DISTRIBUTED SYSTEM DESIGN

COMP6231

Assignment 1

Distributed Course Registration System (DCRS)

Submitted By – Amandeep Singh (40052070)

Contents

Overview

The Distributed Course Registration System (DCRS) is a distributed system used by the university for adding courses to its schedule and enrolling students for these courses. The users of this system are the program advisor and students. For each department in the university, we have a dedicated server aka the COMP server (Computer Science Department), SOEN (Software Engineering Department) and the INSE server (Information System Security Department).

A user logged into the system is uniquely identified by its user id (e.g.: COMPA1452 represents a Computer Science department advisor, INSES3256 represents an Information System Security department student.) A student can enroll, drop courses and view his schedule whereas an Advisor perform all the student operations and even can add/drop courses.

Each server maintains it's internal in-memory database, basically composed of a HashMap, which it uses to store various information like the courses offered in a semester and the students enrolled etc.… A user interacts with the server using the Java RMI connection. The inter server communication is done using UDP sockets.

Each server maintains a log file for all the operation performed on it. A log file per user login is also maintained.

To the make the system more robust, after successful login, the user interaction is performed on a separate thread. Each logged in user communicates with its corresponding department server and performs necessary operations. Since multiple users access a server concurrently, so the proper synchronization of data is implemented in the code for thread safe communication. Moreover, the user input is case insensitive.

System Requirements

- Development Environment: Eclipse Photon
- Programming Language: Java 8
- Class Diagram Creation: Object Aid Eclipse plugin

Running the system

[1] Generate the stub '*rmic remoteObject.EnrollmentImpl*'
[2] Start the RMI server '*start rmiregistry*'
[3] Start the COMP_Server, SOEN_Server, INSE_Server
[4] Start the client program 'login'.

Working

Three different servers (COMP, SOEN, INSE) are started, which then start their own UDP servers for socket communication. Each of these servers bind their object reference in the registry using a unique name.

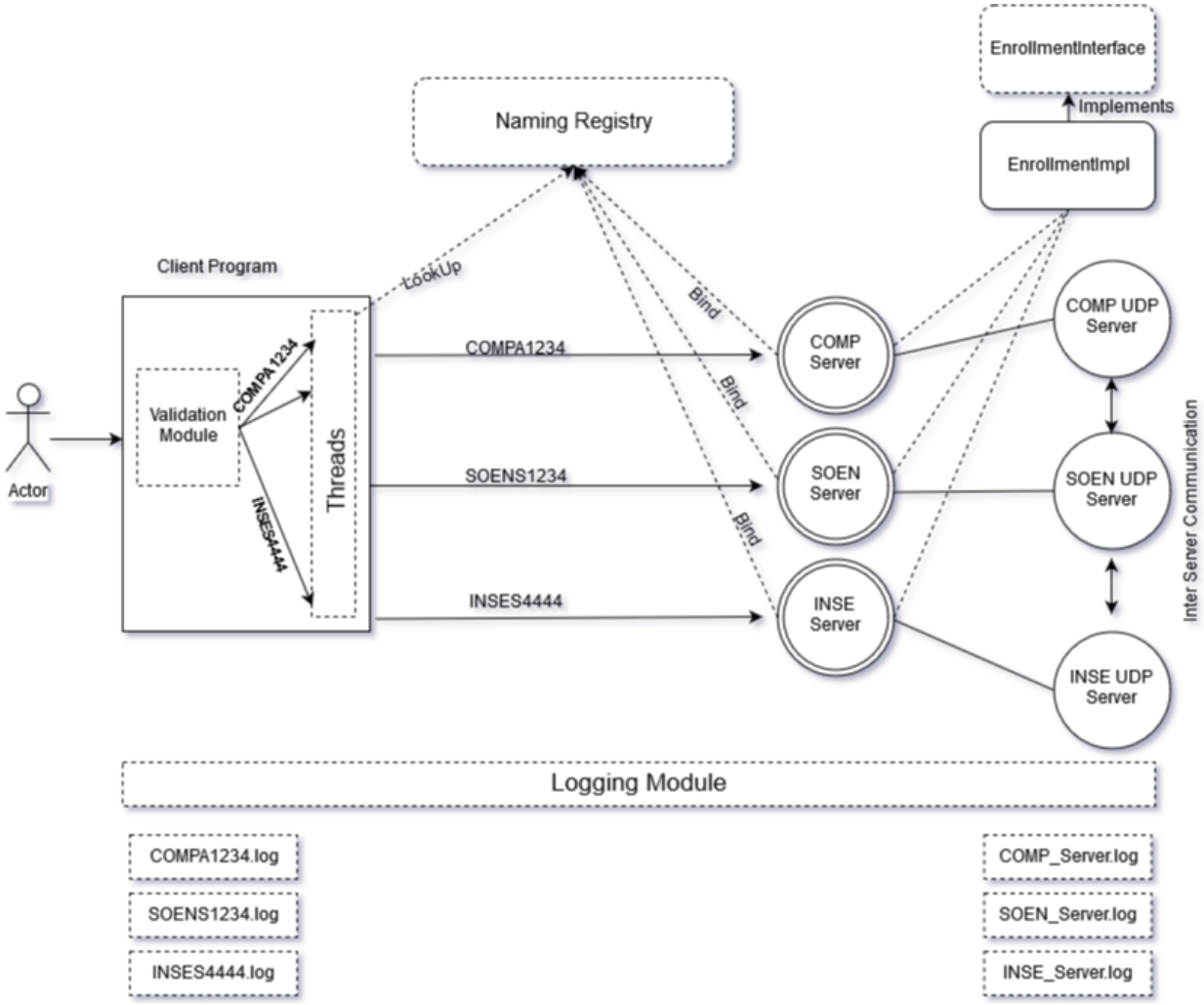
Apart from these servers, there's a client program which handles the user interaction. On the client program, a user log-in using its unique id. This unique id is validated to identify the department, role and id of the user.

