Object Oriented Programming

Lab Manual 05

Introduction

After a week of rigorous coding, Welcome back!

You have learned all about the Classes, Constructors, and member functions in the previous lab manuals. Let's move on to the next, new, and interesting concepts.

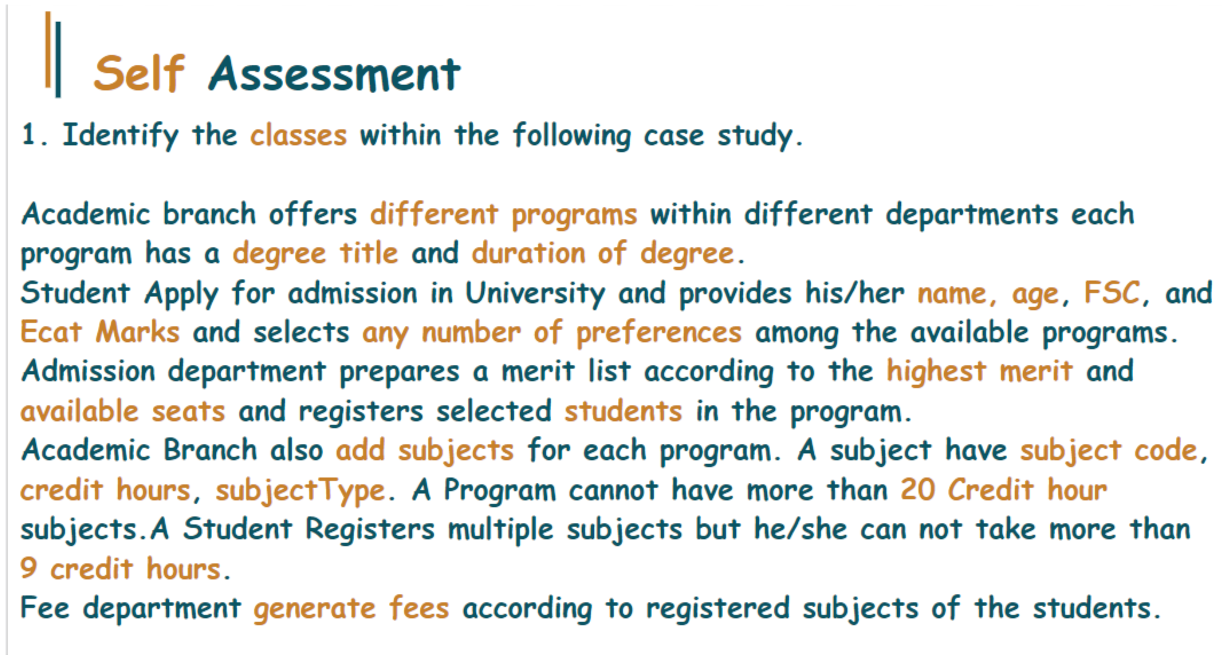
Students, In Object-Oriented Programming, the Class is a combination of data members and member functions. In this Lab, we will learn about the **3 Tier Model** in our program to achieve the object's oriented philosophy.

These coding Layers include

- Business Logic Layer
- User Interface Layer
- Data Logic Layer

University Admission Management System

Read the following question carefully.



Self Assessment

1. Identify the classes within the following case study.

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. A Program cannot have more than 20 Credit hour subjects.A Student Registers multiple subjects but he/she can not take more than 9 credit hours.
Fee department generate fees according to registered subjects of the students.

Try out yourself.

Don't worry.

There is a solution on the next page.

Identification of Classes

By looking at the above-mentioned self-assessment you can extract the following possible class-like structures from the given statement.

- Student Class
- Subject Class
- Program Class
- Separate DL Classes
- Seprate UI Classes

Don't Worry. There is a solution ahead. First Try out yourself.

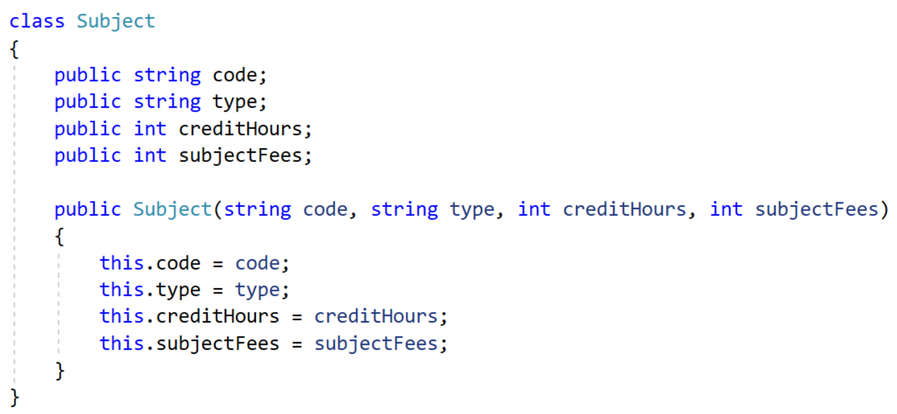
Let's Start with fun coding.

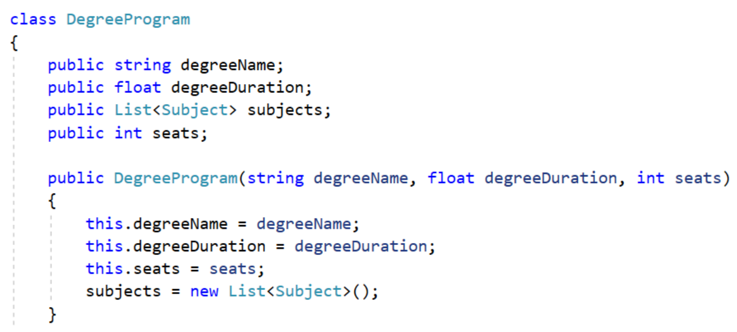
University Admission Management System (Through OOP)

Now that you have identified the classes in your program, it is time to start coding.

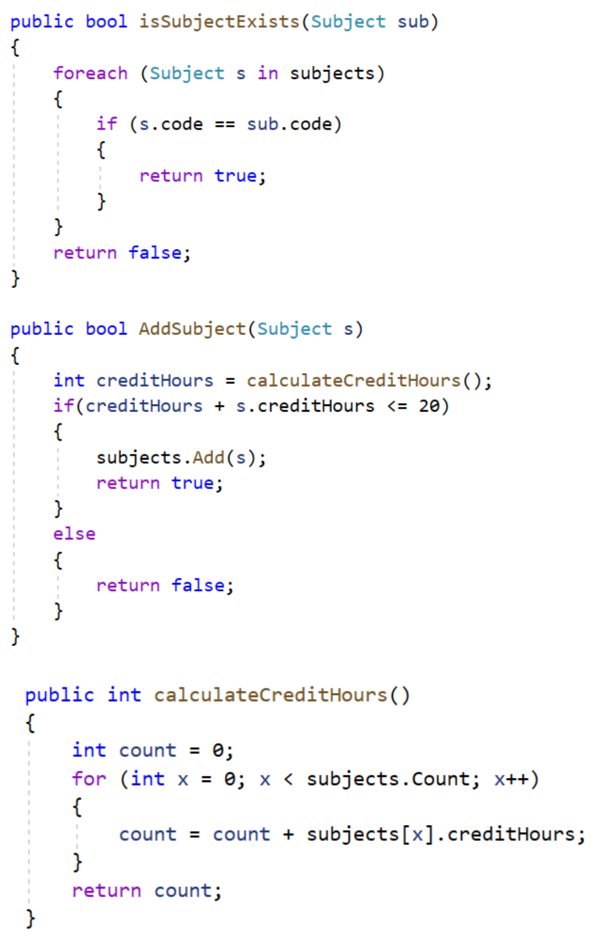
Solution:

Sr. #	Action	Description
Let us define the code for the **BL folder**.		

