Assignment 2:

Distributed Event Management System (DEMS) using CORBA and JavaIdl.

Submitted by:

Name: **Shubham Ranadive**

Student ID: **40083991**

Name: **Krisha Patel**

Student ID: **40084336**

Distributed Event Management System

(1)About System

There are two types of the users for this system:

(1) Manager

(2) Customer

In this system, there are three servers: Montreal server, Ottawa server and Toronto server which handle their own events database solely.

(2) Operation this system can perform

- **Operations for Manager**
  - Add Event: Manager of one city can add events in their database with Event Type as Seminar, Conference and Trade shows and their respective booking capacity.
  - Remove Event: Manager can also remove any event from their database at any time.
  - List Availability: Manager can see the availability for any Event Type from the all servers.

- **Operations for Customer**
  - Book Event: Customer can book event in their own city multiple time if there is enough booking capacity available for the particular event. They can also book events from the other city but for the particular customer, they can only book 3 events outside their own city in a given month.
  - Cancel Event: Customer can also cancel events from their booked events.
  - Get Schedule: Customer can also see their booked events in this section.
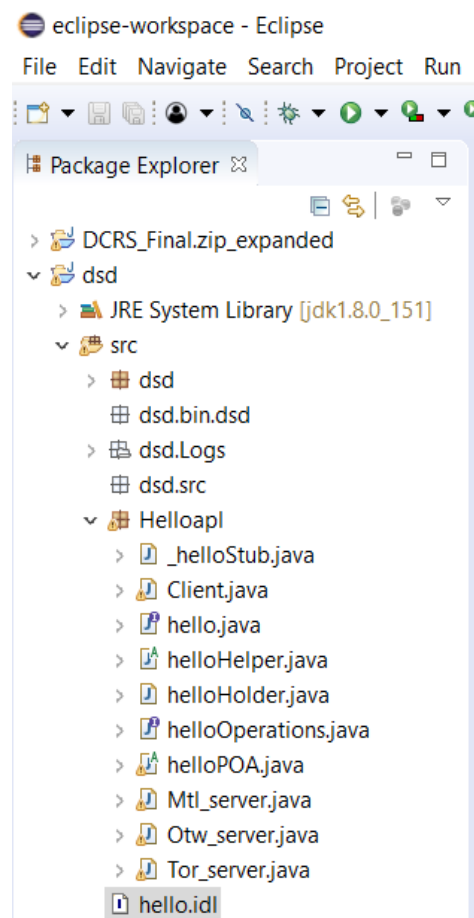 - Swap Event: Customer can swap events by booking new ones by cancelling old ones.

- **All the above operations of the customer can also be performed by the manager.**

(3)Methods and Protocols Used.

In this system, CORBA and Java Idl is used to communicate between server and client. Also, the UDP Protocol is used to do inter-server communication.

(4)Architecture

The architecture used is client-server method. Client can be manager or a customer. There are three servers for three cities: **Montreal, Ottawa and Toronto.**

Hello.idl has all the interfaces for CORBA to function, it allows transparency as if client is calling its own local methods but instead client calls methods of the server through it.

helloStub.java works as client's proxy which helps to send request to the server and get response as per client's language.

helloHelper.java provides auxiliary functionality needed to support a CORBA object in the context of the Java language.

helloHolder.java this class holds (contains) a reference to an object that implements the Hello interface.

helloOperations.java looks at this file to make sure that the method signatures correspond to what you expect.

helloPOA.java The Java class helloImplPOA.java is the skeleton, the server-side proxy, combined with the portable object adapter.

Client.java handles all the operations to be performed by customer and manager. It makes necessary calls with servers and the message from it.

Mtl_server.java is responsible for doing all the operations related to the Montreal city and giving the final message output to the client. This is an implementation(helloImpl) for helloPOA.

Tor_server.java is responsible for doing all the operations related to the Toronto city and giving the final message output to the client. This is another implementation(helloImpl) for helloPOA

Otw_server.java is responsible for doing all the operations related to the Ottawa city and giving the final message output to the client. This is another implementation(helloImpl) for helloPOA

(5)Data Structure

We have used **Hash Map** to store data on server side; it stores data in form of key-value pair. We have also used **Array list** for storing multiple values for the same key.

