**Assignment: Multiple Classes with Association**



Session: 2021 – 2024

**Submitted by:**

Ali Haider 2021-CS-38

**Supervised by:**

Sir.Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

Table of Contents

1-University Admission Management System………………………………………………………………………….4

1.1-Case Study………………………………………………………………………………………………………………….........4

1.2-Domain Model with Class Names……………………………………………………………………………………….4

1.3-Domain Model with Relation and Constraints……………………………………………………………………………………………………………………………….5

1.4-Domain Model with Multiplicity…………………………………………………………………………………………6

1.5-Class Diagram ……………………………………………………………………………………………………………………..7

1.7-Complete Code…………………………………………………………………………………………………………………8

2-Ocean Navigation…………………………………………………………………………. 24

2.1-Case Study…………………………………………………………………………. 24

2.2-Domain Model with Class Names…………………………………………………………………………. 24

2.3-Domain Model with Relation and Constraints…………………………………………………………………………. 24

2.4-Domain Model with Multiplicity…………………………………………………………………………. 24

2.5-Class Diagram …………………………………………………………………………. 25

2.7-Complete Code…………………………………………………………………………. 25

3-Point and Line…………………………………………………………………………. 33

3.1-Case Study ………………………………………………………………………….33

3.2-Domain Model with Class Names…………………………………………………………………………. 33

3.3-Domain Model with Relation and Constraints…………………………………………………………. 34

3.4-Domain Model with Multiplicity…………………………………………………………………………. 34

3.5-Class Diagram …………………………………………………………… 35

3.7-Complete Code ………………………………………………………………………….35

4-Point of Sale ………………………………………………………………………….42

4.1-Case Study ………………………………………………………………………….42

4.2-Domain Model with Class Names…………………………………………………………………………. 42

4.3-Domain Model with Relation and Constraints………………………………………………………………………….42

4.4-Domain Model with Multiplicity…………………………………………………………………………. 43

4.5-Class Diagram …………………………………………………………………………. 44

4.7-Complete Code…………………………………………………………………………. 45

5-Telsa’s Coffee Shop…………………………………………………………………………. 57

5.1-Case Study ………………………………………………………………………….57

5.2-Domain Model with Class Names…………………………………………………………………………. 57

5.3-Domain Model with Relation and Constraints………………………………………………………………………… 58

5.4-Domain Model with Multiplicity…………………………………………………………………………. 58

5.5-Class Diagram …………………………………………………………………………. 59

5.7-Complete Code…………………………………………………………………………. 60

6-Game Object ………………………………………………………………………………………………….67

6.1-Case Study …………………………………………………………………………. ……………………..67

6.2-Domain Model with Class Names…………………………………………………………………………. 67

6.3-Domain Model with Relation and Constraints………………………………………………………………………… 67

6.4-Domain Model with Multiplicity…………………………………………………………………………. 67

6.5-Class Diagram with Attributes and Functions…………………………………………………………………… 68

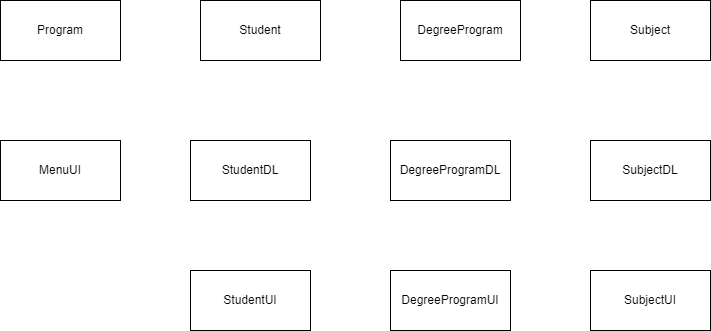
6.7-Complete Code………………………………………………………………………….68

**1-UAMS**

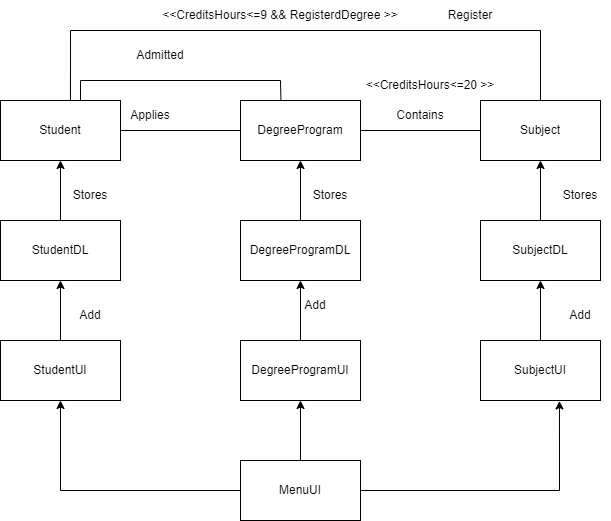
**1.1-Case Study Scenario**

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.

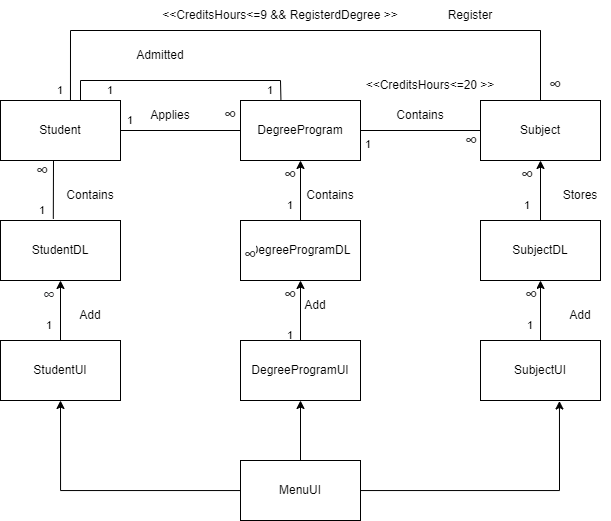
**1.2- Domain Model with only Class Name**

****

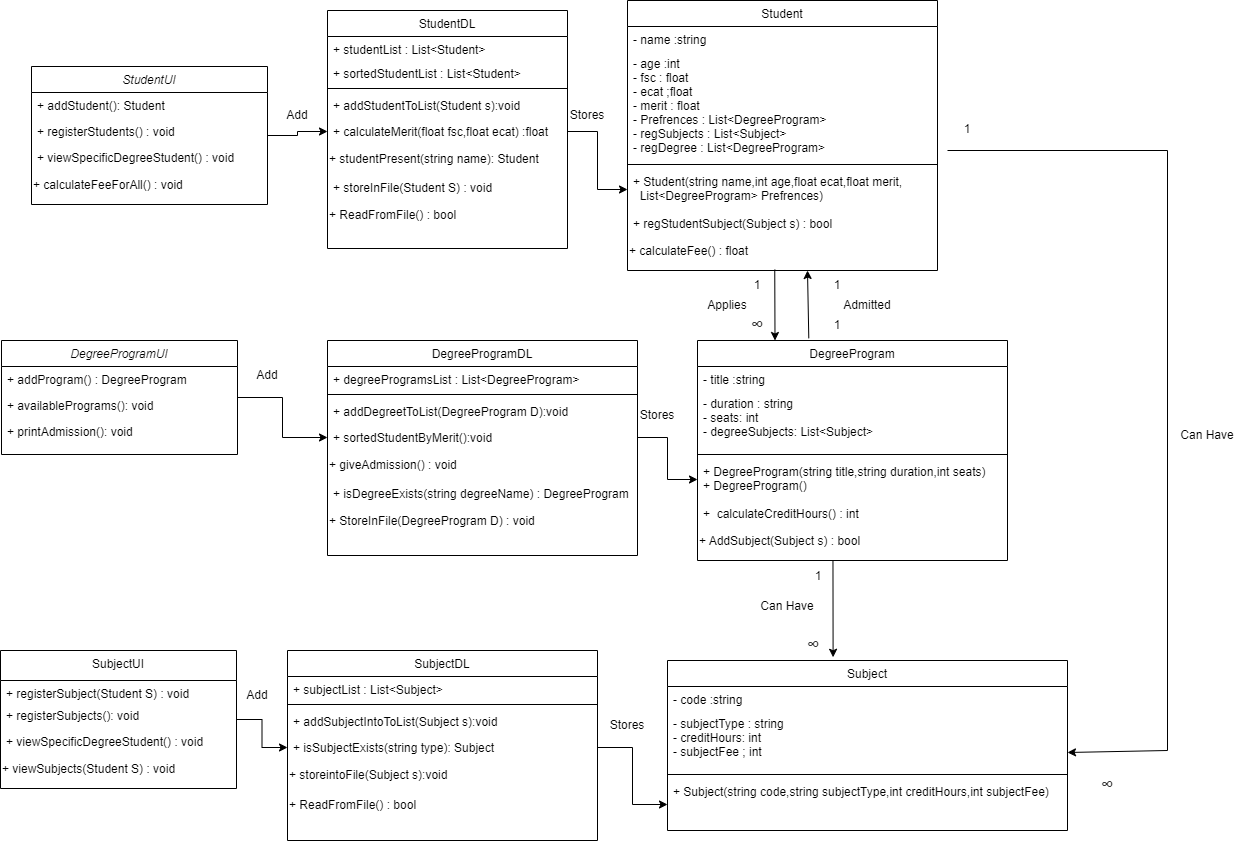
**1.3- Domain Model with Constraints and Relation**

