*Group members:*

Lê Thanh Phương Nam - ITITWE19025

Đặng Khải Dương - ITITWE19010

Đinh Bình Thanh Thông - ITITWE19027

**Group Report Lab 7&8**

**Organization:**

Phương Nam -**Topic 3&4**

Khải Dương -**Topic 1&2**

Thanh Thông -**Topic 5&6**

**Software requirements**

-ArgoUML, StarUML, java…

**Topic 1: The New Student Enrollment Process**

**1.** **USE CASE DIAGRAM:**

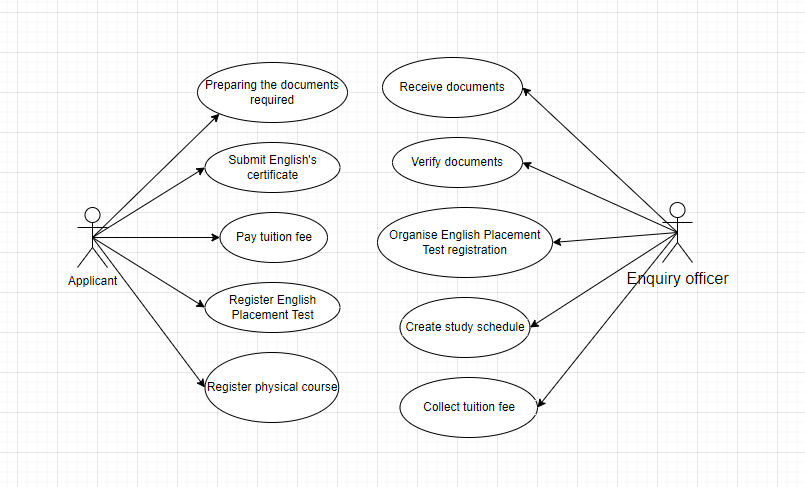
**a) Candidate Actors**

Applicant: someone who has graduated from high school and passed a national or school examination

OAA: Office of Academic Affair, The office in charge of the English Placement Test and student enrollment

OFP: Office of Finance and Planning, the office in charge of student tuition fees

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Applicant | Successfully be a new student after the enrollment process |
| OAA | Successfully enrolled new students |
| OFP | Successfully collecting tuition fees for the first semester of students |

****

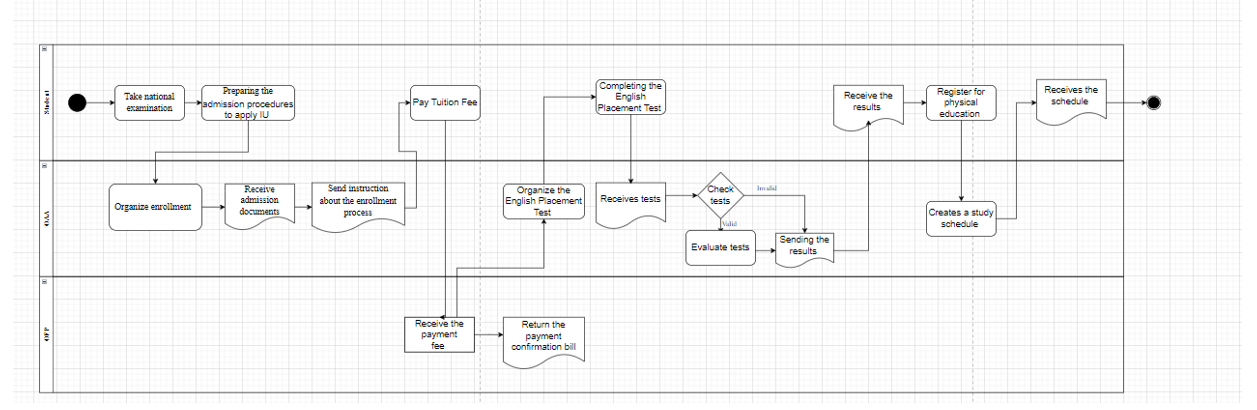
**2.** **ACTIVITY DIAGRAM:**

1. **Identify Candidate use case (Activities)**

|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **Actors** | **Description** |
| UC\_1 | Applicant | Apply for university |
| UC\_2 | Submit enrollment document (required documents, English certificate) to the university |
| UC\_3 | Attend ceremony |
| UC\_4 | OAA | Receive admission form from acceptance students |
| UC\_5 | Confirm validity and send enrollment document to Head Department |
| UC\_6 | Announce required admission score and entrance score. |
| UC\_7 | Approve enrollment to validated students |
| UC\_8 |  | Organize ceremony to welcome freshmen |

1. **Identify the Start point and Endpoint of each use case**

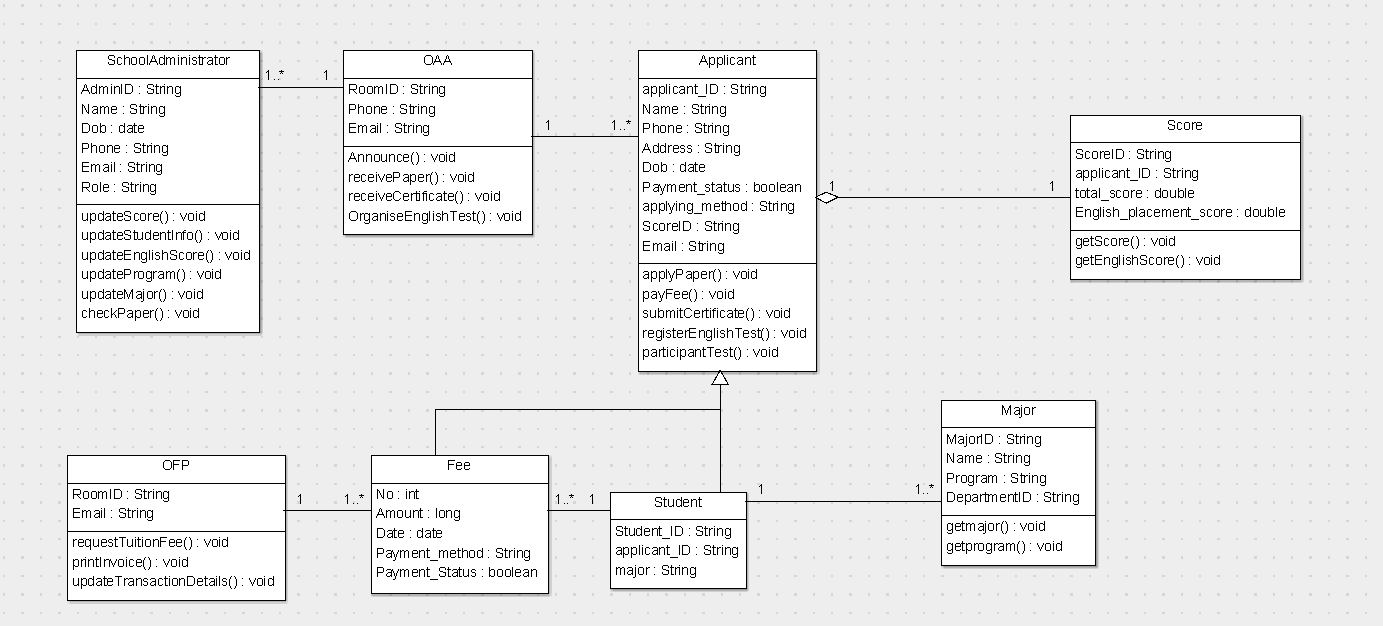
|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **The Start Point** | **The End Point** |
| UC\_1 | Apply for university | Receive acceptance letter from the university |
| UC\_2 | Submit enrollment document to the University | Matriculated students submit all the required documents |
| UC\_3 | Attend ceremony | Attend ceremony |
| UC\_4 | Receive admission form from acceptance students | The submitted document is clarified and validated |
| UC\_5 | Confirm validity and send enrollment document to Head Department | The office received all documents |
| UC\_6 | Announce required admission score and entrance score. | Announce required admission score and entrance score. |
| UC\_7 | Approve enrollment to validated students | Approve enrollment to validated students |
| UC\_8 | welcome freshmen | Organize ceremony |

****

**3.** **CLASS DIAGRAM:**

This diagram consists of the following classes, attributes, and operations.

|  |  |  |
| --- | --- | --- |
| **Classes** | **Attributes** | **Operations** |
| SchoolAdministrator | AdminID: String  Name: String  Dob: date  Phone: String  Email: String  Role: String | updateScore() : void  updateStudentInfo() : void  updateEnglishScore() : void  updateProgram() : void  updateMajor() : void  checkPaper() : void |
| OAA | RoomID : String  Phone : String  Email : String | Announce() : void  receivePaper() : void  receiveCertificate() : void  OrganiseEnglishTest() : void |
| Applicant | applicant\_ID: String  Name: String  Dob: date  Phone: String  Email: String  Address: String  Payment\_status: Boolean  applying\_method: String  ScoreID: String | applyPaper() : void  payFee() : void  submitCertificate() : void  registerEnglishTest() : void  participantTest() : void |
| OFP | RoomID : String  Email : String | requestTuitionFee() : void  printInvoice() : void  updateTransactionDetails() : void |
| Fee | No: int  Amount: long  Date: date  Payment\_method : String  Payment\_Status: boolean |  |
| Student | Student\_ID: String  applicant\_ID: String  major: String |  |
| Major | MajorID : String  Name : String  Program : String  DepartmentID : String | getmajor() : void  getprogram() : void |
| Score | ScoreID : String  applicant\_ID : String  total\_score : double  English\_placement\_score : double | getScore() : void  getEnglishScore() : void |

****

**Code (abstract): using** [**ArgoUML**](https://www.bing.com/videos/search?q=argouml+tutorial&docid=608043554099435551&mid=B642DD9024C6F536AD1EB642DD9024C6F536AD1E&view=detail&FORM=VIRE&msclkid=9cbe87a1cf3711ec93795e04c88c28c9) **to generate**

**Applicant.java**

package topic1;

import java.util.Date;

import java.util.Vector;

public class Applicant {

public String applicant\_ID, Name, Phone, Address, applying\_method, ScoreID, Email;

public Date Dob;

public boolean Payment\_status;

public OAA myOAA;

public Score myScore;

public Vector mystudent;

public Student myStudent;

public void applyPaper() {

}

public void payFee() {

}

public void submitCertificate() {

}

public void registerEnglishTest() {

}

public void participantTest() {

}

}

**Fee.java**

package topic1;

import java.util.Date;

public class Fee extends Applicant {

public int No;

public long Amount;

public Date Date;

public String Payment\_method;

public boolean Payment\_Status;

public OFP myOFP;

public Student myStudent;

}

**Major.java**

package topic1;

public class Major {

public String MajorID, Name, Program, DepartmentID;

public Student myStudent;

public void getmajor() {

}

public void getprogram() {

}

}

**OAA.java**

package topic1;

import java.util.Vector;

public class OAA {

public String RoomID, Phone, Email;

/\*\*

\*

\* @element-type Applicant

\*/

public Vector myApplicant,mySchoolAdministrator;

