Distributed System Lab Assignment

Distributed Course Registration System

Syeed Abrar Zaoad
Roll: 23
Department of Computer Science and Engineering
University of Dhaka

Tauhid Tanjim
Roll: 58
Department of Computer Science and Engineering
University of Dhaka

Submitted To Professor:
Dr. Upama Kabir
and
Dr. Mosarrat Jahan
Department of Computer Science and Engineering
University of Dhaka

November 3, 2019

Contents

1 Introduction

In this assignment,we have designed a distributed course registration system using RMI Java. RMI(Remote Method Invocation) is a distributed system mechanism that allows programmers to use Java programming language and development environments, so that objects on different computers can communicate with each other in a distributed network.A thread can call the method on a remote object.For transparency on the client and server side, using stubs and skeletons remote object is implemented.

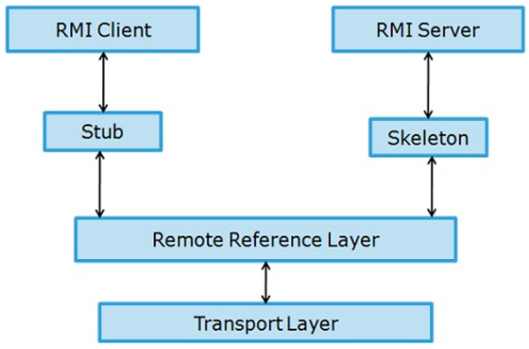


Figure 1: Architecture of RMI

In precise, client calls a remote method using stub. Stub is accountable for constructing and sending the message consisting the name of a method and the marshalled parameters. Now, skeleton receives the message,then unmarshals parameters and invokes the desired method on the server. The skeleton marshals the given value (or exceptions) with the message and sends it to client stub. The stub reassembles the return parcel and sends it to the client.Pictorial representation has been given in figure 1.

2 UML diagram

Here is the Unified Modeling Language(UML) diagram of this assignment.

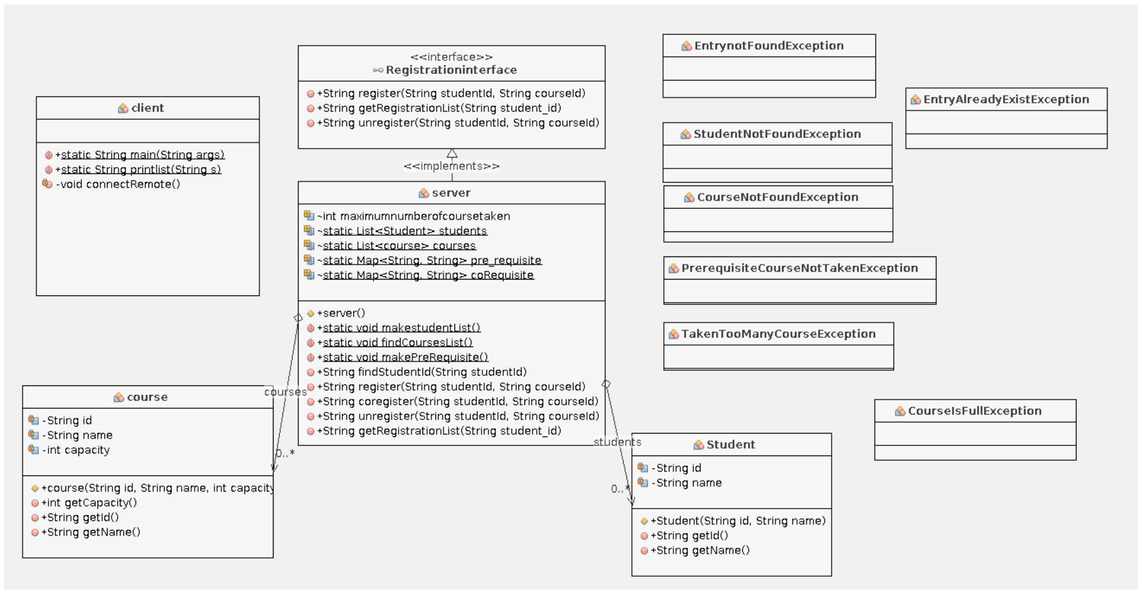

client
+static String main(String args)
+static String printlist(String s)
-void connectRemote()
<<interface>>
Registrationinterface
+String register(String studentId, String courseId)
+String getRegistrationList(String student_id)
+String unregister(String studentId, String courseId)
<<implements>>
server
~int maximumnumberofcoursetaken
~static List<Student> students
~static List<course> courses
~static Map<String, String> pre_requisite
~static Map<String, String> coRequisite
+server()
+static void makestudentList()
+static void findCoursesList()
+static void makePreRequisite()
+String findStudentId(String studentId)
+String register(String studentId, String courseId)
+String coregister(String studentId, String courseId)
+String unregister(String studentId, String courseId)
+String getRegistrationList(String student_id)
course
-String id
-String name
-int capacity
+course(String id, String name, int capacity
+int getCapacity()
+String getId()
+String getName()
courses
0..*
students
0..*
Student
-String id
-String name
+Student(String id, String name)
+String getId()
+String getName()
EntrynotFoundException
EntryAlreadyExistException
StudentNotFoundException
CourseNotFoundException
PrerequisiteCourseNotTakenException
TakenTooManyCourseException
CourseIsFullException

Figure 2: UML

3 Data Structure

We have used array,arraylist,map for our system.

