Software Analysis And Design

Project Phase 2

Kemal Kağan Orhan 05160000785
Güneş Hacıhalil 05170000770
Enes Yaşarbaş 05180000939

We have the two use cases of students registering courses and faculty member's assigning grades from our phase 1 of project. Administrative staff opening courses for semester is a new use case that we wrote.

Use Case for Registering for Courses

Primary Actor: Student

Stakeholders and Interests:
-Faculty: Wants their students to be able to register for courses without a problem.
-University: Want the process of registering go smoothly for their every faculty.

Preconditions:
-There should be an empty space for courses with limited students
-The user should have spare credit score not to hit semesterly credit score limit.

Success Guarantee (Post Conditions)
-The user now have registered for the desired courses.
-The system now has one less student who has to register before the registration week end.

Use Case For Registering Class	
Actions Of User	Responses From System
1. The user opens the registering page for courses.	
	2. The system asks for U-ID and password for verification.
3. User enters their U-ID and password.	
	4. If system returns positive for verification, the system show the courses that have opened in the current semester.
5. The user makes the choices of desired courses with following the rules of registering. (Check Extensions)	
	6. If the rules of the registering courses have not been broken, the chosen courses are sent to counselor for final confirmation.
	7.With counselor's affirmation, the system registers the student to courses.

8. The user is now registered to courses.	

Extensions (Alternative Flow)

*a At any time, systems fails;

- System reboots itself and starts over.

3. If verification has failed, the system throws user to 2nd step.

Rules of Registration:

- If the student has reach the credit quota or close to the credit quota and the desired courses credit surpasses the credit quota, the system will not let that course to be registered.

-If desired course has a student limit and the limit has been hit, the system will not let that course to be registered.

-In the case of earlier year's course that the user have been failed, the system automatically registers that course.

6. If counselor does not approve register, the system throws user to 5th step.

Special Requirements

-The course has to be opened by university to be registered by students.

-The user have to be a student of university.

Technology and Data Variations List

- User must have a device that is able to connect to the internet.
- The data set of courses and the students who have been registered.

Frequency of Occurrence

Four times in a year. (Week of registration for fall and spring semesters, Re-registration week for fall and spring semester.)

Use Case for Assigning Grades

Primary Actor: Faculty

Stakeholders and Interests

- Student: Wants their work to fairly graded.
- University: Wants to calculate the final grade of students to check if the student is passing or not.

Preconditions

- Student must be attending to semesterly course.
- Faculty Member must already graded the work to assign in the system.

Success Guarantee (Post Conditions)

- The grade is now entered to system.

Use Case For Assigning Grades	
Actions Of User	Responses From System
1. The user opens the system of grading page.	
	2. The system asks for U-ID and password to access the course system.
3. User inputs the U-ID and password.	
	4. After verification success, the system will let user the grading page.
	5. System asks which course to grades assigned.
6. User chooses the semesterly course that their teaching.	
	7. System opens the semesterly course and asks for which student's grade will be assigned
8. User chooses the student with grade	
	9. System asks for the grade.
10. User enters the grade of student	
	11. System assigns the grade to student and asks if there is more grade to assign.
	12. If user answers with negative response, the system will exit assigning system. If user answers with positive response, the system throws user to 7th step.

Extensions (Alternative Flow)
*a At any time, systems fails;
- System reboots itself and starts over.

4. If the verification has failed, the system throws user to 3rd step.
G1. The grades have to be numbers between 1 to 100. semester score will be calculated from grades after the user (faculty member) finalizes the semesterly course.

Technology and Data Variations List
- User must have a device that is able to connect to the internet.
-Student data list of semesterly course
-Grade list of students.

Frequency of Occurrence
-Can occur few times while in the university is active in the semester.

Use Case for Opening Courses For Semester

Primary Actor: Administrative Staff

Stakeholders and Interests:
-Faculty: Wants to find to which classes they are teaching
-Student: Wants to register to semesterly opened course

Preconditions:
-For opening a course; location, semester and teacher should predetermined.

Success Guarantee (Post Conditions)
-The user now have opened a course for semester.
-The faculty member's now have classes to teach.