****

**1.4- Domain Model with Multiplicity**

****

**1.5- Class Diagram :**

****

**1.7- Complete Code**

**1.7.1-Business Logic**

**1.7.1.1-Student Class**

class Student

{

private string name;

private int age;

private float fsc;

private float ecat;

private float merit;

private List<DegreeProgram> Prefrences = new List<DegreeProgram>();

private List<Subject> regSubjects = new List<Subject>();

public Student(string name, int age, float fsc, float ecat, float merit, List<DegreeProgram> Prefrences)

{

this.name = name;

this.age = age;

this.fsc = fsc;

this.ecat = ecat;

this.Prefrences = Prefrences;

this.merit = merit;

this.regSubjects = new List<Subject>();

this.regDegree = null;

}

public int getCreditHours()

{

int count = 0;

foreach (Subject sub in regSubjects)

{

count = count + sub.getCreditHours();

}

return count;

}

public bool regStudentSubject(Subject s)

{

int stCH = getCreditHours();

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

{

regSubjects.Add(s);

return true;

}

else

{

return false;

}

}

public float calculateFee()

{

float fee = 0;

if (regDegree != null)

{

foreach (Subject sub in regSubjects)

{

fee = fee + sub.getSubjectFees();

}

}

return fee;

}

public void setStudentName(string name)

{

this.name = name;

}

public void setAge(int age)

{

this.age = age;

}

public void setFsc(int fsc)

{

this.fsc = fsc;

}

public void setEcat(int ecat)

{

this.ecat = ecat;

}

public void setMerit(float merit)

{

this.merit = merit;

}

public void setRegSubjects(List<Subject> regSubject)

{

this.regSubjects = regSubject;

}

public void setRegDegree(DegreeProgram regDegree)

{

this.regDegree = regDegree;

}

public void setPreferance(List<DegreeProgram> Preferences)

{

this.Prefrences = Preferences;

}

//Get Functions

public string getName()

{

return name; ;

}

public int getAge()

{

return age;

}

public float getfsc()

{

return fsc;

}

public float getecat()

{

return ecat;

}

public float getMerit()

{

return merit;

}

public List<Subject> getRegSubjects()

{

return regSubjects;

}

public DegreeProgram GetRegDegree()

{

return regDegree;

}

public List<DegreeProgram> getPrefrences()

{

return Prefrences;

}

}

1.7.1.2- DegreeProgram Class

class DegreeProgram

{

private string title;

private string duration;

private int seats;

private List<Subject> degreeSubjects = new List<Subject>();

public DegreeProgram(string title,string duration,int seats)

{

this.title = title;

this.duration = duration;

this.seats = seats;

degreeSubjects = new List<Subject>();

}

public DegreeProgram()

{

}

public int calculateCreditHours()

{

int count = 0;

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

{

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

}

return count;

}

public bool AddSubject(Subject s)

{

int creditHours = calculateCreditHours();

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

{

degreeSubjects.Add(s);

return true;

}

else

{

return false;

}

}

public void setTitle(string title)

{

this.title = title;

}

public void setDuration(string duration)

{

this.duration = duration;

}

public void setSeats(int seats)

{

this.seats = seats;

}

public void setDegreeSubjects(List<Subject> degreeSubjects)

{

this.degreeSubjects = degreeSubjects;

}

//\_\_\_\_\_\_\_\_\_\_\_Getter Functions

public string getTitle()

{

return title;

}

public string getDuration()

{

return duration;

}

public int getSeats()

{

return seats;

}

public List<Subject> getDegreeSubjects()

{

return degreeSubjects;

}

}

1.7.1.3- Subject Class

class Subject

{

private string code;

private string subjectType;

private int creditHours;

private int subjectFees;

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

{

this.code = code;

this.creditHours = creditHours;

this.subjectType = subjectType;

this.subjectFees = subjectFees;

}

public void setSubjectCode(string code)

{

this.code = code;

}

public void setsubjectType(string subjectType)

{

this.subjectType = subjectType;

}

public void setCreditHour(int creditHours)

{

this.creditHours = creditHours;

}

public void setEcat(int subjectFees)

{

this.subjectFees = subjectFees;

}

public string getCode()

{

return code; ;

}

public string getSubjectType()

{

return subjectType;

}

public int getCreditHours()

{

return creditHours; ;

}

public float getSubjectFees()

{

return subjectFees;

}

}

1.7.2- Data Layer

1.7.2.1- StudentDL

class StudentDL

{

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

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

public static void addStudentToList(Student s)

{

studentList.Add(s);

}

public static float calculateMerit(float fsc,float ecat)

{

float merit = (fsc / 1100) \* 60 + (ecat / 400) \* 40;

return merit;

}

public static Student studentPresent(string name)

{

foreach (Student S in StudentDL.studentList)

{

if (name == S.getName())

{

return S;

}

}

return null;

}

public static void storeInFile(Student S)

{

string path = "studentLists.txt";

StreamWriter file = new StreamWriter(path,true);

string degreesNames="";

for (int i = 0; i < S.getPrefrences().Count-1; i++)

{

degreesNames = degreesNames + S.getPrefrences()[i].getTitle() + ";";

}

degreesNames = degreesNames + S.getPrefrences()[S.getPrefrences().Count - 1].getTitle();

file.WriteLine(S.getName() + "," + S.getAge() + "," + S.getfsc() + "," + S.getecat() + "," + degreesNames);

file.Flush();

file.Close();

}

public static bool ReadFromFile()

{

string path = "studentLists.txt";

StreamReader file = new StreamReader(path);

string record;

if(File.Exists(path))

{

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

{

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

string name = splittedRecord[0];

int age = int.Parse(splittedRecord[1]);

float fsc = float.Parse(splittedRecord[2]);

float ecat = float.Parse(splittedRecord[3]);

float merit = StudentDL.calculateMerit(fsc, ecat);

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, fsc,ecat,merit,preferences);

studentList.Add(s);

}

file.Close();

return true;

}

else

{

return false;

}

}

}

1.7.2.2- DegreeProgramDL

class DegreeProgramDL

{

public static List<DegreeProgram> degreeProgramsList = new List<DegreeProgram>();

public static void addDegreeToList(DegreeProgram D)

{

degreeProgramsList.Add(D);

}

static public void sortedStudentByMerit()

{

StudentDL.sortedStudentList = StudentDL.studentList.OrderByDescending(o => o.getMerit()).ToList();

}

public static void giveAdmission()

{

foreach (Student S in StudentDL.sortedStudentList)

{

foreach (DegreeProgram D in S.getPrefrences())

{

if (D.getSeats() > 0 && S.GetRegDegree() == null)

{

S.setRegDegree(D);

D.setSeats(D.getSeats()-1);

break;

}

}

}

}

public static DegreeProgram isDegreeExists(string degreeName)

{

foreach (DegreeProgram d in degreeProgramsList)

{

if (d.getTitle() == degreeName)

{

return d;

}

}

return null;

}

public static void StoreInFile(DegreeProgram D)

{

string path = "DegreesList.txt";

StreamWriter file = new StreamWriter(path);

string SubjectNames = "";

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

{

SubjectNames = SubjectNames + D.getDegreeSubjects()[i].getSubjectType() + ";";

}

SubjectNames = SubjectNames + D.getDegreeSubjects()[D.getDegreeSubjects().Count-1].getSubjectType() + ";";

file.WriteLine(D.getTitle() + "," + D.getDuration() + "," + D.getSeats() + "," + SubjectNames);

file.Flush();

file.Close();

}

public static bool ReadfromFile()

{

string path = "DegreesList.txt";

StreamReader file = new StreamReader(path);

string record;

if(File.Exists(path))

{

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

{

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

string title = splittedRecord[0];

string duration = splittedRecord[1];

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

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

List<DegreeProgram> preferences = new List<DegreeProgram>();

DegreeProgram d = new DegreeProgram(title, duration, seats);

for (int x = 0; x < splittedRecordforSubject.Length; x++)

{

Subject s = SubjectDL.isSubjectExists(splittedRecordforSubject[x]);

if (s != null)

{

d.AddSubject(s);

}

}

addDegreeToList(d);

}

file.Close();

return true;

}

return false;

}

}

1.7.2.3- SubjectDL

class SubjectDL

{

public static List<Subject> subjectList = new List<Subject>();

public static void addSubjectIntoList(Subject s)

{

subjectList.Add(s);

}

public static Subject isSubjectExists(string type)

{

foreach (Subject s in subjectList)

{

if (s.getSubjectType()== type)

{

return s;

}

}

return null;

}

public static void storeintoFile(Subject s)

{

string path = "SubjectsList.txt";