1	 `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;`
	`    }`
	`}`
		● **Subject** Class (BL)

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

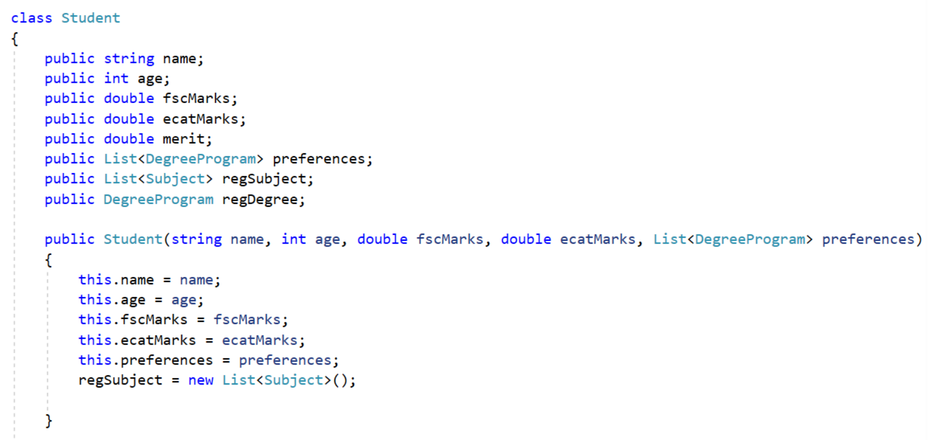
2(a)



```csharp
public bool isSubjectExists(Subject sub)
{
    foreach (Subject s in subjects)
    {
        if (s.code == sub.code)
        {
            return true;
        }
    }
    return false;
}

public bool AddSubject(Subject s)
{
    int creditHours = calculateCreditHours();
    if(creditHours + s.creditHours <= 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].creditHours;
     }
     return count;
 }
```

- Member Function in **DegreeProgram** Class (BL)

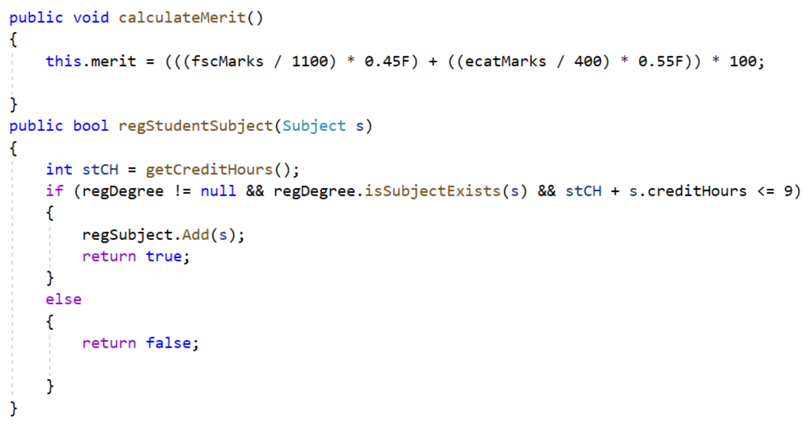
3



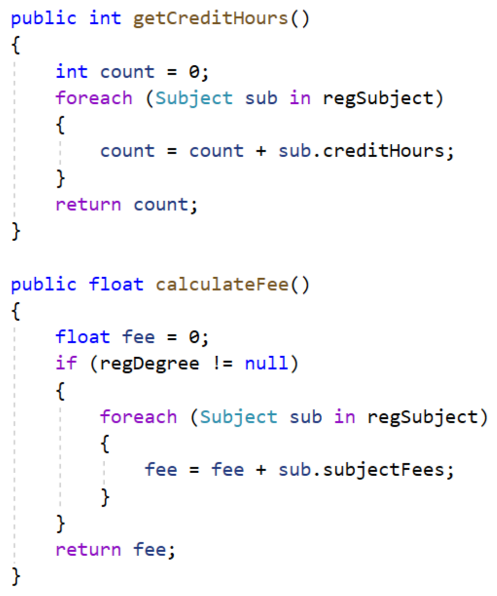
```csharp
class Student
{
    public string name;
    public int age;
    public double fscMarks;
    public double ecatMarks;
    public double merit;
    public List<DegreeProgram> preferences;
    public List<Subject> regSubject;
    public DegreeProgram regDegree;

    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>();

    }
```

- **Student** Class (BL)

3(a)



```
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;

    }
}
```

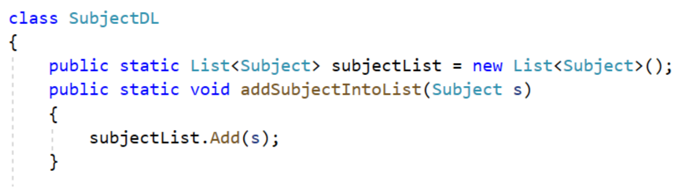


```
public int getCreditHours()
{
    int count = 0;
    foreach (Subject sub in regSubject)
    {
        count = count + sub.creditHours;
    }
    return count;
}

public float calculateFee()
{
    float fee = 0;
    if (regDegree != null)
    {
        foreach (Subject sub in regSubject)
        {
            fee = fee + sub.subjectFees;
        }
    }
    return fee;
}
```

- Member Functions for **Student** Class (BL)