After successful login, a separate thread is started to handle the requests from this user. Since a thread is a light weight process, so this make the system more robust and responsive.

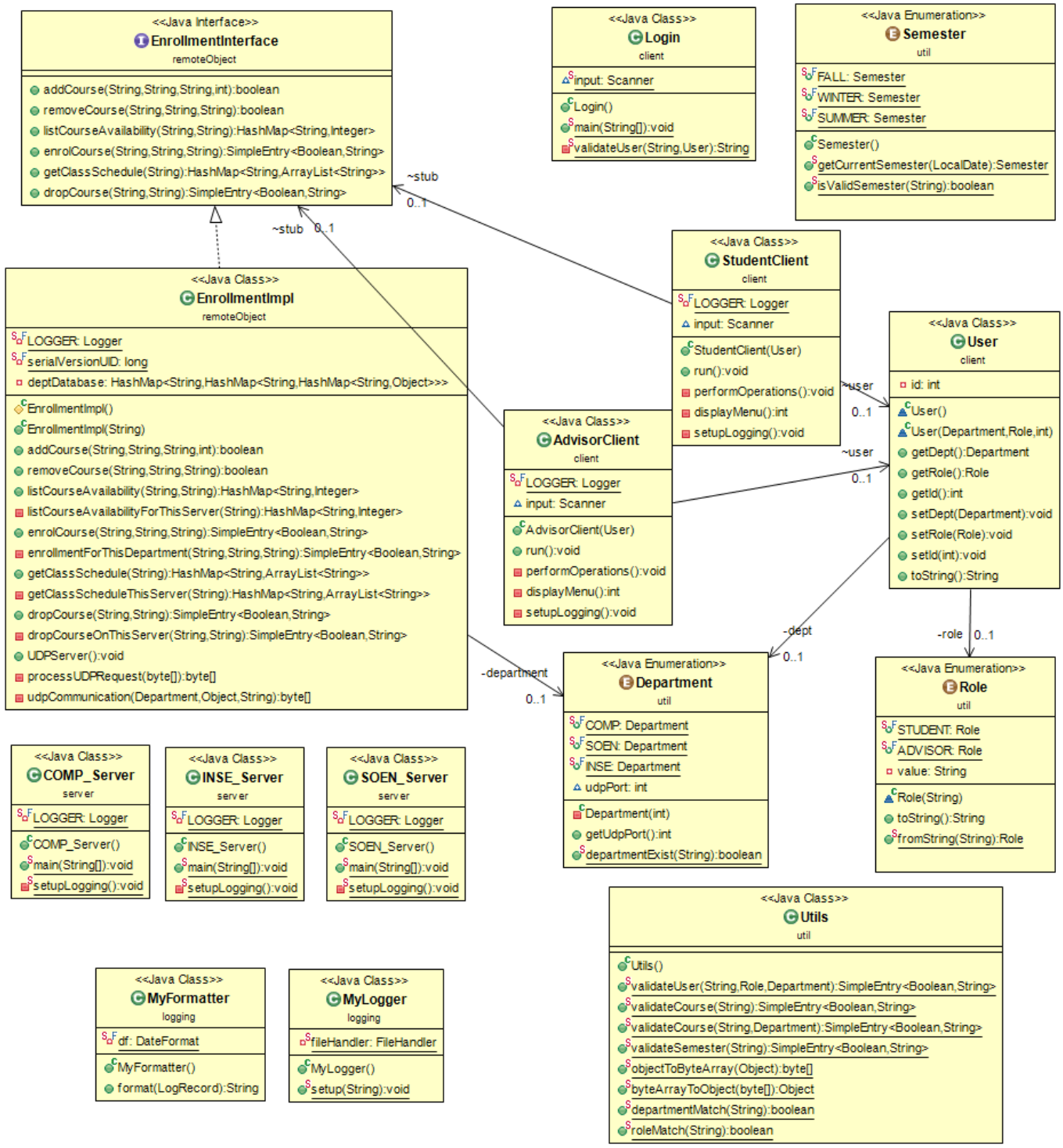
In the thread, we consult the registry to get reference of its corresponding server. Then after, based on the user type (advisor or student), a list of available operations is displayed.