StreamWriter f = new StreamWriter(path, true);

f.WriteLine(s.getCode() + "," + s.getSubjectType() + "," + s.getCreditHours() + "," + s.getSubjectFees());

f.Flush();

f.Close();

}

public static bool ReadFromFile()

{

string path = "SubjectsList.txt";

StreamReader file = new StreamReader(path);

string record;

if (File.Exists(path))

{

while ((record = file.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);

}

file.Close();

return true;

}

else

{

return false;

}

}

}

1.7.3- User Interface

1.7.3.1- MenuUI

class MenuUI

{

static public void header()

{

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

Console.WriteLine(" ");

Console.WriteLine("---------- UAMS --------");

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

Console.WriteLine(" ");

}

static public string menu()

{

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 Student of Specific Program");

Console.WriteLine("[6]-Register Subject of Specific Student");

Console.WriteLine("[7]-Caluclulate Fees for all Registered Students");

Console.WriteLine("[8]-Exit");

Console.Write("Your Option......");

string op = Console.ReadLine();

return op;

}

public static void clearscreen()

{

Console.WriteLine("Press any to Continue........");

Console.ReadKey();

Console.Clear();

}

}

1.7.3.2- StudentUI

class StudentUI

{

static public Student addStudent()

{

string prefre;

List<DegreeProgram> pref = new List<DegreeProgram>();

Console.Clear();

MenuUI.header();

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

string name = Console.ReadLine();

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

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

Console.Write("Enter Student fsc Marks : ");

float fsc = float.Parse(Console.ReadLine());

Console.Write("Enter Student Ecat marks : ");

float ecat = float.Parse(Console.ReadLine());

float merit = StudentDL.calculateMerit(fsc, ecat);

Console.WriteLine(merit);

DegreeProgramUI.availablePrograms();

Console.Write("Enter How Number of Prefrences : ");

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

Console.WriteLine("Enter Programs");

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

{

prefre = Console.ReadLine();

bool flag = false;

foreach (DegreeProgram d in DegreeProgramDL.degreeProgramsList)

{

if (prefre == d.getTitle() && !(pref.Contains(d)))

{

pref.Add(d);

flag = true;

}

}

if (flag == false)

{

Console.WriteLine("Enter Valid Degree Program Name");

i--;

}

}

Student stu = new Student(name, age, fsc, ecat, merit, pref);

Console.Clear();

return stu;

}

public static void registerStudents()

{

if (DegreeProgramDL.degreeProgramsList.Count > 0)

{

Console.WriteLine("Name\t\tAge\t\tFSC\t\tECAT");

foreach (Student S in StudentDL.studentList)

{

if (S.GetRegDegree() != null)

{

Console.WriteLine(S.getName() + "\t\t" + S.getAge() + "\t\t" + S.getfsc() + "\t\t" + S.getecat());

}

}

}

else

{

Console.WriteLine("No Registered Student");

}

MenuUI.clearscreen();

}

public static void viewSpecificDegreeStudent()

{

if (DegreeProgramDL.degreeProgramsList.Count > 0)

{

bool flag = false;

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

string name = Console.ReadLine();

Console.WriteLine("Name\t\tAge\t\tFSC\t\tECAT");

foreach (Student S in StudentDL.studentList)

{

if (S.GetRegDegree() != null)

{

if (name == S.GetRegDegree().getTitle())

{

flag = true;

Console.WriteLine(S.getName() + "\t\t" + S.getAge() + "\t\t" + S.getfsc() + "\t\t" + S.getecat());

}

}

}

if(flag==false)

{

Console.WriteLine("No Such Degree Available");

}

}

else

{

Console.WriteLine("No Registered Student");

}

MenuUI.clearscreen();

}

public static void calculateFeeForAll()

{

if (DegreeProgramDL.degreeProgramsList.Count > 0)

{

foreach (Student s in StudentDL.studentList)

{

if (s.getRegSubjects() != null)

{

Console.WriteLine(s.getName() + " has " + s.calculateFee() + " fees");

}

}

}

else

{

Console.WriteLine("No Registered Student....");

}

MenuUI.clearscreen();

}

}

1.7.3.3- DegreeProgramUI

class DegreeProgramUI

{

public static DegreeProgram addProgram()

{

List<Subject> degreeSubjects = new List<Subject>();

Console.Clear();

MenuUI.header();

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

string name = Console.ReadLine();

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

string dura = Console.ReadLine();

Console.Write("Enter Seats for Degree : ");

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

Console.Write("Enter How Many subjects : ");

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

DegreeProgram D = new DegreeProgram(name, dura, seats);

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

{

Console.Write("Enter Subject[" + (i + 1) + "] code : ");

string code = Console.ReadLine();

Console.Write("Enter Subject[" + (i + 1) + "] Type : ");

string type = Console.ReadLine();

Console.Write("Enter Subject[" + (i + 1) + "] Credit Hours : ");

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

Console.Write("Enter Subject[" + (i + 1) + "] Fees : ");

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

Subject S = new Subject(code, type, credit, fees);

if (D.AddSubject(S))

{

degreeSubjects.Add(S);

SubjectDL.addSubjectIntoList(S);

SubjectDL.storeintoFile(S);

Console.WriteLine("Subject Added");

}

else

{

Console.WriteLine("Subject Not Added");

Console.WriteLine("20 credit hour limit exceeded");

i--;

}

}

MenuUI.clearscreen();

return D;

}

public static void availablePrograms()

{

Console.Write("Available Degree Programs : ");

foreach (DegreeProgram temp in DegreeProgramDL.degreeProgramsList)

{

Console.Write(temp.getTitle() + " ");

}

Console.WriteLine(" ");

}

public static void printAdmission()

{

foreach (Student S in StudentDL.studentList)

{

if(S.GetRegDegree() != null)

{

Console.WriteLine(S.getName() + " got admission in " + S.GetRegDegree().getTitle());

}

else

{

Console.WriteLine(S.getName() + " did not got admission ");

}

}

MenuUI.clearscreen();

}

}

1.7.3.4- SubjectUI

class SubjectUI

{

public static void registerSubject(Student S)

{

Console.Write("How Many Subject want to Enter : ");

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

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

{

bool flag = false;

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

string code = Console.ReadLine();

foreach (Subject sub in S.GetRegDegree().getDegreeSubjects())

{

if (sub.getCode() == code && !(S.getRegSubjects().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 Code.....");

}

}

}

public static void registerSubjects()

{

if (DegreeProgramDL.degreeProgramsList.Count > 0)

{

Console.Write("Enter Name : ");

string name = Console.ReadLine();

Student S = StudentDL.studentPresent(name);

if (S != null)

{

viewSubjects(S);

registerSubject(S);

}

}

else

{

Console.WriteLine("No Registered Student....");

}

MenuUI.clearscreen();

}

public static void viewSubjects(Student S)

{

if (S.GetRegDegree() != null)

{

Console.WriteLine("Code\t\tType");

foreach (Subject sub in S.GetRegDegree().getDegreeSubjects())

{

Console.WriteLine(sub.getCode() + "\t\t" + sub.getSubjectType());

}

}

}

}

**1.7.4- Driver Program**

class Program

{

static void Main(string[] args)

{

if (SubjectDL.ReadFromFile())

{

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

}

if (DegreeProgramDL.ReadfromFile())

{

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

}

if (StudentDL.ReadFromFile())

{

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

}

while (true)

{

string opt = MenuUI.menu();

if (opt == "1")

{

if (DegreeProgramDL.degreeProgramsList.Count > 0)

{

Student S = StudentUI.addStudent();

StudentDL.addStudentToList(S);

StudentDL.storeInFile(S);

}

else

{

Console.WriteLine("No Programs Available....");

MenuUI.clearscreen();

}

}

else if (opt == "2")

{

DegreeProgram D = DegreeProgramUI.addProgram();

DegreeProgramDL.addDegreeToList(D);

DegreeProgramDL.StoreInFile(D);

}

else if (opt == "3")

{

DegreeProgramDL.sortedStudentByMerit();

DegreeProgramDL.giveAdmission();

DegreeProgramUI.printAdmission();

}

else if (opt == "4")

{

StudentUI.registerStudents();

}

else if (opt == "5")

{

StudentUI.viewSpecificDegreeStudent();

}

else if (opt == "6")

{

SubjectUI.registerSubjects();

}

else if (opt == "7")

{

StudentUI.calculateFeeForAll();

}

else if (opt == "8")

{

break;

}

else

{

Console.WriteLine("Wrong Input......");

MenuUI.clearscreen();

}

}

}

}

**2. Ocean Navigation :**

**2.1-Case Study Scenario:**

In ocean navigation, locations are measured in degrees and minutes of latitude and longitude. Thus if you’re lying off the mouth of Papete Harbor in Tahiti, your location is 149 degrees 34.8 minutes west longitude, and 17 degrees 31.5 minutes south latitude. This is written as 149°34.8’ W, 17°31.5’ S. There are 60 minutes in a degree. (An older system also divided a minute into 60 seconds, but the modern approach is to use decimal minutes instead.) Longitude is measured from 0 to 180 degrees, east or west from Greenwich, England, to the international dateline in the Pacific. Latitude is measured from 0 to 90 degrees, north or south from the equator to the poles.