Use Case For Opening Courses For Semester	
Actions Of User	Responses From System
1. The user opens the system of admin staff page.	
	2. The system asks for U-ID and password to access the course system.
3. User inputs the U-ID and password.	
	4. After verification success, the system will let user the choices page.
5. User chooses to open a course for semester.	
	6. System asks for the Course Description, teacher, semester and location for opening.
7. User inputs the required information to system.	
	8. System asks if there is any more action to happen.
9. User input negative response and closes system.	

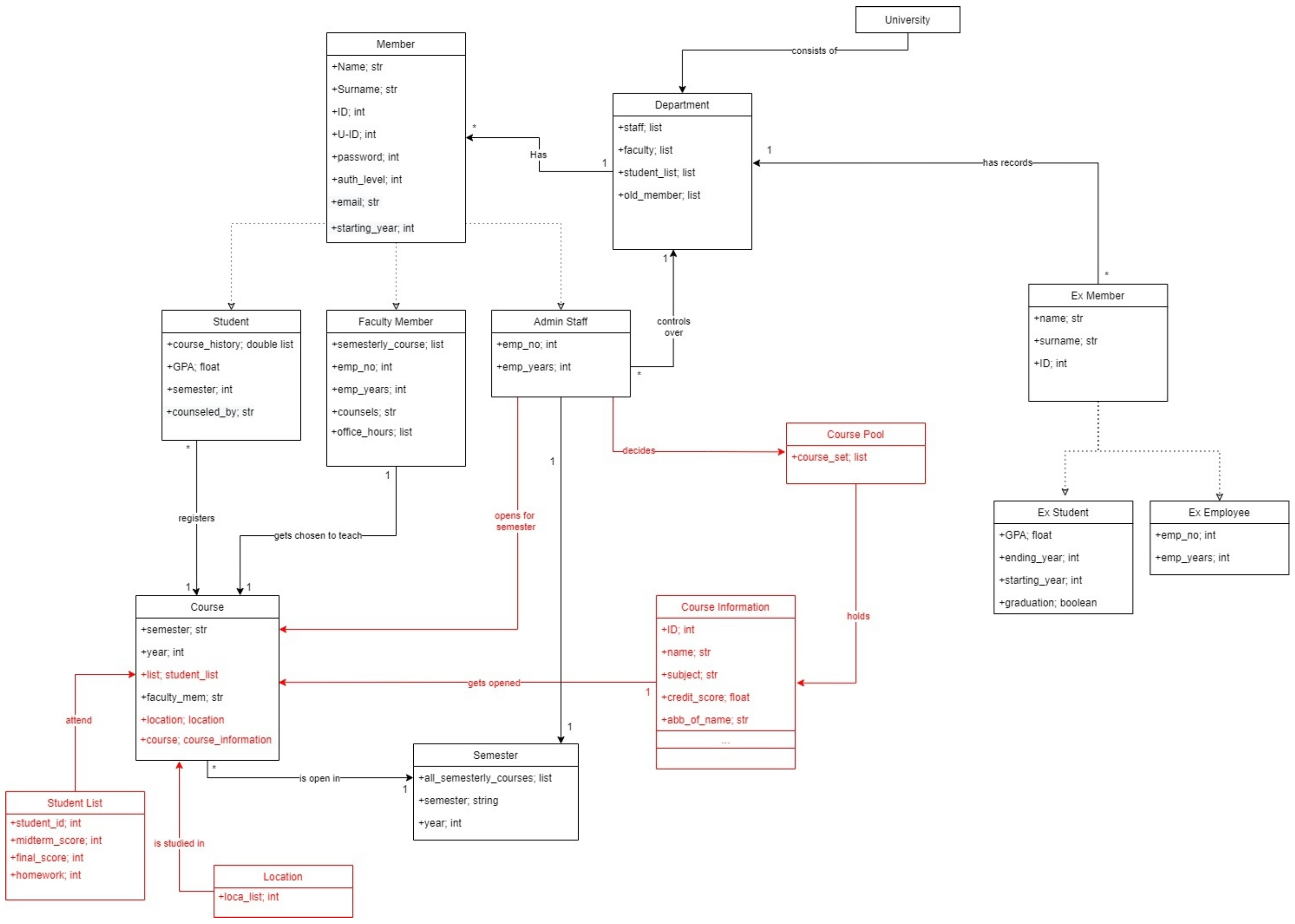
Extensions (Alternative Flow)
*a At any time, systems fails;
- System reboots itself and starts over.

