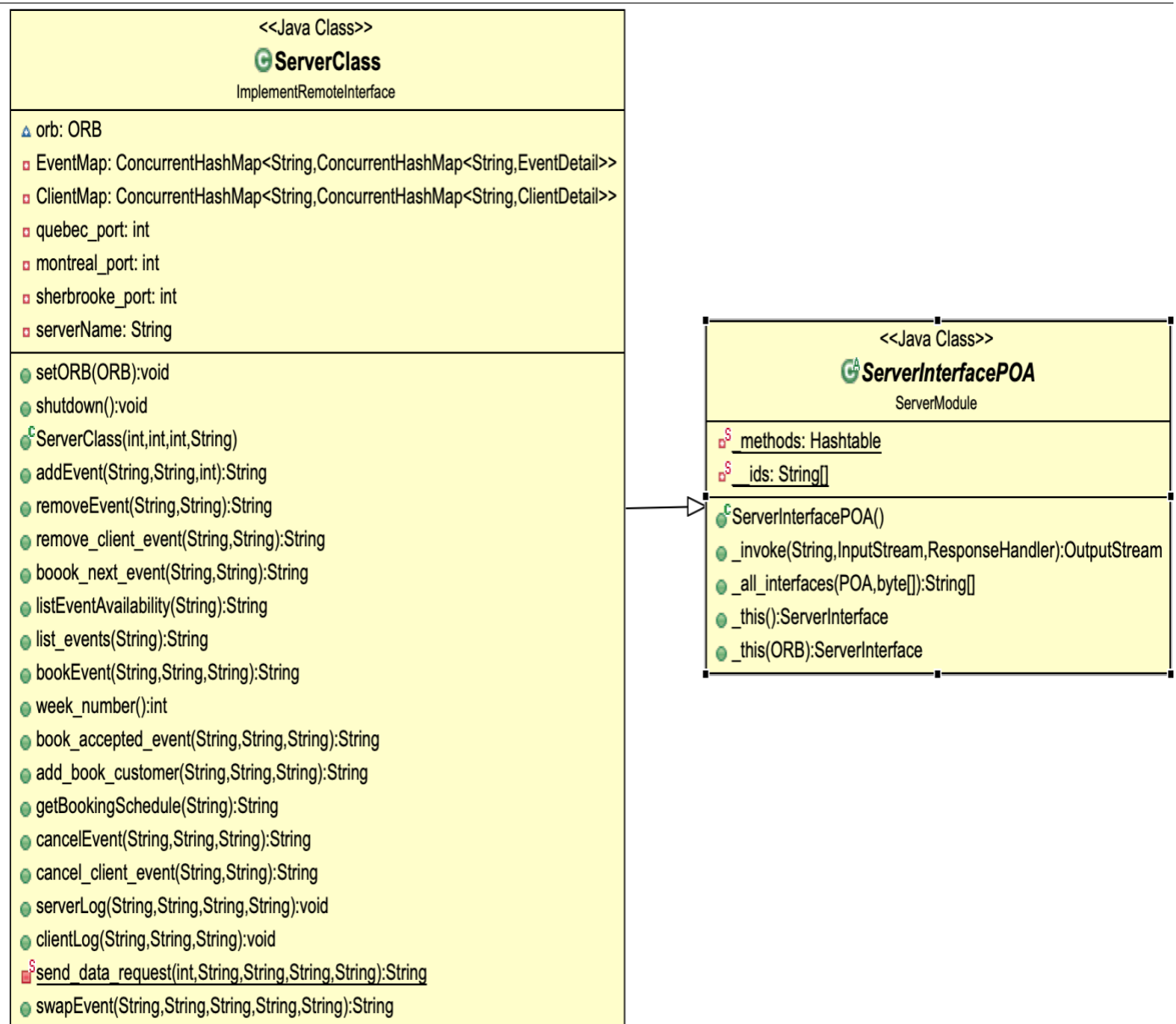
We have 3 servers, Montreal, Quebec and Sherbrook. Therefore, we have one class for implementation of servers and we instantiate each of servers, and pass the required parameters for running the servers through the constructor.

In this assignment in order to perform some of these functions we need some inter server communication. Therefore, we implemented additional functions, which you can observe bellow.