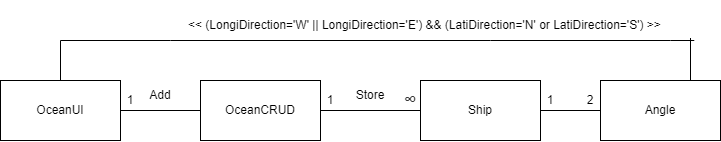
**2.2- Domain Model with Only Class Name**

**classwithname.png**

**2.3- Domain Model with Only Relations and Constraints**



**2.4- Domain Model with Multiplicity**

****

**2.5- Class Diagram**

** 2.7 Complete Code**

**2.7.1- Business Logic**

**2.7.1.1- Ship**

class Ship

{

private string name;

private Angle latitude;

private Angle longitude;

public Ship(string name,Angle longitude,Angle latitude)

{

this.name = name;

this.latitude = latitude;

this.longitude = longitude;

}

public Ship(Angle longitude, Angle latitude)

{

this.latitude = latitude;

this.longitude = longitude;

}

public void setName(string name)

{

this.name = name;

}

public string getName()

{

return name;

}

public Angle getLatitiude()

{

return latitude;

}

public void setLatitude(Angle latitude)

{

this.latitude = latitude;

}

public Angle getLongitude()

{

return longitude;

}

public void setLongitude(Angle longitude)

{

this.longitude=longitude;

}

}

**2.7.1.2- Angle**

class Angle

{

private int degree;

private float minutes;

private char direction;

public Angle(int degree, float minutes, char direction)

{

this.degree = degree;

this.minutes = minutes;

this.direction = direction;

}

public Angle()

{

}

public void setvalue(int d,float m,char direct)

{

degree = d;

m = minutes;

direction = direct;

}

public void disply()

{

Console.WriteLine(degree + "\u00b0" + minutes + "\u00b0" + direction);

}

public void setDegree(int degree)

{

this.degree = degree;

}

public int getDegree()

{

return degree;

}

public void setMinutes(float minutes)

{

this.minutes = minutes;

}

public float getMinutes()

{

return minutes;

}

public void setDirection(char direction)

{

this.direction = direction;

}

public char getDirection()

{

return direction;

}

public bool setDirectionofLongi(char direction)

{

if (direction == 'W' || direction == 'E')

{

this.direction = direction;

return true;

}

else

{

return false;

}

}

public bool setDirectionofLati(char direction)

{

if (direction == 'N' || direction == 'S')

{

this.direction = direction;

return true;

}

else

{

return false;

}

}

}

**2.7.2-Data Layer**

**2.7.2.1- OceanCRUD**

class OceanCRUD

{

static public List<Ship> Ships = new List<Ship>();

static public void addShipstoLists(Ship s)

{

Ships.Add(s);

}

static public string ship\_name(Angle longi,Angle lati)

{

foreach (Ship s in Ships)

{

if(s.getLongitude().getDegree()==longi.getDegree() && s.getLongitude().getMinutes()==longi.getMinutes() && s.getLongitude().getDirection()==longi.getDirection() && s.getLatitiude().getDegree() == lati.getDegree() && s.getLatitiude().getMinutes() == lati.getMinutes() && s.getLatitiude().getDirection() == lati.getDirection())

{

return s.getName();

}

}

return null;

}

static public void StoreInFile()

{

string path = "ships.txt";

StreamWriter file = new StreamWriter(path,true);

foreach (Ship S in Ships)

{

file.WriteLine(S.getName() + "," + S.getLongitude().getDegree() + "," + S.getLongitude().getMinutes() + "," + S.getLongitude().getDirection() + "," + S.getLatitiude().getDegree() + "," + S.getLatitiude().getMinutes() + "," + S.getLatitiude().getDirection());

}

file.Flush();

file.Close();

}

static public bool ReadFromFile()

{

string path = "ships.txt";

StreamReader file = new StreamReader(path);

string record = "";

if(File.Exists(path))

{

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

{

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

string name = splittedRecord[0];

int LogiD = int.Parse(splittedRecord[1]);

float LogiM = float.Parse(splittedRecord[2]);

char LogiDirec = char.Parse(splittedRecord[3]);

Angle A1 = new Angle(LogiD, LogiM, LogiDirec);

int LatiD = int.Parse(splittedRecord[4]);

float LatiM = float.Parse(splittedRecord[5]);

char LatiDirec = char.Parse(splittedRecord[6]);

Angle A2 = new Angle(LatiD, LatiM, LatiDirec);

Ship S = new Ship(name, A1, A2);

addShipstoLists(S);

}

file.Close();

return true;

}

return false;

}

}

**2.7.3- User Interface**

**2.7.3.1- OceanUI**

class OceanUI

{

static public void header()

{

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

Console.WriteLine(" ");

Console.WriteLine(" Ocean Navigation ");

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

Console.WriteLine(" ");

}

static public string menu()

{

header();

Console.WriteLine("[1]- Add Ship");

Console.WriteLine("[2]- View Ship Position");

Console.WriteLine("[3]- View Ship Serial Number");

Console.WriteLine("[4]- Change Ship Position");

Console.WriteLine("[5]- Exit");

Console.Write("Your Option....");

string o = Console.ReadLine();

return o;

}

static public Ship Add\_ship()

{

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

string name = Console.ReadLine();

Console.WriteLine("Enter Ship Longitude....");

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

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

Console.Write("Enter Longitude Minutes : ");

float LongiMin = float.Parse(Console.ReadLine());

char LongiDir;

int x = 0;

Angle A = new Angle();

do

{

if(x>0)

{

Console.WriteLine("Longitude Direction Could be Only W(West) or E(East");

}

Console.Write("Enter Longitude Direction : ");

LongiDir = char.Parse(Console.ReadLine());

x++;

}

while (!A.setDirectionofLongi(LongiDir));

Angle longi = new Angle(LongiDeg, LongiMin, LongiDir);

Console.WriteLine("Enter Ship Latitude....");

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

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

Console.Write("Enter Latitude Minutes : ");

float LatiMin = float.Parse(Console.ReadLine());

char LatiDir;

x = 0;

do

{

if (x > 0)

{

Console.WriteLine("Latitude Direction Could be Only S(South) or N(North");

}

Console.Write("Enter Latitude Direction : ");

LatiDir = char.Parse(Console.ReadLine());

x++;

}

while (!A.setDirectionofLati(LatiDir));

Angle lati = new Angle(LatiDeg, LatiMin, LatiDir);

Ship S = new Ship(name, longi, lati);

Console.Clear();

return S;

}

static public void disply\_ship()

{

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

string s = Console.ReadLine();

bool flag = false;

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

{

if (s == OceanCRUD.Ships[i].getName())

{

flag = true;

Console.WriteLine("Ship is at " + OceanCRUD.Ships[i].getLongitude().getDegree() + "\u00b0" + OceanCRUD.Ships[i].getLongitude().getMinutes() + "\u00b0" + OceanCRUD.Ships[i].getLongitude().getDirection() + " and " + OceanCRUD.Ships[i].getLatitiude().getDegree() + "\u00b0" + OceanCRUD.Ships[i].getLatitiude().getMinutes() + "\u00b0" + OceanCRUD.Ships[i].getLatitiude().getDirection());

}

}

if (flag == false)

{

Console.Write("No Such Ship Avilable");

}

Console.ReadKey();

Console.Clear();

}

static public string searchhByAngle()

{

Console.WriteLine("Enter Ship Longitude....");

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

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

Console.Write("Enter Longitude Minutes : ");

float LongiMin = float.Parse(Console.ReadLine());

Console.Write("Enter Longitude Direction : ");

char LongiDir = char.Parse(Console.ReadLine());

Angle longi = new Angle(LongiDeg, LongiMin, LongiDir);

Console.WriteLine("Enter Ship Latitude....");

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

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

Console.Write("Enter Latitude Minutes : ");

float LatiMin = float.Parse(Console.ReadLine());

Console.Write("Enter Latitude Direction : ");

char LatiDir = char.Parse(Console.ReadLine());

Angle lati = new Angle(LatiDeg, LatiMin, LatiDir);

string name\_Ship = OceanCRUD.ship\_name(longi,lati);

return name\_Ship;

}

static public void replace\_ship()

{

bool flag = false;

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

string s = Console.ReadLine();

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

{

if (s == OceanCRUD.Ships[i].getName())

{

flag = true;

Console.WriteLine("Enter Ship Longitude....");

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

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

Console.Write("Enter Longitude Minutes : ");

float LongiMin = float.Parse(Console.ReadLine());

Console.Write("Enter Longitude Direction : ");

char LongiDir = char.Parse(Console.ReadLine());

Angle longi = new Angle(LongiDeg, LongiMin, LongiDir);

Console.WriteLine("Enter Ship Latitude....");

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

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

Console.Write("Enter Latitude Minutes : ");

float LatiMin = float.Parse(Console.ReadLine());

Console.Write("Enter Latitude Direction : ");

char LatiDir = char.Parse(Console.ReadLine());