/\*\*

\*

\* @element-type SchoolAdministrator

\*/

public void Announce() {

}

public void receivePaper() {

}

public void receiveCertificate() {

}

public void OrganiseEnglishTest() {

}

}

**OFP.java**

package topic1;

import java.util.Vector;

public class OFP {

public String RoomID, Email;

/\*\*

\*

\* @element-type Fee

\*/

public Vector myFee;

public void requestTuitionFee() {

}

public void printInvoice() {

}

public void updateTransactionDetails() {

}

}

**SchoolAdministrator.java**

package topic1;

import java.util.Date;

public class SchoolAdministrator {

public String AdminID, Name, Phone, Email, Role;

public Date Dob;

public OAA myOAA;

public void updateScore() {

}

public void updateStudentInfo() {

}

public void updateEnglishScore() {

}

public void updateProgram() {

}

public void updateMajor() {

}

public void checkPaper() {

}

}

**Score.java**

package topic1;

public class Score {

public String ScoreID, applicant\_ID;

public double total\_score, English\_placement\_score;

public Applicant myApplicant;

public void getScore() {

}

public void getEnglishScore() {

}

}

**Student.java**

package topic1;

import java.util.Vector;

public class Student extends Applicant {

public String Student\_ID, applicant\_ID, major;

/\*\*

\*

\* @element-type Major

\*/

public Vector myMajor;

public Applicant myApplicant;

public Vector myapplicant;

/\*\*

\*

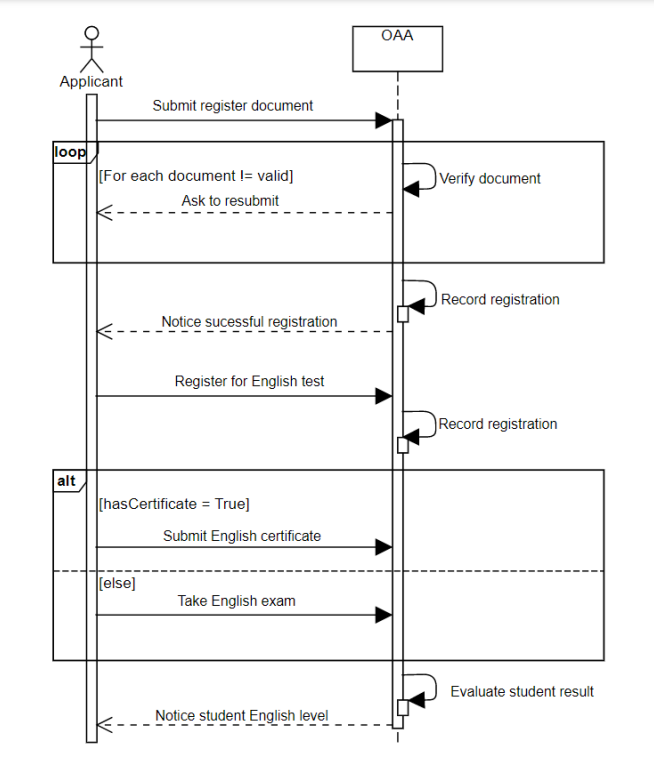
\* @element-type Fee

\*/

public Vector myFee;

}

**4.** **SEQUENCE DIAGRAM:**

****

**Topic 2: The student stand-downs/suspensions/ exclusions/expulsions process**

**1.** **USE CASE DIAGRAM:**

**a) Candidate Actors**

Advisor: A advisor in charge of following and supporting the students.

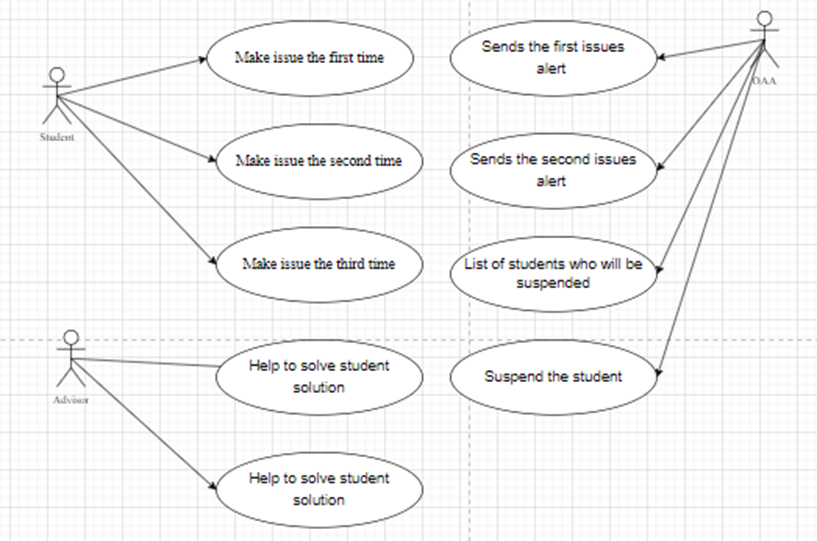
OAA The office in charge of creating a list of students who violate the regulation and deciding the appropriate suspension for each student

OFP: the office in charge of tuition fees from students.

**b) Goal of Actor**

The goal in Context: Make rules for students who are on hold, such as suspensions, exclusions, and expulsions, as well as solutions for these students.

|  |  |
| --- | --- |
| Actor | Goal |
| Student | Students 3 times have low results, owe more than the prescribed amount, and have not paid tuition fees |
| OAA | Completing the student suspension process |
| Advisor | Complete monitoring, reminders, and notifications to OAA |

****

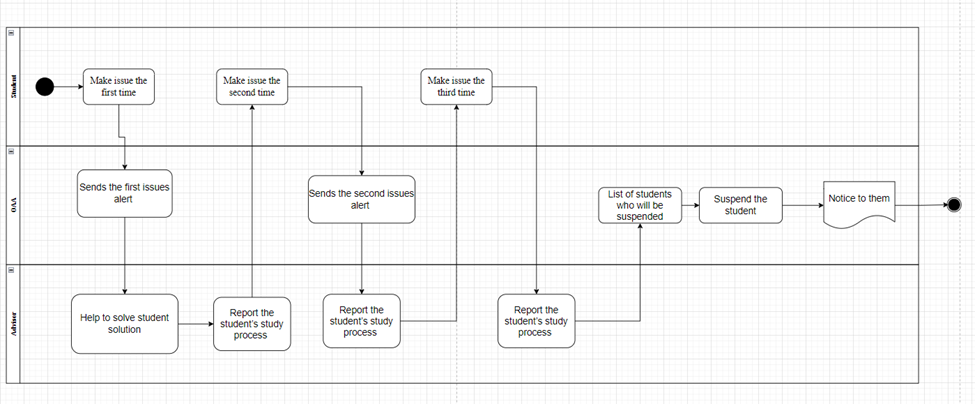
**2.** **ACTIVITY DIAGRAM:**

1. **Identify Candidate use case (Activities)**

|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **Actor** | **Description** |
| 1: Make the first situation | Student | For the first semesters, students have an overall score of less than 35/100.  Students who owe more than 24 credits from the start of the course for the first time.  Students who have not paid their tuition fees for the first time |
| 2: Make a second situation |  | Students have an overall score of less than 35/100 for the upcoming semesters.  Students who owe more than 24 credits from the start of the course for the second time.  Students who have not paid their tuition fees for the second time |
| 3: Make a third situation |  | Students have an overall score of less than 35/100 for the upcoming semesters.  Students who owe more than 24 credits from the start of the course for the second time.  Students who have not paid their tuition fees for the second time |
| 4: Send the first alert | OAA | The office sends the first issue alert to students on the list and their Advisor. |
| 5: Send the first alert |  | The office sends the second issue warning to students on the list and their Advisor. |
| 6: List suspended students |  | The Office compiles a list of students who have been suspended. |
| 7: Decides to suspend and notice |  | The office decides to suspend these students and sends out notices to their home addresses, email addresses, and phone numbers. |
| 8: Work with students and report the first time | Advisor | The advisor will help to solve student solutions for the first time and report the student’s study process to OAA |
| 9: Report the second time |  | The advisor reports the student’s study process to OAA |
| 10: Report third time |  | The advisor reports the student’s study process to OAA |

1. **Identify the Start point and Endpoint of each use case**

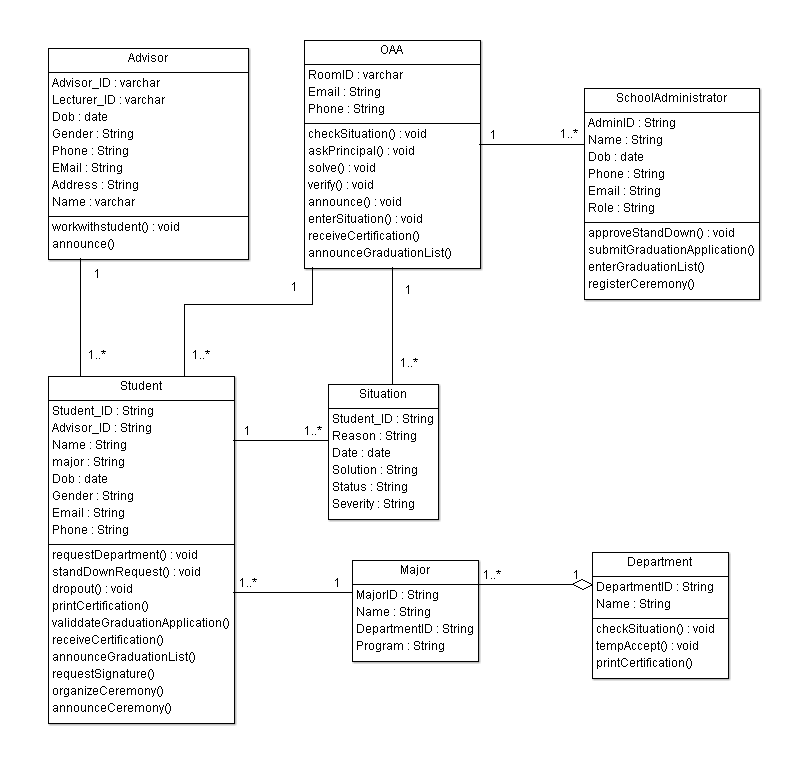
|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **The Start Point** | **The End Point** |
| UC\_1 | Make the first situation | Make the first situation |
| UC\_2 | Make the second situation | Make the second situation |
| UC\_3 | Make the third situation | Make the third situation |
| UC\_4 | Send the first alert | Send the first alert |
| UC\_5 | Send the second alert | Send the second alert |
| UC\_6 | List suspended students | List suspended students |
| UC\_7 | Decides to suspend | Notice to Student |
| UC\_8 | Work with student | report first time |
| UC\_9 | Report the second time | Report the second time |
| UC\_10 | Report third time | Report third time |

****

**3.** **CLASS DIAGRAM:**

This diagram consists of the following classes, attributes, and operations.