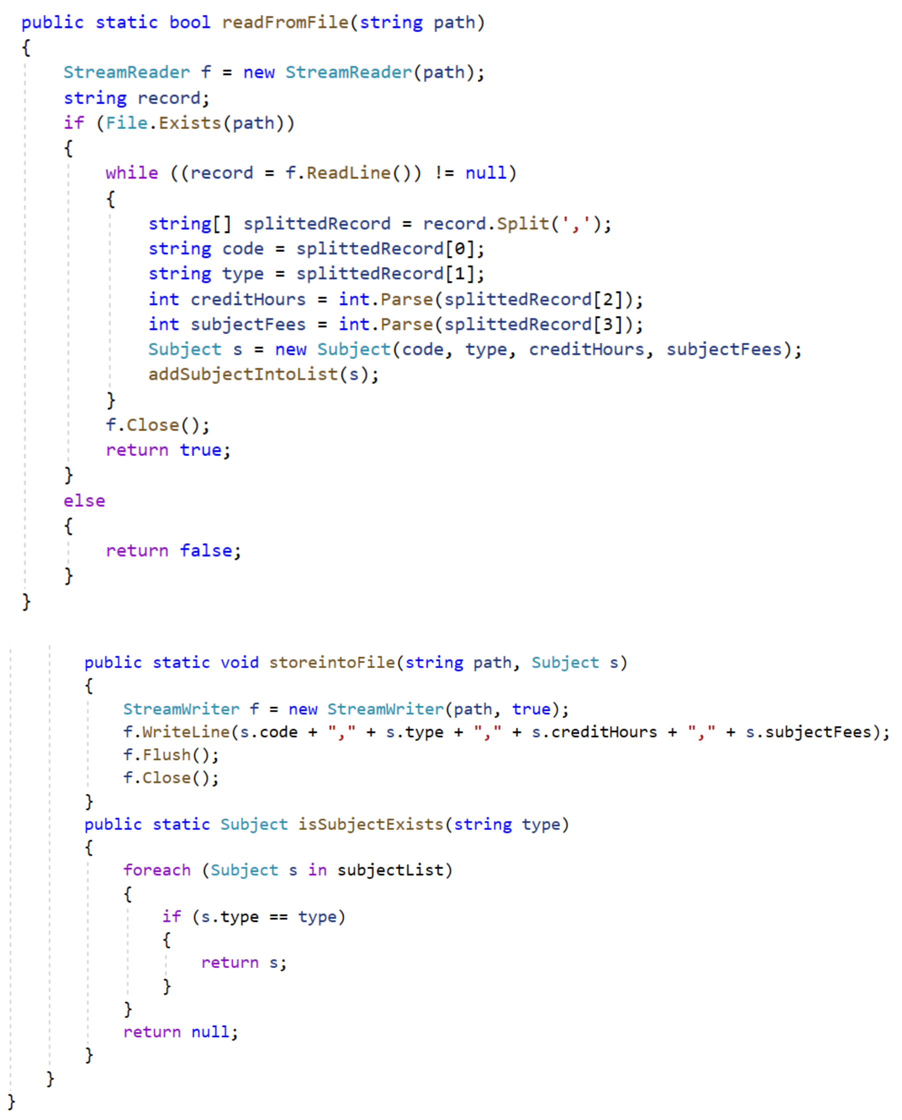
Let us now implement the **Data Logic Layer** (DL folder Classes) for this project.

4.



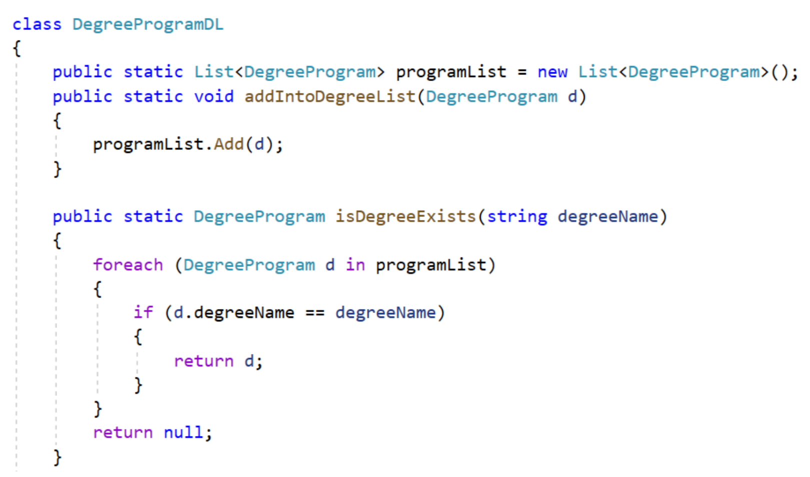
```
class SubjectDL
{
    public static List<Subject> subjectList = new List<Subject>();
    public static void addSubjectIntoList(Subject s)
    {
        subjectList.Add(s);
    }
```

- **SubjectDL** Class
- Member Functions of **SubjectsDL** Class



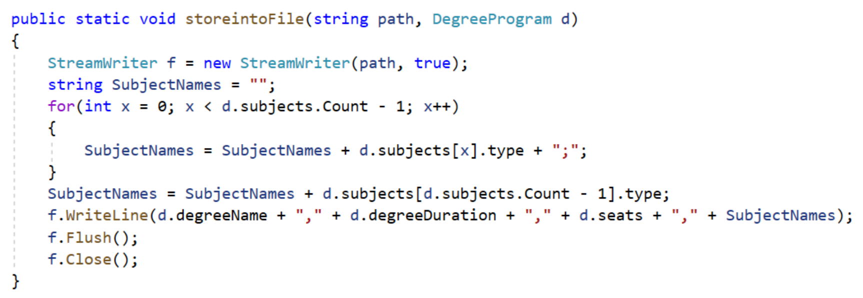
```
public static bool readFromFile(string path)
{
    StreamReader f = new StreamReader(path);
    string record;
    if (File.Exists(path))
    {
        while ((record = f.ReadLine()) != null)
        {
            string[] splittedRecord = record.Split(',');
            string code = splittedRecord[0];
            string type = splittedRecord[1];
            int creditHours = int.Parse(splittedRecord[2]);
            int subjectFees = int.Parse(splittedRecord[3]);
            Subject s = new Subject(code, type, creditHours, subjectFees);
            addSubjectIntoList(s);
        }
        f.Close();
        return true;
    }
    else
    {
        return false;
    }
}

        public static void storeintoFile(string path, Subject s)
        {
            StreamWriter f = new StreamWriter(path, true);
            f.WriteLine(s.code + "," + s.type + "," + s.creditHours + "," + s.subjectFees);
            f.Flush();
            f.Close();
        }
        public static Subject isSubjectExists(string type)
        {
            foreach (Subject s in subjectList)
            {
                if (s.type == type)
                {
                    return s;
                }
            }
            return null;
        }
    }
}
```

5.

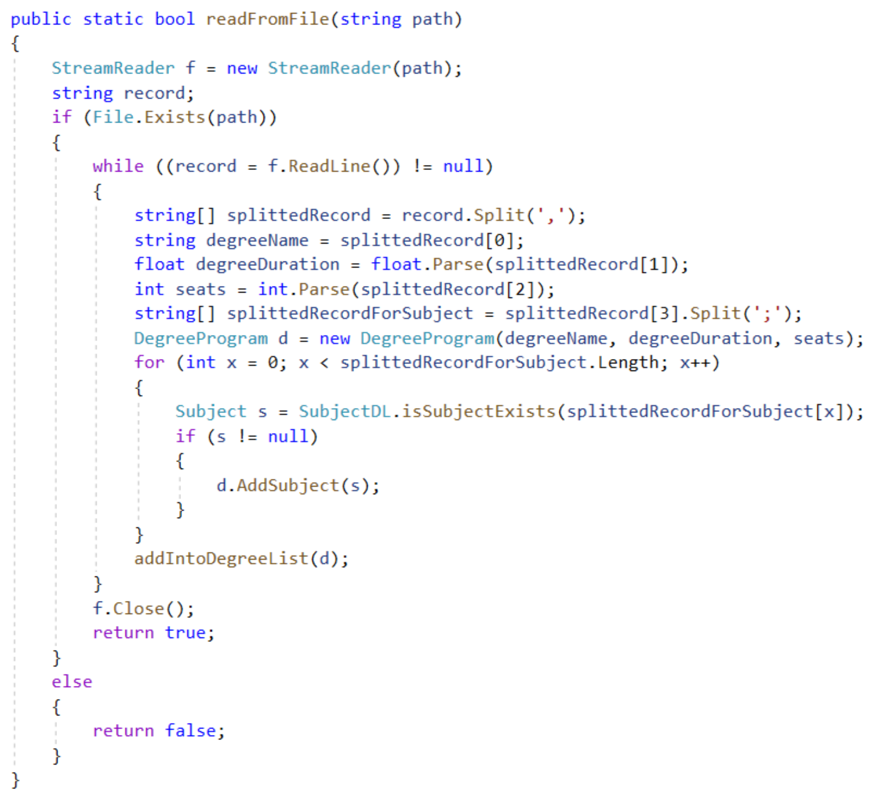


```
class DegreeProgramDL
{
    public static List<DegreeProgram> programList = new List<DegreeProgram>();
    public static void addIntoDegreeList(DegreeProgram d)
    {
        programList.Add(d);
    }

    public static DegreeProgram isDegreeExists(string degreeName)
    {
        foreach (DegreeProgram d in programList)
        {
            if (d.degreeName == degreeName)
            {
                return d;
            }
        }
        return null;
    }
```