Angle lati = new Angle(LatiDeg, LatiMin, LatiDir);

Ship S = new Ship(s,longi, lati);

OceanCRUD.Ships.RemoveAt(i);

OceanCRUD.Ships.Insert(i, S);

Console.Clear();

break;

}

}

if (flag == false)

{

Console.WriteLine("No Such Ships Avilable");

Console.ReadKey();

Console.Clear();

}

}

}

**2.7.3- Driver Program**

static void Main(string[] args)

{

if(OceanCRUD.ReadFromFile())

{

Console.WriteLine("Read Data Succesfully");

}

while (true)

{

string op = OceanUI.menu();

if(op=="1")

{

Ship S = OceanUI.Add\_ship();

OceanCRUD.addShipstoLists(S);

}

else if(op=="2")

{

OceanUI.disply\_ship();

}

else if (op == "3")

{

string name=OceanUI.searchhByAngle();

if(name!=null)

{

Console.WriteLine("Ship Name is : " + name);

}

else

{

Console.WriteLine("No Such Ship Available");

}

Console.ReadKey();

Console.Clear();

}

else if (op == "4")

{

OceanUI.replace\_ship();

}

else if (op == "5")

{

OceanCRUD.StoreInFile();

break;

}

else

{

Console.WriteLine("Wrong Input.....");

Console.ReadLine();

Console.Clear();

}

}

}

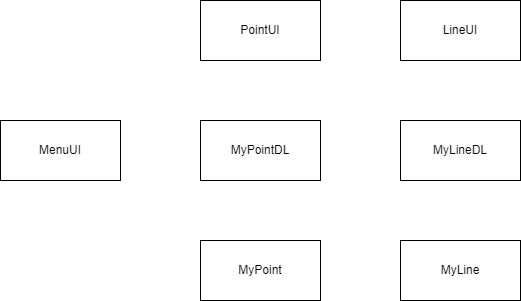
**3. Point and Line**

**3.1-Case Study Scenario:**

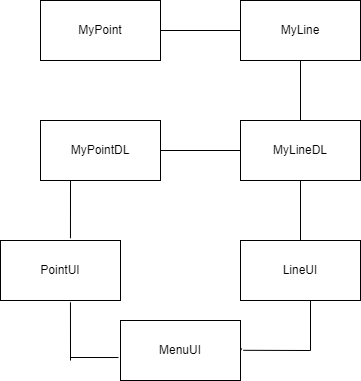
## A class called MyPoint, which models a 2D point with x and y coordinates, is designed as shown in the class diagram. It contains

## ● Two instance variables x (int) and y (int). A default (or "no-argument" or "no-arg") constructor that constructs a point at the default location of (0, 0). A parameterized constructor that constructs a point with the given x and y coordinates. Getter and setter for the instance variables x and y. A method setXY() to set both x and y. A method called distanceWithCords(int x, int y) that returns the distance from this point to another point at the given (x, y) coordinates. A method distanceWithObject(MyPoint another) that returns the distance from this point to the given MyPoint instance (called another). Another method distanceFromZero() method that returns the distance from this point to the origin (0,0). A class called MyLine, which models a line with a begin point at (x1, y1) and an end point at (x2, y2), is designed as shown in the class diagram. The MyLine class uses two MyPoint instances (written in the earlier exercise) as its begin and end points. Write the MyLine class. Also write a test driver to test all the public methods in the MyLine class.

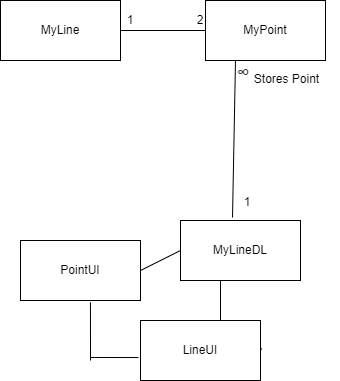
**3.2- Domain Model with only Class Name**

****

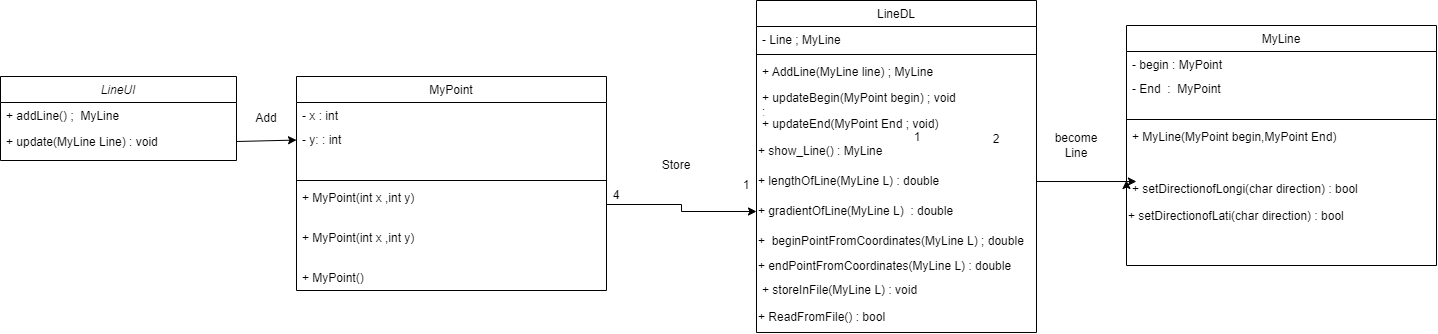
**3.3- Domain Model with Relations and Constraints**

****

**3.4- Domain Model with Multiplicity**



**3.5- Class Diagram**



**3.7- Complete Code**

**3.7.1-Business Logic**

**3.7.1.1- My Point Class**

class MyPoint

{

private int x;

private int y;

public MyPoint()

{

}

public MyPoint(int x,int y)

{

this.x = x;

this.y = y;

}

public void setX(int x)

{

this.x = x;

}

public int getX()

{

return x;

}

public void setY(int y)

{

this.y = y;

}

public int getY()

{

return y;

}

public void setXY(int x,int y)

{

this.x = x;

this.y = y;

}

**3.7.1.2- My Line Class**

class MyLine

{

private MyPoint Begin;

private MyPoint End;

public MyLine(MyPoint Begin, MyPoint End)

{

this.Begin = Begin;

this.End = End;

}

public MyLine()

{

}

public void setBegin(MyPoint Begin)

{

this.Begin = Begin;

}

public MyPoint getBegin()

{

return Begin;

}

public void setEnd(MyPoint End)

{

this.End = End;

}

public MyPoint getEnd()

{

return End;

}

}

**3.7.2- Data Layer**

**3.7.2.1- MyPointDL**

class MyPointDL

{

public static MyPoint Point = new MyPoint();

}

**3.7.2.2- MyLineDL**

class MyLineDL

{

public static MyLine Line = new MyLine();

public static MyLine AddLine(MyLine line)

{

Line = line;

return Line;

}

public static void updateBegin(MyPoint begin)

{

Line.setBegin(begin);

}

public static void updateEnd(MyPoint End)

{

Line.setEnd(End);

}

public static MyLine show\_Line()

{

return Line;

}

public static double lengthOfLine(MyLine L)

{

double length = Math.Pow((L.getBegin().getX() - L.getBegin().getY()),2) + Math.Pow((L.getEnd().getX() - L.getEnd().getY()),2);

length = Math.Sqrt(length);

return length;

}

public static double gradientOfLine(MyLine L)

{

double gradient = (L.getEnd().getY() - L.getBegin().getY()) / (L.getEnd().getX() - L.getBegin().getX());

return gradient;

}

public static double beginPointFromCoordinates(MyLine L)

{

double length = Math.Pow((- L.getBegin().getX()), 2) + Math.Pow((- L.getBegin().getX()), 2);

length = Math.Sqrt(length);

return length;

}

public static double endPointFromCoordinates(MyLine L)

{

double length=Math.Pow((L.getEnd().getX()), 2) + Math.Pow((L.getEnd().getY()), 2);

length = Math.Sqrt(length);

return length;

}

public static void StoreInFile(MyLine L)

{

string path = "lines.txt";

StreamWriter file = new StreamWriter(path);

file.WriteLine(L.getBegin().getX() + "," + L.getBegin().getY() + "," + L.getEnd().getX() + "," + L.getEnd().getY());

file.Flush();

file.Close();

}

public static bool ReadFromFile()

{

string path = "lines.txt";

StreamReader file = new StreamReader(path);

string record="";

if(File.Exists(path))

{

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

{

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

int x1 = int.Parse(splittedRecord[0]);

int y1 = int.Parse(splittedRecord[1]);

int x2 = int.Parse(splittedRecord[2]);

int y2 = int.Parse(splittedRecord[3]);

MyPoint P1 = new MyPoint(x1, y1);

MyPoint P2 = new MyPoint(x2, y2);

MyLine L = new MyLine(P1, P2);

MyLineDL.Line = MyLineDL.AddLine(L);

}

file.Close();

return true;

}

else

{

return false;