|  |  |  |
| --- | --- | --- |
| **Classes** | **Attributes** | **Operations** |
| Student | Student\_ID: String  Advisor\_ID : String  major : String  Name : String  Dob : Date  Email : String  Phone : String  Gender : String | requestDepartment() : void  standDownRequest() : void  dropout() : void |
| Advisor | Advisor\_ID: varchar  Lecturer\_ID : varchar  Dob: date  Gender: String  Phone: String  EMail: String  Address: String  Name: varchar | workwithstudent() : void |
| Major | MajorID: String  Name: String  DepartmentID: String  Program: String |  |
| Department | DepartmentID : String  Name : String | checkSituation() : void  tempAccept() : void |
| OAA | RoomID: String  Email: String  Phone: String | checkSituation() : void  askPrincipal() : void  solve() : void  verify() : void  announce() : void  enterSituation() : void |
| SchoolAdministrator | AdminID: String  Name: String  Dob: date  Phone: String  Email: String  Role: String | approveStandDown() : void |

****

**Code (abstract): using** [**ArgoUML**](https://www.bing.com/videos/search?q=argouml+tutorial&docid=608043554099435551&mid=B642DD9024C6F536AD1EB642DD9024C6F536AD1E&view=detail&FORM=VIRE&msclkid=9cbe87a1cf3711ec93795e04c88c28c9) **to generate**

**Advisor.java**

package topic2;

import java.util.Date;

import java.util.Vector;

public class Advisor {

public String Advisor\_ID, Lecturer\_ID,Gender, Phone,EMail,Address,Name;

public Date Dob;

public Vector my,myStudent;

/\*\*

\*

\* @element-type Student

\*/

public void workwithstudent() {

}

}

**Department.java**

package topic2;

import java.util.Vector;

public class Department extends Student {

public String DepartmentID,Name;

public Vector myStudent,myMajor;

/\*\*

\*

\* @element-type Major

\*/

public void checkSituation() {

}

public void tempAccept() {

}

}

**Major.java**

package topic2;

import java.util.Vector;

public class Major extends Department {

public String MajorID ,Name, DepartmentID,Program;

public Student myStudent;

public Vector mydepartment;

public Department myDepartment;

}

**OAA.java**

package topic2;

import java.util.Vector;

public class OAA {

public String RoomID,Email,Phone;

public SchoolAdministrator mySchoolAdministrator;

/\*\*

\*

\* @element-type Situation

\*/

public Vector myschoolAdministrator,mySituation,myStudent;

/\*\*

\*

\* @element-type Student

\*/

public void checkSituation() {

}

public void askPrincipal() {

}

public void solve() {

}

public void verify() {

}

public void announce() {

}

public void enterSituation() {

}

}

**SchoolAdministrator.java**

package topic2;

import java.util.Date;

import java.util.Vector;

public class SchoolAdministrator {

public String AdminID,Name,Phone,Email,Role;

public Date Dob;

public Vector myoaa;

public OAA myOAA;

public void approveStandDown() {

}

}

**Situation.java**

package topic2;

import java.util.Date;

public class Situation {

public String Student\_ID,Reason,Solution,Status,Severity;

public Date Date;

public Student one,myStudent;

public OAA myOAA;

}

**Student.java**

package topic2;

import java.util.Date;

import java.util.Vector;

public class Student {

public String Student\_ID,Advisor\_ID,Name,major,Gender,Email,Phone;

public Date Dob;

public Advisor myAdvisor;

public Vector one, many,mySituation, myMajor, myoaa;

/\*\*

\*

\* @element-type Situation

\*/

public OAA myOAA;

public void requestDepartment() {

}

public void standDownRequest() {

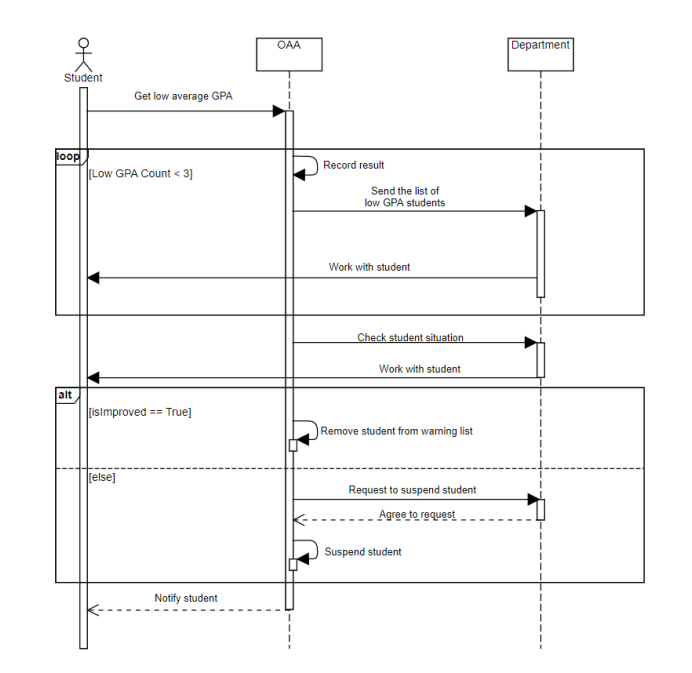
}

public void dropout() {

}

}

**4.** **SEQUENCE DIAGRAM:**



**Topic 3: The Graduation Application Process**

**1.** **USE CASE DIAGRAM:**

**a) Candidate Actors**

Student: A student who studies at International University and is qualified to graduate

OAA: the office in charge of the graduation process for students

Department: the office in charge of the thesis for students.

OFP: the office in charge of checking tuition debt

**b) Goal of Actor**

The goal in Context: The student successfully applies graduation ceremony

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Student | The student has been graduated successfully |
| OAA | Successful Announcement of graduation results and send Successfully an inviting Ceremony letter to Student |
| OFP | Successfully checking tuition debt and receiving tuitiofeesee (If any) |
| Department | Successfully send results of the student in the list’s thesis defense |

****

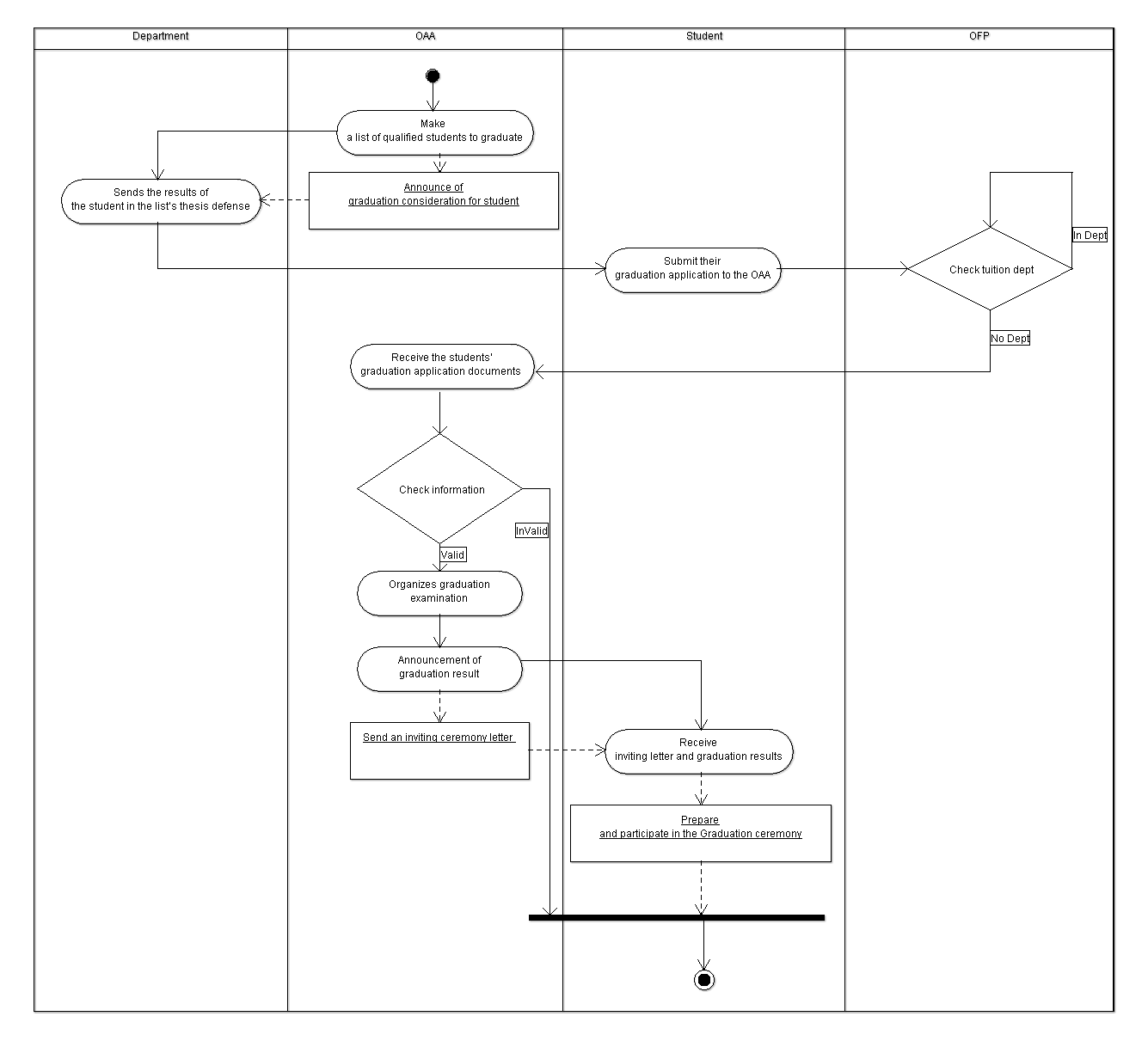
**2.** **ACTIVITY DIAGRAM:**

1. **Identify Candidate use case (Activities)**

|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **Actor** | **Description** |
| UC\_1: Make a list of qualified students to graduate | OAA | Make a list of qualified students to graduate  Announce graduation consideration for students in the list via email, website, edusoft |
| UC\_2: Receive the students’ graduation application documents |  | Receive the students’ graduation application documents and check the information |
| UC\_3: Organizes graduation examination |  | Organizes graduation examination |
| UC\_4: Announcement of graduation result |  | Announcement of graduation results and send an inviting letter to Student |
| UC\_5: Sends the results of  the student in the list’s thesis defense | Department | Sends the results of the student in the list’s thesis defense to the OAA:   * Dissertation defense record. * Application for graduation according to the form. * Documents are attached as prescribed. |
| UC\_6: Check tuition dept | OFP | Check the Students’ tuition dept and receive tuition fee (If any) |
| UC\_7: Submit their graduation application to the OAA | Student | Submit their graduation application to the OAA:   * Graduation application form. * Scientific background:   + 3 3×4 photos (white background, taken within 6 months, dressed politely).  + Photocopy of birth certificate.  + Copy of university diploma with notarized transcripts within 6 months.  + English certificate according to the regulations of the school.  + Dissertation defense file |
| UC\_8: Receive inviting letter and graduation results |  | Receive inviting letter and graduation results. Prepare and participate in the Graduation ceremony. |