There are **five hash maps** used by us per server. Here example for Montreal's server is given below:

(1) MTLEvents

This hashmap is used to stores the events details for the particular city. It stores Event Type, Event Id and its booking capacity. This hashmap has Event Type as a key, EventID as sub-key and bookingcapacity as its value.

(2) Mtl_Con

This hashmap is used to stores the details of the customer who has booked Conference event in Montreal city. This hashmap has Customer ID as a key and its value consists of array list of the Event ID that customer has booked.

(3) Mtl_Ts

This hashmap is used to stores the details of the customer who has booked Trade Show event in Montreal city. This hashmap has Customer ID as a key and its value consists of array list of the Event ID that customer has booked.

(4) Mtl_Sem

This hashmap is used to stores the details of the customer who has booked Seminar event in Montreal city. This hashmap has Customer ID as a key and its value consists of array list of the Event ID that customer has booked.
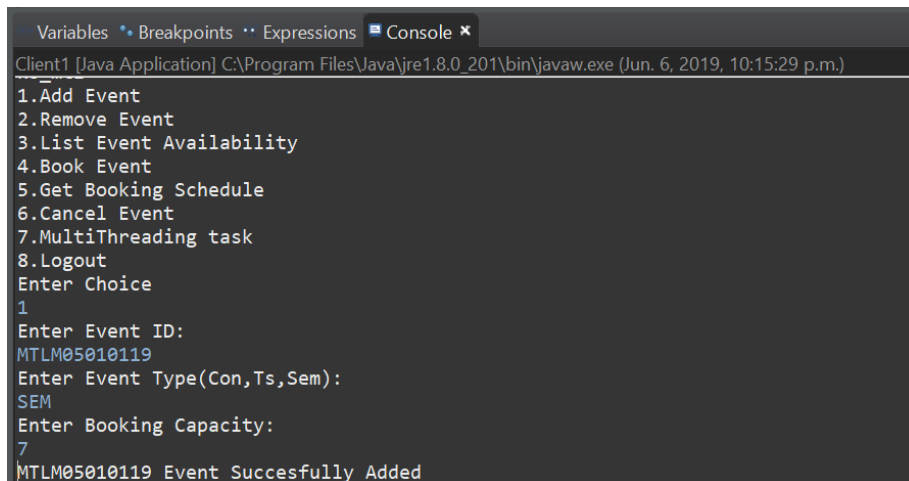
(5) Counter

This hashmap is used to keep track on the events booked by the customer outside their own city. The hashmap has Customer Id as a key and value of array list which consists of twelve fields for the twelve months in a year.
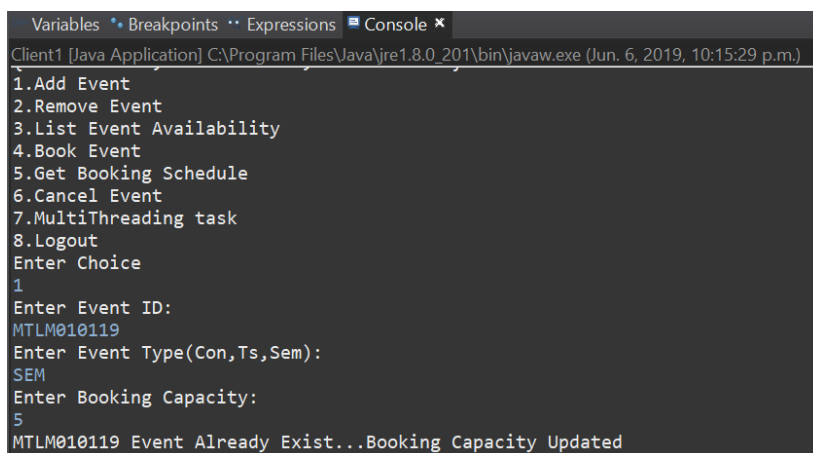
(6)Test Scenarios.

(1) Add event

  (i) This is the test scenario for successful adding of an event

  (ii) This test scenario shows that the booking capacity will be updated if the particular event is already present.