- **DegreeProgramDL** Class
- Member functions for **DegreeProgramDL** Class

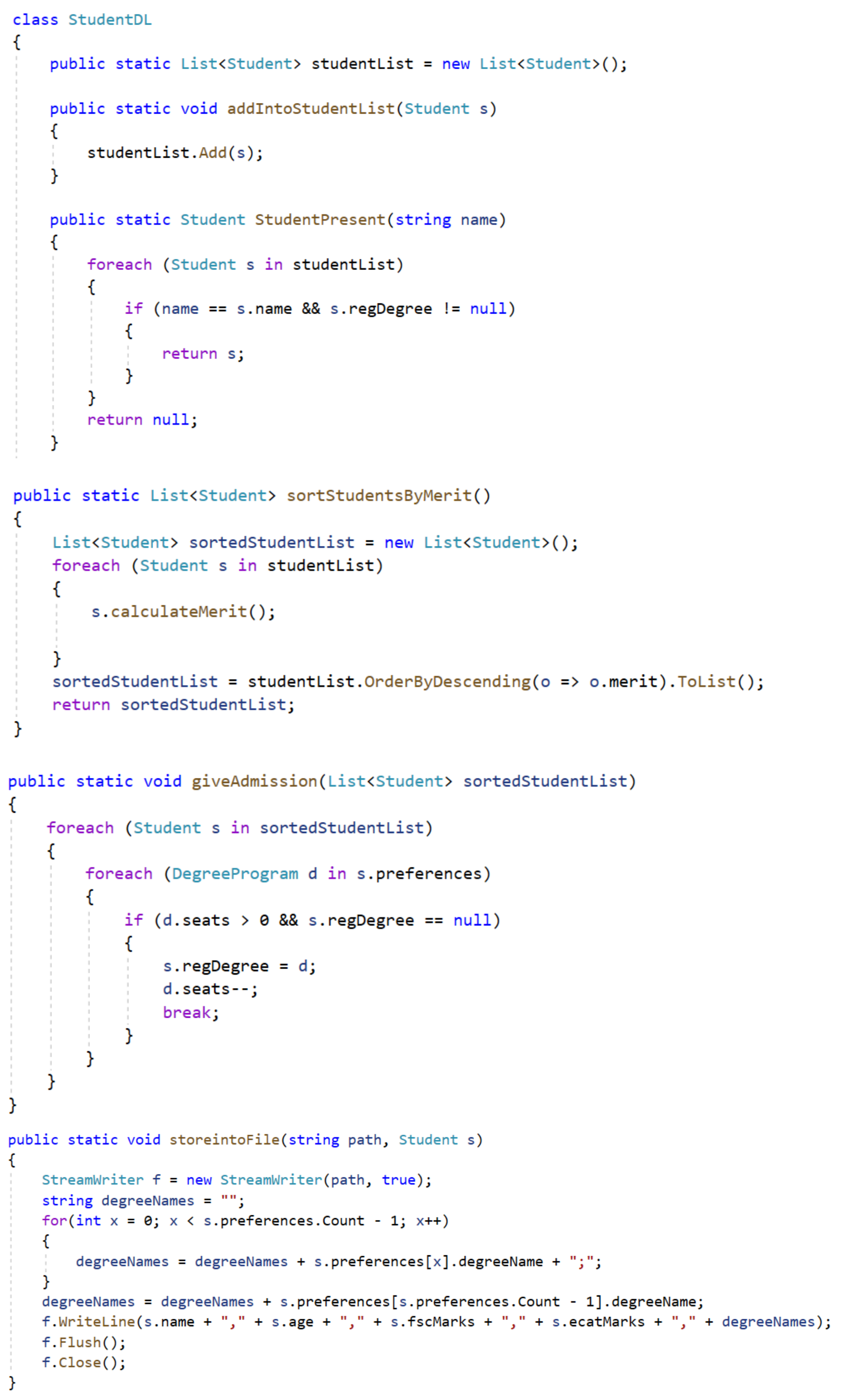


```
public static void storeintoFile(string path, DegreeProgram d)
{
    StreamWriter f = new StreamWriter(path, true);
    string SubjectNames = "";
    for(int x = 0; x < d.subjects.Count - 1; x++)
    {
        SubjectNames = SubjectNames + d.subjects[x].type + ";";
    }
    SubjectNames = SubjectNames + d.subjects[d.subjects.Count - 1].type;
    f.WriteLine(d.degreeName + "," + d.degreeDuration + "," + d.seats + "," + SubjectNames);
    f.Flush();
    f.Close();
}
```



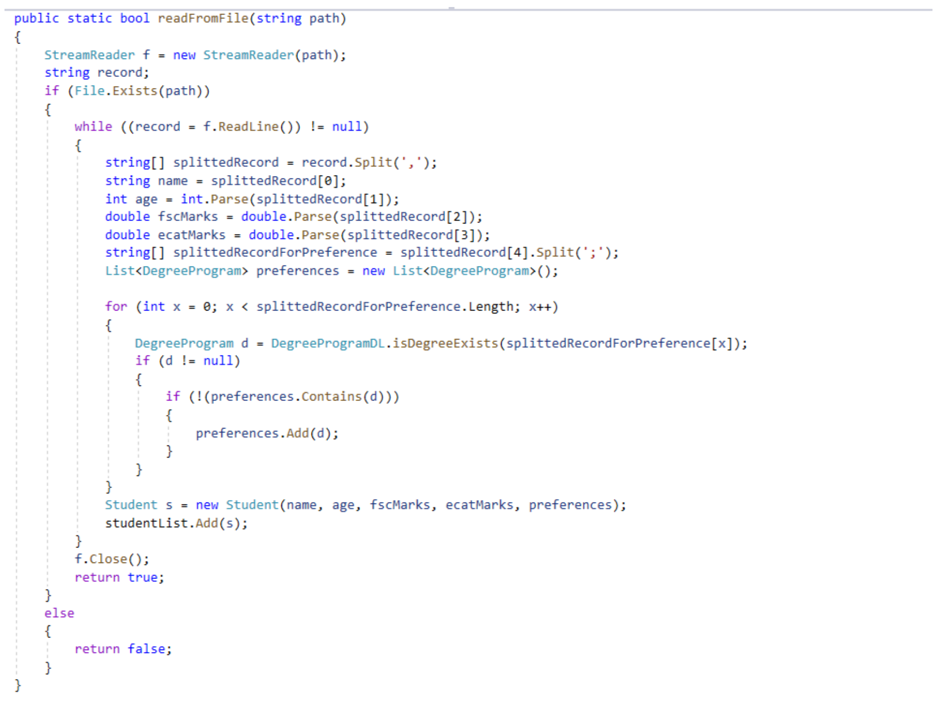
```
public static bool readFromFile(string path)
{
    StreamReader f = new StreamReader(path);
    string record;
    if (File.Exists(path))
    {
        while ((record = f.ReadLine()) != null)
        {
            string[] splittedRecord = record.Split(',');
            string degreeName = splittedRecord[0];
            float degreeDuration = float.Parse(splittedRecord[1]);
            int seats = int.Parse(splittedRecord[2]);
            string[] splittedRecordForSubject = splittedRecord[3].Split(';');
            DegreeProgram d = new DegreeProgram(degreeName, degreeDuration, seats);
            for (int x = 0; x < splittedRecordForSubject.Length; x++)
            {
                Subject s = SubjectDL.isSubjectExists(splittedRecordForSubject[x]);
                if (s != null)
                {
                    d.AddSubject(s);
                }
            }
            addIntoDegreeList(d);
        }
        f.Close();
        return true;
    }
    else
    {
        return false;
    }
}
```