1. **Identify the Start point and Endpoint of each use case**

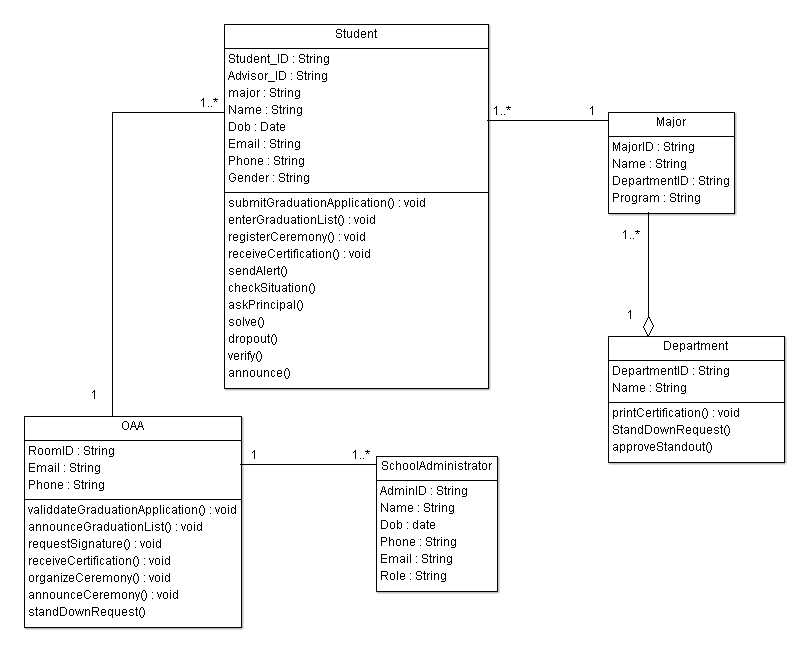
|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **The Start Point** | **The End Point** |
| UC\_1 | Make a list of qualified students to graduate | Announce graduation consideration for students |
| UC\_2 | Receive the students’ graduation application documents | Check the information |
| UC\_3 | Organizes graduation examination | Organizes graduation examination |
| UC\_4 | Announcement of graduation results | Send an inviting letter to Student |
| UC\_5 | Sends the results of the student in the list’s thesis defense | Sends the results of the student in the list’s thesis defense |
| UC\_6 | Check the Students’ tuition dept | Receive tuition fee (If any) |
| UC\_7 | Submit their graduation application to the OAA | Submit their graduation application to the OAA |
| UC\_8 | Receive inviting letter and graduation results | Prepare and participate in the Graduation ceremony |

****

**3.** **CLASS DIAGRAM:**

This diagram consists of the following classes, attributes, and operations.

|  |  |  |
| --- | --- | --- |
| **Classes** | **Attributes** | **Operations** |
| Student | Student\_ID : String  Advisor\_ID : String  major : String  Name : String  Dob : Date  Email : String  Phone : String  Gender : String | submitGraduationApplication() : void  enterGraduationList() : void  registerCeremony() : void  receiveCertification() : void |
| Major | MajorID: String  Name: String  DepartmentID: String  Program: String |  |
| Department | DepartmentID : String  Name : String | printCertification() : void |
| OAA | RoomID: String  Email: String  Phone: String | validdateGraduationApplication() : void  announceGraduationList() : void  requestSignature() : void  receiveCertification() : void  organizeCeremony() : void  announceCeremony() : void |
| SchoolAdministrator | AdminID: String  Name: String  Dob: date  Phone: String  Email: String  Role: String |  |

****

**Code (abstract): using** [**ArgoUML**](https://www.bing.com/videos/search?q=argouml+tutorial&docid=608043554099435551&mid=B642DD9024C6F536AD1EB642DD9024C6F536AD1E&view=detail&FORM=VIRE&msclkid=9cbe87a1cf3711ec93795e04c88c28c9) **to generate**

**Student.java:**

package topic3;

import java.util.Date;

public class Student {

public String Student\_ID,Advisor\_ID, major, Name, Email, Phone, Gender;

public Date Dob;

public Department myDepartment;

public OAA myOAA;

public Major myMajor;

public void submitGraduationApplication() {

}

public void enterGraduationList() {

}

public void registerCeremony() {

}

public void receiveCertification() {

}

}

**Major.java:**

package topic3;

import java.util.Vector;

public class Major {

public String MajorID,Name,DepartmentID,Program;

public Vector one, mydepartment, myStudent;

public Department myDepartment;

/\*\*

\*

\* @element-type Student

\*/

}

**Department.java:**

package topic3;

import java.util.Vector;

public class Department {

public String DepartmentID, Name;

public Vector myStudent, many, myMajor;

/\*\*

\*

\* @element-type Major

\*/

public void printCertification() {

}

}

**OAA.java:**

package topic3;

import java.util.Vector;

public class OAA {

public String RoomID, Email, Phone;

public Vector myStudent, mySchoolAdministrator, many;

/\*\*

\*

\* @element-type Student

\*

\* @element-type SchoolAdministrator

\*/

public void validdateGraduationApplication() {

}

public void announceGraduationList() {

}

public void requestSignature() {

}

public void receiveCertification() {

}

public void organizeCeremony() {

}

public void announceCeremony() {

}

}

**SchoolAdministrator.java:**

package topic3;

import java.util.Date;

import java.util.Vector;

public class SchoolAdministrator {

public String AdminID, Name, Phone, Email, Role;

public Date Dob;

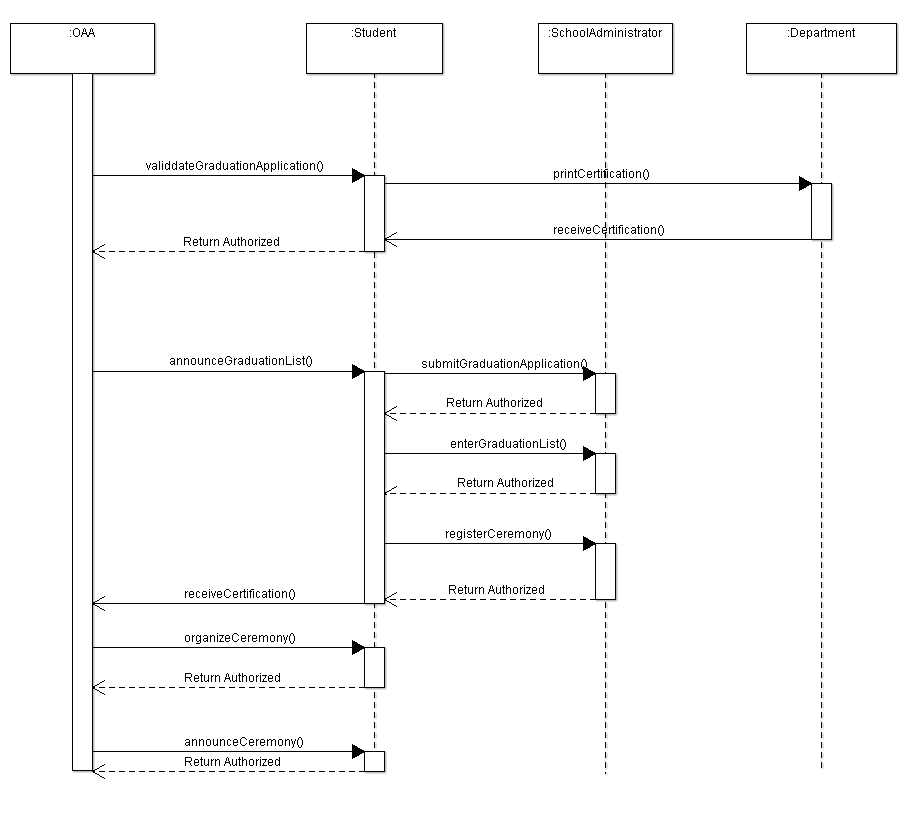
public OAA myOAA;

public Vector one;

}

**4.** **SEQUENCE DIAGRAM:**

First, the major office must have a list of students who will be eligible to graduate once their thesis results have been submitted to the school. Students can then file a graduate application if they do not owe any tuition expenses. After that, the profile information will be reviewed. If your application is approved, you will receive an invitation letter to attend the graduation ceremony.



**Topic 4: The Course Registration Process.**

**1.** **USE CASE DIAGRAM:**

1. **Candidate Actors**

Student: Students who are studying at IU and want to register for courses

OAA: the office is responsible for managing the process

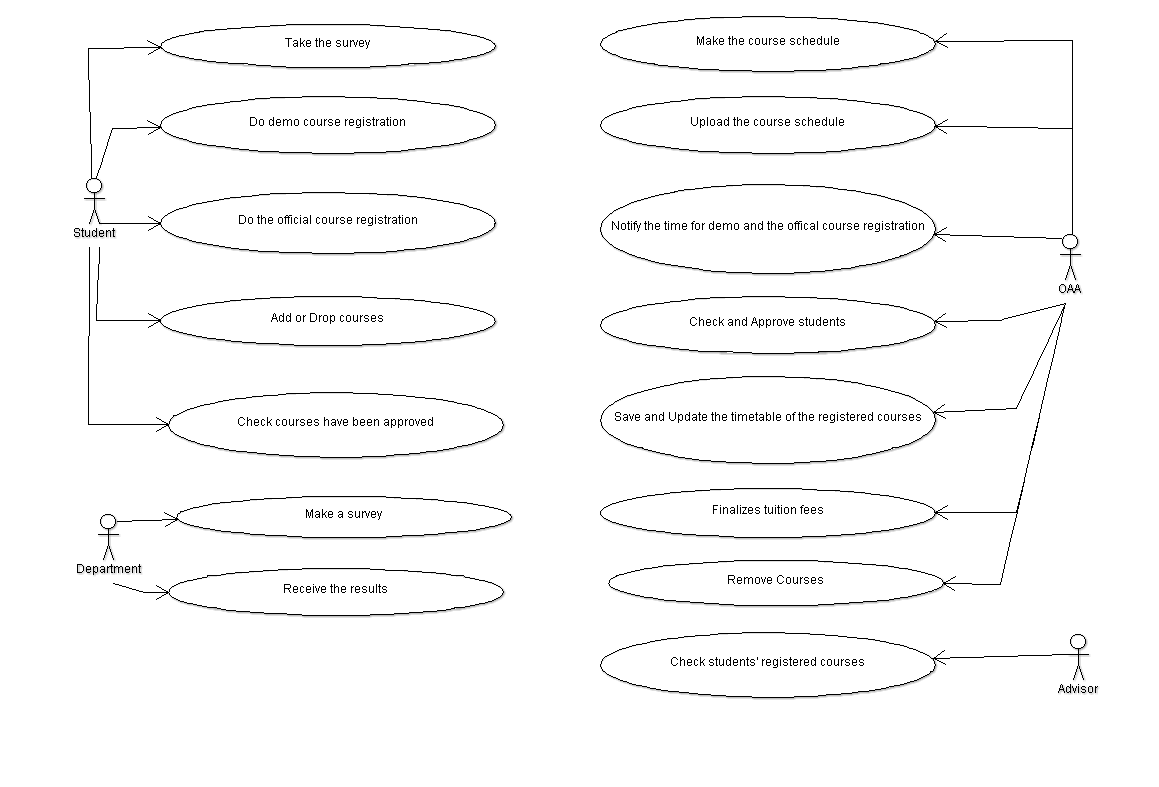
Advisor: A advisor in charge of following and supporting the student

Department: The office in charge of the study process of students in their major

1. **Goal of Actor**

The goal in Context: Students must register for the courses between the minimum and the maximum number of credits for a semester

