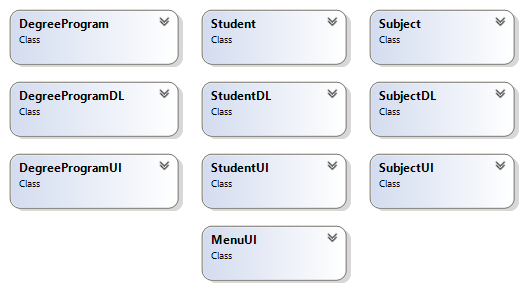
**Step 1: Case Study Scenario**

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

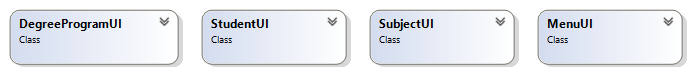
**Step 4: Domain Model with Multiplicity**

1

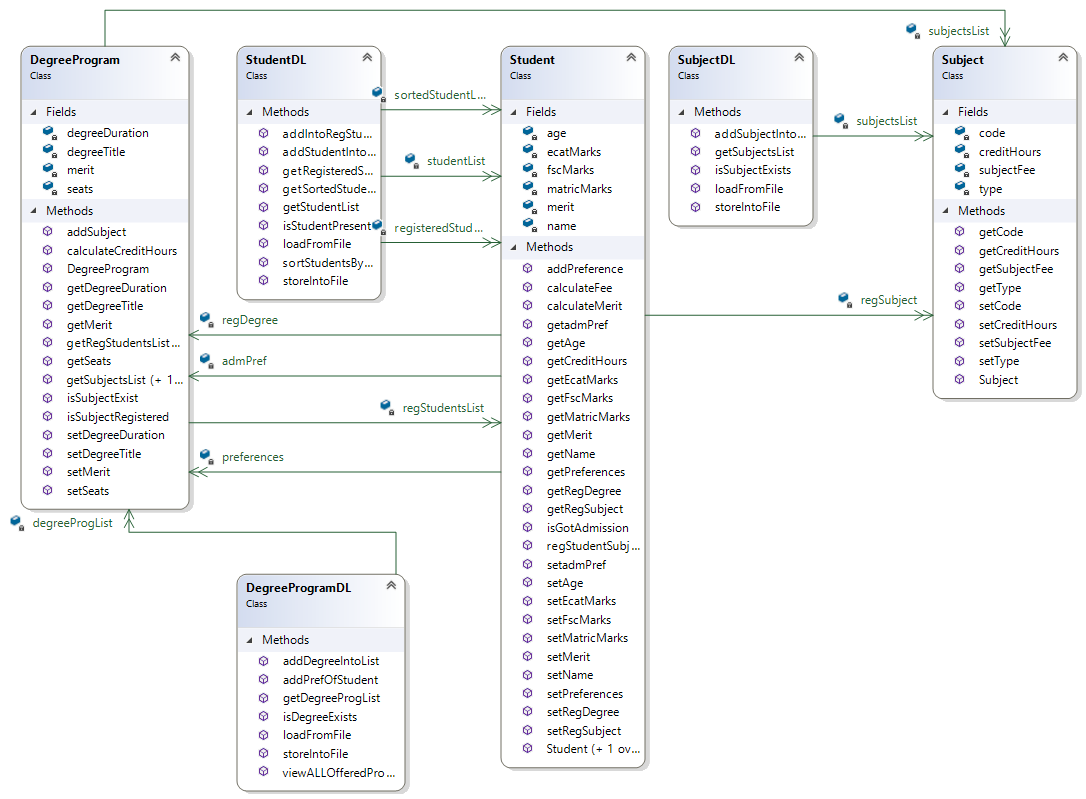
Register

Apply

Offers



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



∞

1

1

∞

∞

1

1

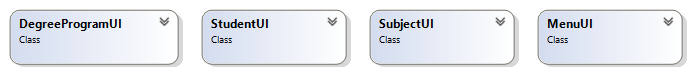
∞

1

1

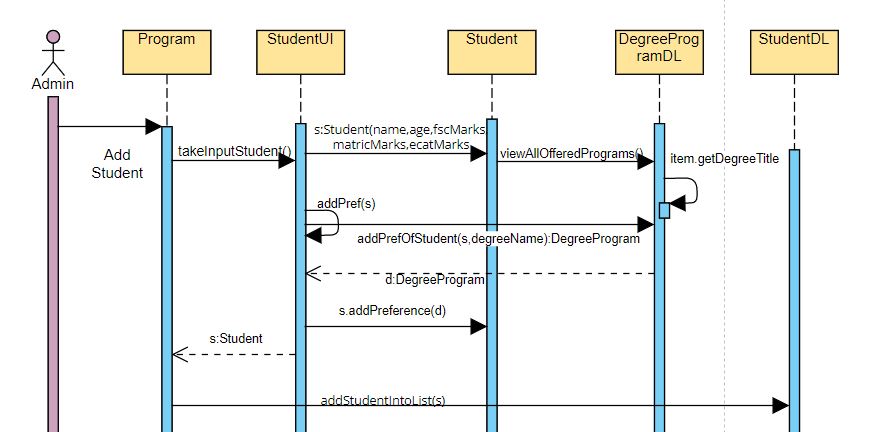
∞

1

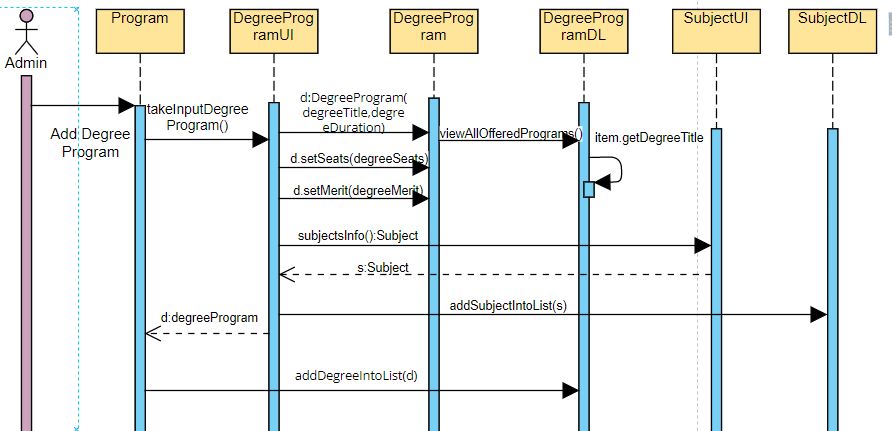


**Step 6: Sequence Diagrams**

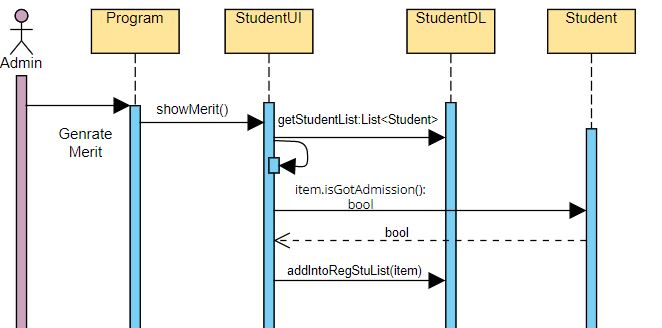
**Option 1: Add Student**

****

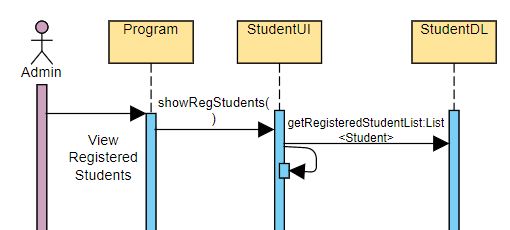
**Option 2: Add Degree Program**

****

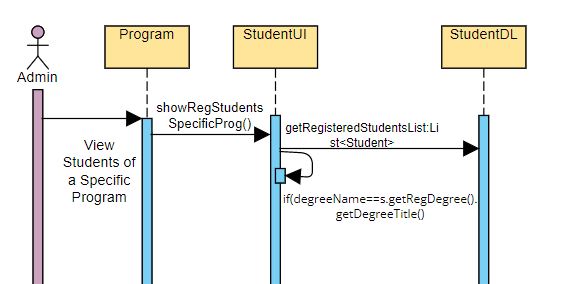
**Option 3: Generate Merit**

****

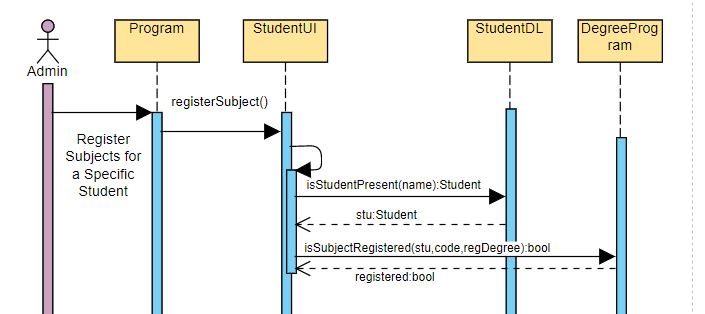
**Option 4: View Registered Students**

****

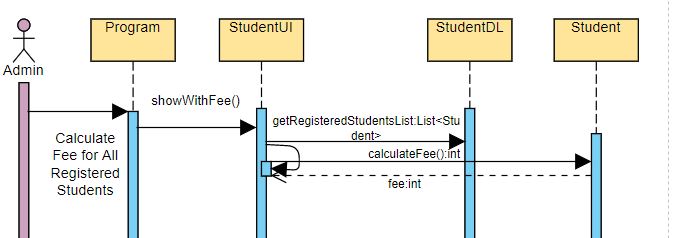
**Option 5: View Students of a Specific Program**

****

**Option 6: View Subjects for a Specific Student**

****

**Option 7: Calculate fee for all registered students**

****

**Full Code**

**BL Code: DegreeProgram.cs**

class DegreeProgram

{

    private string degreeTitle;

    private double degreeDuration;

    private double merit;

    private int seats;

    private List<Subject> subjectsList;

