Case Study:
UAMS
Domain Model Solution

Identify the Classes

Academic branch offers different programs within different departments each program has a degree title and duration of degree.
Student Apply for admission in University and provides his/her name, age, FSC, and Ecat Marks and selects any number of preferences among the available programs.
Admission department prepares a merit list according to the highest merit and available seats and registers selected students in the program.
Academic Branch also add subjects for each program. A subject have subject code, credit hours, subjectType, and subjectFee A Program cannot have more than 20 Credit hour subjects.A Student Registers multiple subjects but only from his enrolled program's subject but he/she can not take more than 9 credit hours.
Fee department generate fees according to registered subjects of the students.

Step 1: Identify the Classes which have attributes

Academic branch offers different programs within different departments each program has a degree title and duration of degree.

Student Apply for admission in University and provides his/her name, age, FSC, and Ecat Marks and selects any number of preferences among the available programs.

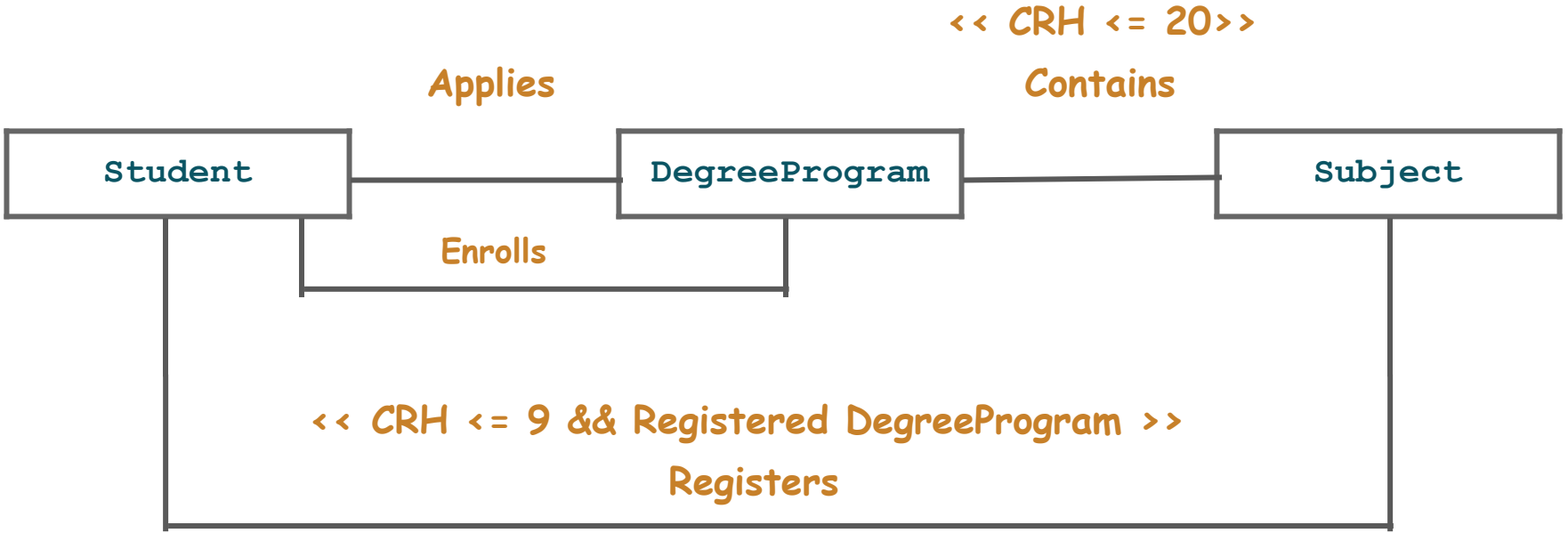
Admission department prepares a merit list according to the highest merit and available seats and registers selected students in the program.

Academic Branch also add subjects for each program. A subject have subject code, credit hours, subjectType, and subjectFee. A Program cannot have more than 20 Credit hour subjects.A Student Registers multiple subjects but only from his enrolled program's subject but he/she can not take more than 9 credit hours.