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Major Office | Successfully get the results from the survey and report successfully to OAA |
| Student | Successfully register courses for next semester |
| OAA | Successfully finalize tuition fees |
| Advisor | Successfully check students' registered courses |

****

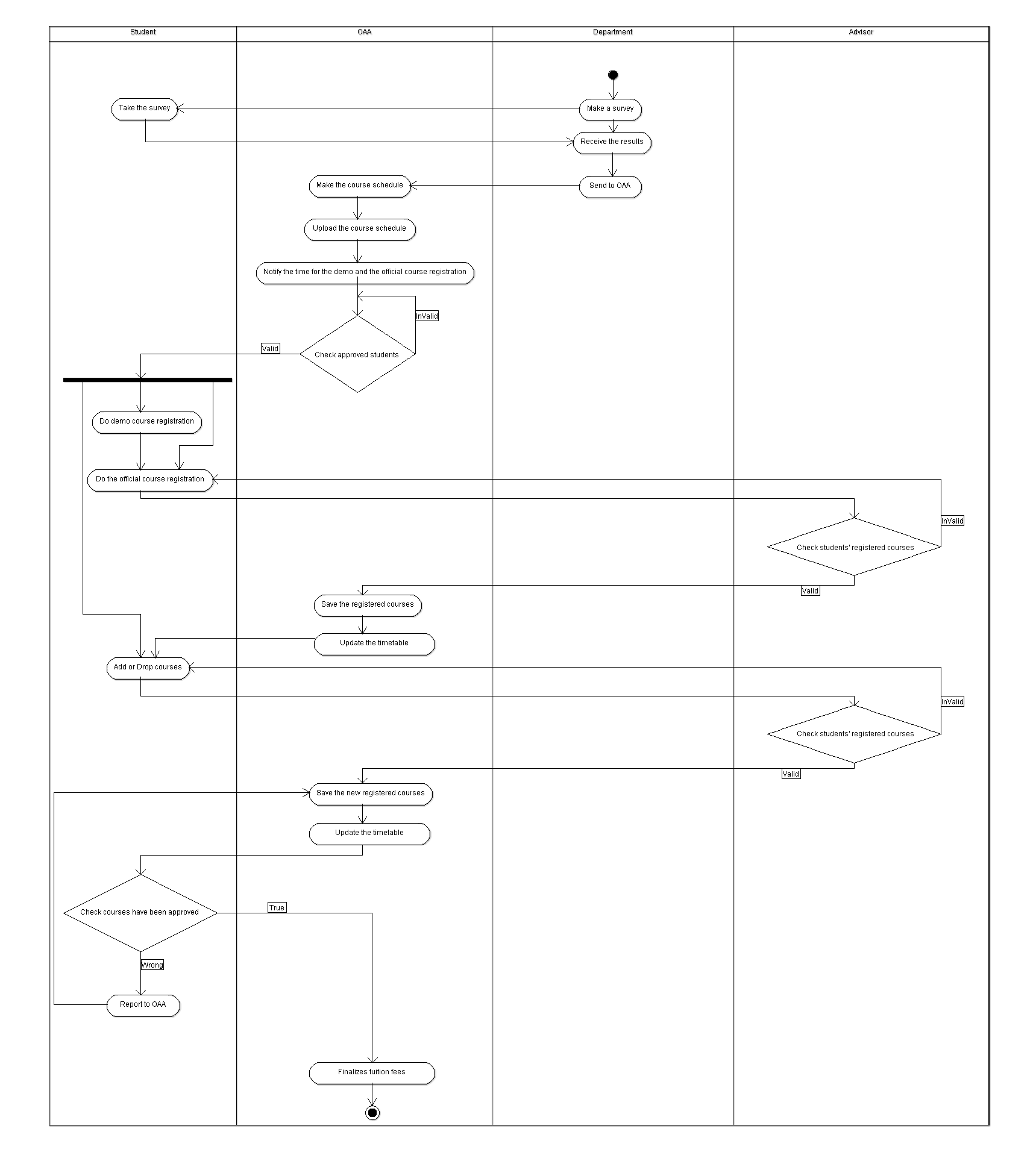
**2.** **ACTIVITY DIAGRAM:**

1. **Identify Candidate use case (Activities)**

|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **Actor** | **Description** |
| UC\_1: Make a survey | Department | Make a survey for students to do to desire the number of classes, students, and lecturers… and post it on the fan page of each school website. |
| UC\_2: Receive the results | Get the results of the survey, then send them to OAA |
| UC\_3: Take the survey | Student | Take the survey and choose the desired course to apply for the next semester |
| UC\_4: Do demo course registration | Students who were approved, do demo course registration to plan the timetable on Edusoft |
| UC\_5: Do the official course registration | Do the official course registration on the Edusoft website and check the automatically updated timetable with successfully registered courses. |
| UC\_6: Add or Drop courses | Students can add or drop courses at the add-drop course week on the Edusoft website and check the new updated timetable |
| UC\_7: Check courses have been approved | Check courses have been approved for the tuition fee |
| UC\_8: Make the course schedule | OAA | Make the course schedule based on the received result of the survey and the schedules of lecturers |
| UC\_9: Upload the course schedule | Upload the course schedule, the number of classes, the number of students, the lecturer, room |
| UC\_10: Notify the time for the demo and the official course registration | Notify the time for the demo and the official course registration on Edusoft |
| UC\_11: Check and Approve students | Check and approve students who satisfy all the conditions to take the registered courses |
| UC\_12: Save and Update the timetable of the registered courses | Save the registered courses on the system and update the timetable |
| UC\_13: Save and Update the timetable again the registered courses | Save and Update the timetable again the registered courses on the system |
| UC\_14: Finalizes tuition fees |  | Finalizes tuition fees for each students base on their registered courses |
| UC\_15: Check students' registered courses | Advisor | Check students' registered courses, if there are no problems, report them to OAA |
| UC\_16: Check again students' registered courses | Check again students' registered courses, if there are no problems, report them to OAA |

1. **Identify the Start point and Endpoint of each use case**

|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **The Start Point** | **The End Point** |
| UC\_1: | Make the survey | Post the survey on the school fan page |
| UC\_2: | Get the results of the survey | Send the survey to OAA |
| UC\_3: | Take the survey | Choose the desired course |
| UC\_4: | Do demo course registration | Plan the timetable |
| UC\_5: | Do the official course registration | Check the automatically updated timetable |
| UC\_6: | Add or Drop courses | Check the new updated timetable |
| UC\_7: | Check courses have been approved | Check courses have been approved |
| UC\_8: | Receive the result of the survey | Make the course schedule |
| UC\_9: | Upload the course schedule | Upload the course schedule |
| UC\_10: | Notify the time for the demo and the official course registration | Notify the time for the demo and the official course registration |
| UC\_11: | Check students who satisfy all the conditions to take the registered courses | Approve students who satisfy all the conditions to take the registered courses |
| UC\_12: | Save the registered courses | Update the timetable |
| UC\_13: | Save again registered courses | Update the timetable again registered courses |
| UC\_14: | Finalizes tuition fees for students | Finalizes tuition fees for students |
| UC\_15: | Check students' registered courses | If there are no problems, report them to OAA |
| UC\_16: | Check again students' registered courses | If there are no problems, report again them to OAA |

****

**3.** **CLASS DIAGRAM:**

|  |  |  |
| --- | --- | --- |
| **Classes** | **Attributes** | **Operations** |
| Lecturer | Lecturer\_ID : String  Name : String  Gender : String  Phone : String  Email : String  Address : String  Dob : Date | receiveCourseList()  receiveClassList() |
| Department | DepartmentID : String  Name : String | createCourse()  receiveClassList()  checkExpandCondition()  requestEditInformation() |
| Major | MajorID: String  Name: String  DepartmentID: String  Program: String |  |
| Course | CourseID: String  Name: String  Credit: int  Price: long  RoomID: String |  |
| CourseRegistration | dateRegistration : date |  |
| Student | StudentID: String  Advisor\_ID: String  Name: String  Dob: date  Major: String |  |
| OAA | RoomID : String  Phone : String  Email : String | receiveCourseInformation() : void  arrangeClasses() : void  editInformation() : void  editCourse() : void  editClass() : void  removeClass() : void |
| Advisor | Advisor\_ID: String  Lecturer\_ID: String  Name: String  Dob: date  Gender: String  Email: String  Phone: String | approveRegistration() : void |

**Code (abstract): using** [**ArgoUML**](https://www.bing.com/videos/search?q=argouml+tutorial&docid=608043554099435551&mid=B642DD9024C6F536AD1EB642DD9024C6F536AD1E&view=detail&FORM=VIRE&msclkid=9cbe87a1cf3711ec93795e04c88c28c9) **to generate**

**Lecturer.java:**

package topic4;

import java.util.Date;

import java.util.Vector;

public class Lecturer {

public String Lecturer\_ID, Name, Gender, Phone, Email, Address;

public Date Dob;

public Vector mydepartment, one;

public Department myDepartment;

public void receiveCourseList() {

}

public void receiveClassList() {

}

}

**Department.java:**

package topic4;

import java.util.Vector;

public class Department {

public String DepartmentID, Name;

public Vector myLecturer, many, myMajor;

public OAA myOAA;

/\*\*

\*

\* @element-type Lecturer

\*/

/\*\*

\*

\* @element-type Major

\*/

public void createCourse() {

}

public void receiveClassList() {

}

public void checkExpandCondition() {

}

public void requestEditInformation() {

}

}

**Major.java:**

package topic4;

import java.util.Vector;

public class Major {

public String MajorID, Name, DepartmentID, Program;

public Vector myCourse, one, mydepartment, many, myStudent;

public Department myDepartment;

/\*\*

\*

\* @element-type Course

\*/

}

**Course.java:**

package topic4;

import java.util.Vector;

public class Course extends Major {

public String CourseID, Name, RoomID;

public int Credit;

public long Price;

public Vector mymajor, one, many, myCourseRegistration;

public Major myMajor;

/\*\*

\*

\* @element-type CourseRegistration

\*/

}

**CourseRegistration.java:**

package topic4;

import java.util.Date;

import java.util.Vector;

public class CourseRegistration {

public Date dateRegistration;

public Vector one;

public Course myCourse;

public Student myStudent;

}

**Student.java:**

package topic4;

import java.util.Date;

import java.util.Vector;

public class Student {

public String StudentID, Advisor\_ID, Name, Major;

public Date Dob;

/\*\*

\*

\* @element-type CourseRegistration

\*/

public Vector myCourseRegistration, one, many;

public Major myMajor;

public Advisor myAdvisor;

public OAA myOAA;

public void receiveClassList() {

}

public void login() {

}

public void demoRegistration() {

}

public void requestExtraClasses() {

}

public void receiveOfficialSchedule() {

}

}

**OAA.java:**

package topic4;

import java.util.Vector;

public class OAA {

public String RoomID, Phone, Email;

public Department myDepartment;

