CONCORDIA UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE AND SOFTWARE
ENGINEERING

COMP 6231, Fall 2018

Instructor: R. Jayakumar

ASSIGNMENT 1

Distributed Course Registration System (DCRS)
using Java RMI

BY:

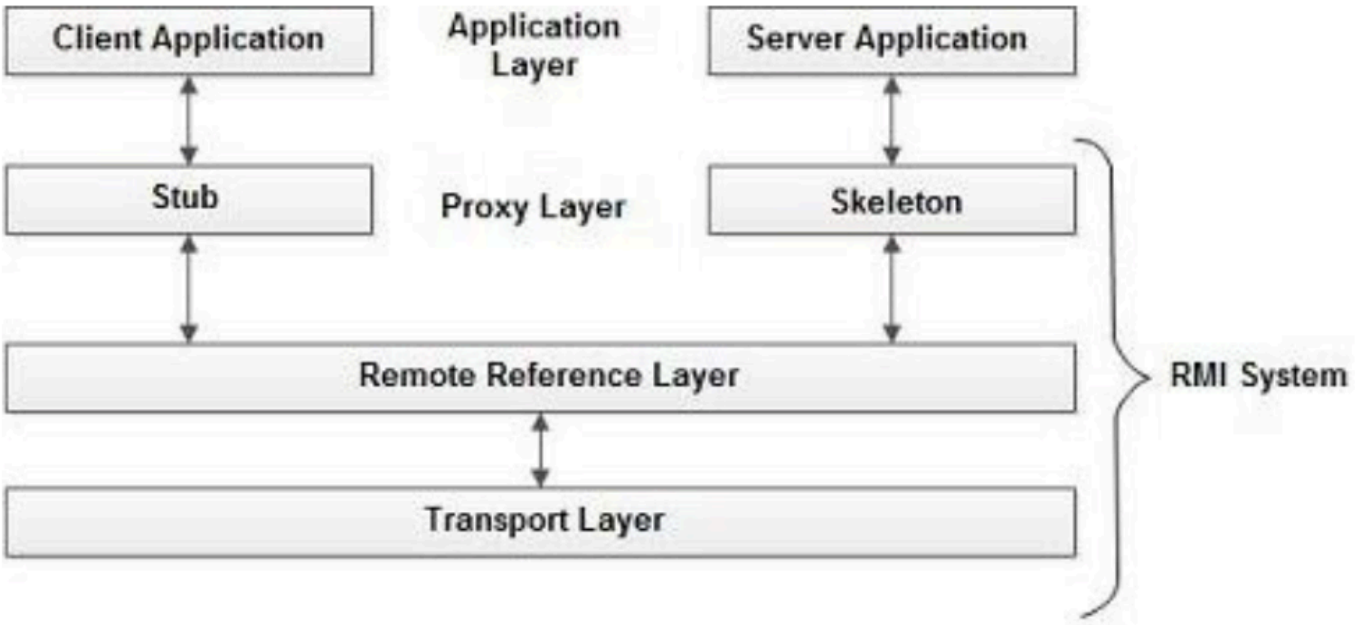
Name : Deep Patel

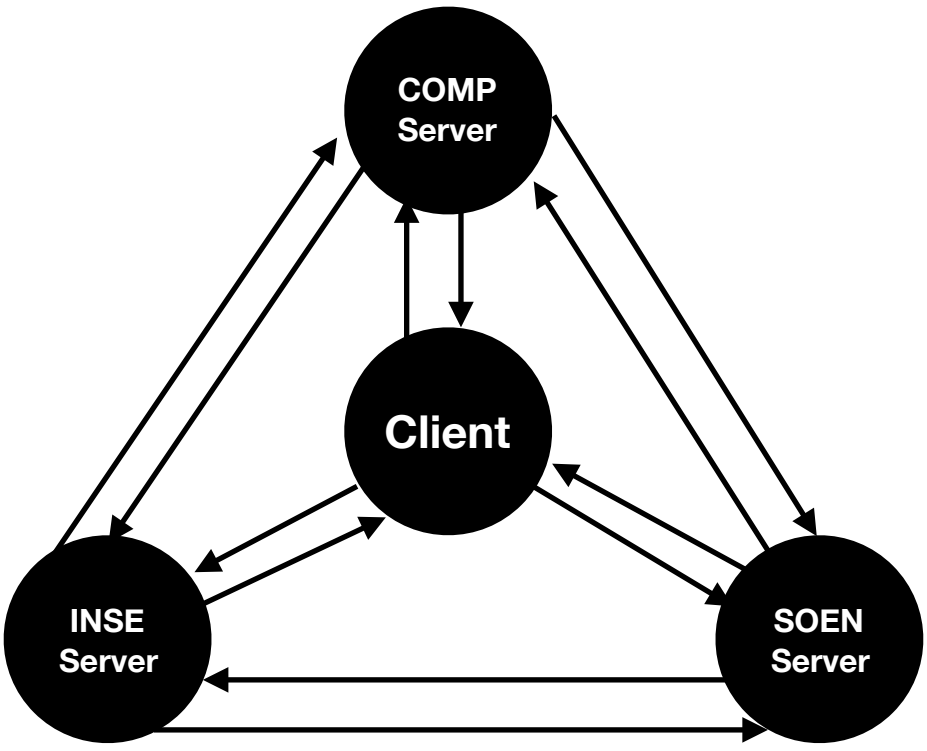
Student Id :

INDEX

1. Architecture : JAVA RMI

[1]





2. Data Structure :

HashMap:

1.

HashMap<String, Integer> compfallsubject = new HashMap<String, Integer>();

This Hashmap is used to store remain course capacity.

ex. compsummersubject , soenfallsubject , compwintersubject etc.

2.

HashMap<String, String> compfallstudent = new HashMap<String, String>();

This Hashmap is used to store students information.

Ex. soenfallstudent , insefallstudent etc.

3.

HashMap<String, ArrayList<String>> compadvisor = new HashMap<String,
ArrayList<String>>();

ArrayList<String> compfallsub = new ArrayList<String>();

ArrayList<String> compwintersub = new ArrayList<String>();

ArrayList<String> compsummersub = new ArrayList<String>();

This hashmap is used to store advisor's information as well as courses added by that advisor in all semesters. Here specific ArrayList stores the courses added by advisors in specific semester.

3. Methods:

addCourse (courseID, semester):

This method is advisor specific only. When an advisor invokes this method then course inserts into the appropriate hashmap. The advisor can only add his/her department's course only. the advisor must specify the capacity of the course. This method does not require inter-server communication.

removeCourse (courseID, semester):

This method is advisor specific only. When an advisor invokes this method then courses remove into the appropriate hashmap. The advisor can only remove his/her department's course only. This method does not require inter-server communication. It must remove this course in all students database who are enrolled in this course in any semester.

listCourseAvailability (semester):

This method is advisor specific only and It is inter-server communication method. When advisor invokes this method by entering semester, advisor's department server fetches course available for this semester with its remaining space as well as the server sends course availability request to another 2 servers and those 2 servers will do computation and return output to the 1st server. At the last, all departments courses for that semester with its availability are displayed on clients terminal.

enrolCourse (studentID, courseID, semester):

This method can invoke by both students and advisors. The user with his department's courses can directly enroll with his department's server but he can enroll another department's course through another specific servers only. Server request another server to enroll student in the specific course. After enrolling it replies a message of operation is successfully or not. Here in both cases, several computations are done as if student is already enrolled or not?, max department vise course (2), max term vise course (3) etc.