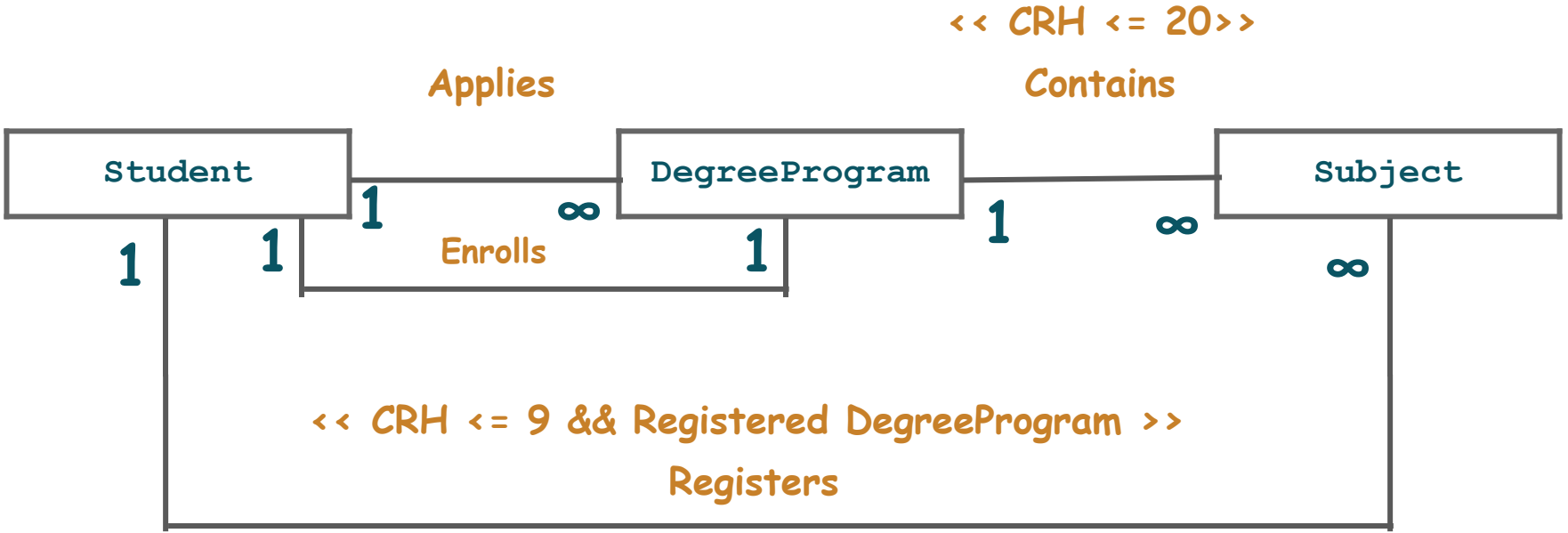
Fee department generate fees according to registered subjects of the students.