    private List<Student> regStudentsList;

    public DegreeProgram (string degreeTitle , double degreeDuration)

    {

        this.degreeTitle = degreeTitle;

        this.degreeDuration = degreeDuration;

        subjectsList = new List<Subject>();

    }

    //GETTERS

    public string getDegreeTitle ()

    {

        return degreeTitle;

    }

    public double getDegreeDuration ()

    {

        return degreeDuration;

    }

    public double getMerit ()

    {

        return merit;

    }

    public int getSeats ()

    {

        return seats;

    }

    public List<Subject> getSubjectsList ()

    {

        return subjectsList;

    }

    public List<Student> getRegStudentsList ()

    {

        return regStudentsList;

    }

    //SETTERS

    public void setDegreeTitle (string degreeTitle)

    {

        this.degreeTitle = degreeTitle;

    }

    public void getSubjectsList (List<Subject> subjectsList)

    {

        this.subjectsList = subjectsList;

    }

    public void getRegStudentsList (List<Student> regStudentsList)

    {

        this.regStudentsList = regStudentsList;

    }

    public void setSeats (int seats)

    {

        this.seats = seats;

    }

    public void setMerit (double merit)

    {

        this.merit = merit;

    }

    public void setDegreeDuration (double degreeDuration)

    {

        this.degreeDuration = degreeDuration;

    }

    //Behaviours

    public int calculateCreditHours ()

    {

        int count = 0;

        for (int i = 0 ; i < subjectsList.Count ; i++)

        {

            count += subjectsList[i].getCreditHours();

        }

        return count;

    }

    public bool addSubject (Subject s)

    {

        int creditHrs = calculateCreditHours();

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

        {

            subjectsList.Add(s);

            return true;

        }

        else

        {

            //  Console.WriteLine("Limit Exceeded");

            return false;

        }

    }

    public bool isSubjectExist (Subject s)

    {

        foreach (Subject sub in subjectsList)

        {

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

            {

                return true;

            }

        }

        return false;

    }

    public static bool isSubjectRegistered (Student stu , string code , DegreeProgram regDegree)

    {

        foreach (Subject s in stu.getRegDegree().subjectsList)

        {

            if (s.getCode() == code && !(stu.getRegSubject().Contains(s)))

            {

                stu.regStudentSubject(s);

                return true;