Image for Add new Event


```
  Variables  •• Breakpoints  •• Expressions  ▣ Console ✕
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 10:15:29 p.m.)
1.Add Event
2.Remove Event
3.List Event Availability
4.Book Event
5.Get Booking Schedule
6.Cancel Event
7.MultiThreading task
8.Logout
Enter Choice
1
Enter Event ID:
MTLM05010119
Enter Event Type(Con,Ts,Sem):
SEM
Enter Booking Capacity:
7
MTLM05010119 Event Succesfully Added
```

Image for update Event capacity


```
  Variables  •• Breakpoints  •• Expressions  ▣ Console ✕
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 10:15:29 p.m.)
1.Add Event
2.Remove Event
3.List Event Availability
4.Book Event
5.Get Booking Schedule
6.Cancel Event
7.MultiThreading task
8.Logout
Enter Choice
1
Enter Event ID:
MTLM010119
Enter Event Type(Con,Ts,Sem):
SEM
Enter Booking Capacity:
5
MTLM010119 Event Already Exist...Booking Capacity Updated
```

(2) List Availability

This test scenario shows the particular Event Type and their booking capacity from all the three cities.

```
  Variables  • Breakpoints  •• Expressions   ▣ Console ✕
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 10:15:29 p.m.)
MTLM05010119 Event Successfully Added
1.Add Event
2.Remove Event
3.List Event Availability
4.Book Event
5.Get Booking Schedule
6.Cancel Event
7.MultiThreading task
8.Logout
Enter Choice
3
Enter Event Type:
SEM
{MTLM010119=5, MTLM05010119=7, MTLA010119=7, MTLE010119=8}
{TORM020119=4, TORA020119=6, TORE020119=8}
{OTWM030119=4, OTWA030119=6, OTWE030119=4}
```

(3) Remove Event

This test scenario shows the successful remove event

```
  Variables  • Breakpoints  •• Expressions   ▣ Console ✕
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 10:30:42 p.m.)
ks_mtl
1.Add Event
2.Remove Event
3.List Event Availability
4.Book Event
5.Get Booking Schedule
6.Cancel Event
7.MultiThreading task
8.Logout
Enter Choice
2
Enter Event ID:
MTLM010119
Enter Event Type:
SEM
MTLM010119 Event has been removed.
```

(4) Book Event

(i) This is the test scenario for successful booking of an event.

```
   Variables •• Breakpoints •• Expressions ▣ Console ✕
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 9:47:11 p.m.)
ks_mtl
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Logout
Enter your Choice=
1
Enter Event ID:MTLM010119
Enter Event Type:SEM
Booking for MTLM010119 is succesful for the MTLC1234
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Logout
Enter your Choice=
```

(ii) This test scenario shows for the particular event if it is already booking by the customer.

```
   Variables •• Breakpoints •• Expressions ▣ Console ✕
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 9:47:11 p.m.)
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Logout
Enter your Choice=
1
Enter Event ID:MTLM010119
Enter Event Type:SEM
You have already book the event
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Logout
Enter your Choice=
4
```

(iii) This test scenario shows for the particular event if the is no booking capacity left.

```
 Variables  • Breakpoints  •• Expressions  ☐ Console ✖
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 9:55:21 p.m.)
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Logout
Enter your Choice=
1
Enter Event ID:OTWM030119
Enter Event Type:SEM
You have already registered in 3 events outside Montreal
```

(iv) This test scenario shows for the customer who is trying to book the event outside thier own city more than three times in a given month.

```
 Variables  • Breakpoints  •• Expressions  ☐ Console ✖
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 9:51:08 p.m.)
ks_mtl
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Logout
Enter your Choice=
1
Enter Event ID:MTLM010119
Enter Event Type:SEM
This event is full.
```

(5) Get Schedule

This test scenarios shows the events booked by the customer.

```
 Variables  • Breakpoints  •• Expressions  ☐ Console ✖
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 9:51:08 p.m.)
ks_mtl
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Logout
Enter your Choice=
1
Enter Event ID:MTLM010119
Enter Event Type:SEM
This event is full.
```

(6) Cancel Event

(i) This test shows the successful cancelation of the event which was booked by the customer.

```
Variables ·• Breakpoints ·· Expressions ▣ Console ✕
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 10:04:51 p.m.)
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Logout
Enter your Choice=
3
Enter Event ID to drop:
MTLA010119
Enter Event Type:SEM
MTLA010119 Event has been removed.
```

(ii) This test shows the cancelation of the event which was not booked by the customer.

```
Variables ·• Breakpoints ·· Expressions ▣ Console ✕
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 10:10:24 p.m.)
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Logout
Enter your Choice=
3
Enter Event ID to drop:
MTLM010119
Enter Event Type:SEM
You haven't booked this event.
```