Step 2: Draw Domain Model: Write Classes name only

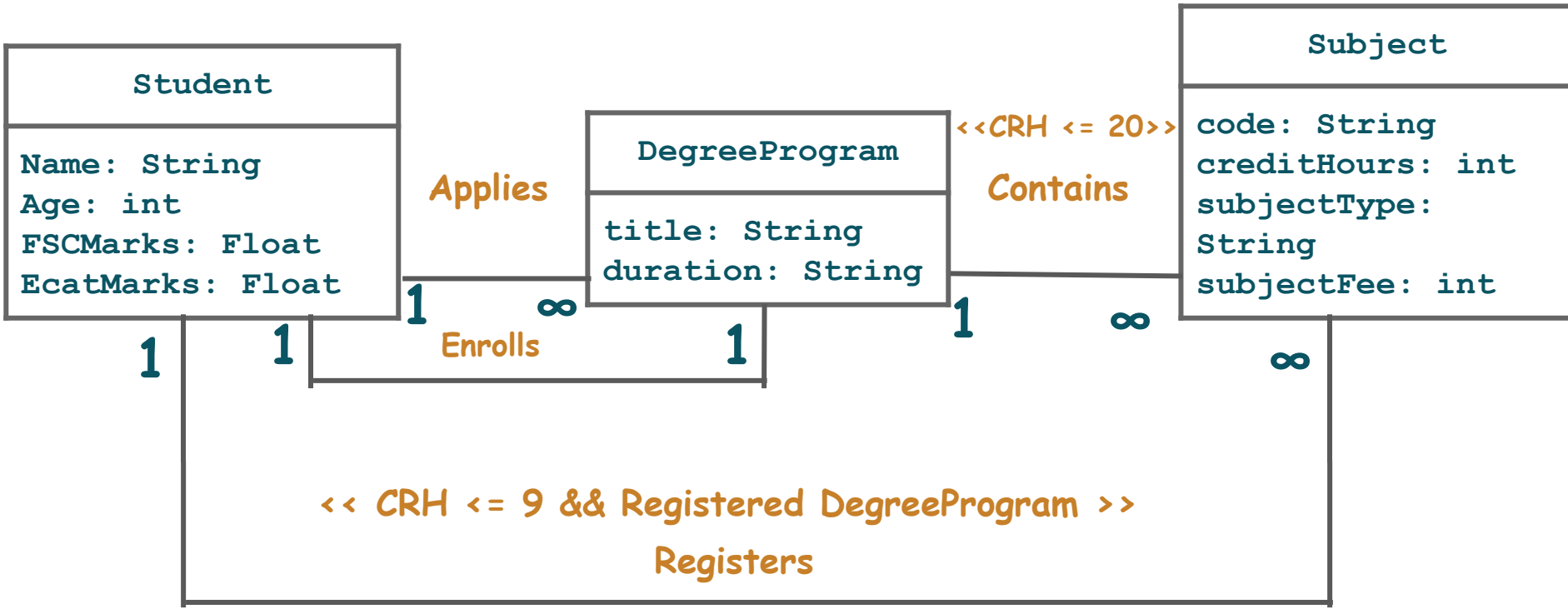
Step 3: Draw Domain Model: Add Relations and Constraints



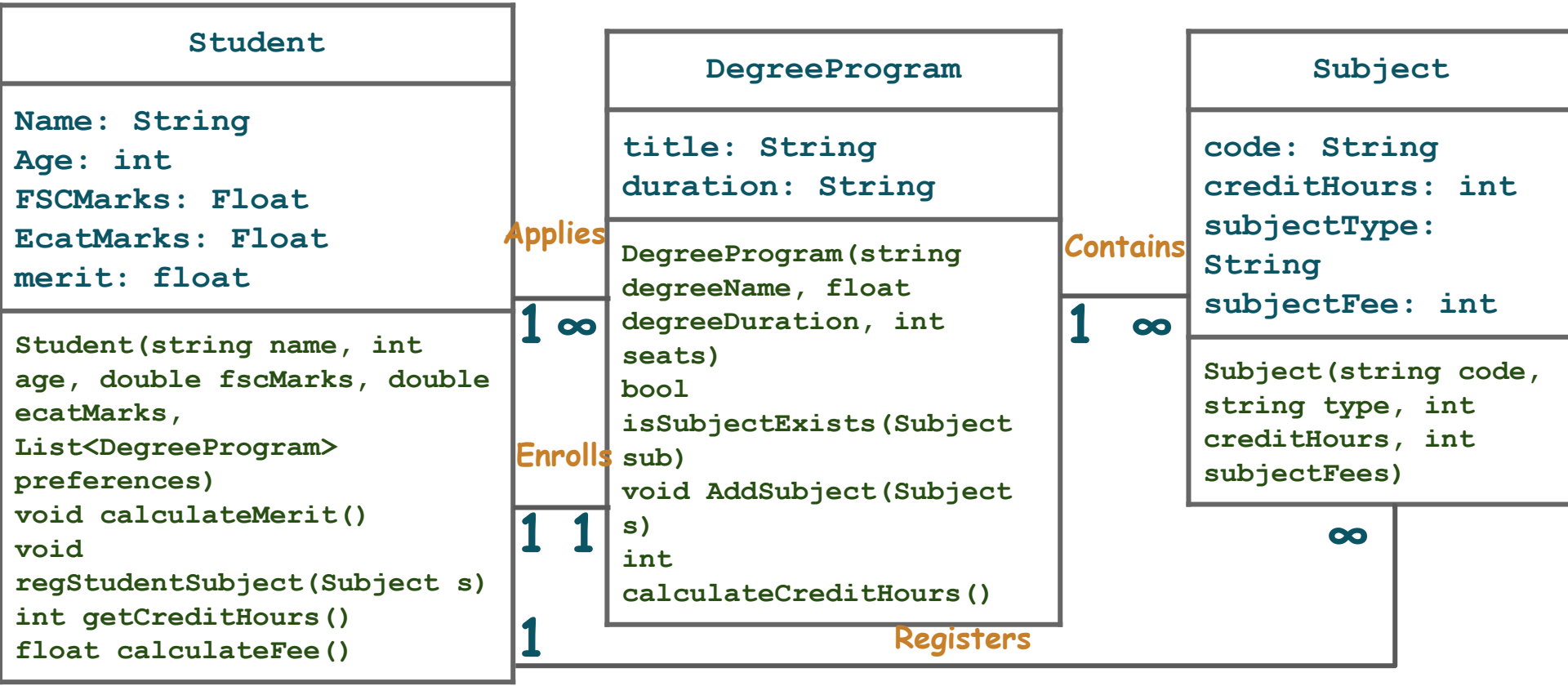
Step 4: Draw Domain Model: Add Multiplicity