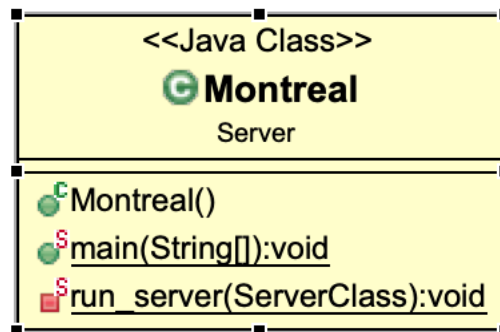
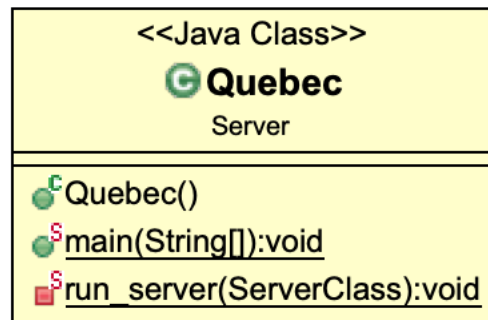
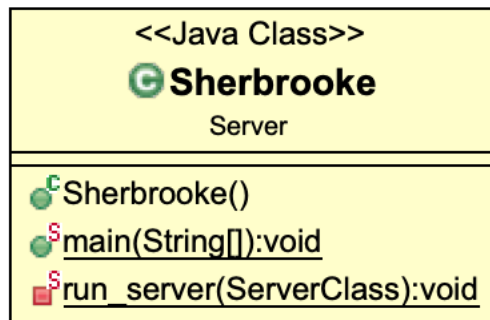
## Class Diagram

Server side:

This class is implemented in order to handle the requests from others servers through UDP and requests from different users. In order to handle the requests through CORBA from users, we have 7 main functions. Additionally, some other functions such as, remove\_client, list\_events and etc. in order to handle the requests through UDP.

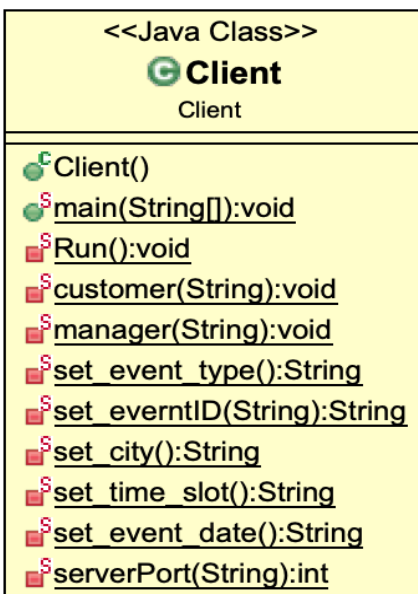


In each server we insatiate the ServerClass and will pass the proper port and name through the constructors.



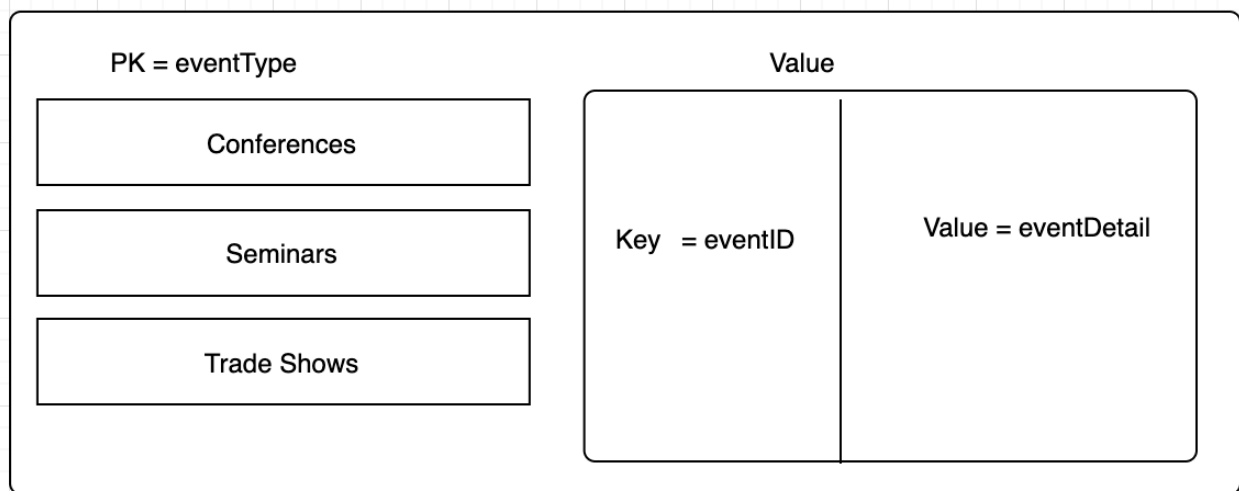
Client side:

This class is implemented in order to handle the interface, where user can log in with their own ID(ManagerID, CustomerID) and perform their actions on events.

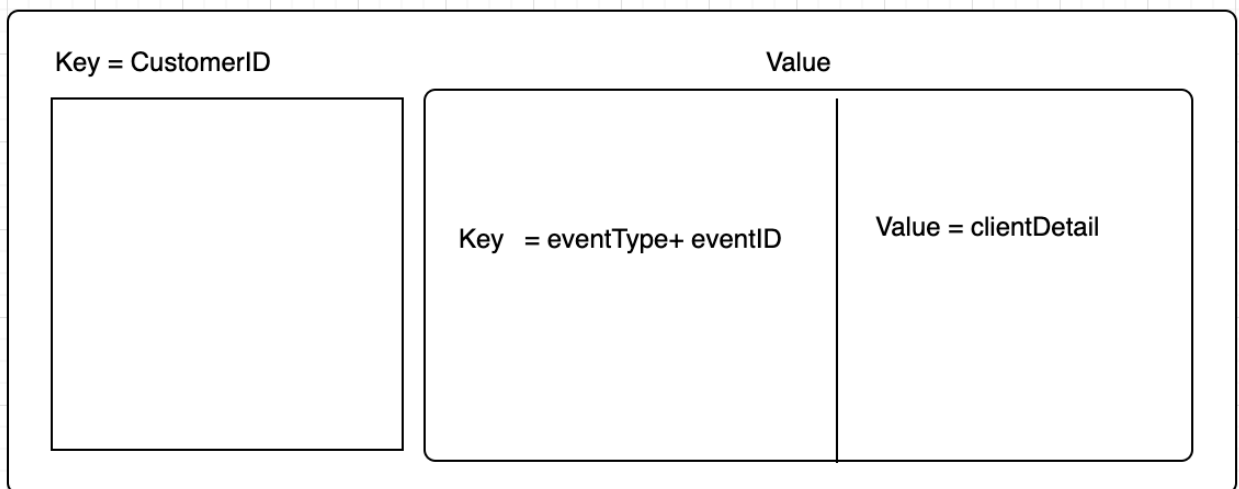


Database:

Our database for server is a Hashmap, where the key is eventType and value of Hashmap is another Hashmap, where eventID is the key and the value of Sub-Hashmap is an object(eventDetails).

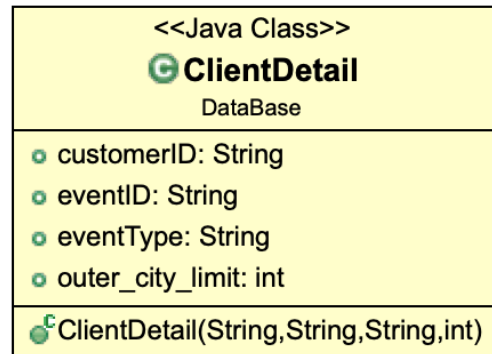
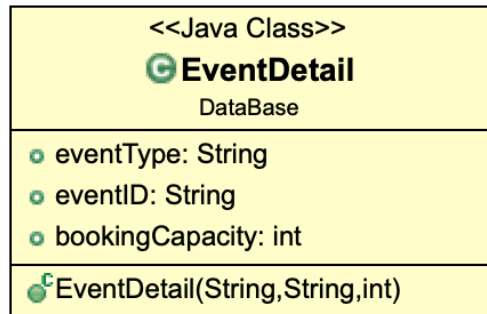


We implemented another hashMap in order to store the events related to the customer, where customerID is the key of the main HashMap and the value of main HashMap is consists of another sub-Hashmap, where the key is an combination of eventType and eventID and the value is an object(clientDetail).