            }

        }

        return false;

    }

}

**BL Code: Student.cs**

class Student

  {

      private string name;

      private int age;

      private int fscMarks;

      private int matricMarks;

      private int ecatMarks;

      private double merit;

      private List<DegreeProgram> preferences;

      private List<Subject> regSubject;

      private DegreeProgram admPref;

      private DegreeProgram regDegree;

      public Student (string name , int age , int fscMarks , int matricMarks , int ecatMarks)

      {

          this.name = name;

          this.age = age;

          this.fscMarks = fscMarks;

          this.matricMarks = matricMarks;

          this.ecatMarks = ecatMarks;

          preferences = new List<DegreeProgram>();

          regSubject = new List<Subject>();

          regDegree = null;

      }

      public Student (string name , int age , int fscMarks , int ecatMarks)

      {

          this.name = name;

          this.age = age;

          this.fscMarks = fscMarks;

          this.ecatMarks = ecatMarks;

          preferences = new List<DegreeProgram>();

          regSubject = new List<Subject>();

          regDegree = null;

      }

      public string getName ()

      {

          return name;

      }

      public int getAge ()

      {

          return age;

      }

      public int getFscMarks ()

      {

          return fscMarks;

      }

      public int getMatricMarks ()

      {

          return matricMarks;

      }

      public int getEcatMarks ()

      {

          return ecatMarks;

      }

      public double getMerit ()

      {

          return merit;

      }

      public List<DegreeProgram> getPreferences ()

      {

          return preferences;

      }

      public List<Subject> getRegSubject ()

      {

          return regSubject;

      }

      public DegreeProgram getadmPref ()

      {

          return admPref;

      }

      public DegreeProgram getRegDegree ()

      {

          return regDegree;

      }

      public void setName (string name)

      {

          this.name = name;

      }

      public void setAge (int age)

      {

          this.age = age;

      }

      public void setFscMarks (int fscMarks)

      {

          this.fscMarks = fscMarks;

      }

      public void setMatricMarks (int matricMarks)

      {

          this.matricMarks = matricMarks;

      }

      public void setEcatMarks (int ecatMarks)

      {

          this.ecatMarks = ecatMarks;

      }

      public void setMerit (double merit)

      {

          this.merit = merit;

      }

      public void setPreferences (List<DegreeProgram> preferences)

      {

          this.preferences = preferences;

      }

      public void setRegSubject (List<Subject> regSubject)

      {

          this.regSubject = regSubject;

      }

      public void setadmPref (DegreeProgram admPref)

      {

          this.admPref = admPref;

      }

      public void setRegDegree (DegreeProgram regDegree)

      {

          this.regDegree = regDegree;

      }

      public void addPreference (DegreeProgram d)

      {

          preferences.Add(d);

      }

      public int getCreditHours ()

      {

          int ch = 0;

          //    Console.WriteLine(regSubject.Count);

          for (int i = 0 ; i < regSubject.Count ; i++)

          {

              ch += regSubject[i].getCreditHours();

              //  Console.WriteLine(ch);

              //Console.WriteLine(regSubject[i].creditHours);

          }

          //  Console.WriteLine(ch);

          return ch;

      }

      public int calculateFee ()

      {

          int fee = 0;

          if (regDegree != null)

          {

              foreach (Subject sub in regSubject)

              {

                  fee += sub.getSubjectFee();

              }

          }

          // int studentFee = getCreditHours() \* 3000;

          //   Console.WriteLine(studentFee);

          return fee;

      }

      public double calculateMerit ()

      {

          /\* Console.WriteLine(fscMarks);

           Console.WriteLine(ecatMarks);\*/

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

          //            Console.WriteLine(merit);

          return merit;

      }

      public bool regStudentSubject (Subject s)

      {

          int stCH = getCreditHours();

          if (regDegree != null && preferences[0].isSubjectExist(s) && stCH + s.getCreditHours() <= 9)

          {

              regSubject.Add(s);

              return true;

          }

          return false;

          /\* else

           {

               //  Console.WriteLine("A student cannot have more than 9 subjects or wrong code");

           }\*/

      }

      public bool isGotAdmission ()

      {

          double merit = calculateMerit();

          for (int i = 0 ; i < preferences.Count ; i++)