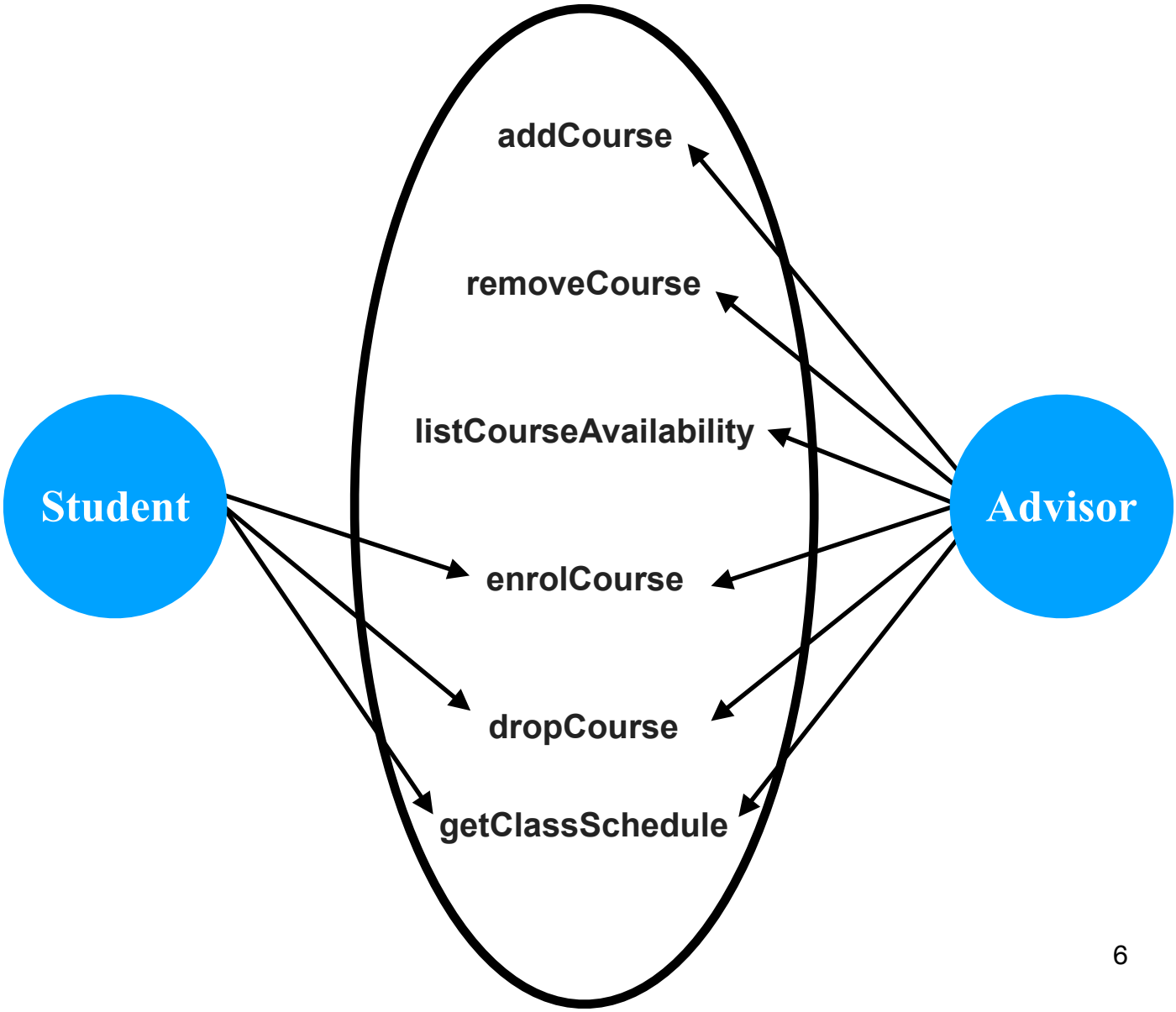
dropCourse (studentID, courseID):

This method can invoke by both students and advisors. If user wants to drop another department's course then server must request that another department's server to drop this student and that server replies whether the student successfully drops a course or not.

getClassSchedule (studentID):

When user invokes this method, it must print all subjects taken by student in all semester on the console. The server must request to another 2 servers to give their department's courses which were enrolled by the student and those other 2 servers return courses enrolled by the student.

