COMP 6231

Distributed System Design

Assignment 1 Report

Distributed Class Management System (DCMS) using Java RMI

Submitted to: Professor Mohamed Taleb

By

Stallone Mecwan: 40161375

Jay Patel: 40163706

TABLE OF CONTENTS

Technique used

- We used **Java RMI (Remote Method Invocation)** to implement the communication between Manager/Clients and servers (MTL, LVL and DDO).
- We used **UDP** to implement the communication between servers for counting Records on each server
- We used **HashMap** to store the records of teachers and students.
- We used **multithreading** technique so that multiple clients can act simultaneously.
- We used **synchronization and semaphore** technique to keep the integrity of data while modifying it, so the server can maximize the concurrency.

Design architecture

CenterServer

Main interface file that defines all the operations that can be used by the managers(Clients of this system).

- createTRecord
- createSRecord
- getRecordCounts
- editRecord
- displayAllRecords
- displayRecord

MTL/LVL/DDOClass

These files implement the CenterServer and extend java.rmi.server.UniCastRemoteObject. All the functions defined in the CenterServer are implemented here. Some other functions are also defined here, i.e. validRecordID(), dateFormatChecker() (Date format checker for attribute statusDate of Student Record), setUpHashMap() (initialising HashMap here) and findRecord().

MTL/LVL/DDOServer

These files are responsible for getting the server up and running.

Server name, number and port are assigned here. The specific <server>Class's object is created here and binding of registry takes place.

As a starting point, some records are created here (by some default managerID like MTL/LVL/DDO0000) to fill the initial database. There are direct method invocations here, so we have also put validations in the MTL/LVL/DDOClass files.

MTL/LVL/DDOClient

These files contain client specific code. Here choices are provided to the user (manager) for which operation is to be performed. Validation of inputs is provided in these files.

ManagerClient

This file is the starting point of the application. This file asks for the mangerID whose prefix (first 3 characters) are used to run the server. If the managerID is MTL1234, then MTL Server will be up and running and so on for other two servers. Here, validation of managerID takes place, if it is not in the format MTL/LVL/DDO followed by four digits, then it won't work.

Record

Record class used to create TeacherRecord and StudentRecord (subclasses). It has two main attributes of Record, RecordID and Name. Also, a toString() method is defined to appropriately format and print the data for the Client.

TeacherRecord

This class is used for storing various details of Teachers. It stores the following details of a student: First Name, Last Name, Address, Phone number, Specialization, Location. Similar to the Record class, Student Record class has a toStringT() method which formats and prints the data for the client.

StudentRecord

This class is used for storing various details of Students. It stores the following details of a student: First Name, Last Name, Courses he is registered in, status, status date. Similar to the Record class, Student Record class has a toStringS() method which formats and prints the data for the client.

Log

This file is used to create log files and add information of all activities taking place in the application.

Server logs are named as MTL.txt, LVL.txt, DDO.txt and are saved in the root folder (src).
Client logs are saved as per the managerID used and are saved in the folder of "Logs". The path to the Logs folder has to be set in the ManagerClient file before running the application.

Before starting the application

We have used Intellij IDEA Community edition (version 2020.3.2) and JDK 11 for development of this application.

Open ManagerClient.java file. Replace the log file location that is **exactly highlighted in blue (till /Logs/) in the image below** to the location (path) where you have extracted the folder in your computer.

ManagerClient.java

```
String prefix = "";
if (input_string.length() > 3) {
    prefix = input_string.substring(0, 3);
}//substring containing first 3 characters

while (!check_valid_managerID(input_string, prefix)) {
    System.out.println("Manager ID not valid." + " Insert a valid ManagerID");
    input_string = in.next();
    if (input_string.length() > 3) {
        prefix = input_string.substring(0, 3);
    }
}
LogObject = new Log(
        fname: "D:/Study/Masters/COMP 6231/Assignments/Distributed Class Management System using Java RMI/src/Logs"
        + input_string + ".txt");
LogObject.logger.info( msg: input_string + " has logged onto the " + prefix + " Server");
```

Starting the application

1) Open command prompt
2) cd into the folder extracted
3) Run “javac ManagerClient.java”
4) start rmiregistry
5) run “java ManagerClient”

Data structures

Array List

All the details of students and teachers are stored in an array list

Hashmap

It consists of an id (string type) as the key and values are Array Lists that contain the records of students and teachers itself.