          {

              if (preferences[i].getMerit() <= merit)

              {

                  DegreeProgram temp = preferences[0];

                  preferences[0] = preferences[i];

                  admPref = preferences[0];

                  regDegree = preferences[0];

                  int seats = preferences[0].getSeats();

                  seats--;

                  preferences[0].setSeats(seats);

                  return true;

              }

          }

          admPref = preferences[0];

          return false;

      }

  }

**U**

**BL Code: Product.cs**

**BL Code: Subject.cs**

class Subject

   {

       private string code;

       private string type;

       private int creditHours;

       private int subjectFee;

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

       {

           this.code = code;

           this.type = type;

           this.creditHours = creditHours;

           this.subjectFee = subjectFee;

       }

       public string getCode ()

       {

           return code;

       }

       public string getType ()

       {

           return type;

       }

       public int getCreditHours ()

       {

           return creditHours;

       }

       public int getSubjectFee ()

       {

           return subjectFee;

       }

       public void setCode (string code)

       {

           this.code = code;

       }

       public void setType (string type)

       {

           this.type = type;

       }

       public void setSubjectFee (int subjectFee)

       {

           this.subjectFee = subjectFee;

       }

       public void setCreditHours (int creditHours)

       {

           this.creditHours = creditHours;

       }

   }

**DL Code: DegreeProgramDL.cs**

class DegreeProgramDL

  {

      private static List<DegreeProgram> degreeProgList = new List<DegreeProgram>();

      public static List<DegreeProgram> getDegreeProgList ()

      {

          return degreeProgList;

      }

      public static void addDegreeIntoList (DegreeProgram d)

      {

          degreeProgList.Add(d);

      }

      public static DegreeProgram isDegreeExists (string degreeName)

      {

          foreach (DegreeProgram d in degreeProgList)

          {

              if (d.getDegreeTitle() == degreeName)

              {

                  return d;

              }

          }

          return null;

      }

      public static DegreeProgram addPrefOfStudent (Student s , string degreeName)

      {

          foreach (DegreeProgram item in degreeProgList)

          {

              if (item.getDegreeTitle() == degreeName)

              {

                  return item;

              }

          }

          return null;

      }

      public static void viewALLOfferedPrograms ()

      {

          Console.WriteLine("Available Programs:");

          foreach (DegreeProgram item in degreeProgList)

          {

              Console.WriteLine(item.getDegreeTitle());

          }

      }

      public static void storeIntoFile (string path , DegreeProgram d)

      {

          StreamWriter f = new StreamWriter(path , true);

          string subjectName = "";

          for (int i = 0 ; i < d.getSubjectsList().Count ; i++)

          {

              if (i != d.getSubjectsList().Count - 1)

              {

                  subjectName += d.getSubjectsList()[i].getType() + ";";

              }

              else

              {

                  subjectName += d.getSubjectsList()[i].getType();

              }

          }

          f.WriteLine(d.getDegreeTitle() + "," + d.getDegreeDuration() + "," + d.getSeats() + "," + subjectName);

          f.Flush();

          f.Close();

      }

      public static bool loadFromFile (string path)

      {

          StreamReader f = new StreamReader(path);

          string record;

          if (File.Exists(path))

          {

              while ((record = f.ReadLine()) != null)

              {

                  string[] splittedRecord = record.Split(',');

                  string degreeTitle = splittedRecord[0];

                  double degreeDuration = double.Parse(splittedRecord[1]);

                  int seats = int.Parse(splittedRecord[2]);

                  string[] splittedRecordSub = splittedRecord[3].Split(';');

                  DegreeProgram d = new DegreeProgram(degreeTitle , degreeDuration);

                  d.setSeats(seats);

                  for (int i = 0 ; i < splittedRecordSub.Length ; i++)

                  {

                      Subject s = SubjectDL.isSubjectExists(splittedRecordSub[i]);

                      if (s != null)

                      {

                          d.addSubject(s);

                      }

                  }

                  addDegreeIntoList(d);

              }

              f.Close();

              return true;

          }

          else

          {

              return false;

          }

      }

  }

**DL Code: StudentDL.cs**

class StudentDL

 {

     private static List<Student> studentList = new List<Student>();

