DISTRIBUTED SYSTEM DESIGN

COMP6231

Assignment 2

Distributed Course Registration System (DCRS) using Java IDL (CORBA)

Submitted By – Amandeep Singh (40052070)

Contents

Overview

This assignment is the continuation of the previous work started on the Distributed Course Registration System (DCRS). Here we need to implement the DCRS in the CORBA programming language using the Java IDL. The system remains completely same as the previous one apart from one additional functionality added to swap the course.

The swap course functionality is available to both the students and their advisors. Using it, a student who is already enrolled in one course can drop it and get enrolled in some other course, it can be within the same department or cross department. The advisor can also perform this operation on behalf of the student. For the cross-department course swap, the inter server communication is done through the UDP sockets.

The swap course operation basically requires 3 operations:

1. Check availability of the new course.
2. Drop the old course
3. Enroll in the new course.

All these operations need to be performed atomically and in this specific order i.e. if any one operation fails all the previous operations needs to be rolled back. Thus, assuring atomicity of swap operation is one of the development areas which requires significant work in this project.

Since, this project is built upon the previous project we did in assignment 1. So, we need to find out any efficient and fast way to refactor the previous project from Java RMI to CORBA which requires minimum code changes. This was also one of the key challenging areas to work on in this project.

System Requirements

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

Running the Project

- Start the ORB from the command prompt: "*start orbd -ORBInitialPort 1050*"
- Add the cmd arguments, "*-ORBInitialPort 1050 -ORBInitialHost localhost*" through eclipse to the following files:
    - COMP_Server.java
    - SOEN_Server.java
    - INSE_Server.java
    - Login.java
- Start the COMP_Server, SOEN_Server, INSE_Server
- Start the client program 'login'.

Working

The working for this project is exactly similar to the previous one, apart from the additional functionality of swap course.

For swapping a course, we need to make sure the atomicity of this operation, particularly when the swap is inter-department. For this, I have used the *Reentrant Lock* [1]. The reentrant lock offers more features than the normal synchronized blocks, such as fairness of the lock etc.

We have the following 3 cases for swap course:

- *Drop*: Same Department; *Enroll*: Same Department
  This case is simple to handle, we just need to synchronize the code for drop and enroll methods.
- *Drop*: Different Department; *Enroll*: Same Department
  From the synchronized swap course method, we check the course availability in the same department. Since I'm using the same reentrant lock, so when a lock is obtained after checking the course availability in the swap course method, it also locks the enroll user method, so that none other than the current thread can execute the enroll user method. All other threads requesting access to enroll user method are put on wait.
  Once the new course availability is confirmed, a UDP request is made to the other department, for dropping the old course. After this, the user is enrolled in the new course and the lock is released.
- *Drop*: Same Department; *Enroll*: Different Department
  This case is quite tricky to handle, since we need to check the new course availability on different server, then drop the course on this server and then finally enroll into the new course on the other department.
  To handle is situation, atomically, from the swap course, we directly move the other department server through the UDP request. Then once on the other server, we perform the operations as: check for the course availability, if available, obtain the lock, make a UDP request (from within the UDP call), to drop the course. If successful, then the student is enrolled into the new course on this different department, the lock is released, and the result is returned back in the form of reply to the 1st UDP call. This way, the swap operation is handled atomically.