6.



```
class StudentDL
{
    public static List<Student> studentList = new List<Student>();

    public static void addIntoStudentList(Student s)
    {
        studentList.Add(s);
    }

    public static Student StudentPresent(string name)
    {
        foreach (Student s in studentList)
        {
            if (name == s.name && s.regDegree != null)
            {
                return s;
            }
        }
        return null;
    }

public static List<Student> sortStudentsByMerit()
{
    List<Student> sortedStudentList = new List<Student>();
    foreach (Student s in studentList)
    {
        s.calculateMerit();

    }
    sortedStudentList = studentList.OrderByDescending(o => o.merit).ToList();
    return sortedStudentList;
}

public static void giveAdmission(List<Student> sortedStudentList)
{
    foreach (Student s in sortedStudentList)
    {
        foreach (DegreeProgram d in s.preferences)
        {
            if (d.seats > 0 && s.regDegree == null)
            {
                s.regDegree = d;
                d.seats--;
                break;
            }
        }
    }
}

public static void storeintoFile(string path, Student s)
{
    StreamWriter f = new StreamWriter(path, true);
    string degreeNames = "";
    for(int x = 0; x < s.preferences.Count - 1; x++)
    {
        degreeNames = degreeNames + s.preferences[x].degreeName + ";";
    }
    degreeNames = degreeNames + s.preferences[s.preferences.Count - 1].degreeName;
    f.WriteLine(s.name + "," + s.age + "," + s.fscMarks + "," + s.ecatMarks + "," + degreeNames);
    f.Flush();
    f.Close();
}
```

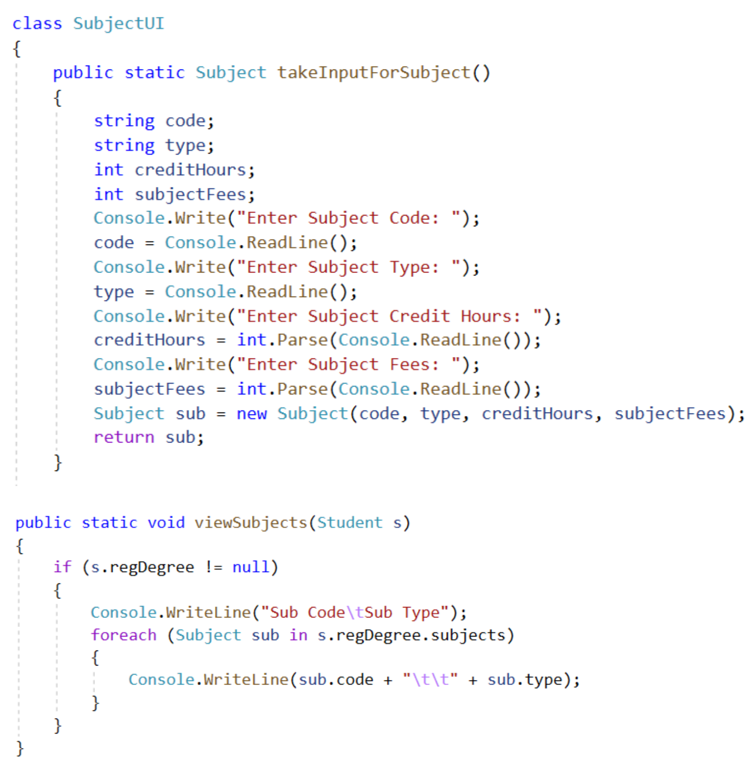
- **StudentDL** Class
- Member functions for **StudentDL** Class



```
public static bool readFromFile(string path)
{
    StreamReader f = new StreamReader(path);
    string record;
    if (File.Exists(path))
    {
        while ((record = f.ReadLine()) != null)
        {
            string[] splittedRecord = record.Split(',');
            string name = splittedRecord[0];
            int age = int.Parse(splittedRecord[1]);
            double fscMarks = double.Parse(splittedRecord[2]);
            double ecatMarks = double.Parse(splittedRecord[3]);
            string[] splittedRecordForPreference = splittedRecord[4].Split(';');
            List<DegreeProgram> preferences = new List<DegreeProgram>();

            for (int x = 0; x < splittedRecordForPreference.Length; x++)
            {
                DegreeProgram d = DegreeProgramDL.isDegreeExists(splittedRecordForPreference[x]);
                if (d != null)
                {
                    if (!(preferences.Contains(d)))
                    {
                        preferences.Add(d);
                    }
                }
            }
            Student s = new Student(name, age, fscMarks, ecatMarks, preferences);
            studentList.Add(s);
        }
        f.Close();
        return true;
    }
    else
    {
        return false;
    }
}
```

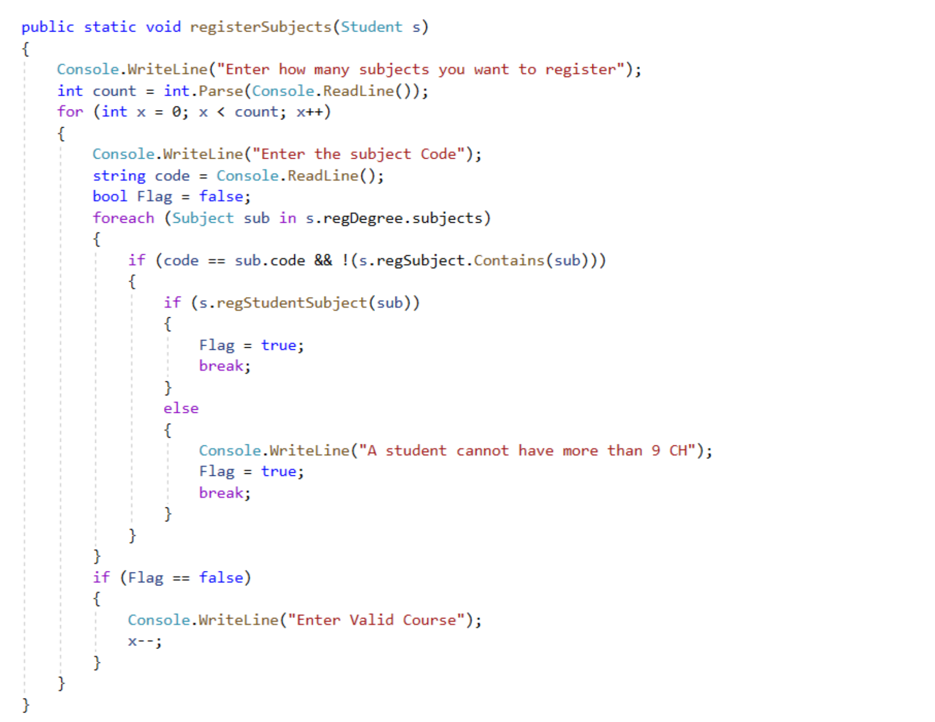
Let us now implement the **User Interface Layer** (UI folder Classes) for this project.