Step 5: Draw Class Diagram: Add Attributes



Step 6: Draw Class Diagram: Add Functions



Step 7: Convert Class Diagram to Code

Student Class Attributes

```
class Student
{
    public string name;
    public int age;
    public double fscMarks;
    public double ecatMarks;
    public double merit;
    public List<DegreeProgram> preferences;
    public List<Subject> regSubject;
    public DegreeProgram regDegree;
}
```

Step 7: Convert Class Diagram to Code

Student Class Behaviours

```
    public Student(string name, int
age, double fscMarks, double ecatMarks,
List<DegreeProgram> preferences)
    {
    }

    public void calculateMerit()
    {
    }

    public int getCreditHours()
    {
    }

    public float calculateFee()
    {
    }
```

```
public void regStudentSubject(Subject s)
    {
        int stCH = getCreditHours();
        if (regDegree != null &&
regDegree.isSubjectExists(s) && stCH +
s.creditHours <= 9)
        {
            regSubject.Add(s);
        }
        else
        {
            Console.WriteLine("A student
cannot have more than 9 CH or Wrong
Subject");
        }
    }
```

Step 7: Convert Class Diagram to Code

Student Class Behaviours

```
    public Student(string name, int
age, double fscMarks, double ecatMarks,
List<DegreeProgram> preferences)
    {
    }

    public void calculateMerit()
    {
    }

    public int getCreditHours()
    {
    }

    public float calculateFee()
    {
    }
```

```
public void regStudentSubject(Subject s)
    {
        int stCH = getCreditHours();
        if (regDegree != null &&
regDegree.isSubjectExists(s) && stCH +
s.creditHours <= 9)
        {
            regSubject.Add(s);
        }
        else
        {
            Console.WriteLine("A student
cannot have more than 9 CH or Wrong
Subject");
        }
    }
```

What is Wrong with this Approach?

Step 7: Convert Class Diagram to Code

Subject Class Attributes and Behaviours

```
class Subject
{
    public string code;
    public string type;
    public int creditHours;
    public int subjectFees;

    public Subject(string code, string type,
int creditHours, int subjectFees)
    {
        this.code = code;
        this.type = type;
        this.creditHours = creditHours;
        this.subjectFees = subjectFees;
    }
}
```

Step 7: Convert Class Diagram to Code

DegreeProgram Class Attributes and Behaviours

```
class DegreeProgram
{
    public string degreeName;
    public float degreeDuration;
    public List<Subject> subjects;
    public int seats;
}
```

Step 7: Convert Class Diagram to Code

DegreeProgram Class Behaviours

```
public DegreeProgram(string degreeName, float
degreeDuration, int seats)
    {
        this.degreeName = degreeName;
        this.degreeDuration = degreeDuration;
        this.seats = seats;
        subjects = new List<Subject>();
    }

    public int calculateCreditHours()
    {
    }

     public bool isSubjectExists(Subject sub)
    {
    }
}
```

```
    public void AddSubject(Subject s)
    {
        int creditHours =
calculateCreditHours();
        if(creditHours + s.creditHours
<= 20)
        {
            subjects.Add(s);
        }
        else
        {
            Console.WriteLine("20 credit
hour limit exceeded");
        }
    }
```