- **static List<Student>students = new ArrayList <Student>()** This arraylist is used to store student name,id (information) from Student class which is structured in that class.

- **static List<Course>courses = new ArrayList<Course>()**
  This arraylist is used to store course name,course id,course capacity (information) from Course class which is structured in that class.

- **static Map<String,String>pre_requisite = new HashMap<String,String>()**
  This hashmap is used to map the course id(key) with pre-requisite course id(value) to check pre-requisite course taken or not.

- **static Map<String,String>coRequisite = new HashMap<String,String>()**
  This hashmap is used to map the course id(key) with co-requisite courses id(value) to check co-requisite courses needed to be taken or register.

4 Methods

The methods,we needed to implemented in this assignment have been described below.

- **boolean Register(String studentId, String courseId):** This method is used to check whether a student can register for a course or not. It takes Student ID and course ID as input and checks if the student has fulfilled all types of criteria or not for that course.Then if yes, we register the student for that course.

- **boolean unRegister(String studentId, String courseId):** This method is used to check a student is already registered for a course or not. It takes student ID and course ID as a input parameter to check if the student has registered in that course or not.Then if yes, we unregister the student from that course.

- **String gettRegistrationList (String studentID):** This method takes studentID as a input parameter and return the student ID and the courses that he/she is taken in the current semester.

5 Exceptions

The exceptions needed to handle in this assignment are:

- **TakenTooManyCourseException:**If a student has exceeds his limit of taking course then this exception occurs.
- **EntrynotFoundException:**While unregistering if the course not found then this exception is given.
- **CourseIsFullException:**If the capacity of the particular course exceeds this exception is given.
- **PrerequisiteCourseNotTakenException:**While registering if the prerequisite course not found for a particular course this exception occurs.
- **StudentNotFoundException:**While registering,if the student is not found then this exception occurs.
- **CourseNotFoundException:**While registering,if the course is not found then this exception occurs.
- **EntryAlreadyExistException:**While registering,if the registration has already done,then this exception occurs.

6 Database:

6.1 Course information

courses	capacity	pre-requisite	co-requisite
CSE 1201	5	CSE 1101	CSE 1102
CSE 2101	5	CSE 1201	None
MATH 4102	6	MATH 3101	MATH 4101
CSE 2202	3	CSE 2101	None
CSE 3101	3	CSE 2203	CSE 2204
STAT 3205	2	None	None
CSE 1101	5	None	CSE 1201
CSE 1102	5	CSE 1101	None
MATH 3101	5	none	none
Math 4101	5	none	MATH 4102
CSE 2204	3	none	CSE 3101

6.2 Student information

Student name	student id
kashob Kumar Roy	201901
Tauhid Tanjim	201958
Musfiq Shohan	201905
Syeed Abrar Zaoad	201923
Pranto Hasan	201927
Mashrur Rashik	201929
sadia Afrin meem	201902

6.3 Course information

Course name	Course id
algorithm 2	SE 1201
Fundamental of Programming	CSE 2101
numerial analysis	MATH 4102
Programing Language	CSE 2202
Object Oriented Programming	CSE 3101
Probability and Statistics	STAT 3205
algorithm 1	CSE 1101
Introduction to Computer Science	CSE 1102
Integration and Differentiation	MATH 3101
Linear Algebra	Math 4101
Software Engineering	CSE 2204

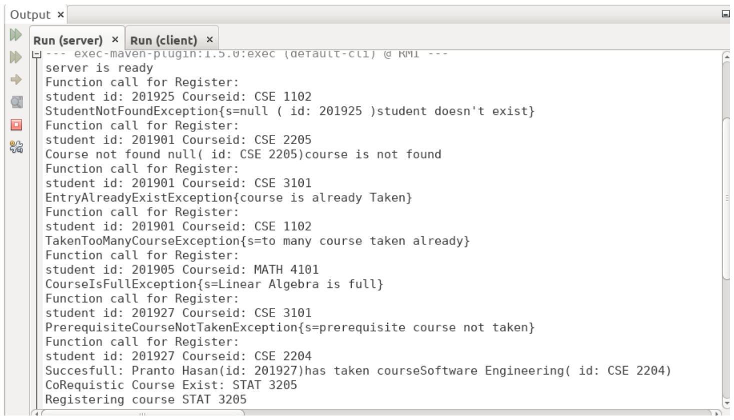
6.4 Already Taken Courses

Student Id	Course Id
201901	CSE 1101, CSE 1201,Math 4101,CSE 1101
201905	CSE 1101
201902	STAT 3205
201923	CSE 4101
201927	none
201929	CSE 2101