Test Scenarios

The ManagerClient is used to start the application. According to the ManagerID provided, the server will run (prefix of the ID).

After this, the manager will be allowed to use the operations displayed to him in a menu based way.

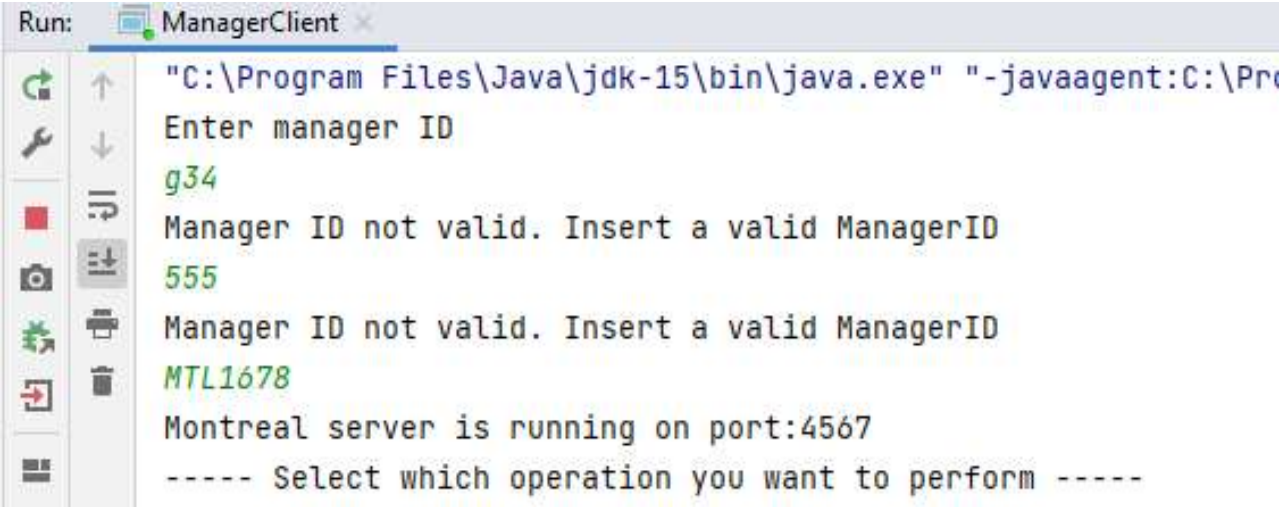
For the entire application we have created functions that test several inputs:

- validRecordID()
- hasNumbers()
- dateFormatChecker()
- hasAlpha()

These functions test the conditions on the input provided by the user to the system.

Some examples are as follows:

- Manager ID only valid in the format (MTL/LVL/DDO followed by four digits)



- First Name and Last name are validated in the same way. (Both can not be empty and can't contain numbers)

```
Enter First name:
St12
A name can not contain numbers, please insert valid input.
Stallone
Enter Last name:
we12
A name can not contain numbers, please insert valid input.
Mecwan
Enter Address:
```

- Phone number must be in this format only, otherwise it would not be accepted

```
Enter Phone number in the format (514-888-9999):
8866429090
Invalid phone number
Try another
514-898-0000
Specializtion courses:
```

- Only these values in status are accepted: (active, Active, Inactive, inactive)

```
Enter Status of the student:
act
Status inserted is not accepted, please enter (active or inactive).
inactive
Enter status date: (format - dd/mm/yyyy)
```

- Status date would be only accepted in this format dd/mm/yyyy:

```
Enter status date: (format - dd/mm/yyyy)
4/4/98
Invalid date format, please insert again in this format dd/mm/yyyy
04/16/1998
Invalid date format, please insert again in this format dd/mm/yyyy
```

The first case representing that 04/04/1998 is not provided

The second case representing that month (mm) cannot exceed 12.