The diagram below indicates our object classes, which we used in our HashMaps above. In EventDetail we have additional attribute, bookingCapacity, which indicates the capacity of each events.

In the ClientDetail we have additional attribute, outer\_city\_limit, in order to memorize how many orders a customer had outside of his own city.



Test Scenario:

Test No	Test Scenario	Test Case	Output
1	Login as an employee	MTLM1111 entered as an ID for employee	Please enter number of the action: 1.Add new event 2.Remove an event 3.Check availability of an event 4.log in as a customer 5.Exit
2	Add new event	Adding an event with capacity 4: MTLA101010	Event added toMTL
3	Add second event	Adding another event with	Event added toMTL

		capacity 3: MTLA101010	
4	List of availability	Check the all available event for particular eventType, in this case Conference	List of availability for Conferences: MTLA200920 3, MTLA101010 4,
5	Login as a customer	QUEC2345 entered as an ID for customer	Please enter number of the action: 1.Book an event 2.Get booking schedule 3.Cancel event 4.Exit
6	Book an event	Try to book the event from other server: MTLA101010	BOOKING_APPROVED
7	Get booking schedule	Check which event customer has booked	EventType:CONFERENCES EventID:MTLA101010
8	Event capacity	Check if the capacity of event after booking became less	MTLA200920 3, MTLA101010 3
9	Book an event	Try to book the same event from other server: MTLA101010	ERR_RECORD_EXISTS
10	Book an event	Try to book another event from other server: MTLA200920	BOOKING_APPROVED
11	Book an event	Try to book another event from other server: MTLA301220	BOOKING_APPROVED



12	Book an event	Try to book an event from other server for more than 3 times: MTLA301220	This customer has already booked 3 times from other cities!
13	Remove event	Remove event MTLA101010 and check if it will book the next available event	<p>Before removing the event:</p> <p>EventType:CONFERENCES EventID:MTLA101010            EventType:CONFERENCES EventID:MTLA200920            EventType:CONFERENCES EventID:MTLA301220</p> <p>After removing the event:</p> <p>EventType:CONFERENCES EventID:MTLA200920            EventType:CONFERENCES EventID:MTLA120222            EventType:CONFERENCES EventID:MTLA301220</p>
14	Cancel event	Cancel an event MTLA301220	<p>EventType:CONFERENCES EventID:MTLA200920            EventType:CONFERENCES EventID:MTLA120222</p>
15	Get booking schedule	Check the booking capacity for the event after canceling MTLA301220	<p>Before canceling the event:</p> <p>MTLA120222 2, MTLA301220 2, MTLA200920 2,</p> <p>After canceling the event:</p> <p>MTLA120222 2, MTLA301220 3, MTLA200920 2,</p>
16	Manager login on behalf of customer	Manager login on behalf of customer MTLC2345	Please enter number of the action: 1.Book an event 2.Get booking schedule 3.Cancel event 4.Exit
17	Book an non-existing event	MTLC2345 tries to book and event MTLE101010, Which is not exists	ERR_NO_RECORD!

Test case for testing concurrency and atomicity:

- Manager sends two requests at the same time to add two different events (MTLA101010 with capacity of 3 and MTLE202020 with capacity of 2) with the eventType of Conferences.

- 5 different Customers try to book the event MTLA101010 with eventType of Conferences.
- Manager removes two events.

Result:

```
addEvent("MTLE202020", "Conferences", 1) -->Event added toMTL
addEvent("MTLA101010", "Conferences", 2) -->Event added toMTL
-----
bookEvent("MTLC5678", "MTLA101010", "Conferences") -->BOOKING_APPROVED
bookEvent("MTLC1234", "MTLA101010", "Conferences") -->BOOKING_APPROVED
bookEvent("MTLC8765", "MTLA101010", "Conferences") -->BOOKING_APPROVED
bookEvent("MTLC4321", "MTLA101010", "Conferences") -->ERR_NO_CAPACITY
bookEvent("MTLC0000", "MTLA101010", "Conferences") -->ERR_NO_CAPACITY
removeEvent("MTLA101010", "Conferences") -->Event with eventID:MTLA101010 and eventType: CONFERENCES for clients has been removed for server

removeEvent("MTLE202020", "Conferences") -->Event with eventID:MTLE202020 and eventType: CONFERENCES for clients has been removed for server
```