7



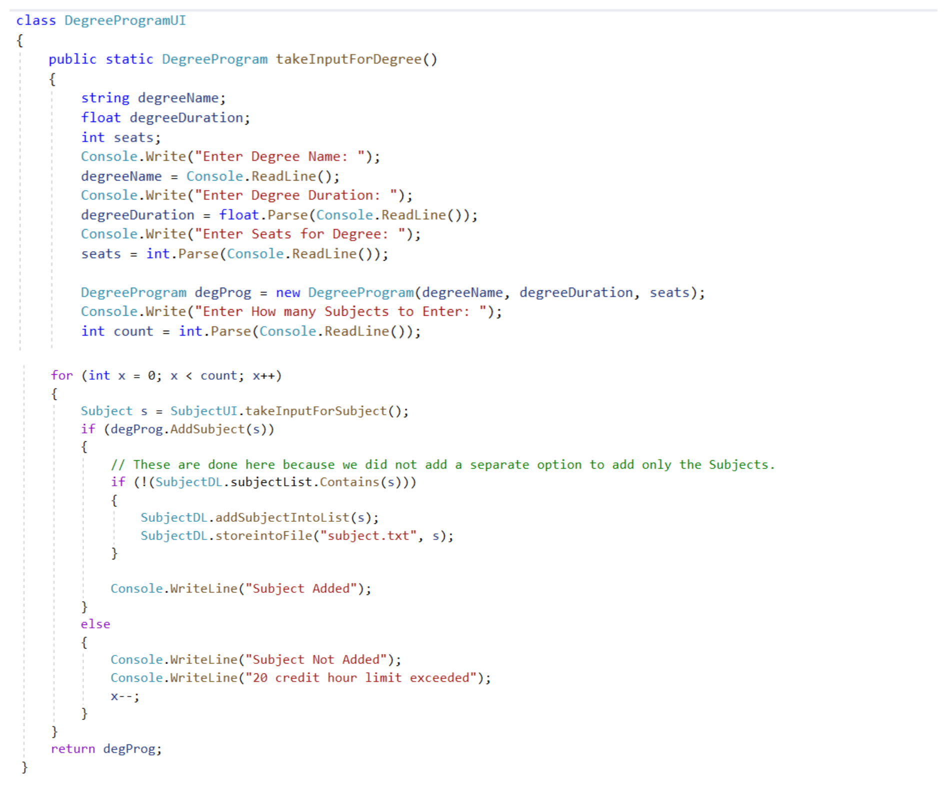
```
class SubjectUI
{
    public static Subject takeInputForSubject()
    {
        string code;
        string type;
        int creditHours;
        int subjectFees;
        Console.Write("Enter Subject Code: ");
        code = Console.ReadLine();
        Console.Write("Enter Subject Type: ");
        type = Console.ReadLine();
        Console.Write("Enter Subject Credit Hours: ");
        creditHours = int.Parse(Console.ReadLine());
        Console.Write("Enter Subject Fees: ");
        subjectFees = int.Parse(Console.ReadLine());
        Subject sub = new Subject(code, type, creditHours, subjectFees);
        return sub;
    }

public static void viewSubjects(Student s)
{
    if (s.regDegree != null)
    {
        Console.WriteLine("Sub Code\tSub Type");
        foreach (Subject sub in s.regDegree.subjects)
        {
            Console.WriteLine(sub.code + "\t\t" + sub.type);
        }
    }
}
```

- **SubjectUI** Class
- Member functions for **SubjectUI** Class



```
public static void registerSubjects(Student s)
{
    Console.WriteLine("Enter how many subjects you want to register");
    int count = int.Parse(Console.ReadLine());
    for (int x = 0; x < count; x++)
    {
        Console.WriteLine("Enter the subject Code");
        string code = Console.ReadLine();
        bool Flag = false;
        foreach (Subject sub in s.regDegree.subjects)
        {
            if (code == sub.code && !(s.regSubject.Contains(sub)))
            {
                if (s.regStudentSubject(sub))
                {
                    Flag = true;
                    break;
                }
                else
                {
                    Console.WriteLine("A student cannot have more than 9 CH");
                    Flag = true;
                    break;
                }
            }
        }
        if (Flag == false)
        {
            Console.WriteLine("Enter Valid Course");
            x--;
        }
    }
}
```

8



```
class DegreeProgramUI
{
    public static DegreeProgram takeInputForDegree()
    {
        string degreeName;
        float degreeDuration;
        int seats;
        Console.Write("Enter Degree Name: ");
        degreeName = Console.ReadLine();
        Console.Write("Enter Degree Duration: ");
        degreeDuration = float.Parse(Console.ReadLine());
        Console.Write("Enter Seats for Degree: ");
        seats = int.Parse(Console.ReadLine());

        DegreeProgram degProg = new DegreeProgram(degreeName, degreeDuration, seats);
        Console.Write("Enter How many Subjects to Enter: ");
        int count = int.Parse(Console.ReadLine());

    for (int x = 0; x < count; x++)
    {
        Subject s = SubjectUI.takeInputForSubject();
        if (degProg.AddSubject(s))
        {
            // These are done here because we did not add a separate option to add only the Subjects.
            if (!(SubjectDL.subjectList.Contains(s)))
            {
                SubjectDL.addSubjectIntoList(s);
                SubjectDL.storeintoFile("subject.txt", s);
            }

            Console.WriteLine("Subject Added");
        }
        else
        {
            Console.WriteLine("Subject Not Added");
            Console.WriteLine("20 credit hour limit exceeded");
            x--;
        }
    }
    return degProg;
}
```