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

     private static List<Student> registeredStudentsList = new List<Student>();

     public static List<Student> getStudentList ()

     {

         return studentList;

     }

     public static List<Student> getRegisteredStudentsList ()

     {

         return registeredStudentsList;

     }

     public static List<Student> getSortedStudentList ()

     {

         return sortedStudentList;

     }

     public static void sortStudentsByMerit ()

     {

         foreach (Student s in registeredStudentsList)

         {

             s.calculateMerit();

         }

         sortedStudentList = registeredStudentsList.OrderByDescending(o => o.getMerit()).ToList();

     }

     public static Student isStudentPresent (string name)

     {

         foreach (Student s in studentList)

         {

             if (name == s.getName() && s.isGotAdmission())

             {

                 return s;

             }

         }

         return null;

     }

     public static void addStudentIntoList (Student s)

     {

         studentList.Add(s);

     }

     public static void addIntoRegStuList (Student s)

     {

         registeredStudentsList.Add(s);

     }

     public static bool loadFromFile (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]);

                 int fscMarks = int.Parse(splittedRecord[2]);

                 int ecatMarks = int.Parse(splittedRecord[3]);

                 string[] splittedRecordPref = splittedRecord[4].Split(';');

                 Student s = new Student(name , age , fscMarks , ecatMarks);

                 for (int i = 0 ; i < splittedRecordPref.Length ; i++)

                 {

                     DegreeProgram d = DegreeProgramDL.isDegreeExists(splittedRecordPref[i]);

                     if (d != null)

                     {

                         s.addPreference(d);

                     }

                 }

                 addStudentIntoList(s);

             }

             f.Close();

             return true;

         }

         else

         {

             return false;

         }

     }

     public static void storeIntoFile (string path , Student s)

     {

         StreamWriter f = new StreamWriter(path , true);

         string degreeName = "";

         for (int i = 0 ; i < s.getPreferences().Count ; i++)

         {

             if (i != s.getPreferences().Count - 1)

             {

                 degreeName += s.getPreferences()[i].getDegreeTitle() + ";";

             }

             else

             {

                 degreeName += s.getPreferences()[i].getDegreeTitle();

             }

         }

         f.WriteLine(s.getName() + "," + s.getAge() + "," + s.getFscMarks() + "," + s.getEcatMarks() + "," + degreeName);

         f.Flush();

         f.Close();

     }

 }

**DL Code: SubjectDL.cs**

class SubjectDL

    {

        private static List<Subject> subjectsList = new List<Subject>();

        public static List<Subject> getSubjectsList ()

        {

            return subjectsList;

        }

        public static void addSubjectIntoList (Subject s)

        {

            subjectsList.Add(s);

        }

        public static Subject isSubjectExists (string type)

        {

            foreach (Subject s in subjectsList)

            {

                if (s.getType() == type)

                {

                    return s;

                }

            }

            return null;

        }

        public static void storeIntoFile (string path , Subject s)

        {

            StreamWriter f = new StreamWriter(path , true);

            f.WriteLine(s.getCode() + "," + s.getType() + "," + s.getCreditHours() + "," + s.getSubjectFee());

            f.Flush();

            f.Close();

        }

        public static bool loadFromFile (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 subjectFee = int.Parse(splittedRecord[3]);

                    Subject s = new Subject(code , type , creditHours , subjectFee);

                    addSubjectIntoList(s);

                }

                f.Close();

                return true;

            }

            else

            {

                return false;

            }

        }

    }

**UI Code: DegreeProgramUI.cs**

class DegreeProgramUI

   {

       public static DegreeProgram takeInputDegreeProgram ()

       {

           string degreeTitle;

           double degreeDuration, degreeMerit;

           int degreeSeats, subCount;

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

           degreeTitle = (Console.ReadLine());

           Console.WriteLine("Enter Degree Duration:");

           degreeDuration = double.Parse(Console.ReadLine());

           // Object

           DegreeProgram d = new DegreeProgram(degreeTitle , degreeDuration);

           // Behaviors

           Console.WriteLine("Enter Seats for degree:");

           degreeSeats = int.Parse(Console.ReadLine());