In particular, a user only communicates with its corresponding server. But if the operation requires data from other departmental servers, then the user server makes a UDP request to the target server and gets the required data using a socket communication.

Architecture



Class Diagram



Key Features

1. Java 8 Lambdas

   The system code uses Java 8 *Lambdas*. The lambdas provide efficient way of writing long codes into just few lines, thus reducing the project size. Also, lambdas improve *readability*.

```
// get courses from the current department
if (deptDatabase.containsKey(semester)) {
    deptDatabase.get(semester).forEach(
            (course, courseDetails) -> result.put(course, (Integer) courseDetails.get(Constants.CAPACITY)
                    - (Integer) courseDetails.get(Constants.STUDENTS_ENROLLED)));
}
```

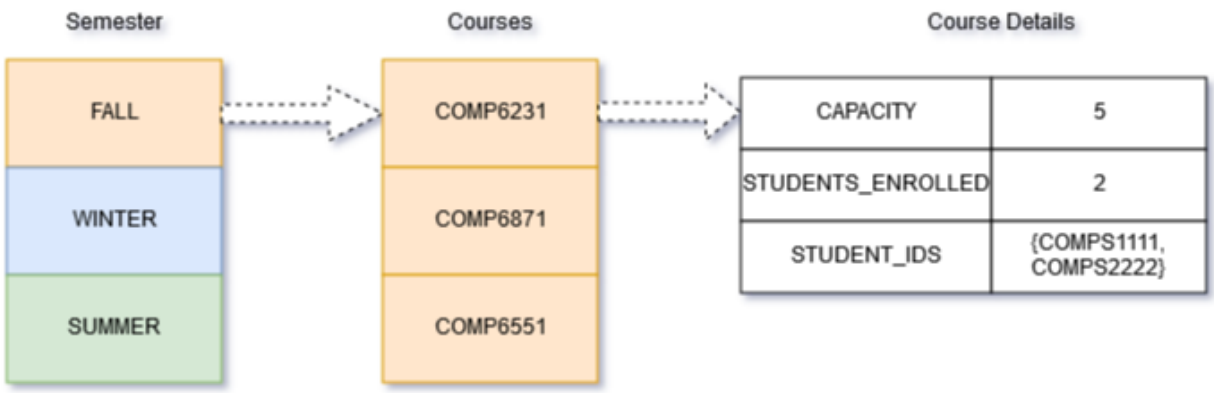
2. Data Type for parameters

   All the parameters passed are of type “String” and thus comply with the interface contract.

3. In - Memory Database

   Each server maintains it's in-memory database. This database is implemented as:

   $$HashMap < String, HashMap < String, HashMap < String, Object >>> \ deptDatabase$$



4. Case Insensitive

   The user input is case insensitive i.e. compa1234 is same as COMPA1234. This makes the system more user friendly.