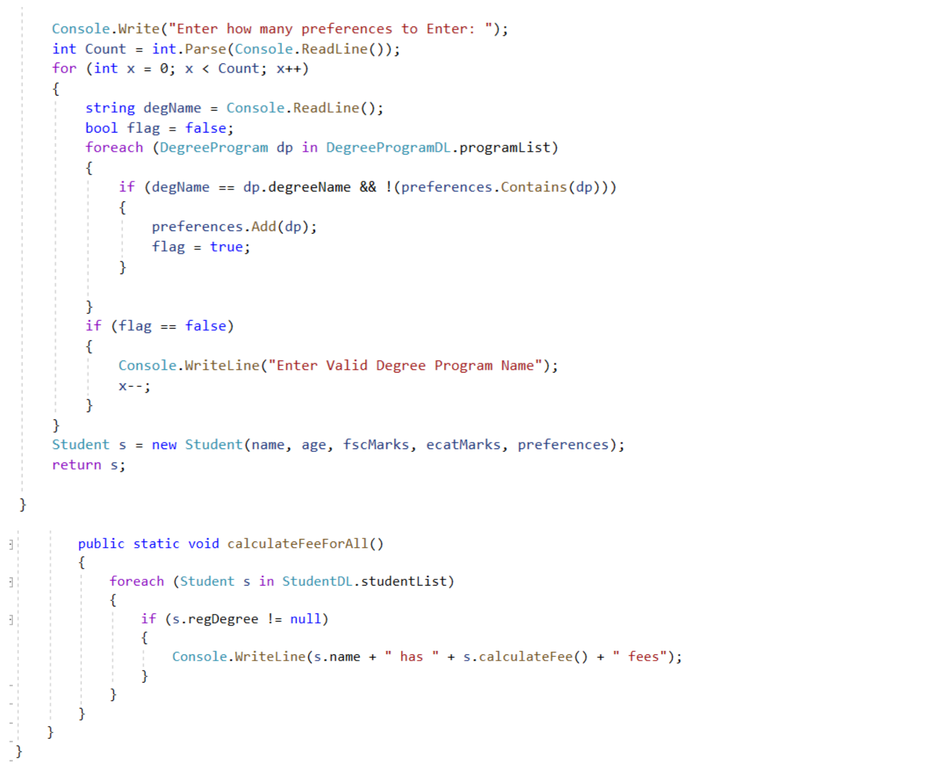
- **DegreeProgramUI** Class
- Member functions for **DegreeProgramUI** Class

```
        public static void viewDegreePrograms()
        {
            foreach (DegreeProgram dp in DegreeProgramDL.programList)
            {
                Console.WriteLine(dp.degreeName);
            }
        }
    }
}
```

9

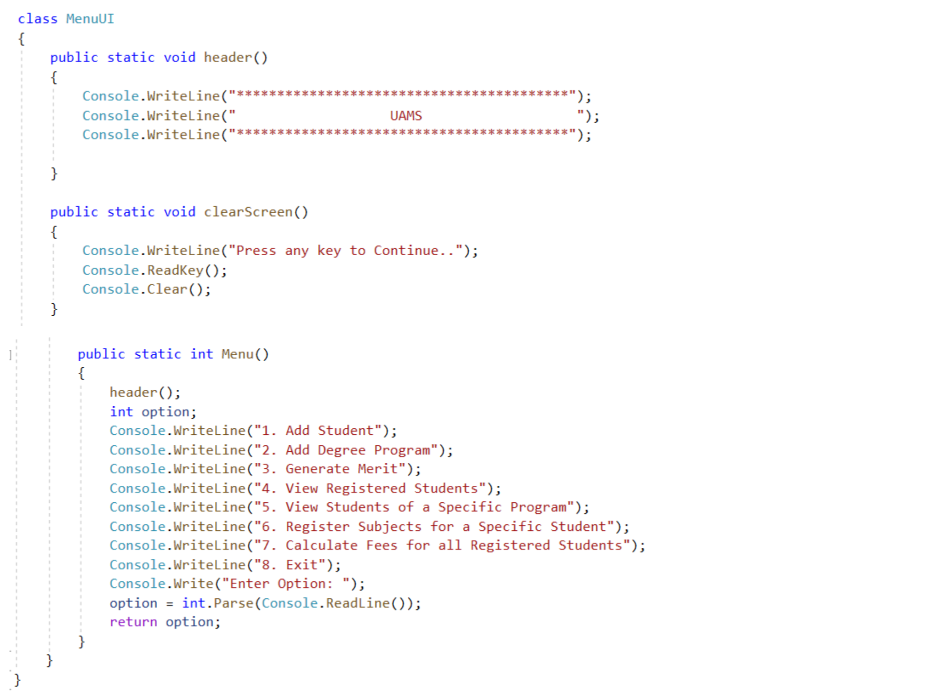
```
class StudentUI
{
    public static void printStudents()
    {
        foreach (Student s in StudentDL.studentList)
        {
            if (s.regDegree != null)
            {
                Console.WriteLine(s.name + " got Admission in " + s.regDegree.degreeName);
            }
            else
            {
                Console.WriteLine(s.name + " did not get Admission");

            }
        }
    }

public static void viewStudentInDegree(string degName)
{
    Console.WriteLine("Name\tFSC\tEcat\tAge");
    foreach (Student s in StudentDL.studentList)
    {
        if (s.regDegree != null)
        {
            if (degName == s.regDegree.degreeName)
            {
                Console.WriteLine(s.name + "\t" + s.fscMarks + "\t" + s.ecatMarks + "\t" + s.age);
            }
        }
    }
}

public static void viewRegisteredStudents()
{
    Console.WriteLine("Name\tFSC\tEcat\tAge");
    foreach (Student s in StudentDL.studentList)
    {
        if (s.regDegree != null)
        {
            Console.WriteLine(s.name + "\t" + s.fscMarks + "\t" + s.ecatMarks + "\t" + s.age);
        }
    }
}

 public static Student takeInputForStudent()
 {
     string name;
     int age;
     double fscMarks;
     double ecatMarks;
     List<DegreeProgram> preferences = new List<DegreeProgram>();
     Console.Write("Enter Student Name: ");
     name = Console.ReadLine();
     Console.Write("Enter Student Age: ");
     age = int.Parse(Console.ReadLine());
     Console.Write("Enter Student FSc Marks: ");
     fscMarks = double.Parse(Console.ReadLine());
     Console.Write("Enter Student Ecat Marks: ");
     ecatMarks = double.Parse(Console.ReadLine());
     Console.WriteLine("Available Degree Programs");
     DegreeProgramUI.viewDegreePrograms();
```

- **StudentUI** Class
- Member functions for **StudentUI** Class



```
        Console.Write("Enter how many preferences to Enter: ");
        int Count = int.Parse(Console.ReadLine());
        for (int x = 0; x < Count; x++)
        {
            string degName = Console.ReadLine();
            bool flag = false;
            foreach (DegreeProgram dp in DegreeProgramDL.programList)
            {
                if (degName == dp.degreeName && !(preferences.Contains(dp)))
                {
                    preferences.Add(dp);
                    flag = true;
                }

            }
            if (flag == false)
            {
                Console.WriteLine("Enter Valid Degree Program Name");
                x--;
            }
        }
        Student s = new Student(name, age, fscMarks, ecatMarks, preferences);
        return s;

    }

        public static void calculateFeeForAll()
        {
            foreach (Student s in StudentDL.studentList)
            {
                if (s.regDegree != null)
                {
                    Console.WriteLine(s.name + " has " + s.calculateFee() + " fees");
                }
            }
        }
    }
}
```