           Console.WriteLine("Enter Merit for degree:");

           degreeMerit = double.Parse(Console.ReadLine());

           d.setMerit(degreeMerit);

           d.setSeats(degreeSeats);

           // Subjects

           Console.WriteLine("Enter How many subjects to enter:");

           subCount = int.Parse(Console.ReadLine());

           // Subjects List

           for (int i = 0 ; i < subCount ; i++)

           {

               Subject s = SubjectUI.subjectsInfo();

               if (d.addSubject(s))

               {

                   // These are done here because we did not add a separate option to add only the Subjects.

                   if (!(SubjectDL.getSubjectsList().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");

                   i--;

               }

           }

           return d;

       }

   }

**UI Code: StudentUI.cs**

class StudentUI

{

    public static Student takeInputStudent ()

    {

        string name;

        int age;

        int fscMarks;

        int matricMarks;

        int ecatMarks;

        Console.WriteLine("Enter Student Name:");

        name = (Console.ReadLine());

        Console.WriteLine("Enter Student Age:");

        age = int.Parse(Console.ReadLine());

        Console.WriteLine("Enter FSC Marks:");

        fscMarks = int.Parse(Console.ReadLine());

        Console.WriteLine("Enter Matric Marks:");

        matricMarks = int.Parse(Console.ReadLine());

        Console.WriteLine("Enter ECAT Marks:");

        ecatMarks = int.Parse(Console.ReadLine());

        Student s = new Student(name , age , fscMarks , matricMarks , ecatMarks);

        // Show all programs

        DegreeProgramDL.viewALLOfferedPrograms();

        addPref(s);

        return s;

    }

    public static void addPref (Student s)

    {

        Console.WriteLine("Enter How Many Preference to Enter:");

        int prefCount = int.Parse(Console.ReadLine());

        for (int i = 0 ; i < prefCount ; i++)

        {

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

            string degreeName = (Console.ReadLine());

            DegreeProgram d = DegreeProgramDL.addPrefOfStudent(s , degreeName);

            if (d != null)

            {

                s.addPreference(d);

                Console.WriteLine("Preference added");

            }

            else

            {

                Console.WriteLine("Wrong Entry");

                i--;

            }

        }

    }

    public static void showMerit ()

    {

        foreach (Student item in StudentDL.getStudentList())

        {

            if (item.isGotAdmission())

            {

                if (!(StudentDL.getRegisteredStudentsList().Contains(item)))

                {

                    StudentDL.addIntoRegStuList(item);

                }

                Console.WriteLine($"{item.getName()} GOT ADMISSION IN {item.getadmPref().getDegreeTitle()}");

            }

            else

            {

                Console.WriteLine($"{item.getName()} DID NOT GOT ADMISSION IN {item.getadmPref().getDegreeTitle()}");

            }

        }

    }

    public static void showRegStudents ()

    {

        /\*foreach (Student item in Student.studentList)

        {

            Console.WriteLine($"{item.getName()} ");

        }\*/

        Console.WriteLine("NAME\tFSC\tECAT\tAGE");

        foreach (Student s in StudentDL.getRegisteredStudentsList())

        {

            Console.WriteLine($"{s.getName()}\t{s.getFscMarks()}\t{s.getEcatMarks()}\t{s.getAge()}");

        }

    }

    public static void showRegStudentsSpecificProg ()

    {

        Console.WriteLine("Enter Degree Name");

        string degreeName = Console.ReadLine();

        Console.WriteLine("NAME\tFSC\tECAT\tAGE");

        foreach (Student s in StudentDL.getRegisteredStudentsList())

        {

            if (degreeName == s.getRegDegree().getDegreeTitle())

            {

                Console.WriteLine($"{s.getName()}\t{s.getFscMarks()}\t{s.getEcatMarks()}\t{s.getAge()}");

            }

        }

    }

    public static void registerSubject ()

    {

        Console.WriteLine("Enter Your Name");

        string name = Console.ReadLine();

        Console.WriteLine("Enter How Many Subjects You want To Register");

        int count = int.Parse(Console.ReadLine());

        for (int i = 0 ; i < count ; i++)

        {

            Console.WriteLine("Enter Subject Code");

            string code = Console.ReadLine();

