REPORT

Name: AKSHAT BISHT
Student ID: 40053762
COMP 6231 : Assignment 1

Objective: The assignment want us to create a student and advisor Distributed Course Registration System (DCRS) using Java RMI.

System Design: To implement this system there are use of 6 classes.

1. Client side communication class: Client.java
2. Computer Science Server class: COMP_server.java
3. Information Security Server class: INSE_server.java
4. Software Engineering Server class: SOEN_server.java
5. A client information data structure class: data_structure.java
6. A course information class: courses_availability.java

1.Client.java

This class is the user interface class. The user inputs their User Identification Number. The main function searchs for the character sequences-"COMP" or "INSE" or "SOEN". If the user ID doesn't contain any of the following sequences then it throws an error. If it does contain any of those strings it then it sends it to the their respective servers to check if the user ID exists or not.
If the ID is of a student the code moves to the student_menu() function, otherwise if it an advisor ID then it will move to the advisor_menu() function.
The student menu contains the 4 options to add a course, drop a course, check course availability and to logout.
The advisor menu contains 4 options, to add a new course, to delete an existing course and to check the availability of all the courses available.

For the addition, deletion and dropping of courses, the format of input is the same:

(COURSE NAME) (COURSE CODE) (COURSE TERM)

Example:
COMP 6231 FALL
SOEN 6441 WINTER

2. Servers: COMP_Server.java, INSE_Server.java, SOEN_Server.java

The server classes hold all the data of that particular department. They process the queries coming from the Client side or from either of the other 2 servers. They all contain a list(Linked List) of courses available along with their terms. This list is a list of a user defined data structure(courses_availability.java) which contains information about the course: term(fall, winter, summer) and course capacity.
When a student of a department is registering or deregistering for courses of the same department, then there is only clinet and server communication, but there is no server to server communication.
The servers inter communicate with each other when the students of one course are registering or de-registering for courses of different departments.

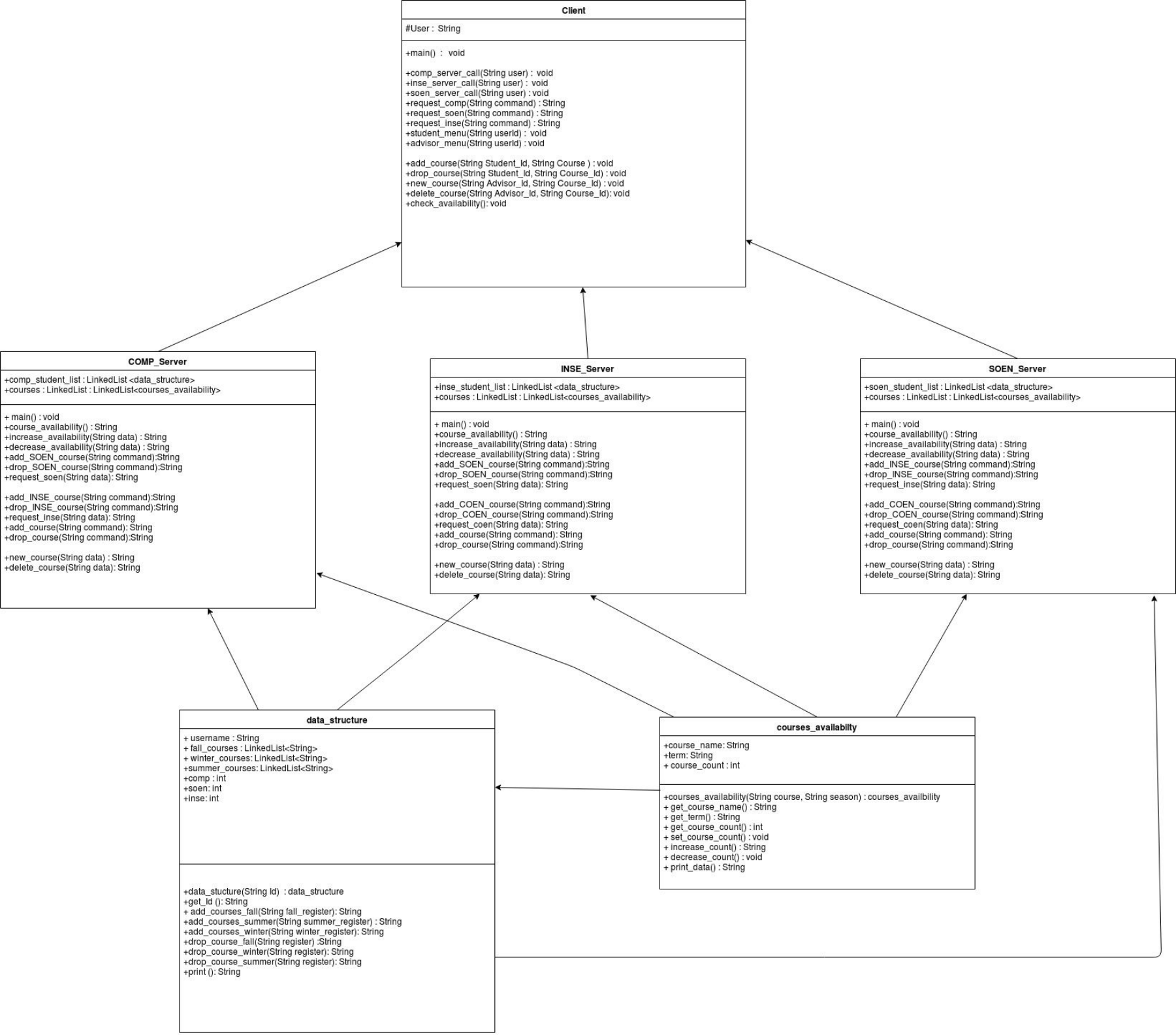
3. data_structure.java
   This class contains the information of the client(Student or Advisor).
   It contains 3 linked list which divided on the basis of different terms(fall,winter,summer).
   3 counter variables are also created to keep check of the fact that a student from 1 department cannot enroll in more than 1 course from each of the different departments.

