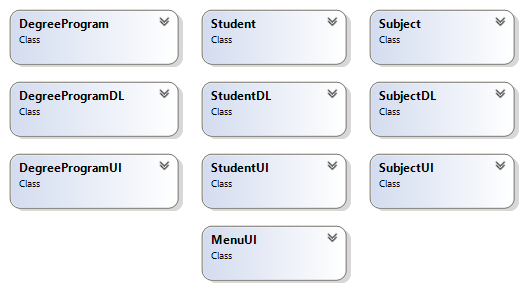
**Step 1: Case Study Scenario**

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: Domain Model with Only the Class Names**



**Step 3: Domain Model with Relations and Constraints**

Credit Hour is less than 9

Credit Hour is less than 19

Register

Apply

Offers

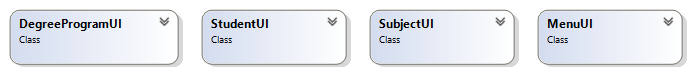
**Step 4: Domain Model with Multiplicity**

1

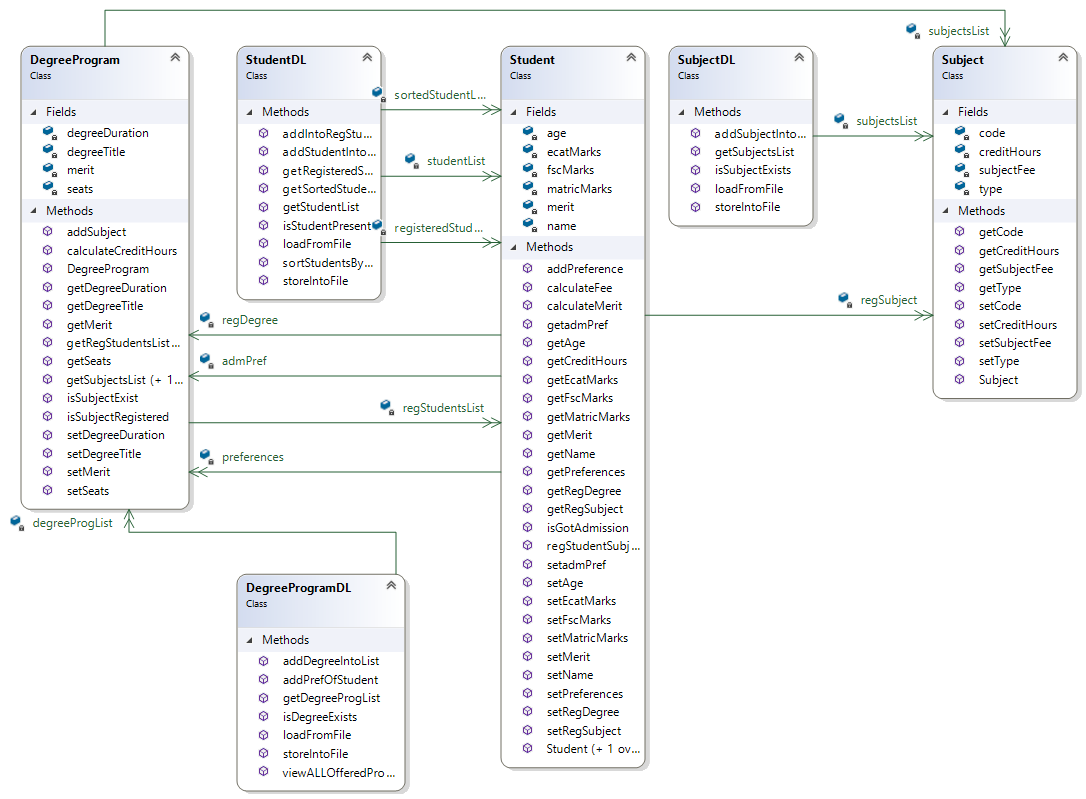
Register

Apply

Offers



**Step 5: Class Diagram Attributes and Functions**



∞

1

1

∞

∞

1

1

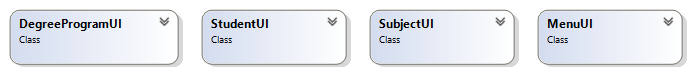
∞

1

1

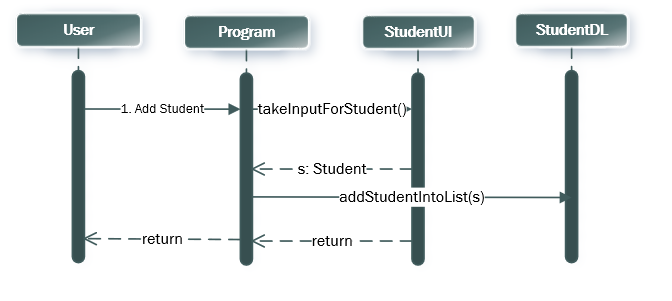
∞

1

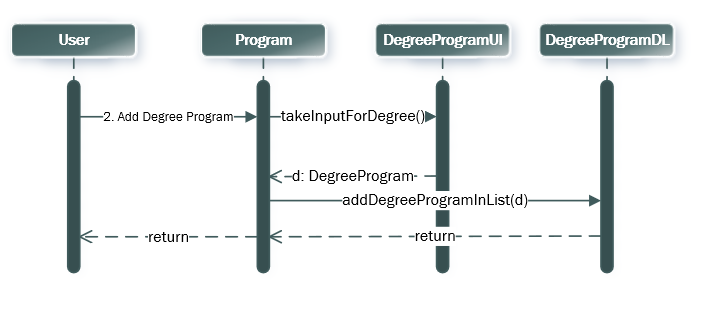


**Step 6: Sequence Diagram**

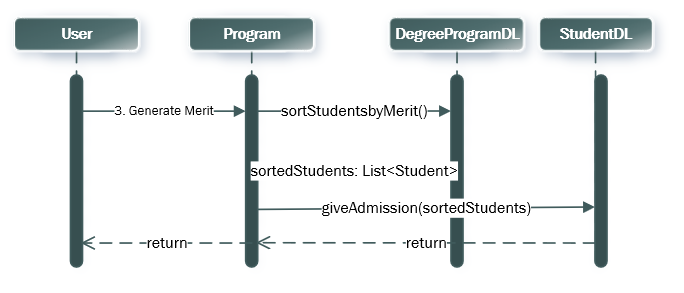
**1: Add Student:**

****

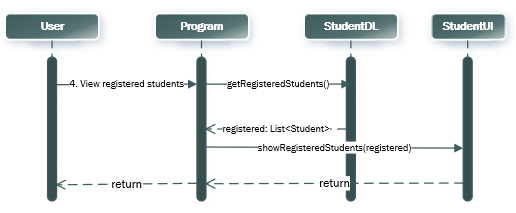