Architecture

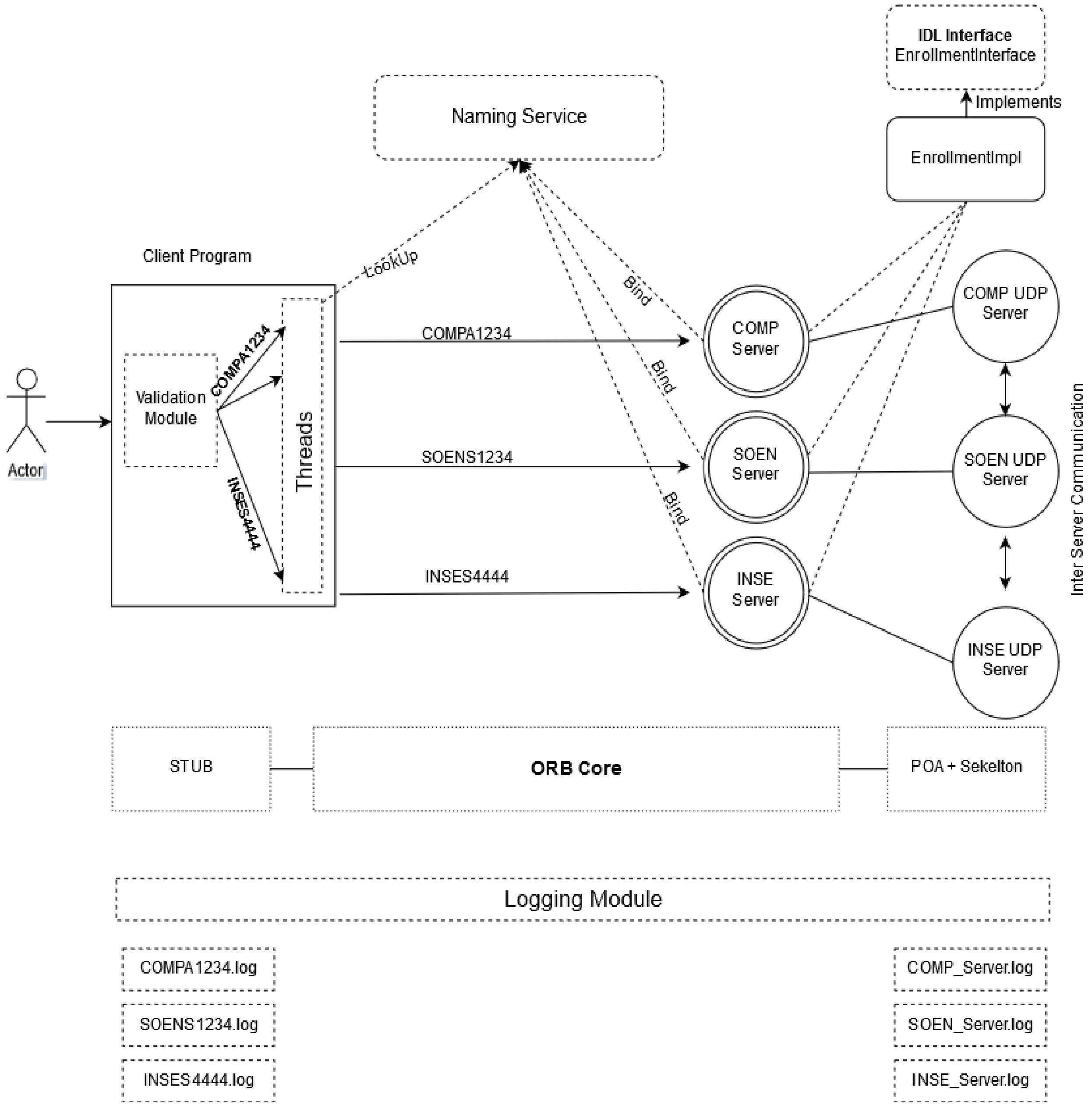
