COMP6231 Assignment1

Name: Kaichen Zhang **ID: 40000160**

Techniques:

In this assignment:

I used **Java RMI(Remote Method Invocation)** to implement the communication between clients (passenger and manager) and servers (MTL,WDC,NDL).

I used **UDP** to implement the communication between servers.

I used **JSON(JavaScript Object Notation)** to store the records performing database.

I used **multithreading** technique to implement that multiple clients can act simultaneously.

I used **synchronization** technique to keep the integrity of data while modifying it, so the server can maximize the concurrency.

I used **serialization** technique to transform the message avoiding the middleware.

Design architecture:

serverInterface package:

DFRSInterface defines interface to be implemented in the servers. Methods including: book a new ticket, add a new flight, edit the existing flight record, getting available dates, and counting the number of tickets which has been booked.

records package:

Defining the flight record data structure, manager record data structure, passenger record data structure.

montrealServer, newdelhiServer, washingtonServer packages:

Containing the server class which implements the interface, providing remote method invocation by implementing java.rmi.server.UniCastRemoteObject and registering server object.

Containing the log.txt file tracking the action of clients.

Containing the flightRecords.json file to store the flight record in the server.

Containing the passengerRecords.json file to store the passengers who have booked a ticket.

clients package:

Containing ManagerClient class which can add a flight, get the number of tickets of three cities, or edit an existing flight.

Containing PassengerClient class which can test multithreading clients booking scenario or acting as a passenger to book an available ticket.

Containing ServerInfo class store the hash map of server, which is invoked by clients to locate server.

Containing Manager.json file to store existing managers.

Containing managerLogs.txt to keep the action of managers.

auxiliary package:

Containing DataIO class implements the reading and writing of data by using the JSON.

Containing RandomString class which can provide random string.

Containing WriteToLogFile class which is used to write the log to corresponding log file.

Data structure:

Array List: FlightRecordsList, PassengerRecordsList and ManagerRecordsList using array list to store record units.

Hash map: PassengerRecordsMap uses passengers first character of his last name as the key to store PassengerRecordsList. And serverMaps uses the server name as key to store the server port number.

Scenarios:

Firstly we should run the server of three different cities.

To simulate the manager, we run the ManagerClient class. In managers.json I set 9 mangers, three of each city, and we can do the operations like :
Add a new flight from departure city to destination city, the system will assign seats number for the flight, a random flight record ID.
Count the number of booked tickets of three cities.
Edit an existing flight by entering its flight record ID, modifying its date and number of seats.

To simulate the passenger, we run the PassengerClient class, and we can do the operations like:

Test multiple clients who are going to buy an available ticket (or can buy many tickets). Action of a specific passenger, who can input the full information to choose a valid flight from data base.

Important part/ Difficulty:

When I try to implement the hash map, I found that hash map is not thread safe and can not be written to the txt file. So I'm trying to use JSON as the data base, which is both good for writing hash map and performing the database.

To keep the integrity of data is important, so I used synchronized key word to the methods which modifies the data.

And implementing the Runnable interface can avoid the limitation of extending Thread class and can also share the resources with other threads.

References:

http://blog.csdn.net/iplayvs2008/article/details/16350919

https://github.com/google/gson

MultiThreading_Synchronization_Tut1[HARPREET SINGH NARULA]

Tutorial_2_JAVA_RMI[HARPREET SINGH NARULA]