/\*\*

\*

\* @element-type Student

\*/

public Vector myStudent, many;

public void receiveCourseInformation() {

}

public void arrangeClasses() {

}

public void editInformation() {

}

public void editCourse() {

}

public void editClass() {

}

public void removeClass() {

}

}

**Advisor.java:**

package topic4;

import java.util.Date;

import java.util.Vector;

public class Advisor extends Lecturer {

public String Advisor\_ID, Lecturer\_ID, Name, Gender, Email, Phone;

public Date Dob;

public Vector many, myStudent;

/\*\*

\*

\* @element-type Student

\*/

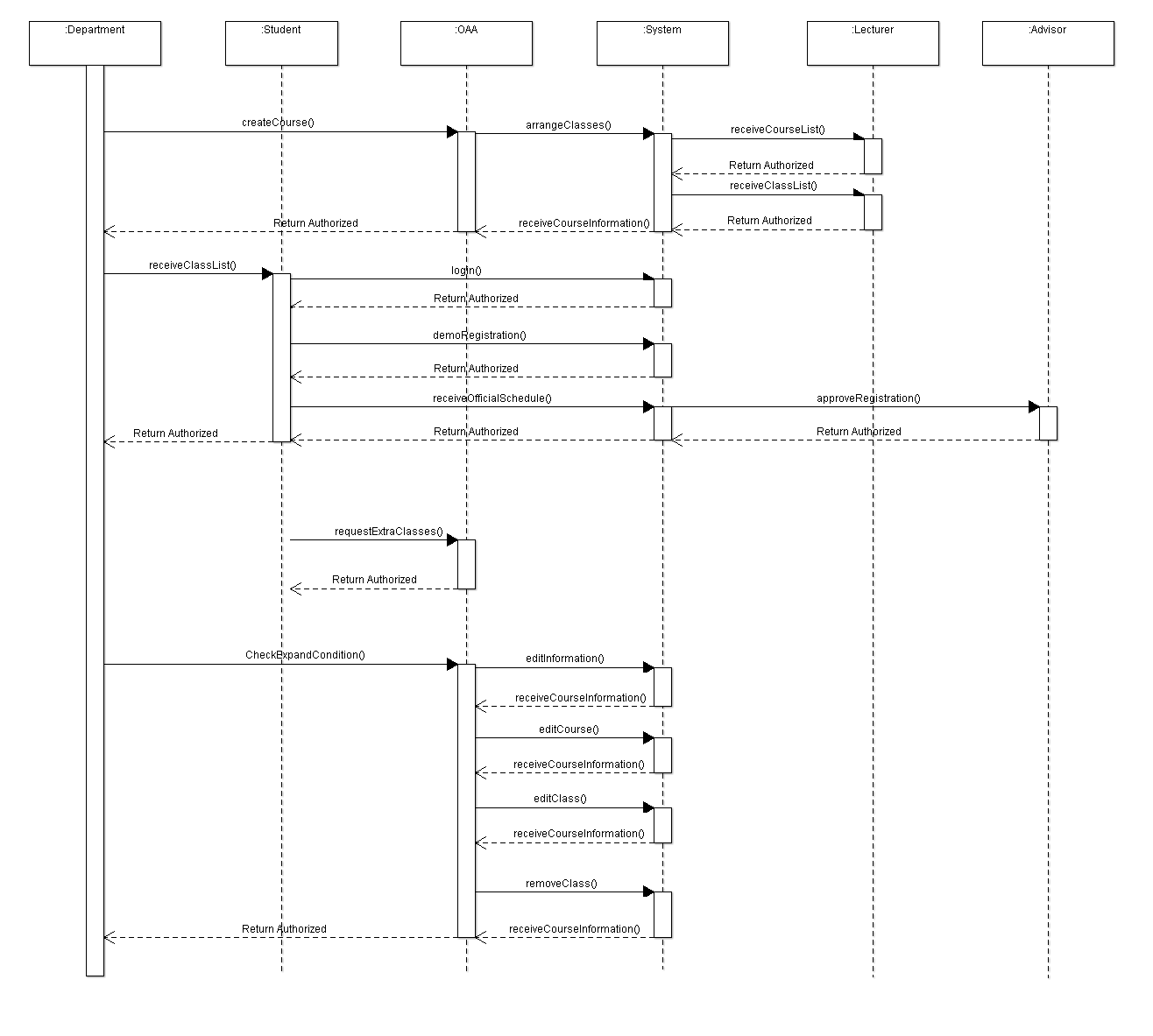
public void approveRegistration() {

}

}

**4.** **SEQUENCE DIAGRAM:**

First, the major office will solicit student feedback on the amount of subjects they wish to enroll in. The school will schedule students to register for the course based on the survey. Only qualified students are eligible to apply for the course. The next step is to verify that the course is genuine before saving it to the system. The next step is to see if the course has been authorized, and if it has, compute the tuition amount.



**Topic 5: The Tuition Fee Payment Process**

**1. USE CASE DIAGRAM:**

1. **Candidate Actors**

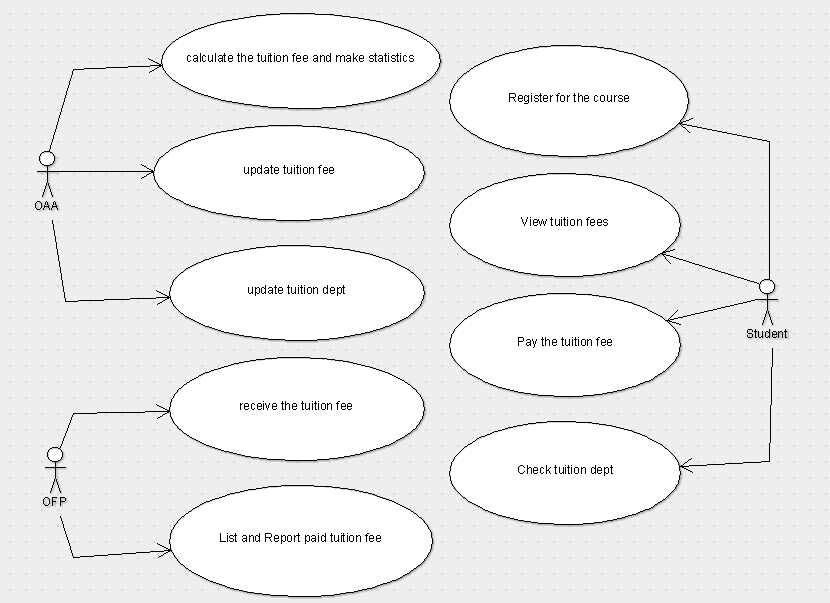
Student: Students who are studying at IU and want to pay the tuition fee after completing the course registration of the Add-Drop week.

OFP: The office receives the student's tuition fee and generates a list report.

1. **Goal of Actor**

Goal in Context: Students must pay tuition for the courses before the Midterm exam week to be allowed to take the exam.

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Student | Successfully complete the tuition fee |
| OFP | The list of students with paid tuition fees was successfully reported. |

****

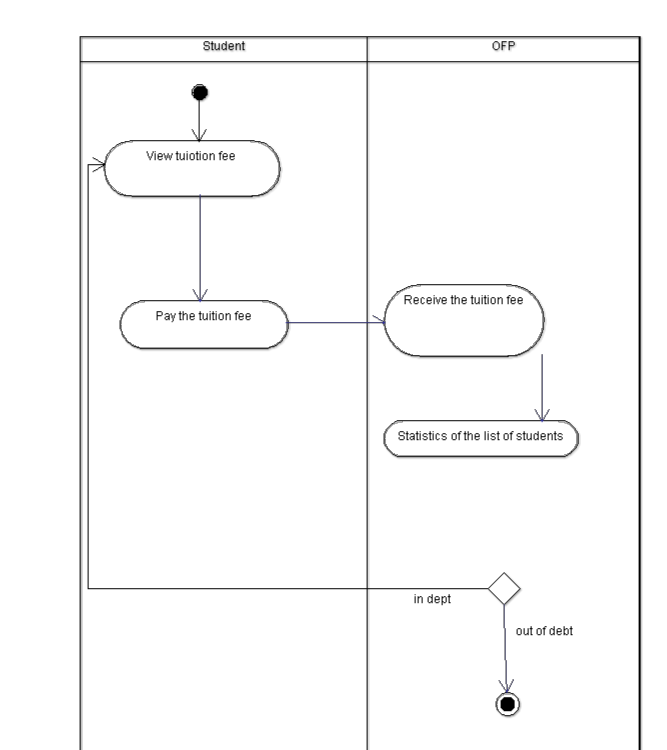
**2. ACTIVITY DIAGRAM:**

1. **Identify Candidate use case (Activities)**

|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **Actor** | **Description** |
| UC\_1: Register for the course | Student | Begin by registering for the course and editing the last course of the week by "Add-Dropping" enough credits as specified on the edusoft website. |
| UC\_2: View tuition fees | Students go to the edusoft website's tuition section to view tuition fees. |
| UC\_3: Pay the tuition fee | Students pay the full tuition fee to OFP before the midterm exam or through OFP International University's banking system and receive a confirmation bill. |
| UC\_4: Check tuition dept | Check the student's outstanding tuition amount on the edusoft website.  The tuition fee payment process is successful if the outstanding amount is 0 VND. |
| UC\_5: Receive the tuition fee | OFP | Receive the payment fee and return the student's payment confirmation bill. |
| UC\_6: List and Report paid tuition fee | Statistics of students who have paid tuition fees, or the amount owed by each student, and notification to OAA |

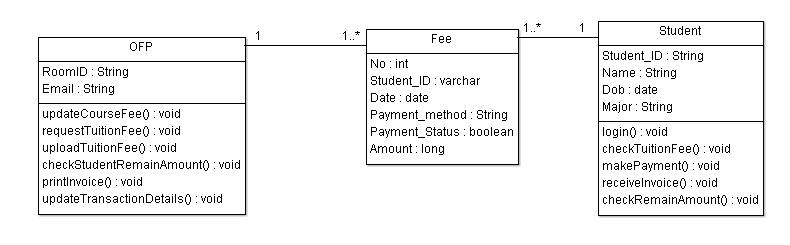
1. **Identify the Start point and Endpoint of each use case**

|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **The Start Point** | **The End Point** |
| UC\_1 | Register for the course | Edit the last course of the week "Add-Drop" with a valid number of credits |
| UC\_2 | Login to the edusoft website | Check tuition fees in the tuition section |
| UC\_3 | Pay the full tuition fee directly to OFP before the midterm exam or banking of OFP International University | Receive confirmation bill |
| UC\_4 | Login to the edusoft website | Check the tuition dept in the tuition section. If the outstanding amount is 0 VND, the tuition fee payment process is successful |
| UC\_5 | Receive the payment fee | Return the payment confirmation bill to the Student |
| UC\_6 | Statistics of the list of students who have paid tuition fees, or the amount owed by each student | Report to OAA |

****

**3. CLASS DIAGRAM**

|  |  |  |
| --- | --- | --- |
| **Classes** | **Attributes** | **Operations** |
| OFP | RoomID: String  Email: String | updateCourseFee(): void  requestTuitionFee(): void  uploadTuitionFee(): void  checkStudentRemainAmount(): void  printlnvoice(): void  updateTransactionDetails(): void |
| Fee | No: int  Student\_ID: varchar  Date: date  Payment\_method: String  Payment\_Status: Boolean  Amount: long |  |
| Student | Stundent\_ID: String  Name: String  Dob: date  Major: String | Login(): void  checkTuitionFee(): void  makePayment(): void  receivelnvoice(): void  checkRemainAmount(): void |