            Student stu = StudentDL.isStudentPresent(name);

            if (stu != null)

            {

                DegreeProgram regDegree = stu.getRegDegree();

                bool registered = DegreeProgram.isSubjectRegistered(stu , code , regDegree);

                if (registered)

                {

                    Console.WriteLine("Subject Registered");

                }

                else

                {

                    Console.WriteLine("A student cannot have more than 9 subjects or wrong code");

                    i--;

                }

            }

            else

            {

                Console.WriteLine("Enter Valid name");

                break;

                i--;

            }

        }

    }

    public static void showWithFee ()

    {

        int fee;

        Console.WriteLine("NAME\tFSC\tECAT\tAGE\tTotal Fee");

        foreach (Student s in StudentDL.getRegisteredStudentsList())

        {

            fee = s.calculateFee();

            Console.WriteLine($"{s.getName()}\t{s.getFscMarks()}\t{s.getEcatMarks()}\t{s.getAge()}\t{fee}");

        }

    }

}

**UI Code: SubjectUI.cs**

class SubjectUI

    {

        public static Subject subjectsInfo ()

        {

            Console.WriteLine("Enter Subject Code:");

            string code = (Console.ReadLine());

            Console.WriteLine("Enter Subject Type:");

            string type = (Console.ReadLine());

            Console.WriteLine("Enter Credit Hours:");

            int ch = int.Parse(Console.ReadLine());

            Console.WriteLine("Enter Subject Fee:");

            int fee = int.Parse(Console.ReadLine());

            Subject s = new Subject(code , type , ch , fee);

            return s;

        }

    }

**UI Code: MenuUI.cs**

class MenuUI

  {

      public static int mainMenu ()

      {

          int op;

          header();

          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 fee for all Registered Students");

          Console.WriteLine("8- Exit");

          op = int.Parse(Console.ReadLine());

          return op;

      }

      public static void header ()

      {

          Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

          Console.WriteLine("             UAMS             ");

          Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

      }

      public static void clearScreen ()

      {

          //Console.WriteLine(StudentDL.registeredStudentsList.Count);

          Console.WriteLine("Press any key to continue...");

          Console.ReadKey();

          Console.Clear();

      }

  }

**Driver Program: Program.cs**

class Program

   {

       static void Main ()

       {

           string subjectPath = "subject.txt";

           string degreePath = "degree.txt";

           string studentPath = "student.txt";

           if (SubjectDL.loadFromFile(subjectPath))

           {

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

           }

           if (DegreeProgramDL.loadFromFile(degreePath))

           {

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

           }

           if (StudentDL.loadFromFile(studentPath))

           {

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

           }

           int op = 0;

           while (op < 8)

           {

               MenuUI.clearScreen();

               op = MenuUI.mainMenu();

               if (op == 1)

               {

                   MenuUI.clearScreen();

                   if (DegreeProgramDL.getDegreeProgList().Count > 0)

                   {

                       MenuUI.header();

                       Student s = StudentUI.takeInputStudent();

                       StudentDL.addStudentIntoList(s);

                       StudentDL.storeIntoFile(studentPath , s);

                   }

               }

               else if (op == 2)

               {

                   MenuUI.clearScreen();

                   MenuUI.header();

                   DegreeProgram deg = DegreeProgramUI.takeInputDegreeProgram();

                   DegreeProgramDL.addDegreeIntoList(deg);

                   DegreeProgramDL.storeIntoFile(degreePath , deg);

               }

               else if (op == 3)

               {

                   MenuUI.clearScreen();

                   MenuUI.header();

                   StudentUI.showMerit();

               }

               else if (op == 4)

               {

                   MenuUI.clearScreen();

                   MenuUI.header();

                   StudentUI.showRegStudents();

               }

               else if (op == 5)

               {

                   MenuUI.clearScreen();

                   MenuUI.header();

                   StudentUI.showRegStudentsSpecificProg();

               }

               else if (op == 6)

               {

                   MenuUI.clearScreen();

                   MenuUI.header();

                   StudentUI.registerSubject();

               }

               else if (op == 7)

               {

                   MenuUI.clearScreen();

                   MenuUI.header();

                   StudentUI.showWithFee();

               }

           }

       }

   }