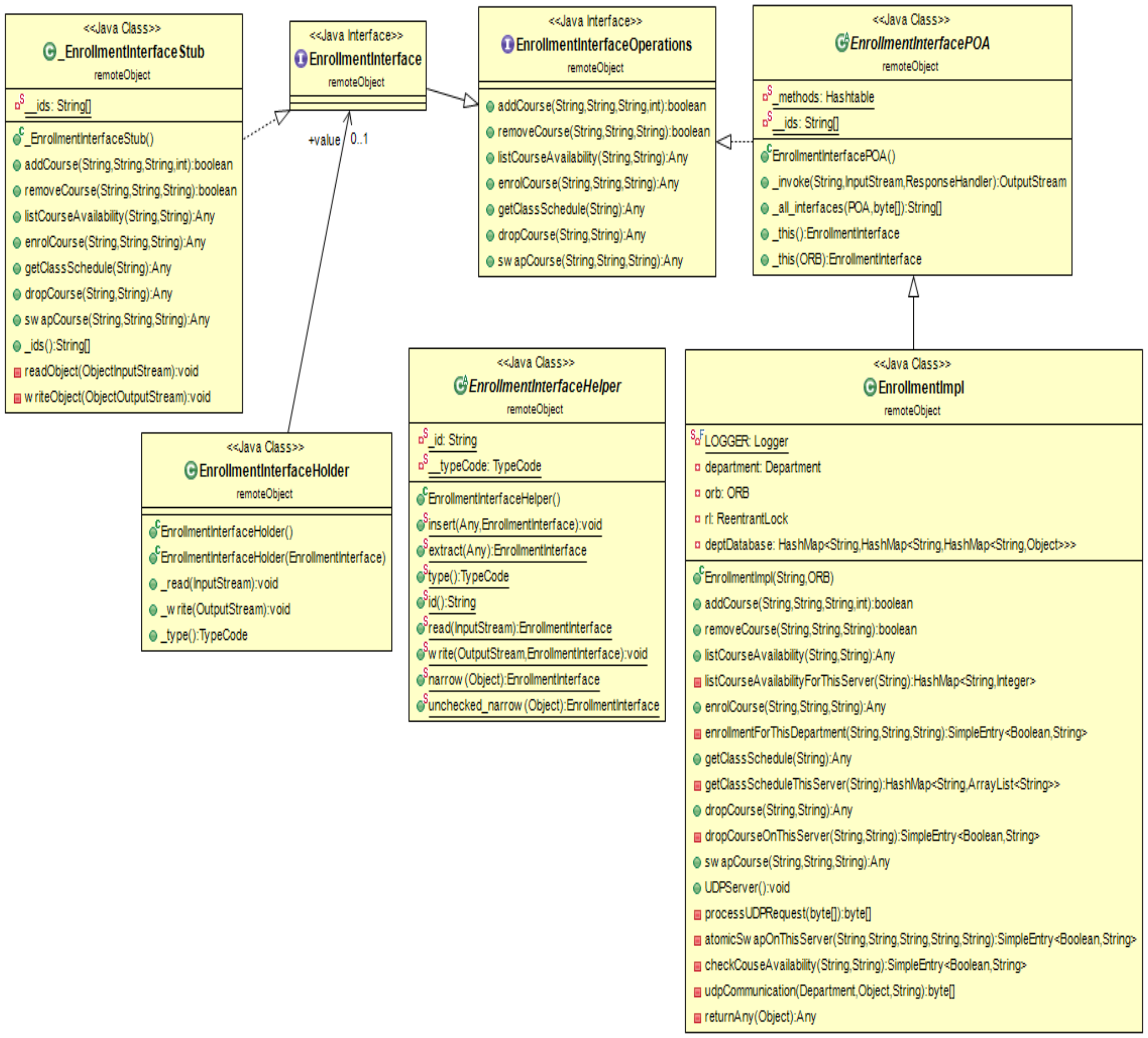