5. Thread safe

   Since each user runs on a thread. So, the system needs to be able to handle concurrent requests and provide thread safety to its data. This is achieved by using ***"synchronized"*** blocks whenever there is a need to update the in-memory database.

```
//synchronizing the write operation to the in-memory database
synchronized(this) {
    this.deptDatabase.put(semester, courses);
}
```

6. Logging

   Custom logger is used to log all the messages. Each user operation is logged into the user specific log file (e.g. SOENS4444.log). The same is true for departmental servers too. (e.g. COMP_Server.log).

7. Custom Data Type for Socket Communication (UDP)

   Instead of simply storing String parameters to the byte array for socket communication, a custom HashMap is used.

$$HashMap < String, Object > \ data$$

   To facilitate this data type, the Utility methods "`objectToByteArray`" and "`byteArrayToObject`" methods are being used for conversion.

Test Cases

- Login

#	Operation	Input	Output	Result
1	Validate Department	ABSCA1236	Your department('ABSC') isn't recognized	PASS
2	Validate User Type	COMPY1234	Your role('Y') isn't recognized.	PASS
3	Validate Number	COMPA_qaw	Your id('_qaw') isn't recognized.	PASS
4	Case Insensitivity	COMPA1234 Compa1234	User can login	PASS

- User Specific Actions

ADVISOR

```
WELCOME TO DISTRIBUTED COURSE REGISTRATION SYSTEM
Please enter your ID : COMPA1234
Login Successful : COMPA1234
--------------------------------
|       Available Operations    |
--------------------------------
|1| Add a course.
|2| Remove a course.
|3| List Courses Availability.
|4| Enroll in Course.
|5| Get Class Schedule.
|6| Drop a Course.
|7| Quit.
Input your operation number : |
```

STUDENT

```
WELCOME TO DISTRIBUTED COURSE REGISTRATION SYSTEM
Please enter your ID : COMPS4444
Login Successful : COMPS4444
--------------------------------
|       Available Operations    |
--------------------------------
|1| Enroll in Course.
|2| Get Class Schedule.
|3| Drop a Course.
|4| Quit.
Input your operation number : |
```

- Advisor Add Course

#	Operation	Input	Output	Result
1	Valid Semester	AUTUM	AUTUM isn't valid semester.	PASS
2	Validate Course Id	comp12345	Seems to be an invalid course (length not equal to 8).	PASS
3	Adding other department course	SOEN6441	You are not authorized for this department('SOEN').	PASS
4	Adding already added course	-	FAILURE = COMP6231 is already offered in FALL semester.	PASS
5	Adding same course to other semester		SUCCESS - Course Added Successfully	PASS

- Advisor Remove Course

#	Operation	Input	Output	Result
1	Valid Semester	rainy	rainy isn't valid semester.	PASS
2	Validate Course Id	comp51486	Seems to be an invalid course (length not equal to 8).	PASS
3	Course Doesn’t exist	Comp6232	FAILURE - comp6232 is not offered in FALL semester.	PASS
4	Remove other department course	SOEN6441	You are not authorized for this department('SOEN').	PASS

- Advisor List Course Availability

#	Operation	Input	Output	Result
1	Valid Semester	HOT	HOT isn't valid semester.	PASS

- Student Enroll Course

#	Operation	Input	Output	Result
1	Valid Semester	COLD	COLD isn't valid semester.	PASS
2	Validate Course Id	comp51486	Seems to be an invalid course (length not equal to 8).	PASS
3	Course Doesn’t exist	COMP6541	COMP6541 is not offered in FALL semester.	PASS
4	Already Enrolled	-	FAILURE - COMPS4444 is already enrolled in COMP6231.	PASS
5	Enrolled in 3 courses for this semester	-	COMPS4444 is already enrolled in 3 courses [COMP6231, COMP6478, COMP6985] for this FALL semester.	PASS
6	Enrolled in 2 off department courses in all the semesters	-	COMPS4444 is already enrolled in 2	PASS

			out-of-department courses.	

- Advisor get Class Schedule

#	Operation	Input	Output	Result
1	Valid StudentId	COMPS258745	Seems to be an invalid id(length not equal to 9).	PASS
2	Student is of his/her department	SOENS5142	You are not authorized for this department('SOEN').	PASS

- Student Drop Course

#	Operation	Input	Output	Result
1	Valid StudentId	COMPS258745	Seems to be an invalid id(length not equal to 9).	PASS
2	Valid course id	comp51486	You are not authorized for this department('SOEN').	PASS

References

[1] Custom Logging: http://www.vogella.com/tutorials/Logging/article.html

[2] Java RMI tutorial: https://www.tutorialspoint.com/java_rmi