}

**3.7.3- User Interface**

**3.7.3.1- MenuUI**

class MenuUI

{

public static void header()

{

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

Console.WriteLine(" ");

Console.WriteLine(" Point Calculation ");

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

Console.WriteLine(" ");

}

static public string menu()

{

header();

Console.WriteLine("[1]-Make a Line");

Console.WriteLine("[2]-Update a Begin Point");

Console.WriteLine("[3]-Update a End Point");

Console.WriteLine("[4]-Show the Update Point");

Console.WriteLine("[5]-Show the End Point");

Console.WriteLine("[6]-Get the Length of Line");

Console.WriteLine("[7]-Get the Gradient of Line");

Console.WriteLine("[8]-Find the Distance of begin point from zero coordinates");

Console.WriteLine("[9]-Find the Distance of End point from zero coordinates");

Console.WriteLine("[10]-Exit");

Console.Write("Your Option......");

string op = Console.ReadLine();

return op;

}

public static void clearScreen()

{

Console.WriteLine("Press any Key to Continue....");

Console.ReadKey();

Console.Clear();

}

}

**3.7.3.2- PointUI**

class PointUI

{

public static MyPoint updateBegin()

{

Console.Write("Enter Begin Point x1 : ");

int x1 = int.Parse(Console.ReadLine());

Console.Write("Enter Begin Point y1 : ");

int y1 = int.Parse(Console.ReadLine());

MyPoint p1 = new MyPoint(x1, y1);

MenuUI.clearScreen();

return p1;

}

public static MyPoint updateEnd()

{

Console.Write("Enter End Point x2 : ");

int x2 = int.Parse(Console.ReadLine());

Console.Write("Enter End Point y2 : ");

int y2 = int.Parse(Console.ReadLine());

MyPoint p1 = new MyPoint(x2, y2);

MenuUI.clearScreen();

return p1;

}

public static void printUpdatePoint()

{

}

public static void printEndPoint()

{

}

}

**3.7.3.3- LineUI**

class LineUI

{

public static MyLine addLine()

{

Console.Write("Enter Begin Point X1 : ");

int x1 = int.Parse(Console.ReadLine());

Console.Write("Enter Begin Point Y1 : ");

int y1 = int.Parse(Console.ReadLine());

MyPoint p1 = new MyPoint(x1, y1);

Console.Write("Enter End Point X2 : ");

int x2 = int.Parse(Console.ReadLine());

Console.Write("Enter End Point Y2 : ");

int y2 = int.Parse(Console.ReadLine());

MyPoint p2 = new MyPoint(x2, y2);

MyLine L = new MyLine(p1, p2);

MenuUI.clearScreen();

return L;

}

public static void print\_update(MyLine Line)

{

Console.WriteLine("Begin-X1\t\tBegin-Y1\t\tEnd-X2\t\tEnd-Y2");

Console.WriteLine(Line.getBegin().getX() + "\t\t\t" + Line.getBegin().getY() + "\t\t\t" + Line.getEnd().getX() + "\t\t" + Line.getEnd().getY());

MenuUI.clearScreen();

}

}

**3.7.4- Main Driver Program**

class Program

{

static void Main(string[] args)

{

if(MyLineDL.ReadFromFile())

{

Console.WriteLine("Read data Succesfully");

}

while (true)

{

string opt = MenuUI.menu();

if (opt == "1")

{

MyLine L = LineUI.addLine();

MyLineDL.Line = MyLineDL.AddLine(L);

}

else if (opt == "2")

{

MyPoint P = PointUI.updateBegin();

MyLineDL.updateBegin(P);

}

else if (opt == "3")

{

MyPoint P = PointUI.updateEnd();

MyLineDL.updateEnd(P);

}

else if (opt == "4")

{

MyLine L= MyLineDL.show\_Line();

LineUI.print\_update(L);

}

else if (opt == "5")

{

MyLine L = MyLineDL.Line;

Console.WriteLine("End Point X2 : " + L.getEnd().getX() );

Console.WriteLine("End Point Y2 :" + L.getEnd().getY());

MenuUI.clearScreen();

}

else if (opt == "6")

{

MyLine L = MyLineDL.Line;

double len = MyLineDL.lengthOfLine(L);

Console.WriteLine("Length of Line Is : " + len);

MenuUI.clearScreen();

}

else if (opt == "7")

{

MyLine L = MyLineDL.Line;

double gra = MyLineDL.gradientOfLine(L);

Console.WriteLine("Gradient of Line Is : " + gra);

MenuUI.clearScreen();

}

else if (opt == "8")

{

MyLine L = MyLineDL.Line;

double beginCoord = MyLineDL.beginPointFromCoordinates(L);

Console.WriteLine("Distance of Begin Point from zero Coordinates : " + beginCoord);

MenuUI.clearScreen();

}

else if (opt == "9")

{

MyLine L = MyLineDL.Line;

double endCoord = MyLineDL.endPointFromCoordinates(L);

Console.WriteLine("Distance of End Point from zero Coordinates : " + endCoord);

MenuUI.clearScreen();

}

else if (opt == "10")

{

MyLine L = MyLineDL.Line;

MyLineDL.StoreInFile(L);

break;

}

else

{

Console.WriteLine("Wrong Input......");

MenuUI.clearScreen();

}

}

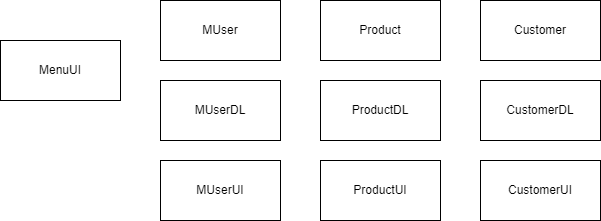
**4. Point of Sales Application**

**4.1-Case Study Scenario:**

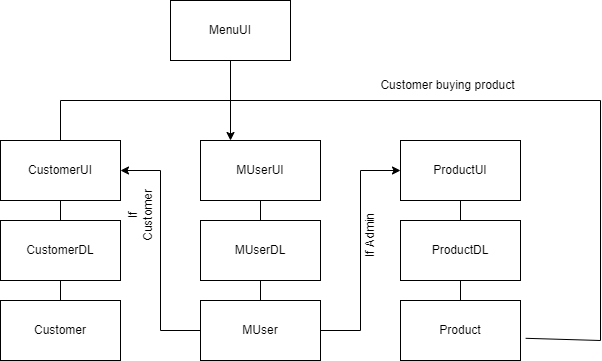
Miss Client wants to develop a software system for her departmental store. She wants this system to have the following functionalities. As an Admin, she can Add Products, View All Products, Find Product with Highest Unit Price, View Sales Tax of All Products, Products to be Ordered. (less than threshold). Following is the information that is required to save for the product. Name of Product. Product Category. Product Price. Available Stock Quantity. Minimum Stock threshold Quantity after which the owner wants to order the product. On All Grocery type of products, the sales tax is 10%, on all fruit types the tax is 5% and if there is any other type the tax is 15%. She also wants that

* The Customers to view all the products
* Customers can buy the products (When a customer buy a product then its
* quantity should decrease from the stock)
* Generate invoice (While calculating the price of the products that the
* customer has bought, sales tax should be applied.)

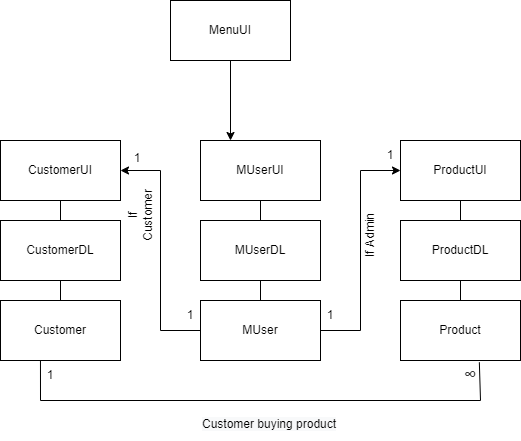
**4.2- Domain Model with only Class Name**