- Location can be only MTL or LVL or DDO.

```
Enter Location:
Either of them (MTL, LVL, DDO)
mtl
Invalid location entered
Try another
edr
Invalid location entered
Try another
MTL
```

- Record ID must be in the format (TR/SR followed by five numbers)

```
3
Enter the recordID of the record you want to edit:
TR100
Invalid recordID, insert a valid ID
TR10TR10
Invalid recordID, insert a valid ID
TR10000
Enter name of the field you want to change:
Enter in this format: address, phone, specialization, location
```

- Below is a demonstration of creating teacher record with its log created

```
1
Enter First name:
ABC
Enter Last name:
XYZ
Enter Address:
Montreal, QC
Enter Phone number in the format (514-888-9999):
514-898-1234
Specializtion courses:
DSD
Enter Location:
Either of them (MTL, LVL, DDO)
DDO
```

```
Enter the recordID of the record to display:
TR10003
Record ID: TR10003  Name: ABC XYZ
Address: Montreal, QC    Phone: 514-898-1234
Specialization: DSD  Location: DDO
```

```
INFO: MTL1212 created a teacher record ID:TR10003
Jun. 13, 2021 7:29:35 P.M. ServerImplementation.MTLClass displayAllRecords
INFO: MTL1212 has displayed all records
Jun. 13, 2021 7:29:42 P.M. ServerImplementation.MTLClass displayRecord
INFO: MTL1212 has displayed record TR10003
```

- Below is a demonstration of edit record function with its log created:

```
3
Enter the recordID of the record you want to edit:
TR10003
Enter name of the field you want to change:
Enter in this format: address, phone, specialization, location
address
What value you want to change to:
Toronto, ON
```

```
5
Enter the recordID of the record to display:
TR10003
Record ID: TR10003  Name: ABC XYZ
Address: Toronto, ON     Phone: 514-898-1234
Specialization: DSD  Location: DDO
```

```
Jun. 13, 2021 7:33:15 P.M. ServerImplementation.MTLClass editRecord
INFO: MTL1212 edited a Teacher record ID:TR10003
Jun. 13, 2021 7:33:23 P.M. ServerImplementation.MTLClass displayRecord
INFO: MTL1212 has displayed record TR10003
```

- Below is a demonstration of creating student record with its log created

```
2
Enter First name:
LMN
Enter Last name:
OPQ
Name courses the student is registered in:
Algorithms
Enter Status of the student:
active
Enter status date: (format - dd/mm/yyyy)
12/12/2020
```

```
4
Record ID: TR10001
Name: Zeal Agrawal
Record ID: SR10000
Name: Stallone Mecwan
Record ID: SR10004
Name: LMN OPQ
Record ID: SR10002
```

```
Jun. 13, 2021 7:34:36 P.M. ServerImplementation.MTLClass createSRecord
INFO: MTL1212 created a student record ID:SR10004
Jun. 13, 2021 7:36:31 P.M. ServerImplementation.MTLClass displayAllRecords
INFO: MTL1212 has displayed all records
```

- Below is the demonstration of displayAllRecords function:

```
4. Display all records
5. Displaying a record
6. Display total Record count of all servers.
7. Exit.
4
Record ID: TR10001
Name: Zeal Agrawal
Record ID: SR10000
Name: Stallone Mecwan
Record ID: SR10002
Name: Meet Patel
Record ID: SR10003
Name: Bhoomi Sehra
Record ID: TR10000
Name: Teja Sehra
Record ID: TR10002
Name: Pavit Singh
Record ID: SR10001
Name: Vandit Thakkar
```

```
Jun. 13, 2021 7:23:52 P.M. ServerImplementation.MTLClass displayAllRecords
INFO: MTL1212 has displayed all records
```

- This is how we can get total number of records of ever server:

```
6. Display total Record count of all servers.
7. Exit.
6
Laval Server was not running, so running it now to get the record count.
Laval server is running on port:4568
DDO Server was not running, so running it now to get the record count.
Dollard des server is running on port:4569
MTL 7, LVL 12, DDO 12.
```

```
Jun. 13, 2021 7:22:03 P.M. ServerImplementation.MTLClass getRecordCounts
INFO: MTL1212 has retrieved the total record count
```

The application requires all the servers to start before getRecordCount method is used due to which if only one server is running, and client tries to retrieve the count, all the remaining servers are started.

Important part/ Difficulty

We initially used HashMap<Character, ArrayList> due to which everytime during creation of records, NullPointerExceptions were thrown.

This was solved by using a HashMap<String, ArrayList>.

The application uses synchronized methods for some operations, while using semaphore alongside. This was due to the fact that removing semaphore and replacing with synchronized blocks made the application run into infinite loop and the console hanged. This was the most difficult part to figure out.