4. courses_availability.java
   This class contains the information of each of the courses(course Identification numbers and
   the number of seats in the course).

The hardest part to implement was to do inter server communication because of certain level of checks to be performed.

Tests Performed:

1. Adding an INSE and SOEN, course for a COMP student and trying to add course from INSE or SOEN.
2. Testing the limit for the foreign department courses.
3. Adding more than 3 courses to a single semester for a student.
4. As an advisor, adding and deleting courses, and deleting those courses.
5. Students deleting and adding the new courses added by the advisor.


Client
#User : String
+main() : void
+comp_server_call(String user) : void
+inse_server_call(String user) : void
+soen_server_call(String user) : void
+request_comp(String command) : String
+request_soen(String command) : String
+request_inse(String command) : String
+student_menu(String userId) : void
+advisor_menu(String userId) : void
+add_course(String Student_Id, String Course) : void
+drop_course(String Student_Id, String Course_Id) : void
+new_course(String Advisor_Id, String Course_Id) : void
+delete_course(String Advisor_Id, String Course_Id): void
+check_availability(): void
COMP_Server
+comp_student_list : LinkedList <data_structure>
+courses : LinkedList : LinkedList<courses_availability>
+ main() : void
+course_availability() : String
+increase_availability(String data) : String
+decrease_availability(String data) : String
+add_SOEN_course(String command):String
+drop_SOEN_course(String command):String
+request_soen(String data): String
+add_INSE_course(String command):String
+drop_INSE_course(String command):String
+request_inse(String data): String
+add_course(String command): String
+drop_course(String command):String
+new_course(String data) : String
+delete_course(String data): String
INSE_Server
+inse_student_list : LinkedList <data_structure>
+courses : LinkedList : LinkedList<courses_availability>
+ main() : void
+course_availability() : String
+increase_availability(String data) : String
+decrease_availability(String data) : String
+add_SOEN_course(String command):String
+drop_SOEN_course(String command):String
+request_soen(String data): String
+add_COEN_course(String command):String
+drop_COEN_course(String command):String
+request_coen(String data): String
+add_course(String command): String
+drop_course(String command):String
+new_course(String data) : String
+delete_course(String data): String
SOEN_Server
+soen_student_list : LinkedList <data_structure>
+courses : LinkedList : LinkedList<courses_availability>
+ main() : void
+course_availability() : String
+increase_availability(String data) : String
+decrease_availability(String data) : String
+add_INSE_course(String command):String
+drop_INSE_course(String command):String
+request_inse(String data): String
+add_COEN_course(String command):String
+drop_COEN_course(String command):String
+request_coen(String data): String
+add_course(String command): String
+drop_course(String command):String
+new_course(String data) : String
+delete_course(String data): String
data_structure
+ username : String
+ fall_courses : LinkedList<String>
+ winter_courses: LinkedList<String>
+summer_courses: LinkedList<String>
+comp : int
+soen: int
+inse: int
+data_stucture(String Id) : data_structure
+get_Id (): String
+ add_courses_fall(String fall_register): String
+add_courses_summer(String summer_register) : String
+add_courses_winter(String winter_register): String
+drop_course_fall(String register) :String
+drop_course_winter(String register): String
+drop_course_summer(String register): String
+print (): String
courses_availability
+course_name: String
+term: String
+ course_count : int
+courses_availability(String course, String season) : courses_availbility
+ get_course_name() : String
+ get_term() : String
+ get_course_count() : int
+ set_course_count() : void
+ increase_count() : String
+ decrease_count() : void
+ print_data() : String