7) Swap Event

a)Successful

```
Client [Java Application] C:\Program Files\Java\jdk1.8.0_151\bin\javaw.exe (Jul
5.Logout
Enter your Choice=
1
Enter Event ID:TORM020119
Enter Event Type:SEM
Booking for TORM020119 is successful for the TORC1234
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Swap Event
5.Logout
Enter your Choice=
4
Enter new Event ID:
TORE040119
Enter new Event Type:TS
Enter old Event ID:
TORM020119
Enter old Event Type:SEM
Events are Swapped Successfully
1.Book Event
```
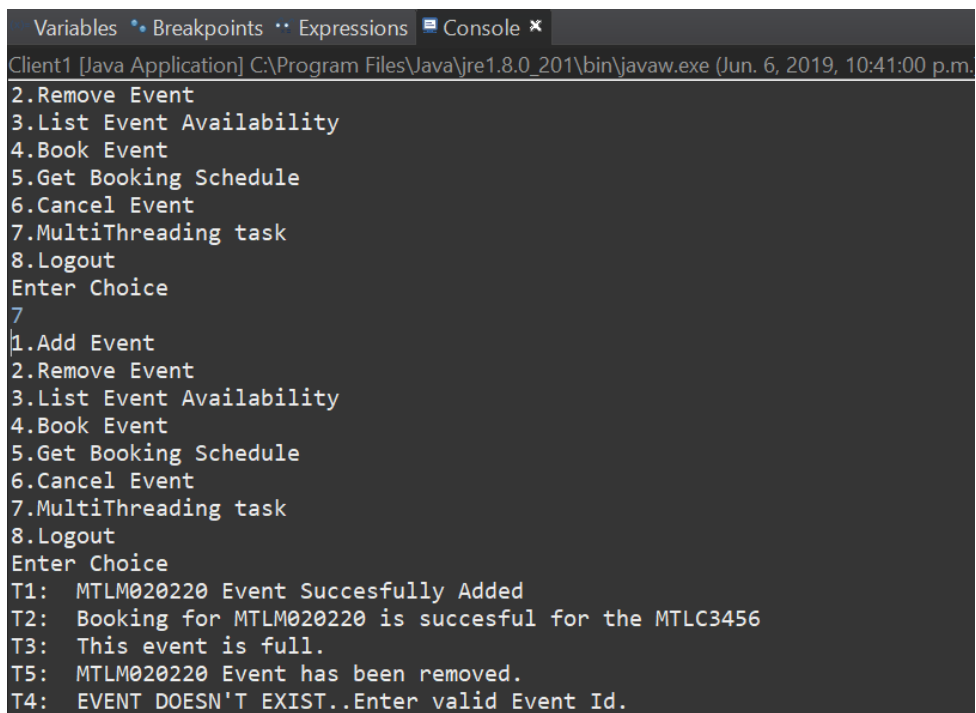
b) Unsuccessful

```
1.Book Event
2.Get Booking Schedule
3.Cancel Event
4.Swap Event
5.Logout
Enter your Choice=
4
Enter new Event ID:
TORA645156
Enter new Event Type:SEM
Enter old Event ID:
TORE040119
Enter old Event Type:TS
enter valid event id ,So swapping unsuccessful.
```

Multithreading:

Multithreading event will some static events to be run at run time.

These events are predefined and can be access by manager of any Server.

This will create an inter-Server connection and will provide different output as we run the multithreading task several times.

```
  Variables  • Breakpoints  •• Expressions  ■ Console ✖
Client1 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Jun. 6, 2019, 10:41:00 p.m.
2.Remove Event
3.List Event Availability
4.Book Event
5.Get Booking Schedule
6.Cancel Event
7.MultiThreading task
8.Logout
Enter Choice
7
1.Add Event
2.Remove Event
3.List Event Availability
4.Book Event
5.Get Booking Schedule
6.Cancel Event
7.MultiThreading task
8.Logout
Enter Choice
T1:  MTLM020220 Event Succesfully Added
T2:  Booking for MTLM020220 is succesful for the MTLC3456
T3:  This event is full.
T5:  MTLM020220 Event has been removed.
T4:  EVENT DOESN'T EXIST..Enter valid Event Id.
```