****

**Code (abstract): using** [**ArgoUML**](https://www.bing.com/videos/search?q=argouml+tutorial&docid=608043554099435551&mid=B642DD9024C6F536AD1EB642DD9024C6F536AD1E&view=detail&FORM=VIRE&msclkid=9cbe87a1cf3711ec93795e04c88c28c9) **to generate**

**Fee.java**

package topic5;

import java.util.Vector;

public class Fee {

public int No;

public varchar Student\_ID;

public date Date;

public String Payment\_method;

public boolean Payment\_Status;

public long Amount;

public Vector one,myOFP,myStudent;

}

**OFP.java**

package topic5;

import java.util.Vector;

public class OFP {

public String RoomID;

public String Email;

public Integer newAttr;

public Vector many,myFee;

/\*\*

\*

\* @element-type Fee

\*/

public void updateCourseFee() {

}

public void requestTuitionFee() {

}

public void uploadTuitionFee() {

}

public void checkStudentRemainAmount() {

}

public void printInvoice() {

}

public void updateTransactionDetails() {

}

}

**Student.java**

package topic5;

import java.util.Vector;

public class Student {

public String Student\_ID,Name,Major;

public date Dob;

public Vector many,myFee;

/\*\*

\*

\* @element-type Fee

\*/

public void login() {

}

public void checkTuitionFee() {

}

public void makePayment() {

}

public void receiveInvoice() {

}

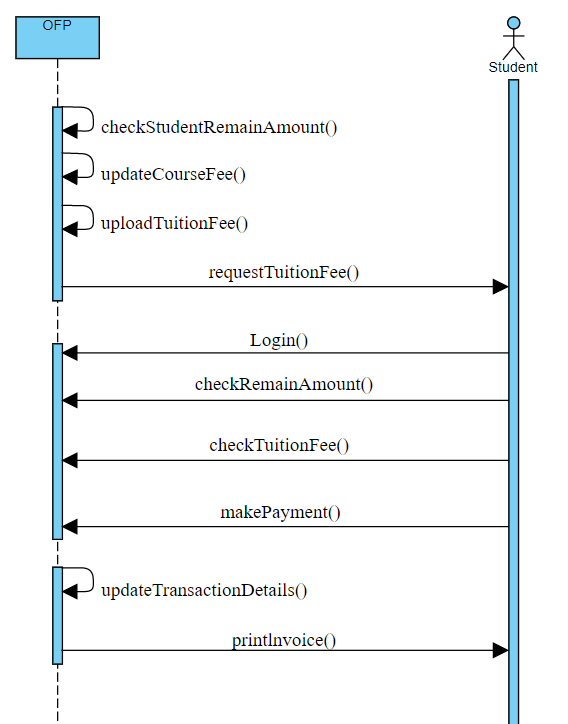
public void checkRemainAmount() {

}

}

**4.** **SEQUENCE DIAGRAM:**

OFP will check the student's debt from the previous semester, update the course fee as soon as the student completes the course modification during the add-drop week, and request payment of the tuition fee. Students who need to pay tuition fees will log in and check the payable amounts, which will include this semester's tuition and the amount owed by the student; after completing the check and payment, the student will receive an invoice.



**Topic 6: The Grading Process for students**

**1. USE CASE DIAGRAM:**

1. **Candidate Actors**

Student: Students who are studying at IU complete quizzes, homework, and midterm and final examinations.

OAA: The office is responsible for managing the process.

Lecturer: Lecturer Who is teaching at IU and mark for the quizzes, homework, midterm, and final examinations.

1. **Goal of Actor**

Goal in Context: Students must pay tuition for the courses before the Midterm exam week to be allowed to take the exam.

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Student | Successfully check the results |
| OAA | Successfully update the final assessment results |
| OFP | Successfully update all scores & distribution percent |

****

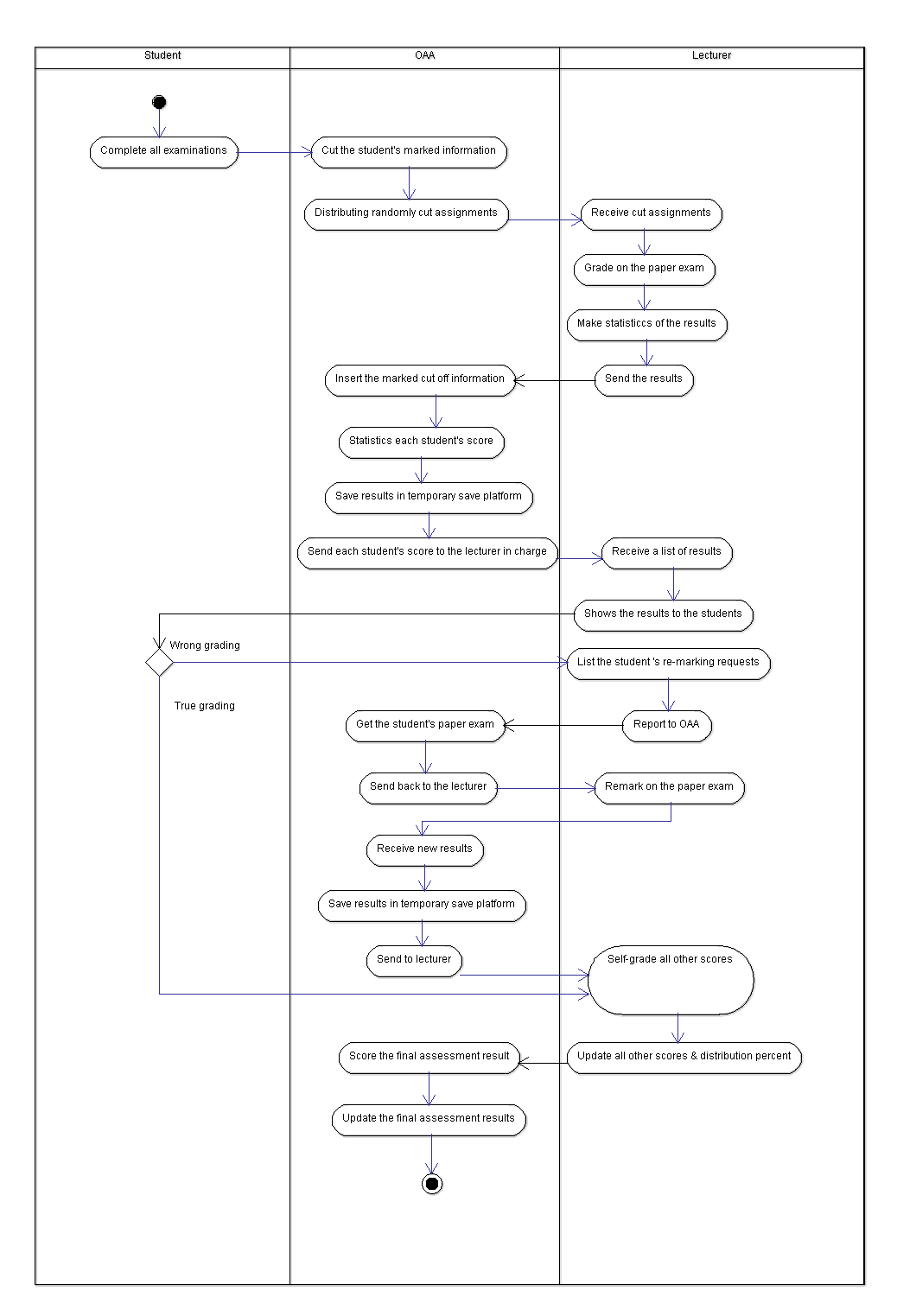
**2. ACTIVITY DIAGRAM:**

1. **Identify Candidate use case (Activities)**

|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **Actor** | **Description** |
| UC\_1: Complete all examinations | Student | Quizzes, homework, midterm, and final exams are all completed by students. |
| UC\_2: Check their results | Students review their grades. If they want to point out a mistake, they must notify the lecturer in charge; otherwise, they should move on to UC-15. |
| UC\_3: Cut the student's marked information | OAA | OAA will cut the student's marked information from the student's paper exam for official college testing. |
| UC\_4: Distributing randomly cut assignments | distributing cut assignments to lecturers at random |
| UC\_5: Insert the marked cut-off information | The marked cut-off information will be inserted into the graded student's paper exam by OAA. |
| UC\_6: Statistics and sends each student's score to the lecturer in charge | Each student's score is sent back to the lecturer in charge of that student by OAA statistics.  OAA saves them in the platform for temporary storage. |
| UC\_7: Get the student's paper exam that requires remarking and send it back to the lecturer | When OAA receives the list, it searches the archives for the correct student's paper exam that needs to be remarked. Then return it to the lecturer who informed you of the remark. |
| UC\_8: Receives new results | The lecturer in charge sends new results to OAA, who saves them in a temporary save platform. |
| UC\_9: Scores the final assessment result | OAA receives statistics and scores the final assessment results using the lecturer's distribution points. |
| UC\_10: Updates the final assessment results | On the edusoft website and the system, OAA updates each student's final assessment results. |
| UC\_11: Mark on the paper exam | Lecturer | Make notes on the exam paper, compile statistics, and send the results to OAA. |
| UC\_12: Shows the results to the students | The lecturer receives a list of OAA results and distributes it to the students under his or her supervision. |
| UC\_13: Lists the student's re-marking requests | The lecturer receives lists and reports the student's re-marking requests to OAA. |
| UC\_14: Remarks on the paper exam | The lecturer provides feedback on the paper exam and updates the OAA results. |
| UC\_15: Self-grade and update all other scores & distribution scores | The lecturer will manually self-grade in-class Quizzes and update all other scores and distribution percentages on the edusoft website ahead of time. |

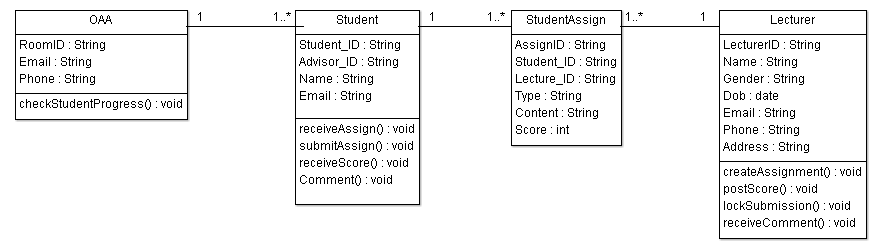
**b.** **Identify the Start point and Endpoint of each use case**

|  |  |  |
| --- | --- | --- |
| **UC\_ID** | **The Start Point** | **The End Point** |
| UC\_1: | Students complete all examinations | Students complete all examinations |
| UC\_2: | Students check their results | If the mistake is to be remarked, they must notify the lecturer in charge |
| UC\_3: | Organize the official exam | Cut the student's marked information from the student's paper exam |
| UC\_4: | Distributing randomly cut assignments to lecturers | Distributing randomly cut assignments to lecturers |
| UC\_5: | Receive the graded student's paper exam | Insert the marked cut off information into the graded student's paper exam |
| UC\_6: | Statistics and sends each student's score back to the lecturer in charge of that student | Save each student's score in the temporarily save platform |
| UC\_7: | Receives the list and goes to the archive to get the correct student’s paper exam that requires remarking | Send the student's paper exam back to the lecturer who notified the re-mark. |
| UC\_8: | Receives new results from the lecturer in charge | Save each new student's score in the temporarily save platform |
| UC\_9: | Receives statistics | Scores for the final assessment |
| UC\_10: | Updates the final assessment results on the edusoft website | Updates the final assessment results on the system |
| UC\_11: | Mark the test, Make statistics | Send the results to OAA |
| UC\_12: | Receives a list of results from OAA | Shows the results to the students he/she oversees |
| UC\_13: | Receives, Lists the student's re-marking requests | Reports the list of student's re-marking requests to OAA |
| UC\_14: | Remarks on the paper exam | Updates the new results for OAA |
| UC\_15: | Self-grade all other scores and distribution scores of each student | Update all other scores and distribution percent of each student |