Figure 1 Project Architecture

Class Diagram for IDL Interfaces and Implementation Classes



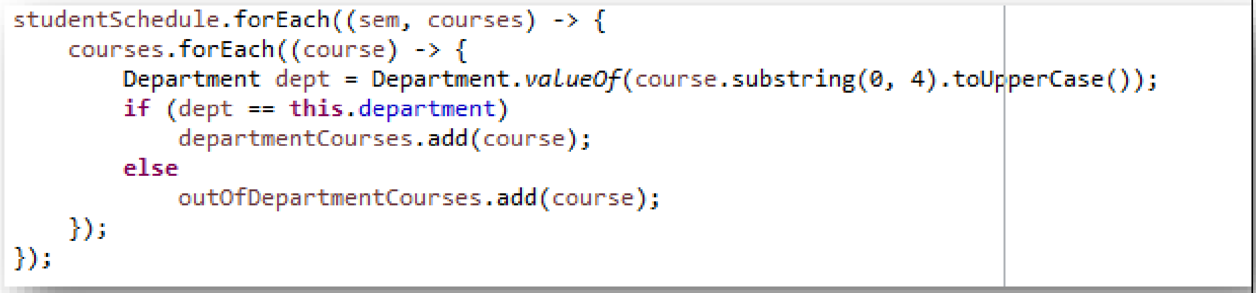
<<Java Class>>
_EnrollmentInterfaceStub
remoteObject
__ids: String[]
_EnrollmentInterfaceStub()
addCourse(String,String,String,int):boolean
removeCourse(String,String,String):boolean
listCourseAvailability(String,String):Any
enrolCourse(String,String,String):Any
getClassSchedule(String):Any
dropCourse(String,String):Any
swapCourse(String,String,String):Any
_ids():String[]
readObject(ObjectInputStream):void
writeObject(ObjectOutputStream):void
<<Java Interface>>
EnrollmentInterface
remoteObject
+value
0..1
<<Java Interface>>
EnrollmentInterfaceOperations
remoteObject
addCourse(String,String,String,int):boolean
removeCourse(String,String,String):boolean
listCourseAvailability(String,String):Any
enrolCourse(String,String,String):Any
getClassSchedule(String):Any
dropCourse(String,String):Any
swapCourse(String,String,String):Any
<<Java Class>>
EnrollmentInterfacePOA
remoteObject
_methods: Hashtable
__ids: String[]
EnrollmentInterfacePOA()
_invoke(String,InputStream,ResponseHandler):OutputStream
_all_interfaces(POA,byte[]):String[]
_this():EnrollmentInterface
_this(ORB):EnrollmentInterface
<<Java Class>>
EnrollmentInterfaceHolder
remoteObject
EnrollmentInterfaceHolder()
EnrollmentInterfaceHolder(EnrollmentInterface)
_read(InputStream):void
_write(OutputStream):void
_type():TypeCode
<<Java Class>>
EnrollmentInterfaceHelper
remoteObject
_id: String
__typeCode: TypeCode
EnrollmentInterfaceHelper()
insert(Any,EnrollmentInterface):void
extract(Any):EnrollmentInterface
type():TypeCode
id():String
read(InputStream):EnrollmentInterface
write(OutputStream,EnrollmentInterface):void
narrow(Object):EnrollmentInterface
unchecked_narrow(Object):EnrollmentInterface
<<Java Class>>
EnrollmentImpl
remoteObject
LOGGER: Logger
department: Department
orb: ORB
rl: ReentrantLock
deptDatabase: HashMap<String,HashMap<String,HashMap<String,Object>>>
EnrollmentImpl(String,ORB)
addCourse(String,String,String,int):boolean
removeCourse(String,String,String):boolean
listCourseAvailability(String,String):Any
listCourseAvailabilityForThisServer(String):HashMap<String,Integer>
enrolCourse(String,String,String):Any
enrollmentForThisDepartment(String,String,String):SimpleEntry<Boolean,String>
getClassSchedule(String):Any
getClassScheduleThisServer(String):HashMap<String,ArrayList<String>>
dropCourse(String,String):Any
dropCourseOnThisServer(String,String):SimpleEntry<Boolean,String>
swapCourse(String,String,String):Any
UDPServer():void
processUDPRequest(byte[]):byte[]
atomicSwapOnThisServer(String,String,String,String,String):SimpleEntry<Boolean,String>
checkCouseAvailability(String,String):SimpleEntry<Boolean,String>
udpCommunication(Department,Object,String):byte[]
returnAny(Object):Any

Figure 2 Class Diagrams

Key Features

Apart from the features of the previous project, following new features are added.

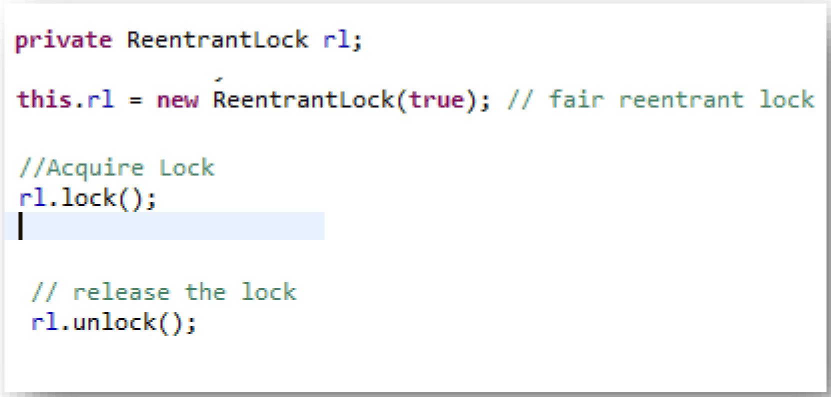
1. Java 8 Lambdas
   As in the previous project, I'm continuing using the Java 8 lambdas.



```java
studentSchedule.forEach((sem, courses) -> {
    courses.forEach((course) -> {
        Department dept = Department.valueOf(course.substring(0, 4).toUpperCase());
        if (dept == this.department)
            departmentCourses.add(course);
        else
            outOfDepartmentCourses.add(course);
    });
});
```