4. If the verification has failed, the system throws user to 3rd step.
7. If one of the information is corrupted, the system throws user to 6th step.

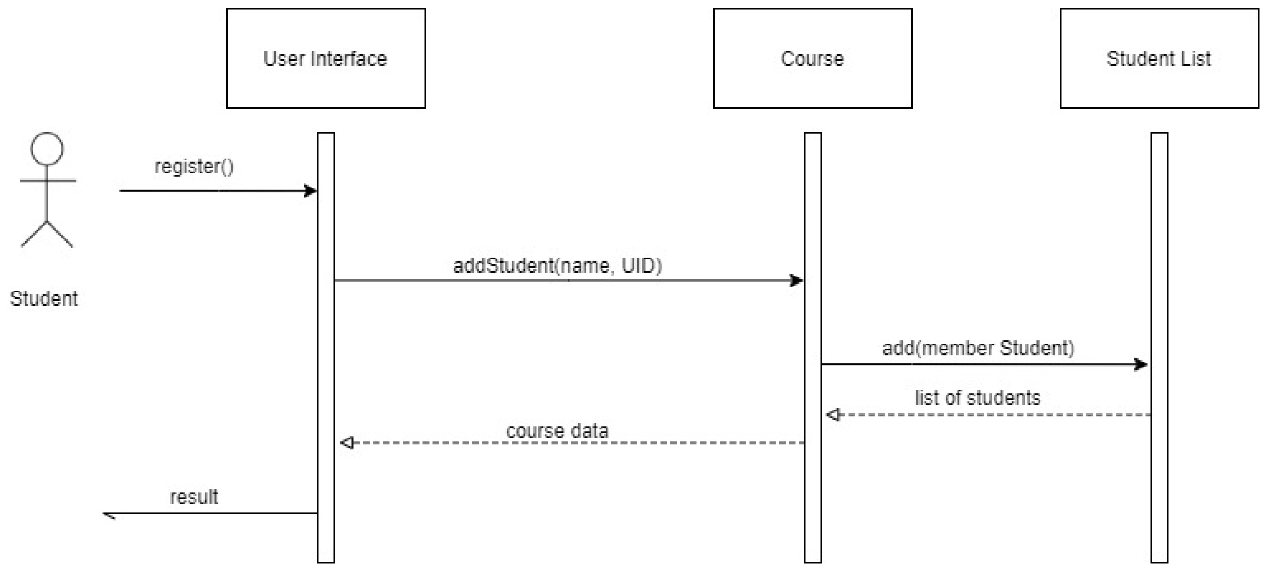
Technology and Data Variations List
- User must have a device that is able to connect to the internet.
- Data set of course pool

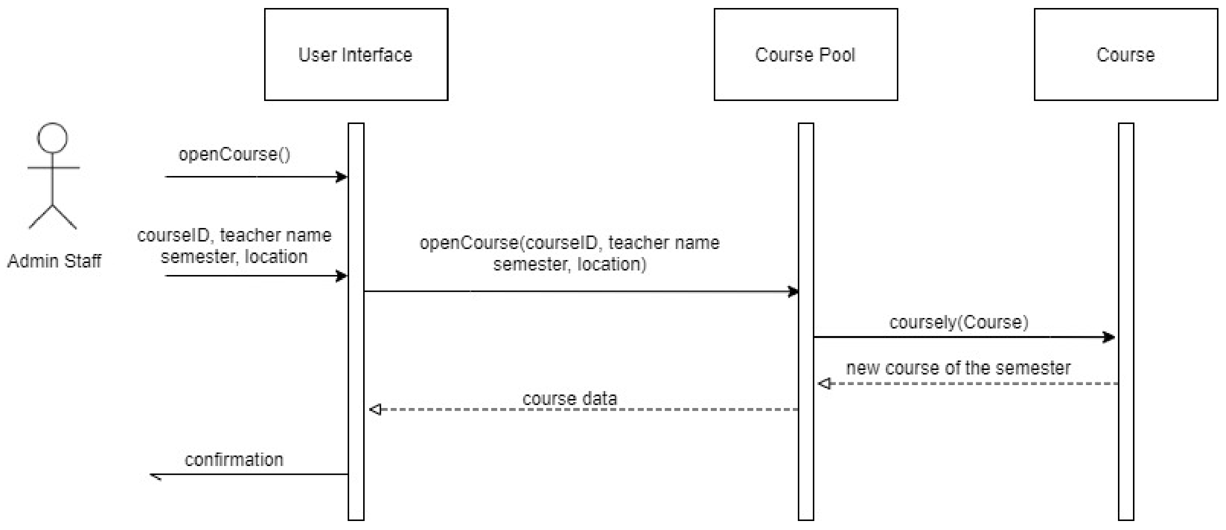
Frequency of Occurrence
-Can occur two times in year. Before the starting of first and second semesters.