Step 8: Main

```
static List<Student> studentList = new List<Student>();
static List<Student> sortedStudentList = new List<Student>();
static List<DegreeProgram> programList = new
List<DegreeProgram>();
static void Main(string[] args)
{
    int option;
    do
    {
        option = Menu();
        clearScreen();
        if (option == 1)
        {
            if (programList.Count > 0)
            {
                Student s = takeInputForStudent();
                addIntoStudentList(s);
            }

        }
        else if (option == 2)
        {
            DegreeProgram d = takeInputForDegree();
            addIntoDegreeList(d);
        }
```

```
else if (option == 3){
    sortStudentsByMerit();
    giveAdmission();
    printStudents();
}
else if (option == 4){
    viewRegisteredStudents();
}
else if (option == 5){
    string degName;
    Console.Write("Enter Degree Name: ");
    degName = Console.ReadLine();
    viewStudentInDegree(degName);
}
else if (option == 6){
    Console.Write("Enter the Student Name: ");
    string name = Console.ReadLine();
    Student s = StudentPresent(name);
    if (s != null){
        s.viewSubjects();
        registerSubjects(s);
    }
}
else if (option == 7){
    calculateFee();
}
clearScreen();
}
while (option != 8);
    Console.ReadKey();
}
```

Step 8: Main

```
static List<Student> studentList = new List<Student>();
static List<Student> sortedStudentList = new List<Student>();
static List<DegreeProgram> programList = new
List<DegreeProgram>();
static void Main(string[] args)
{
    int option;
    do
    {
        option = Menu();
        clearScreen();
        if (option == 1)
        {
            if (programList.Count > 0)
            {
                Student s = takeInputForStudent();
                addIntoStudentList(s);
            }

        }
        else if (option == 2)
        {
            DegreeProgram d = takeInputForDegree();
            addIntoDegreeList(d);
        }
```

What is Wrong with this Code?

```
else if (option == 3){
    sortStudentsByMerit();
    giveAdmission();
    printStudents();
}
else if (option == 4){
    viewRegisteredStudents();
}
else if (option == 5){
    string degName;
    Console.Write("Enter Degree Name: ");
    degName = Console.ReadLine();
    viewStudentInDegree(degName);
}
else if (option == 6){
    Console.Write("Enter the Student Name: ");
    string name = Console.ReadLine();
    Student s = StudentPresent(name);
    if (s != null){
        s.viewSubjects();
        registerSubjects(s);
    }
}
else if (option == 7){
    calculateFee();
}
clearScreen();
}
while (option != 8);
    Console.ReadKey();
}
```

WireFrames: Main Menu

```
****************************************
                UAMS
****************************************
1. Add Student
2. Add Degree Program
3. Generate Merit
4. View Registered Students
5. View Students of a Specific Program
6. Register Subjects for a Specific Student
7. Calculate Fees for all Registered Students
8. Exit
Enter Option:
```

WireFrames: Option 2: Degree Program

```
Enter Degree Name: CE
Enter Degree Duration: 4
Enter Seats for Degree: 1
Enter How many Subjects to Enter: 1
Enter Subject Code: 162
Enter Subject Type: OOP
Enter Subject Credit Hours: 3
Enter Subject Fees: 8000
Press any key to Continue..
```

WireFrames: Option 1: Add Student

```
Enter Student Name: AAA
Enter Student Age: 12
Enter Student FSc Marks: 1000
Enter Student Ecat Marks: 390
Available Degree Programs
CS
Enter how many preferences to Enter: 1
CS
Press any key to Continue..
```

WireFrames: Option 3: Generate Merit

```
AAA got Admission in CS
BBB did not get Admission
CCC got Admission in CE
DDD did not get Admission
Press any key to Continue..
```

WireFrames: Option 4: Registered Student

```
Name     FSC      Ecat     Age
AAA      1000     390      12
CCC      999      380      15
Press any key to Continue..
```

WireFrames: Option 5: Specific Degree

```
Enter Degree Name: CS
Name     FSC      Ecat     Age
AAA      1000     390      12
Press any key to Continue..
```

WireFrames: Option 6: Register Subject

Ask the Student name and then ask for the subject code.
If the conditions are satisfied then student's subject should be registered.

WireFrames: Option 7: Generate Fee

Fees should be generated for all the registered students