**2:Add Degree Program:**

****

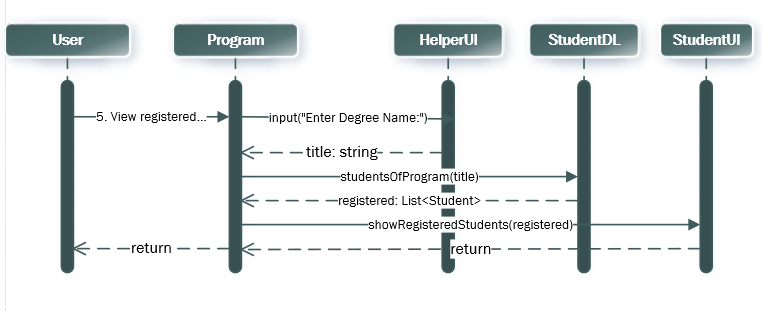
**3: Generate Merit:**

****

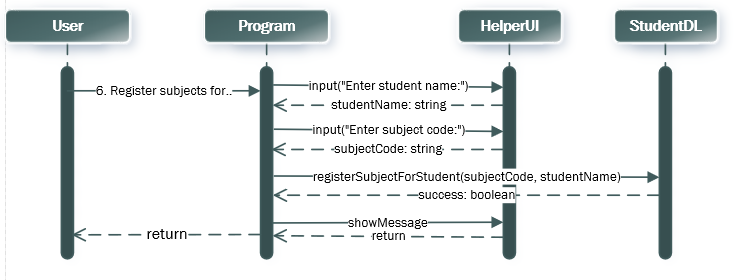
**4: View Registered Student**

****

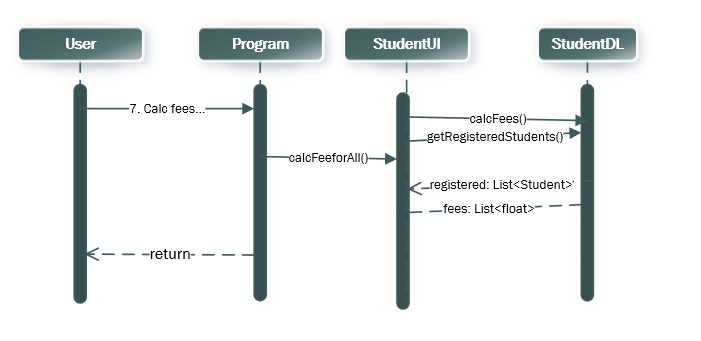
**5:View Student of specific program**

****

**6:Register Subject for students**

****

**7:Calculate fees for all Registered Students:**

****

**BL Code: Student.cs**

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;

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

{

this.name = name;

this.age = age;

this.fscMarks = fscMarks;

this.ecatMarks = ecatMarks;

this.preferences = preferences;

regSubject = new List<Subject>();

}

public void calculateMerit()

{

this.merit = (((fscMarks / 1100) \* 0.45F) + ((ecatMarks / 400) \* 0.55F)) \* 100;

}

public bool regStudentSubject(Subject s)