10

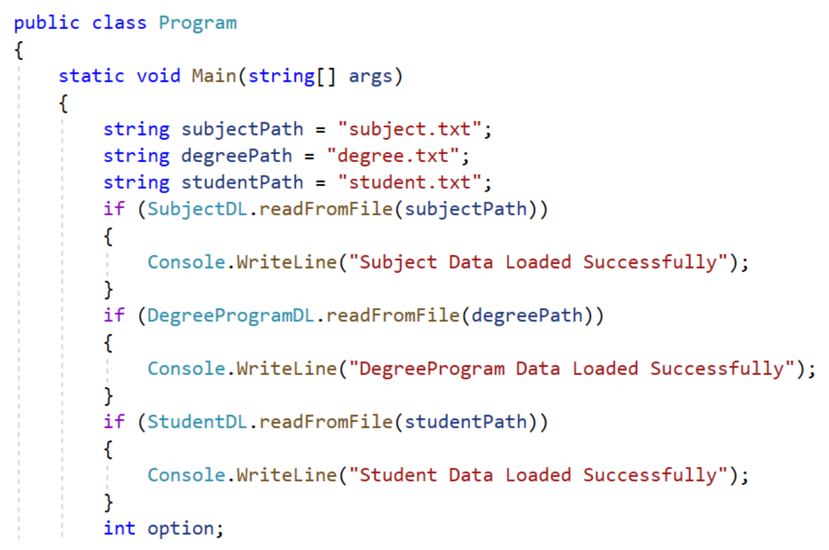


```
class MenuUI
{
    public static void header()
    {
        Console.WriteLine("*****************************************");
        Console.WriteLine("                    UAMS                 ");
        Console.WriteLine("*****************************************");

    }

    public static void clearScreen()
    {
        Console.WriteLine("Press any key to Continue..");
        Console.ReadKey();
        Console.Clear();
    }

        public static int Menu()
        {
            header();
            int option;
            Console.WriteLine("1. Add Student");
            Console.WriteLine("2. Add Degree Program");
            Console.WriteLine("3. Generate Merit");
            Console.WriteLine("4. View Registered Students");
            Console.WriteLine("5. View Students of a Specific Program");
            Console.WriteLine("6. Register Subjects for a Specific Student");
            Console.WriteLine("7. Calculate Fees for all Registered Students");
            Console.WriteLine("8. Exit");
            Console.Write("Enter Option: ");
            option = int.Parse(Console.ReadLine());
            return option;
        }
    }
}
```

- **MenuUI** Class
- Member functions for **MenuUI** Class

Let us now implement the **Main Driver Program** (program.cs file) for this project.

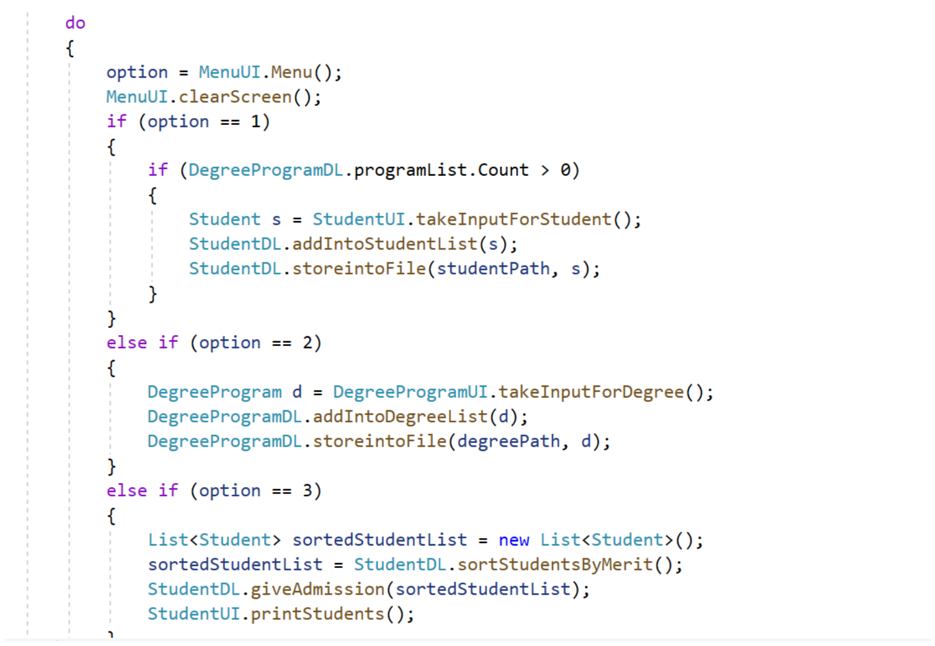
11



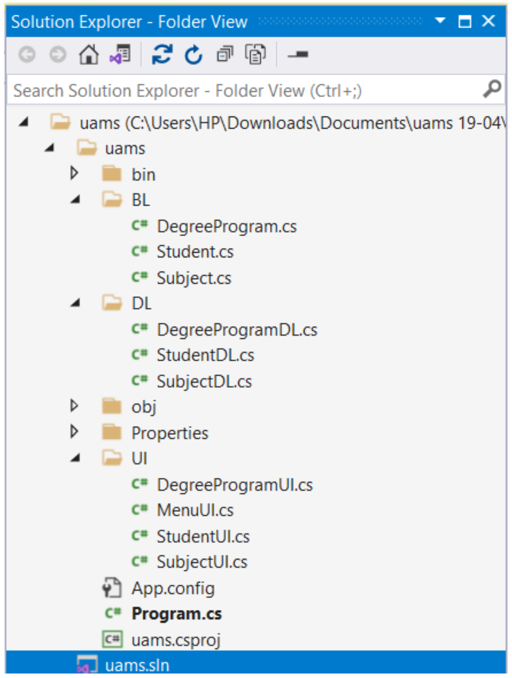
```
public class Program
{
    static void Main(string[] args)
    {
        string subjectPath = "subject.txt";
        string degreePath = "degree.txt";
        string studentPath = "student.txt";
        if (SubjectDL.readFromFile(subjectPath))
        {
            Console.WriteLine("Subject Data Loaded Successfully");
        }
        if (DegreeProgramDL.readFromFile(degreePath))
        {
            Console.WriteLine("DegreeProgram Data Loaded Successfully");
        }
        if (StudentDL.readFromFile(studentPath))
        {
            Console.WriteLine("Student Data Loaded Successfully");
        }
        int option;
```

- **Main Driver Program**

11(a)



```
        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);
}
```



Solution Explorer - Folder View
Search Solution Explorer - Folder View (Ctrl+;)

```
uams (C:\Users\HP\Downloads\Documents\uams 19-04\
  uams
    bin
    BL
      DegreeProgram.cs
      Student.cs
      Subject.cs
    DL
      DegreeProgramDL.cs
      StudentDL.cs
      SubjectDL.cs
    obj
    Properties
    UI
      DegreeProgramUI.cs
      MenuUI.cs
      StudentUI.cs
      SubjectUI.cs
    App.config
    Program.cs
    uams.csproj
  uams.sln
```

- Final Layer wise Directory Structure

Congratulations !!!!! You have made it through and implemented the complete project through the 3 Tier Model. Great Work Students.

Self Assessment Task: Draw the updated Domain Model and Class Diagram for this project now that contains all the classes from the 3-tier model.

Note: Take a 2-minute break if you may and once you are done with that, convert all the assigned projects using this layered approach.

Good Luck and Best Wishes ‼
Happy Coding ahead :)