****

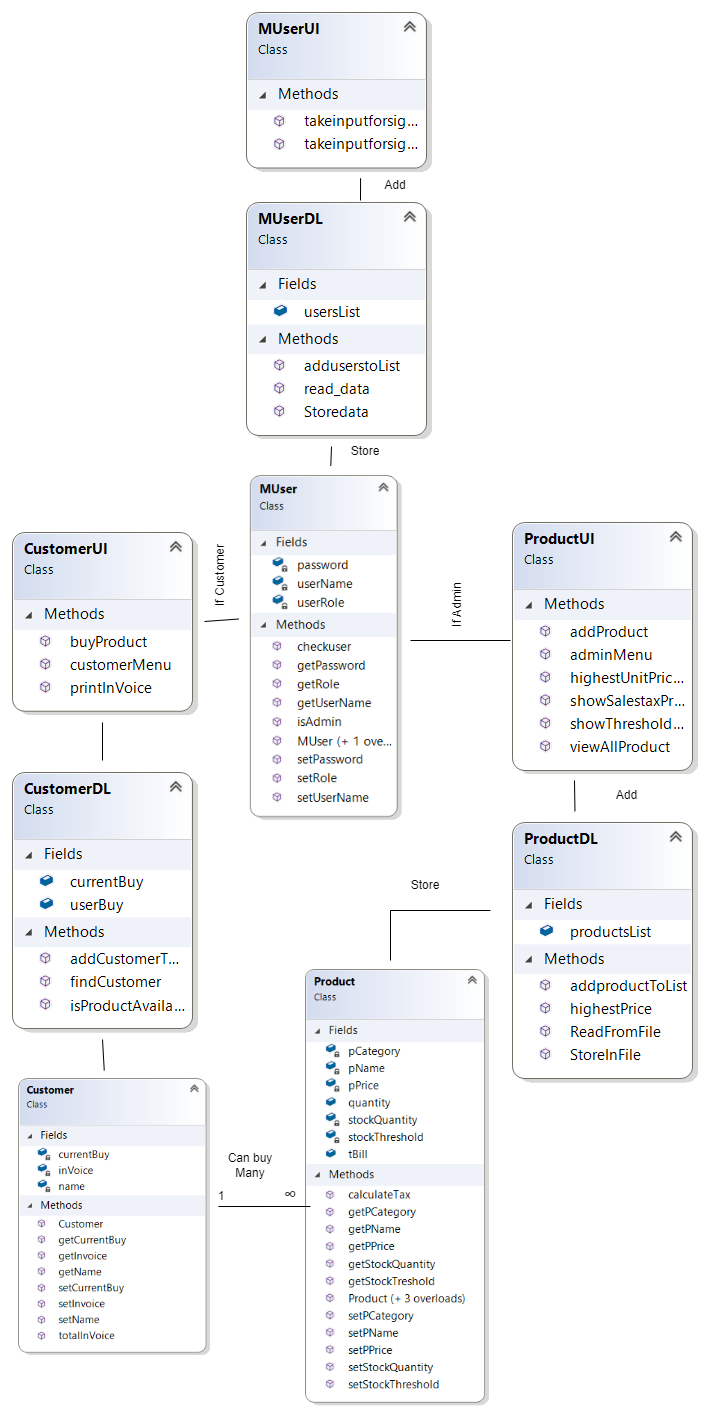
**4.3- Domain Model Relation and Constraints**

****

**4.4- Domain Model with Multiplicity**



**4.5- Class Diagram**

****

4**.7- Complete Code**

**4.7.1-Business Logic**

**4.7.1.1- MUser Class**

class MUser

{

private string userName;

private string password;

private string userRole;

public MUser(string userName, string password, string userRole)

{

this.userName = userName;

this.password = password;

this.userRole = userRole;

}

public MUser(string userName, string password)

{

this.userName = userName;

this.password = password;

this.userRole = "NA";

}

static public MUser checkuser(MUser user)

{

foreach (MUser storedUser in MUserDL.usersList)

{

if (storedUser.userName == user.userName && storedUser.password == user.password)

{

return storedUser;

}

}

return null;

}

static public bool isAdmin(MUser user)

{

if (user.userRole == "Admin" || user.userRole == "admin")

{

return true;

}

return false;

}

public void setUserName(string userName)

{

this.userName = userName;

}

public string getUserName()

{

return userName; ;

}

public void setPassword(string password)

{

this.password = password;

}

public string getPassword()

{

return password;

}

public void setRole(string userRole)

{

this.userRole=userRole;

}

public string getRole()

{

return userRole;

}

**4.7.1.2- Product Class**

class Product

{

private string pName;

private string pCategory;

private int pPrice;

private int stockQuantity;

private int stockThreshold;

public double tBill;

public int quantity;

public Product(string pName,string pCategory,int pPrice,int stockQuantity,int stockThreshold)

{

this.pName = pName;

this.pCategory = pCategory;

this.pPrice = pPrice;

this.stockQuantity = stockQuantity;

this.stockThreshold = stockThreshold;

}

public Product(string pName,string pCategory,int pPrice,int stockQuantity)

{

this.pName = pName;

this.pCategory = pCategory;

this.pPrice = pPrice;

this.stockQuantity = stockQuantity;

}

public Product(string pName, string pCategory, int pPrice, int quantity,double bill)

{

this.pName = pName;

this.pCategory = pCategory;

this.pPrice = pPrice;

this.tBill = bill;

this.quantity = quantity;

}

public Product()

{

}

public float calculateTax()

{

float tax;

if(pCategory=="Grocery")

{

tax = pPrice \* 0.1F;

}

else if (pCategory == "Fruit")

{

tax = pPrice \* 0.05F;

}

else

{

tax = pPrice \* 0.15F;

}

return tax;

}

public void setPName(string pName)

{

this.pName = pName;

}

public string getPName()

{

return pName;

}

public void setPCategory(string pCategory)

{

this.pCategory = pCategory;

}

public string getPCategory()

{

return pCategory;

}

public void setPPrice(int pPrice)

{

this.pPrice = pPrice;

}

public int getPPrice()

{

return pPrice ;

}

public void setStockQuantity(int stockQuantity)

{

this.stockQuantity=stockQuantity;

}

public int getStockQuantity()

{

return stockQuantity;

}

public void setStockThreshold(int stockThreshold)

{

this.stockThreshold = stockThreshold;

}

public int getStockTreshold()

{

return stockThreshold;

}

}

**4.7.1.2- Customer Class**

class Customer

{

private string name;

private double inVoice;

private List<Product> currentBuy = new List<Product>();

public Customer(string name)

{

this.name = name;

}

public double totalInVoice()

{

inVoice = 0;

foreach (Product P in currentBuy)

{

inVoice = inVoice + (((P.calculateTax())\*P.quantity) + (P.getPPrice()\*P.quantity));

}

return inVoice;

}

public void setName(string name)

{

this.name = name;

}

public string getName()

{

return name;

}

public void setInvoice(double inVoice)

{

this.inVoice = inVoice;

}

public double getInvoice()

{

return inVoice;

}

public void setCurrentBuy(List<Product> currentBuy)

{

this.currentBuy = currentBuy;

}

public List<Product> getCurrentBuy()

{

return currentBuy;

}

}

**4.7.2- Data Layar**

**4.7.2.1- MUserDL**

class MUserDL

{

public static List<MUser> usersList = new List<MUser>();

static public void adduserstoList(MUser user)

{

usersList.Add(user);

}

public static void Storedata(MUser u, string path)

{

StreamWriter file = new StreamWriter(path, true);

file.WriteLine(u.getUserName() + "," + u.getPassword() + "," + u.getRole());

file.Flush();

file.Close();

}

public static void read\_data(string path)

{

string record;

StreamReader file = new StreamReader(path);

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

{

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

string name = splittedRecord[0];

string password = splittedRecord[1];

string role = splittedRecord[2];

MUser u = new MUser(name, password, role);

adduserstoList(u);

}

file.Close();

}

}

**4.7.2.2- ProductDL**

class ProductDL

{

public static List<Product> productsList = new List<Product>();

public static void addproductToList(Product P)

{

productsList.Add(P);

}

public static Product highestPrice()

{

int highest = 0;

Product Pro = new Product();

foreach (Product P in productsList)

{

if(P.getPPrice()>highest)

{

highest = P.getPPrice();

Pro = P;

}

}

return Pro;

}

public static void StoreInFile()

{

string path = "products.txt";

StreamWriter file = new StreamWriter(path);

foreach (var P in productsList)

{

file.WriteLine(P.getPName() + "," + P.getPCategory() + "," + P.getPPrice() + "," + P.getStockQuantity() + "," + P.getStockTreshold());

}

file.Flush();

file.Close();

}

public static bool ReadFromFile()

{

string path = "products.txt";

StreamReader file = new StreamReader(path);

string record = "";

if(File.Exists(path))

{

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

{

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

string name = splittedRecord[0];

string category = splittedRecord[1];

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

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

int threshold = int.Parse(splittedRecord[4]);

Product P = new Product(name, category, price, quantity, threshold);

addproductToList(P);

}

file.Close();

return true;

}

return false;

}

}

**4.7.2.3- CustomerDL**

class CustomerDL

{

public static List<Customer> userBuy= new List<Customer>();

public static List<Product> currentBuy = new List<Product>();

static public Product isProductAvailable(string name)

{

foreach (Product P in ProductDL.productsList)

{

if(name==P.getPName())

{

return P;

}

}

return null;

}

public static Customer findCustomer(string userName)

{

foreach (Customer C in userBuy)

{

if(C.getName()==userName)

{

return C;

}

}

return null;

}

public static void addCustomerToList(Customer C)

{

if(!(userBuy.Contains(C)))

{

userBuy.Add(C);

}

}

}

**4.7.3- User Interface**

**4.7.3.1- MenuUI**

class MenuUI

{

public static void header()

{

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

Console.WriteLine(" ");

Console.WriteLine(" Point Of Sale ");

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

Console.WriteLine(" ");

}

public static string menu()

{

Console.Clear();

header();

Console.WriteLine("[1]- Sign In");

Console.WriteLine("[2]- Sign Up");