{

int stCH = getCreditHours();

if (regDegree != null && regDegree.isSubjectExists(s) && stCH + s.creditHours <= 9)

{

regSubject.Add(s);

return true;

}

else

{

return false;

}

}

public int getCreditHours()

{

int count = 0;

foreach (Subject sub in regSubject)

{

count = count + sub.creditHours;

}

return count;

}

public float calculateFee()

{

float fee = 0;

if (regDegree != null)

{

foreach (Subject sub in regSubject)

{

fee = fee + sub.subjectFees;

}

}

return fee;

}

}

**BL Code: ship.cs**

**DL Code: shipDL.cs**

**UI Code: shipUI.cs**

**BL Code: Subject.cs**

class Subject

{

private string code;

private string type;

private int creditHours;

private int subjectFees;

public string getCode()

{

return code;

}

public string getType()

{

return type;

}

public int getCreditHours()

{

return creditHours;

}

public int getSubjectFees()

{

return subjectFees;

}

public void setCode(string code)

{

this.code = code;

}

public void setType(string type)

{

this.type = type;

}

public void setCreditHours(int creditHours)

{

this.creditHours = creditHours;

}

public void setSubjectFees(int subjectFees)

{

this.subjectFees = subjectFees;

}

public Subject(string code, string type, int creditHours, int subjectFees)

{

this.code = code;

this.type = type;

this.creditHours = creditHours;

this.subjectFees = subjectFees;

}

}

**U**

**BL Code: DegreeProgram.cs**

class DegreeProgram

{

private string degreeName;

private float degreeDuration;

private List<Subject> subjects;

private int seats;

public string getDegreeName()

{

return degreeName;

}

public float getDegreeDuration()

{

return degreeDuration;

}

public List<Subject> getSubjects()

{

return subjects;

}

public int getSeats()

{

return seats;

}

public void setDegreeName(string degreeName) {

this.degreeName = degreeName;

}

public void setDegreeDuration(float degreeDuration)

{

this.degreeDuration = degreeDuration;

}

public void setSubjects(List<Subject> subjects)

{

this.subjects = subjects;

}

public void setSeats(int seats)

{

this.seats = seats;

}

public DegreeProgram(string degreeName, float degreeDuration, int seats)

{

this.degreeName = degreeName;

this.degreeDuration = degreeDuration;

this.seats = seats;

subjects = new List<Subject>();

}

public bool isSubjectExists(Subject sub)

{

foreach (Subject s in subjects)

{

if (s.getCode() == sub.getCode())

{

return true;

}

}

return false;

}

public bool AddSubject(Subject s)

{

int creditHours = calculateCreditHours();

if(creditHours + s.getCreditHours() <= 20)

{

subjects.Add(s);

return true;

}

else

{

return false;

}

}

public int calculateCreditHours()

{

int count = 0;

for (int x = 0; x < subjects.Count; x++)

{

count = count + subjects[x].getCreditHours();

}

return count;

}

}

**Driver Program: Program.cs**

public class Program

{

static void Main(string[] args)

{

string subjectPath = "subject.txt";

string degreePath = "degree.txt";

string studentPath = "student.txt";

if (SubjectDL.readFromFile(subjectPath))

{

Console.WriteLine("Subject Data Loaded Successfully");

}

if (DegreeProgramDL.readFromFile(degreePath))

{

Console.WriteLine("DegreeProgram Data Loaded Successfully");

}

if (StudentDL.readFromFile(studentPath))

{

Console.WriteLine("Student Data Loaded Successfully");

}

int option;

do

{

option = MenuUI.Menu();

MenuUI.clearScreen();

if (option == 1)

{

if (DegreeProgramDL.programList.Count > 0)

{

Student s = StudentUI.takeInputForStudent();

StudentDL.addIntoStudentList(s);

StudentDL.storeintoFile(studentPath, s);

}

}

else if (option == 2)

{

DegreeProgram d = DegreeProgramUI.takeInputForDegree();

DegreeProgramDL.addIntoDegreeList(d);

DegreeProgramDL.storeintoFile(degreePath, d);

}

else if (option == 3)

{

List<Student> sortedStudentList = new List<Student>();

sortedStudentList = StudentDL.sortStudentsByMerit();

StudentDL.giveAdmission(sortedStudentList);

StudentUI.printStudents();

}

else if (option == 4)

{

StudentUI.viewRegisteredStudents();

}

else if (option == 5)

{

string degName;

Console.Write("Enter Degree Name: ");

degName = Console.ReadLine();

StudentUI.viewStudentInDegree(degName);

}

else if (option == 6)

{

Console.Write("Enter the Student Name: ");

string name = Console.ReadLine();

Student s = StudentDL.StudentPresent(name);

if (s != null)

{

SubjectUI.viewSubjects(s);

SubjectUI.registerSubjects(s);

}

}

else if (option == 7)

{

StudentUI.calculateFeeForAll();

}

MenuUI.clearScreen();

}

while (option != 8);

}

}