7 Output

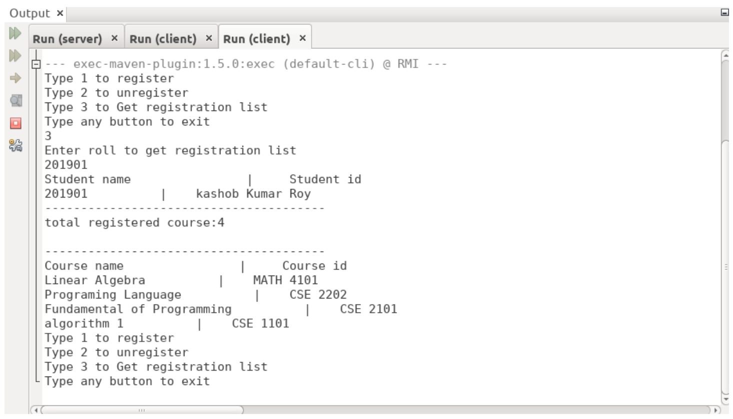
```
Function call for Register:
student id: 201925 Courseid: CSE 1102
StudentNotFoundExceptions=null ( id: 201925 )student doesn't exist
Function call for Register:
student id: 201901 Courseid: CSE 2205
Course not found null( id: CSE 2205)course is not found
Function call for Register:
student id: 201901 Courseid: CSE 3101
EntryAlreadyExistExceptioncourse is already Taken
Function call for Register:
student id: 201901 Courseid: CSE 1102
TakenTooManyCourseExceptions=to many course taken already
Function call for Register:
student id: 201905 Courseid: MATH 4101
CourseIsFullExceptions=Linear Algebra is full
Function call for Register:
student id: 201927 Courseid: CSE 3101
PrerequisiteCourseNotTakenExceptions=prerequisite course not taken
Function call for Register:
student id: 201927 Courseid: CSE 2204
Succesfull: Pranto Hasan(id: 201927)has taken courseSoftware Engineering(
id: CSE 2204)
CoRequistic Course Exist: STAT 3205
Registering course STAT 3205
Function call for corequisite course register:
student id: 201927 Courseid: STAT 3205
Succesfull: Pranto Hasan(id: 201927)has taken courseProbability and Statis-
tics( id: STAT 3205)
Function call for Register:i
student id: 201905 Courseid: MATH 3101
Succesfull: Musfiq Shohan(id: 201905)has taken courseIntegration and Dif-
ferentiation( id: MATH 3101)
Function call for Unegister:
student id: 201901 Courseid: CSE 3101
Succesfully unregistered kashob Kumar Roy(id: 201901)has unregisterd courseOb-
ject Oriented Programming( id: CSE 3101)
```

CoRequistic Course Exist: CSE 2204
Unregistering course CSE 2204
Function call for getRegistrationList:
student id: 201901



```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ RMI ---
server is ready
Function call for Register:
student id: 201925 Courseid: CSE 1102
StudentNotFoundException{s=null ( id: 201925 )student doesn't exist}
Function call for Register:
student id: 201901 Courseid: CSE 2205
Course not found null( id: CSE 2205)course is not found
Function call for Register:
student id: 201901 Courseid: CSE 3101
EntryAlreadyExistException{course is already Taken}
Function call for Register:
student id: 201901 Courseid: CSE 1102
TakenTooManyCourseException{s=to many course taken already}
Function call for Register:
student id: 201905 Courseid: MATH 4101
CourseIsFullException{s=Linear Algebra is full}
Function call for Register:
student id: 201927 Courseid: CSE 3101
PrerequisiteCourseNotTakenException{s=prerequisite course not taken}
Function call for Register:
student id: 201927 Courseid: CSE 2204
Succesfull: Pranto Hasan(id: 201927)has taken courseSoftware Engineering( id: CSE 2204)
CoRequistic Course Exist: STAT 3205
Registering course STAT 3205
```

Figure 3: Server Output



```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ RMI ---
Type 1 to register
Type 2 to unregister
Type 3 to Get registration list
Type any button to exit
3
Enter roll to get registration list
201901
Student name                |     Student id
201901          |    kashob Kumar Roy
---------------------------------------
total registered course:4

---------------------------------------
Course name               |     Course id
Linear Algebra          |    MATH 4101
Programing Language          |    CSE 2202
Fundamental of Programming          |    CSE 2101
algorithm 1          |    CSE 1101
Type 1 to register
Type 2 to unregister
Type 3 to Get registration list
Type any button to exit
```

Figure 4: Client Output