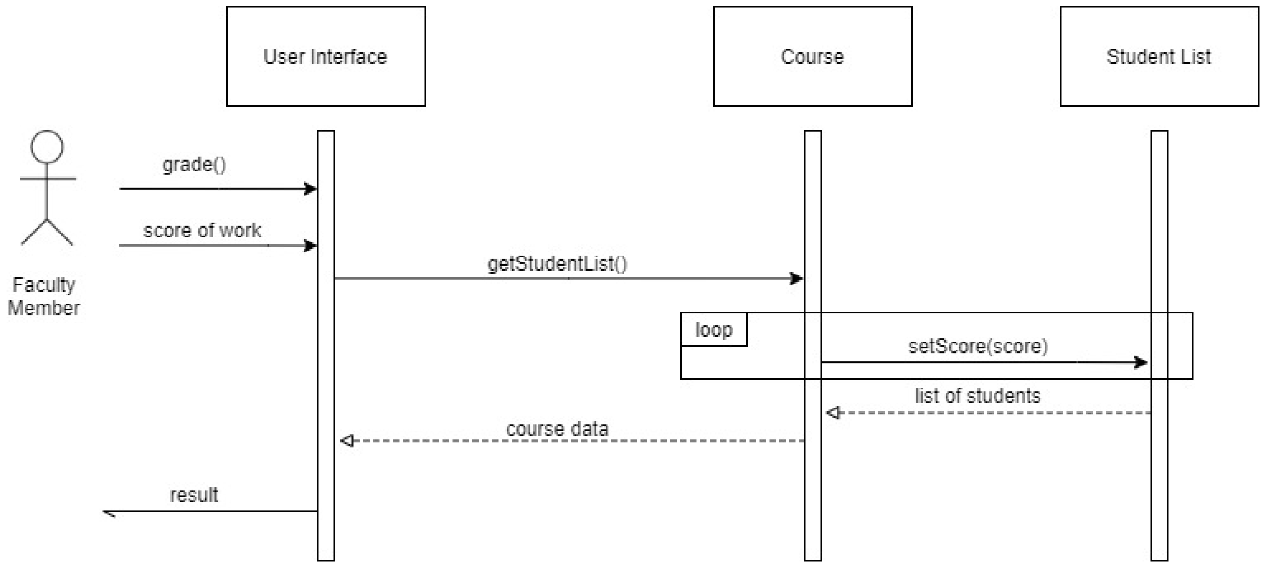
We made some changes in domain modal that will improve the system.



For the uses cases we wrote, lets draw the sequence diagrams.




User Interface
Course Pool
Course
Admin Staff
openCourse()
courseID, teacher name semester, location
openCourse(courseID, teacher name semester, location)
coursely(Course)
new course of the semester
course data
confirmation



Information Expert

Each class and attribute we created, have its own responsibilities regards to how the system works. When creating new objects with these classes, we can gather the information in a easier and more effective fashion.

High Cohesion

By building each method in its own class, we avoided the duplication of codes. Thanks to this we can easily redesign, rewrite and test our code. Result was that, we have a highly cohesive code that is easy to maintain and update.

Low Coupling

With newly added relations and classes we decreased the complexity of the network by using references from other classes. This action we performed, has also decreased the dependency of the classes to each other. As a result we have achieved low coupling.

Creator

In our system there are mainly few creator pattern classes that are responsible for creating objects from other classes. Primarily our class of course is accountable for creating course related attributes. Most of the system has actions on the course which can grant the main creator tag for the course class. Secondly our member class is in charge of creating person based objects.

Our implemented code is based on administrative staff opening a course for the semester. The code also includes few different methods other than opening courses due to easily using the main action. Before opening the course for the semester, the first method add a course description. Second method is for seeing the course description we created. The last and forth method is for testing if the method worked. Our third method is the main method of opening the course for the semester.