4. Use Case Diagram:



5. Test cases:

1. addCourse 2. removeCourse

InterFace.java | ImpInterFace.java | Client.java | Server1.java | Server2.java | Server3.java

```
1  package asmt1_12;
2  import java.io.*;
3  import java.rmi.*;
4  import java.rmi.registry.LocateRegistry;
5  import java.rmi.registry.Registry;
6  import java.util.Date;
```

Console

```
Client (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-10.0.2.jdk/Contents/Home/bin/java (14-Oct-2018, 10:48:59 PM)
Enter Your ID : compa1111

1-> enrol course
2-> drop course
3-> get class schedule
4-> add course
5-> remove course
6-> list course availability

4
Enter semester : fall

Enter course id : comp5555

Enter course capacity : 4
course comp5555 added in fall term
wanna continue? 1/01

1-> enrol course
2-> drop course
3-> get class schedule
4-> add course
5-> remove course
6-> list course availability

5

Enter course id : comp5555
Enter semester : fall
course comp5555 remove in fall term
wanna continue? 1/0
```

3. listCourseAvailability

```
24          output.write("\n"+sdf.format(date)+"  ,");
25          if(n==1)
26          {
```

Console

```
Client (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-10.0.2.jdk/Contents/Home/bin/java (14-Oct-2018, 10:48:59 PM)
1-> enrol course
2-> drop course
3-> get class schedule
4-> add course
5-> remove course
6-> list course availability

6

Enter semester : fall
fall -  comp2222  6, comp3333  6, comp1111  3,   soen1111  4, soen2222  4, soen3333  9,  inse2222  4,

wanna continue? 1/0
```

Writable | Smart Insert | 17 : 5

4. enrolCourse

InterFace.java | ImpInterFace.java | Client.java | Server1.java | Server2.java | Server3.java | a.java

```
1  package asmt1_12;
2  import java.io.*;
3  import java.rmi.*;
4  import java.rmi.registry.LocateRegistry;
5  import java.rmi.registry.Registry;
6  import java.util.Date;
7  import java.text.DateFormat;
8  import java.text.SimpleDateFormat;
9  import java.util.Scanner;
10
11 //simport asmt1_11.InterFace;
12 import asmt1_11.ImpInterFace;
13 import asmt1_11.*;
14
```

Console

```
Client (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-10.0.2.jdk/Contents/Home/bin/java (14-Oct-2018, 10:51:15 PM)
Enter Your ID : comps1111

1-> enrol course
2-> drop course
3-> get class schedule

1

Enter course id : comp3333
Enter semester : fall
comps1111is successfully enrolled in comp3333
wanna continue? 1/01

1-> enrol course
2-> drop course
3-> get class schedule

3

Fall : comp1111 comp3333
winter : soen1111
summer :
wanna continue? 1/0
```

4. dropCourse 5. getClassSchedule

Console

```
Client (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk-10.0.2.jdk/Contents/Home/bin/java (14-Oct-2018, 10:
Enter Your ID : comps1111

1-> enrol course
2-> drop course
3-> get class schedule

2

Enter course id : comp3333
comps1111 is successfully drop in comp3333
wanna continue? 1/01

1-> enrol course
2-> drop course
3-> get class schedule

3

Fall : comp1111
winter : soen1111
summer :
wanna continue? 1/0
```

6. Important/difficult part :

One difficult part of this assignment is inter-server communication . I faced trouble many time while sending data from one server to another server. There was a small problem of how to send different information in one single message and how to separate such information like user id, semester etc in the received message on another server. I faced problems in maintaining server log files also. Most important part of the assignment is method invocation using right server.

7. References:

1. http://ecomputernotes.com/java/what-is-java/rmi-architecture/

What is RMI Architecture? / figure1