Figure 3 Java Lambdas

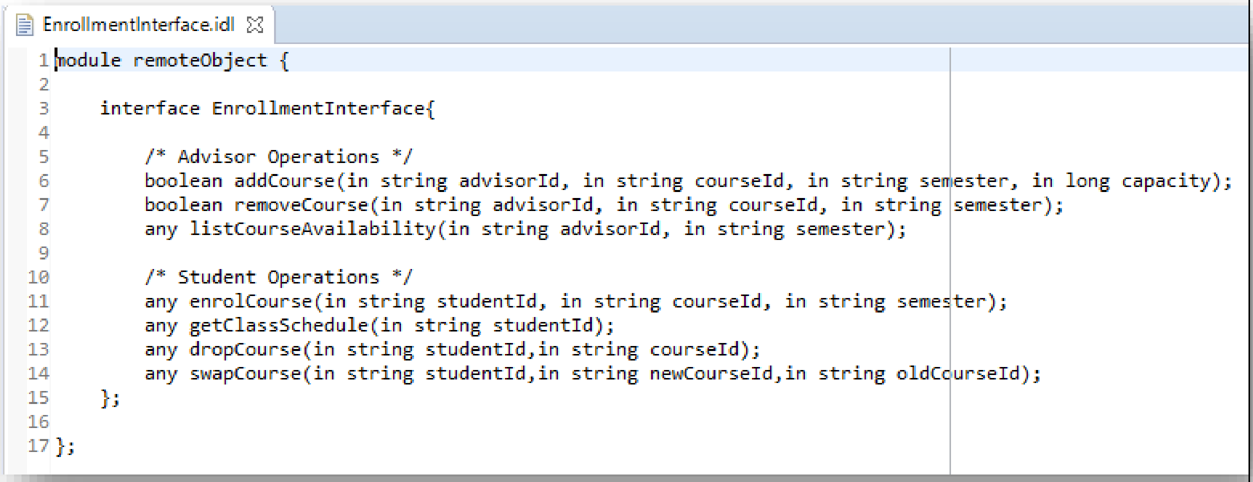
2. Reentrant Lock [*1*]
   As explained in the "*Working*" section (of this report), proper synchronization of operations is obtained using the Reentrant lock [*1*].



```java
private ReentrantLock rl;

this.rl = new ReentrantLock(true); // fair reentrant lock

//Acquire Lock
rl.lock();


 // release the lock
 rl.unlock();
```

Figure 4 Reentrant lock usage

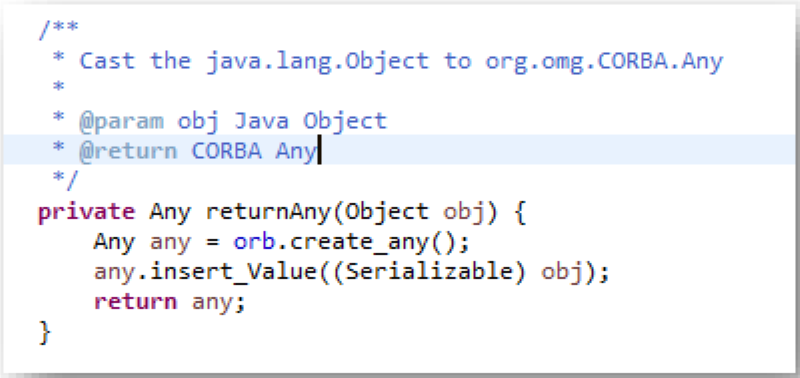
3. <u>IDL Definition</u>



```
EnrollmentInterface.idl
module remoteObject {

    interface EnrollmentInterface{

        /* Advisor Operations */
        boolean addCourse(in string advisorId, in string courseId, in string semester, in long capacity);
        boolean removeCourse(in string advisorId, in string courseId, in string semester);
        any listCourseAvailability(in string advisorId, in string semester);

        /* Student Operations */
        any enrolCourse(in string studentId, in string courseId, in string semester);
        any getClassSchedule(in string studentId);
        any dropCourse(in string studentId,in string courseId);
        any swapCourse(in string studentId,in string newCourseId,in string oldCourseId);
    };

};
```