****

**3. CLASS DIAGRAM:**

|  |  |  |
| --- | --- | --- |
| **Classes** | **Attributes** | **Operations** |
| OAA | RoomID: String  Email: String  Phone: String | CheckStudentProgress(): void |
| Student | Student\_ID: varchar  Advisor\_ID: String  Name: String  Email: String | receiveAssign(): void  submitAssign(): void  receiveScore(): void  comment(): void |
| StudentAssign | AssignID: String  Student\_ID: String  Lecture\_ID: String  Type: String  Content: String  Score: int |  |
| Lecturer | LectureID: String  Name: String  Gender: String  Dob: date  Email: String  Phone: String  Address: String | createAssignment(): void  postscore(): void  lockSubmission(): void  receiveComment(): void |

****

**Code (abstract): using** [**ArgoUML**](https://www.bing.com/videos/search?q=argouml+tutorial&docid=608043554099435551&mid=B642DD9024C6F536AD1EB642DD9024C6F536AD1E&view=detail&FORM=VIRE&msclkid=9cbe87a1cf3711ec93795e04c88c28c9) **to generate**

**Lecture.java**

package topic6;

import java.util.Vector;

import java.util.Date;

public class Lecturer {

public String LecturerID,Name,Gender,Email,Phone,Address;

public Date Dob;

/\*\*

\*

\* @element-type StudentAssign

\*/

public Vector many,myStudentAssign;

public void createAssignment() {

}

public void postScore() {

}

public void lockSubmission() {

}

public void receiveComment() {

}

}

**OAA.java**

package topic6;

import java.util.Vector;

public class OAA {

public String RoomID,Email,Phone;

/\*\*

\*

\* @element-type Student

\*/

public Vector many,myStudent;

public void checkStudentProgress() {

}

}

**Student.java**

package topic6;

import java.util.Vector;

public class Student {

public String Student\_ID,Advisor\_ID,Name,Email;

public OAA one,myStudentAssign,many;

/\*\*

\*

\* @element-type StudentAssign

\*/

public void receiveAssign() {

}

public void submitAssign() {

}

public void receiveScore() {

}

public void Comment() {

}

}

**StudentAssign.java**

package topic6;

public class StudentAssign {

public String AssignID,Student\_ID,Lecture\_ID,Type,Content;

public int Score;

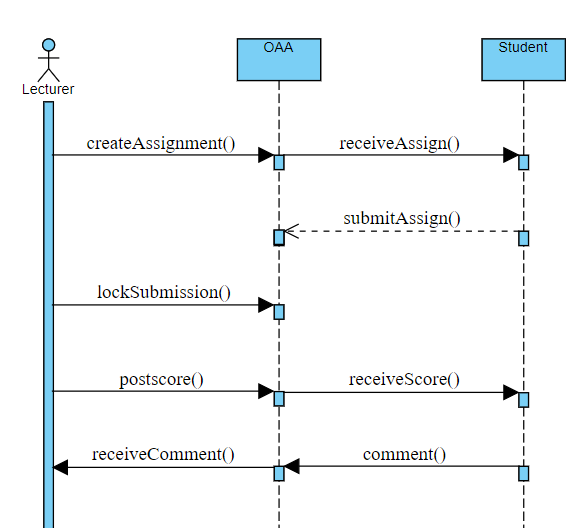
public Student myStudent;

public Lecturer myLecturer;

}

**4. SEQUENCE DIAGRAM:**

The Lecturer creates the Assignment and commits it to OAA, which then distributes it to the students. The student will return to OAA after working with Assign. When the submission deadline approaches, the lecturer can lock it. After grading, the Assign Lecturer submits the score to OAA, where the student can review it and leave comments. Through the OAA, the lecturer can receive feedback.

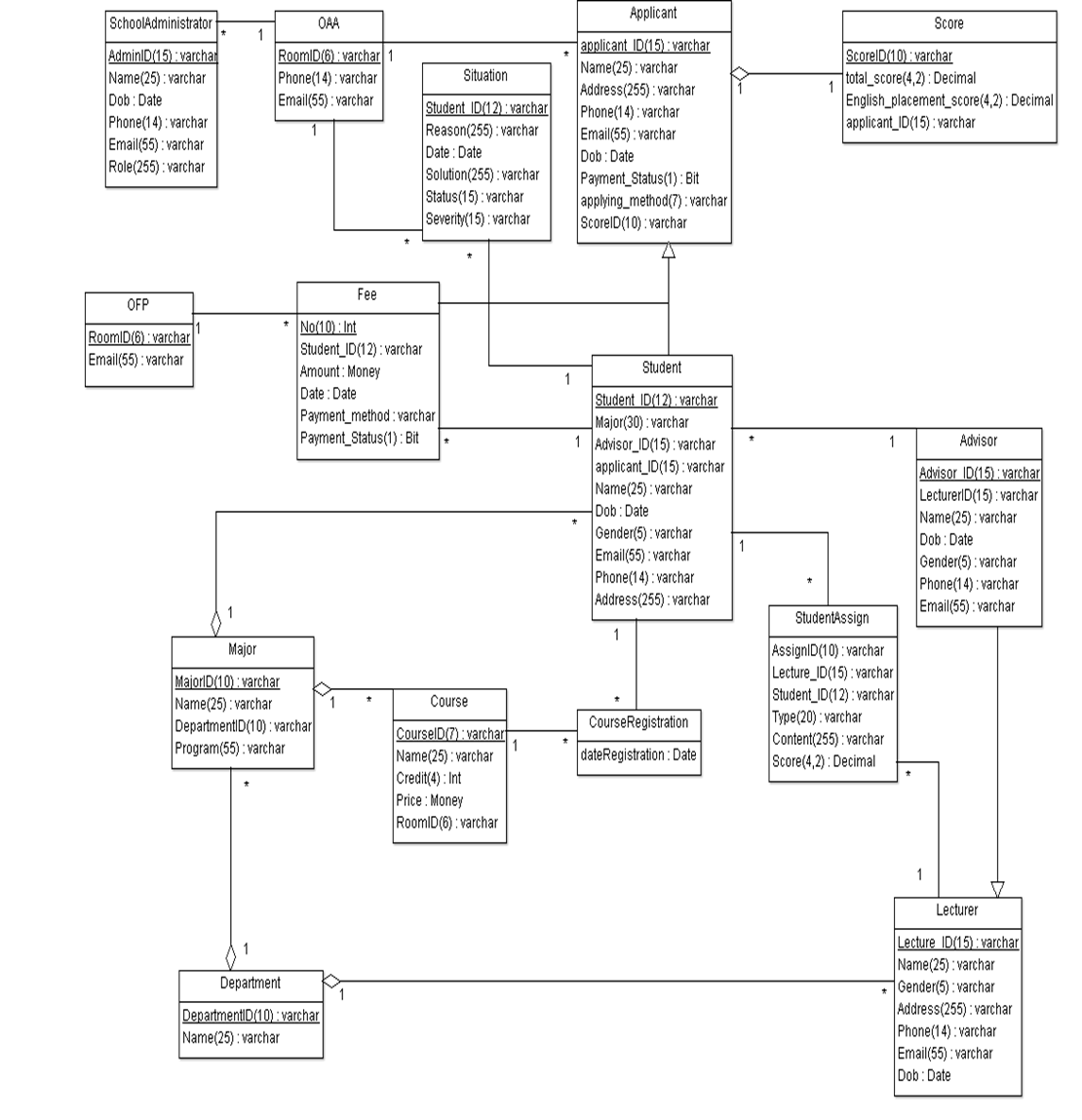
****

**Database designs for 6 topics**

The normal form of the relationship system has the 2NF because all entities have a primary key (1NF) and non-key attributes rely on the key attribute from which the database is retrieved. However, to achieve 3NF, we would need to further normalize the system. To accomplish this, we must make the non-key attribute directly dependent on the key attribute.

**Based on the relationship system and the normalization, we can come up with this database:**

Because all entities have a primary key (1NF) and non-key attributes rely on the key attribute from which the database is retrieved, the normal form of the relationship system has the 2NF. However, to achieve 3NF, the system would need to be further normalized. To achieve this, the non-key attribute must be directly dependent on the key attribute.



**Web interface (frontend)/Backend**

**Code:**

package src;

import java.awt.Color;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

public class Frame extends JFrame{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

public static void main(String[] args) {

new Frame().setVisible(true);

}

public Frame() {

super("OOAD Lab 7&8");

initialize();

}

private void initialize() {

setForeground(Color.CYAN);

setLayout(null);

JLabel iupic = new JLabel("");

iupic.setIcon(new ImageIcon(ClassLoader.getSystemResource("Icons/remain1.png")));

iupic.setBounds(0, 0, 600, 400);

add(iupic);

JMenuBar menuBar = new JMenuBar();

setJMenuBar(menuBar);

JMenu iu = new JMenu("VNU-HCMC-UNIVERSITY");

iu.setForeground(Color.BLUE);

menuBar.add(iu);

JMenuItem Enrollment = new JMenuItem("New Student Enrollment Process");

iu.add(Enrollment);

JMenuItem Suspensions = new JMenuItem("Student Stand-downs/Suspensions/Exclusions/Expulsions Process");

iu.add(Suspensions);

JMenuItem Graduation = new JMenuItem("Graduation Application Process");

iu.add(Graduation);

JMenuItem Course = new JMenuItem("Course Registration Process");

iu.add(Course);

JMenuItem Tuition = new JMenuItem("Tuition Fee Payment Process");

iu.add(Tuition);

JMenuItem Grading = new JMenuItem("Grading Process for Student Process");

iu.add(Grading);

JMenu members = new JMenu("Team Member");

members.setForeground(Color.BLUE);

menuBar.add(members);

JMenuItem nam = new JMenuItem("Lê Thanh Phương Nam - ITITWE19025");

members.add(nam);

JMenuItem duong = new JMenuItem("Đặng Khải Dươnng - ITITWE19010");

members.add(duong);

JMenuItem thong = new JMenuItem("Đinh Bình Thanh Thông - ITITWE19027");

members.add(thong);

setSize(600,400);

setVisible(true);

}

}

**Result**