Unlike the third method, other methods does not contain expectation handling. Our third method does have a handling system regards to use case we written.

Source Code

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package finalsad;
import java.util.*;
import java.io.*;
/**
 *
 * @author pc
 */
public class FinalSAD {

    public static void main(String[] args) {
        ArrayList<member> teachers = new ArrayList();
        ArrayList<member> admins = new ArrayList();
        Scanner sc = new Scanner(System.in);
        course_pool pool = new course_pool();
        ArrayList<course> opened_courses = new ArrayList();
        ArrayList<location> locations = new ArrayList();;
        location loca1= new location();
        location loca2= new location();
        loca1.setLoca("B7");
        loca2.setLoca("B8");
        locations.add(loca1);
        locations.add(loca2);

        admin_staff staff = new admin_staff();
        staff.setMember("Kemal",100,0);
        staff.setAuthority();
        admins.add(staff);
        faculty_member hoca = new faculty_member();
        hoca.setMember("Enes",200,0);
        hoca.setAuthority();
        teachers.add(hoca);

        member temp = new member();
        int active1 = 0;
        while (active1 == 0){
          System.out.println("Please sign in with ID:");
          int signid = sc.nextInt();
          System.out.println("Password:");
          int password = sc.nextInt();

          for (member type: teachers){
            if(signid == type.getUid() && password == type.getPassword()){
              temp = type;
              active1 = 1;
            }
          }
          for (member type: admins){
            if(signid == type.getUid() && password == type.getPassword()){
              temp = type;
              active1 = 1;
            }
          }
          if (temp.getName() == null){
              System.out.println("Error: Wrong UID or Password.");
              System.out.println("Retry? (1 for Yes/2 for No)");
              int retry = sc.nextInt();
              if (retry == 2){
                System.exit(0);
              }
          }
        }
        int active2 = 2;
        while(active2 == 2){
          System.out.println("-------*-------");
          if (temp.getAutharity() == 3){
            System.out.println("Choose action:\n1. Add Course\n2. See Openable
Courses\n3. Open Course For Semester\n4. Opened Courses For Semester");
            int action = sc.nextInt();
            switch(action){
              case 1:
                pool.addtoPool();
                break;
              case 2:
                System.out.println(pool.seeCourses());
                break;
              case 3:
                System.out.println("Enter Course ID:");
                int temp1 = 0;

                boolean flag;
                do{
                  try{
                    Scanner sc2 = new Scanner(System.in);
                    temp1=sc2.nextInt();
                    flag=false;
                  }
                  catch(Exception e){
                  System.out.println("Input Error!\nEnter Course ID:");
                  flag=true;
                  }
                }
                while(flag);
                course_description temp2 = new course_description();
                for (course_description temp3 : pool.courses){
                  if(temp3.getId() == temp1){
                    temp2 = temp3;
                  }
                }
                System.out.println("Choose Teacher:(Enter name)");
                String teach = sc.next();
                member temp5 = new member();
                for (member temp4: teachers){
                  if(teach.equals(temp4.getName())){
                    temp5 = temp4;
                  }
                }
                System.out.println("Choose Location For Course:\n");
                int i = 1;
                for (location loca: locations){
                  System.out.println(i  +". " + loca.getLoca());
                  i++;
                  }
                int k = sc.nextInt();
                k = k-1;
                location temp6 = new location();
                temp6 = locations.get(k);
                System.out.println("Enter Current Year:");
                int year = sc.nextInt();
                System.out.println("Enter Current Semester:");
                String semester = sc.next();
                semester temp7 = new semester();
```

```
                temp7.setSemester(semester);
                temp7.setYear(year);
                course termlycourse = new course();
                termlycourse = pool.openCourse(temp2, temp5, temp6, temp7);
                opened_courses.add(termlycourse);


                break;
              case 4:
                for(course list: opened_courses){
                  System.out.println("-------*-------");
                  System.out.println(list.toString());
                }
                break;
            }

        }
        System.out.println("-------*-------");
        System.out.println("End the system?\n For Yes Press 1\n For No Press 2");
        active2 = sc.nextInt();
      }
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package finalsad;