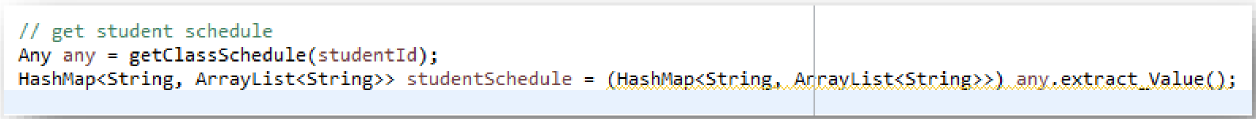
Figure 5 IDL Definition

Proper precautions have been taken to minimize the refactoring work required for switching from Java RMI to CORBA. In the CORBA IDL, apart from the primitive data type, I'm using *'any'* datatype which can easily be casted to previous data type used in assignment 1 as below:



```
/**
 * Cast the java.lang.Object to org.omg.CORBA.Any
 *
 * @param obj Java Object
 * @return CORBA Any
 */
private Any returnAny(Object obj) {
    Any any = orb.create_any();
    any.insert_Value((Serializable) obj);
    return any;
}
```

Figure 6 Conversion from Java Object to CORBA Any



```
// get student schedule
Any any = getClassSchedule(studentId);
HashMap<String, ArrayList<String>> studentSchedule = (HashMap<String, ArrayList<String>>) any.extract_Value();
```

Figure 7 Conversion from CORBA Any to Java Object

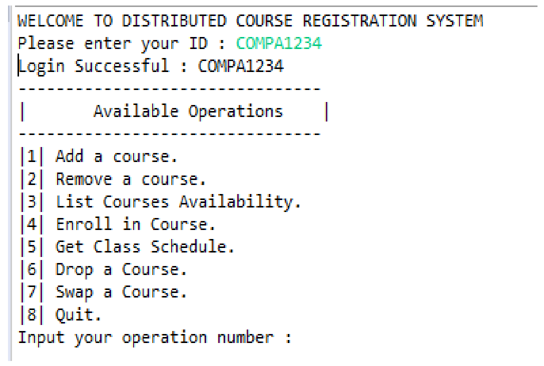
Test Cases

- 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

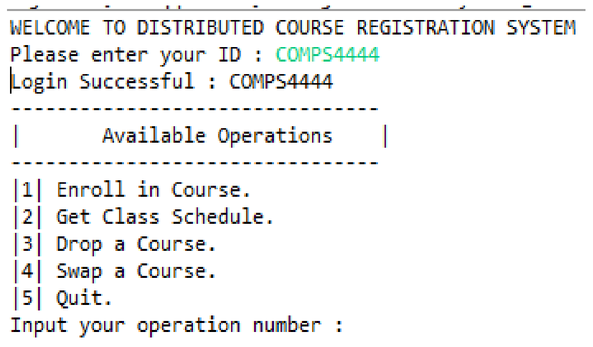
- User Specific Actions

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| Swap a Course.
|8| 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| Swap a Course.
|5| 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
3	Course not offered	COMP9854	COMP9854 isn't offered by the department yet.	PASS

- Swap Course

#	Operation	Input	Output	Result
1	Valid StudentId	COMPS258745	Seems to be an invalid id(length not equal to 9).	PASS
2	Valid New Course id	comp658578	Seems to be an invalid course (length not equal to 8).	PASS
3	Valid old course id	comp658578	Seems to be an invalid course (length not equal to 8).	PASS
4	Student not enrolled in the course to drop	comp6231	COMPS4444 is not enrolled in COMP6231	PASS
5	Student already enrolled in the new course	comp6441	COMPS4444 is already enrolled in COMP6441	PASS

6	New Course is not offered in that semester	(inse6231, inse6441)	INSE6441 is not offered in FALL semester.	PASS
7	New Couse is elective & the student already have enrolled in 2 elective subjects.	-	COMPS4444 is already enrolled in 2 out-of-department courses	PASS
8	New Course is full, old course should not be dropped.	-	COMP6231 is full.	PASS
9	Swap course offered in different semester (SHOULD NOT HAPPEN)	-	-	PASS
10	Swap a course which the student is already enrolled in another semester (SHOULD NOT HAPPEN)	-	-	PASS

References

1. Reentrant Lock: https://www.geeksforgeeks.org/reentrant-lock-java/
2. CORBA Hello World Example: http://www.ejbtutorial.com/corba/tutorial-for-corba-hello-world-using-java