Web Service Implementation of the Distributed Course Registration System (DCRS)

Classes

Course: an instance of this class stores information about a course object which contains capacity, course id, enrolled students, description and etc.
This class implements Serializable as we need to serialize these data and send them over servers.

Student: an instance of this class stores information about a student object. Each student has id, name, 3 different list of enrolled courses including summer, fall and winter, the data type considered for these data are Hashmaps. Id must start with prefix+S.
Also we store capacity per semester and the total capacity for courses that one student can have.

StudentClient: This class invokes the student's operations at the associated server as necessary.
All the necessary methods for this class are implemented as below:

- `public static void enrolCourse (String studentID, String courseID, String semester)`
- `public static String[] getClassSchedule(String studentID)`
- `public static void dropCourse(String studentID, String courseID)`
- `public static void swapCourse (String student_id, String oldCourse_id, String newCourse_id, String semester)`

Based on the student and course id, it invokes method from different server implementation using web services.
For each operation, the StudentClient finds the required information about the associated server from the central repository and invokes the corresponding operation.
For swapCourse() we are using inter server communication that is done using UDP/IP sockets.

Synchronizing is very critical for implanting this method, as at the same time several different clients might want to enroll in a course using enroll method, this might lead to concurrency issues.
So first of all we have to make sure that we have thread –safety while using this method and also we have to somehow manage to make sure that the user can be enrolled in the new course and be dropped from the old course correctly.
For this purpose, I am using Java synchronization definition for my methods and handling connecting different users through multithreading.

In this assignment we have 3 servers which are explained in details below.

AdvisorClient: This class invokes the advisor's operations at the associated server. All the necessary methods for this class are all those that we already have for student client plus the following:

- `public static void addCourse(String courseID, String semester, int capacity, String desc)`
- `public static void removeCourse(String courseID, String semester)`
- `public static void listCourseAvailability (String semester)`

For each operation, the AdvisorClient finds the required information about the associated server from the central repository and invokes the corresponding operation.
For listCourseAvailability we are using inter server communication that is done using UDP/IP sockets.

***About UDP**
Computers in networks communicate through two ways TCP and UDP.
UDP (User Datagram Protocal) sends independent packets of data called datagrams from one computer to another. It is an unreliable method of communication meaning that unlike TCP, it is not a connection-based protocal.
Datagram is an independent, self-contained message sent over the network whose arrival, arrival time and content are not guaranteed.
Ping command which is used to test the connectivity of the network is an example of an application that uses UDP. UDP is said to be a datagram-based

communication. In UDP the datagram packets contains the port number of destination and UDP routes the packet to the right application.

CompPublisher, SoenPublisher, InsePublisher:

These are our main server classes.
We have to assign different ports to our web services in order to be able to run them appropriately, then the url address for each class is as following:

COMP Services: **http://127.0.0.1:10000/COM**
INSE Services: **http://127.0.0.1:20000/INSE**
SOEN Services: **http://127.0.0.1:30000/SOEN**

If we run this publisher classes, we can browse their wsdl configuration which includes of all the xml tags and our service settings such as targetNamespace or service name. We use this information later to be able to create our service and provide an instance of it for further usage.

Each class has also our configuration for UDP connection.

ImplCOMP, ImplSOEN and ImplINSE

These classes contain the main part of implementation of each servers by implementing our Inteface class.

InterfaceService

```
@WebService
@SOAPBinding(style = SOAPBinding.Style.RPC)
public interface InterfaceService
{

    public String addCourse (String courseID, String semester, int capacity, String
desc);
    public String removeCourse (String courseID, String semester);
    public String[] courseAvailability (String semester);

    public String enrolCourse (String studentID, String courseID, String semester);
    public String[] getClassSchedule (String studentID);
    public String dropCourse (String studentID, String courseID);
    public String addUserToEnrolledUser(String name, String id ) throws IOException;

    public boolean hasCapacity(String courseID);
}
```

Data Structures

Two categories of data structures are mainly used in this assignment:

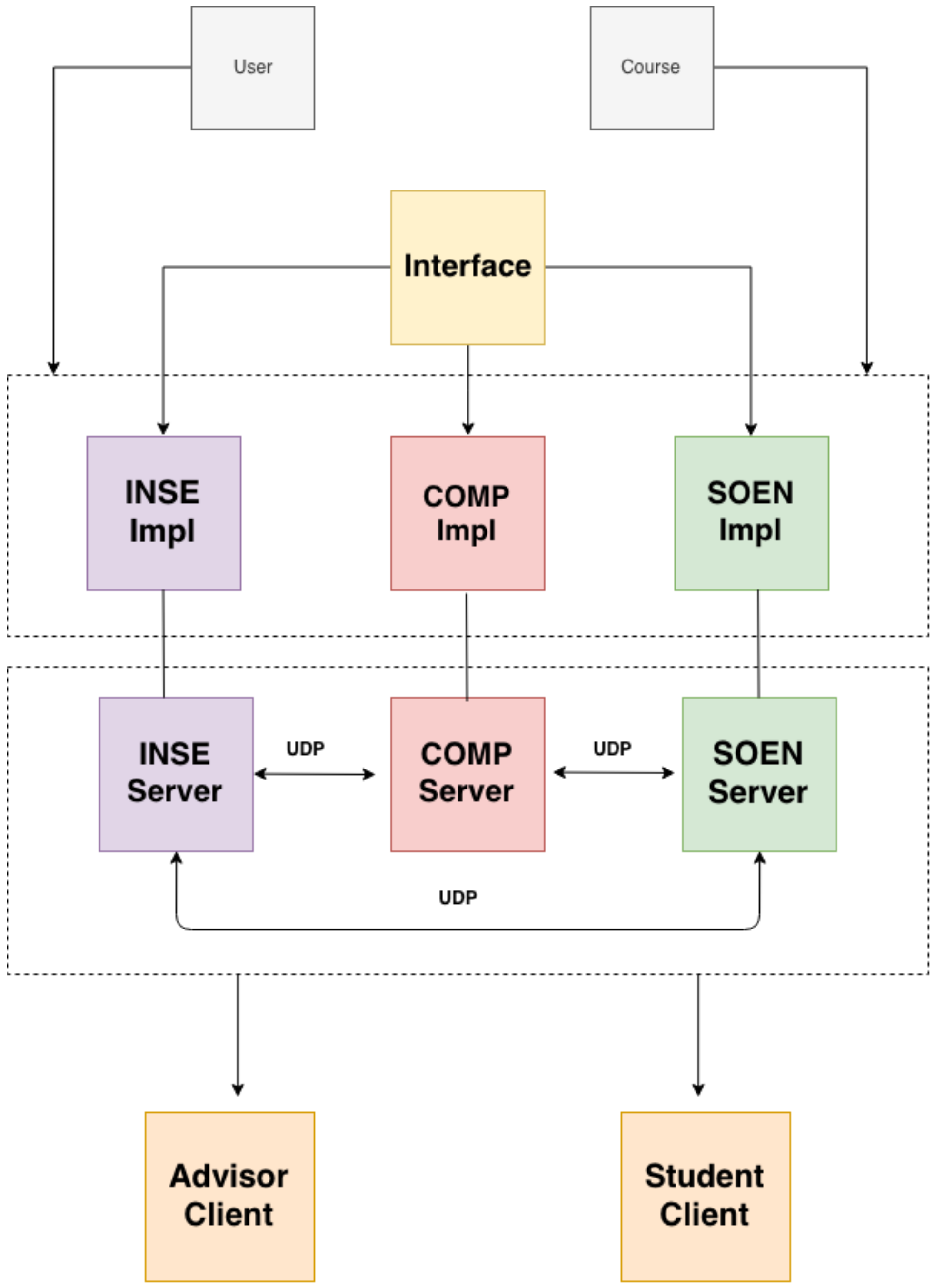
- **Hashmap <key, value>:**
- **Hasmap < key, Hashmap<key,value>**
- **List [value]**
- **String []**

in order to be able to retrieve our instance of our classes like course and user, sometimes we had to store a key as an id and pass an object of that class as the value of this data structures as you can notice it while reading the programs.

Challenges

Most important/difficult part of this assignment for me was to first understand using Web services and the communication between servers and clients, then managing the synchronization part which we are supposed to design a server that maximizes concurrency means that use proper synchronization that allows multiple users to correctly perform operations on the same or different records at the same time. I could manage to overcome this issue using Java Synchronization and Multi-threading.
Besides, since java web services doesn't return Arraylist types, I had to change all those data types to String[] instead which at first was a bit confusing since I didn't realize why my clients can not show any results from servers.

Classes Diagram


User
Course
Interface
INSE Impl
COMP Impl
SOEN Impl
INSE Server
UDP
COMP Server
UDP
SOEN Server
UDP
Advisor Client
Student Client