 /*
 * @author pc
 */
public class member {
    private String name;
    private int uid;
    private int password;
    private int authority;

    public String getName() {
        return name;
    }

    public void setMember(String name, int uid, int password) {
        this.name = name;
        this.uid = uid;
        this.password = password;
    }

    public int getUid() {
        return uid;
    }

    public int getPassword() {
        return password;
    }

    public void setPassword(int password) {
        this.password = password;
    }

    public int getAutharity() {
        return authority;
    }

    public void setAutharity(int autharity) {
        this.authority = autharity;
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package finalsad;

/**
 *
 * @author pc
 */
public class admin_staff extends member {
    void setAuthority()
    {
        admin_staff.super.setAutharity(3);
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package finalsad;

/**
 *
 * @author pc
 */
public class faculty_member extends member {
    void setAuthority()
    {
        faculty_member.super.setAutharity(2);
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package finalsad;

import java.util.*;

/**
 *
 * @author pc
 */
public class course_pool {

    public ArrayList<course_description> courses;
    public course_description courseone;
    private Scanner scan;

    public void addtoPool(){
        courses = new ArrayList();
        course_description temp = new course_description();
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Course ID:");
        int id = sc.nextInt();
        System.out.println("Enter Course Name:");
        String name = sc.next();
        System.out.println("Enter Course Info:");
        String info = sc.next();
        temp.setAll(id, name, info);
        courses.add(temp);
    }

    public String seeCourses(){
        int i = 0;
        String text = new String();
        for (course_description item : courses){
            text += "Course Name: " + item.getName() + " Course ID: " + item.getId() + "
Course Info: " + item.getInfo() + "\n";
            i++;
        }
        return text;
    }

    public course openCourse(course_description temp, member teacher, location
loca, semester year){
        course coursely = new course();
        coursely.setCourse_info(temp);
        coursely.setTeacher(teacher);
        coursely.setRoom(loca);
        coursely.setYear(year);
        return coursely;
    }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package finalsad;

/**
 *
 * @author pc
 */
public class course_description {
    private int id;
    private String name;
    private String info;

    public void setAll(int id, String name, String info){
        this.id = id;
        this.name = name;
        this.info = info;
    }

    /**
     * @return the id
     */
    public int getId() {
        return id;
    }

    /**
     * @return the name
     */
    public String getName() {
        return name;
    }

    /**
     * @return the info
     */
    public String getInfo() {
        return info;
    }

}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package finalsad;

/**
 *
 * @author pc
 */
public class course {
    private course_description course_info;
    private member teacher;
    private location room;
    private semester year;

    /**
     * @return the course_info
     */
    public course_description getCourse_info() {
        return course_info;
    }

    /**
     * @param course_info the course_info to set
     */
    public void setCourse_info(course_description course_info) {
        this.course_info = course_info;
```

```
}

/**
 * @return the teacher
 */
public member getTeacher() {
   return teacher;
}

/**
 * @param teacher the teacher to set
 */
public void setTeacher(member teacher) {
   this.teacher = teacher;
}

/**
 * @return the room
 */
public location getRoom() {
   return room;
}

/**
 * @param room the room to set
 */
public void setRoom(location room) {
   this.room = room;
}

/**
 * @return the year
 */
public semester getYear() {
   return year;
}

/**
 * @param year the year to set
 */
public void setYear(semester year) {
   this.year = year;
}

public String toString(){
        return "Course Name: " + course_info.getName() + "\nTeacher's Name: " +
teacher.getName() + "\nSemester: " + year.getSemester() + "\nLocation: " +
room.getLoca();
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package finalsad;

/**
 *
 * @author pc
 */
public class location {
    private String loca;

    /**
     * @return the loca
     */
    public String getLoca() {
        return loca;
    }

    /**
     * @param loca the loca to set
     */
    public void setLoca(String loca) {
        this.loca = loca;
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package finalsad;

/**
 *
 * @author pc
 */
public class semester {
```

```java
    private int year;
    private String semester;

    /**
     * @return the year
     */
    public int getYear() {
        return year;
    }

    /**
     * @param year the year to set
     */
    public void setYear(int year) {
        this.year = year;
    }

    /**
     * @return the semester
     */
    public String getSemester() {
        return semester;
    }

    /**
     * @param semester the semester to set
     */
    public void setSemester(String semester) {
        this.semester = semester;
    }
}
```