Console.WriteLine("[3]- Exit");

Console.Write("Your Option.....");

string op = Console.ReadLine();

return op;

}

public static void clearScreen()

{

Console.WriteLine("Press any Key to Continue....");

Console.ReadKey();

Console.Clear();

}

}

**4.7.3.2- MUserUI**

class MUserUI

{

public static MUser takeinputforsignin()

{

Console.Clear();

MenuUI.header();

Console.Write("Enter UserName : ");

string name = Console.ReadLine();

Console.Write("Enter Password : ");

string password = Console.ReadLine();

MUser user = new MUser(name, password);

return user;

}

public static MUser takeinputforsignup()

{

Console.Clear();

MenuUI.header();

Console.Write("Enter UserName : ");

string name = Console.ReadLine();

Console.Write("Enter Password : ");

string password = Console.ReadLine();

Console.Write("Enter Role : ");

string role = Console.ReadLine();

MUser user = new MUser(name, password, role);

Console.WriteLine("User Succesfully Added");

MenuUI.clearScreen();

return user;

}

}

**4.7.3.2- ProductUI**

class ProductUI

{

static public string adminMenu()

{

Console.Clear();

MenuUI.header();

Console.WriteLine("[1]-Add Product");

Console.WriteLine("[2]-View All Products");

Console.WriteLine("[3]-Find Product with highest Unit Price");

Console.WriteLine("[4]-View Sales Tax of All Products");

Console.WriteLine("[5]-Product to Be Ordered");

Console.WriteLine("[6]-Exit");

Console.Write("Your Option......");

string op = Console.ReadLine();

return op;

}

static public Product addProduct()

{

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

string pName = Console.ReadLine();

Console.Write("Enter Product Category : ");

string pCategory = Console.ReadLine();

Console.Write("Enter Product Price: ");

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

Console.Write("Enter Available Stock: ");

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

Console.Write("Enter Minimum Threshold Stock : ");

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

Console.WriteLine("Product added Successfully");

MenuUI.clearScreen();

Product P = new Product(pName, pCategory, pPrice, stockQuantity, stockThreshold);

return P;

}

static public void viewAllProduct()

{

Console.WriteLine("Name\t\tCategory\t\tPrice\t\tStockQuantity\t\tStockThreshold");

foreach (Product P in ProductDL.productsList)

{

Console.WriteLine(P.getPName() + "\t\t" + P.getPCategory() + "\t\t\t" + P.getPPrice() + "\t\t" + P.getStockQuantity() + "\t\t\t" + P.getStockTreshold());

}

MenuUI.clearScreen();

}

static public void highestUnitPriceProduct()

{

Product P = ProductDL.highestPrice();

Console.WriteLine("Name\t\tCategory\t\tPrice\t\tStockQuantity\t\tStockThreshold");

Console.WriteLine(P.getPName() + "\t\t" + P.getPCategory() + "\t\t\t" + P.getPPrice() + "\t\t" + P.getStockQuantity() + "\t\t\t" + P.getStockTreshold());

MenuUI.clearScreen();

}

static public void showSalestaxProduct()

{

Console.WriteLine("Name\t\tCategory\t\tPrice\t\tStockQuantity\t\tStockThreshold\t\tTax");

foreach (Product P in ProductDL.productsList)

{

Console.WriteLine(P.getPName() + "\t\t" + P.getPCategory() + "\t\t\t" + P.getPPrice() + "\t\t" + P.getStockQuantity() + "\t\t\t" + P.getStockTreshold() + "\t\t\t" + P.calculateTax());

}

MenuUI.clearScreen();

}

static public void showThresholdProduct()

{

Console.WriteLine("Name\t\tCategory\t\tPrice\t\tStockQuantity\t\tStockThreshold\t\tTax");

foreach (Product P in ProductDL.productsList)

{

if (P.getStockQuantity() <= P.getStockTreshold())

{

Console.WriteLine(P.getPName() + "\t\t" + P.getPCategory() + "\t\t\t" + P.getPPrice() + "\t\t" + P.getStockQuantity() + "\t\t\t" + P.getStockTreshold());

}

MenuUI.clearScreen();

}

}

}

**4.7.3.2- CustomerUI**

class CustomerUI

{

static public string customerMenu()

{

Console.Clear();

MenuUI.header();

Console.WriteLine("[1]-View All the Products");

Console.WriteLine("[2]-Buy the Product");

Console.WriteLine("[3]-Generate Invoice");

Console.WriteLine("[4]-Exit");

Console.Write("Your Option......");

string op = Console.ReadLine();

return op;

}

static public void buyProduct(string userName)

{

Customer C = CustomerDL.findCustomer(userName);

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

string pname = Console.ReadLine();

Product P = CustomerDL.isProductAvailable(pname);

if(P!=null)

{

Console.Write("Enter Product Quantity : ");

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

if (quantity <= P.getStockQuantity())

{

double bill = P.getPPrice() \* quantity;

P.setStockQuantity(P.getStockQuantity() - quantity);

Product P3 = new Product(P.getPName(), P.getPCategory(), P.getPPrice(),quantity,bill);

CustomerDL.currentBuy.Add(P3);

Console.WriteLine("Products Purchased Successfully");

}

else

{

Console.WriteLine("Only " + P.getStockQuantity() + " Products are availble ");

}

}

else

{

Console.WriteLine("Stock is not Available");

}

MenuUI.clearScreen();

}

public static void printInVoice(string userName)

{

Customer C = CustomerDL.findCustomer(userName);

if(C.getCurrentBuy()!=null)

{

Console.WriteLine("Name\t\tInvoice");

Console.WriteLine(C.getName() + "\t\t" + C.totalInVoice());

MenuUI.clearScreen();

}

}

}

**3.7.4- Main Driver Program**

class Program

{

static void Main(string[] args)

{

string path = "ids.txt";

MUserDL.read\_data(path);

if (ProductDL.ReadFromFile())

{

Console.WriteLine("Read Data Succesfully");

}

while (true)

{

string op = MenuUI.menu();

if (op == "1")

{

MUser user = MUserUI.takeinputforsignin();

MUser U = MUser.checkuser(user);

if (U != null)

{

if (MUser.isAdmin(U))

{

while (true)

{

string opt = ProductUI.adminMenu();

if (opt == "1")

{

Product P = ProductUI.addProduct();

ProductDL.addproductToList(P);

}

else if(opt=="2")

{

ProductUI.viewAllProduct();

}

else if (opt == "3")

{

ProductUI.highestUnitPriceProduct();

}

else if (opt == "4")

{

ProductUI.showSalestaxProduct();

}

else if (opt == "5")

{

ProductUI.showThresholdProduct();

}

else if (opt == "6")

{

break;

}

}

}

else

{

Customer C = new Customer(U.getUserName());

CustomerDL.addCustomerToList(C);

while (true)

{

string opti = CustomerUI.customerMenu();

if (opti == "1")

{

ProductUI.viewAllProduct();

}

else if (opti == "2")

{

CustomerUI.buyProduct(U.getUserName());

}

else if (opti == "3")

{

CustomerUI.printInVoice(U.getUserName());

}

else if (opti == "4")

{

CustomerDL.currentBuy.Clear();

break;

}

}

}

}

else

{

Console.WriteLine("No Such User");

MenuUI.clearScreen();

}

}

else if (op == "2")

{

MUser user = MUserUI.takeinputforsignup();

MUserDL.adduserstoList(user);

MUserDL.Storedata(user, path);

}

else if (op == "3")

{

ProductDL.StoreInFile();

break;

}

else

{

Console.WriteLine("Wrong Input");

MenuUI.clearScreen();

}

}

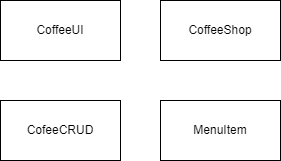
}

**5- Coffee Sop**

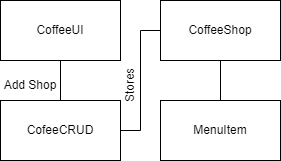
**5.1-Case Study Scenario:**

Create a Class MenuItem, which has three instances 1. name: name of the item. 2. type: whether food or a drink.3. price: price of the item. Write a class called CoffeeShop, which has three instance variables:1. name : a string (basically, of the shop)2. menu : an list of items (of object type), with each item containing the item (name of the item),type (whether food or a drink) and price.3. orders : an empty list of string type.And a parameterized constructor which takes the name of the CoffeeShop as a parameter.and eight methods:1. addMenuItem: adds the menu item in the list of menu. addOrder: adds the name of the item to the end of the orders list if it exists on the menu.Otherwise, return "This item is currently unavailable!"3. fulfillOrder: if the orders list is not empty, return "The {item} is ready!" and make the list empty.if the order list is empty, return "All orders have been fulfilled!". 4. listOrders: returns the list of orders taken, otherwise null. 7.dueAmount: returns the total amount due for the orders taken.6. cheapestItem: returns the name of the cheapest item on the menu.7. drinksOnly: returns only the item names of type drink from the menu.8. foodOnly: returns only the item names of type food from the menu.

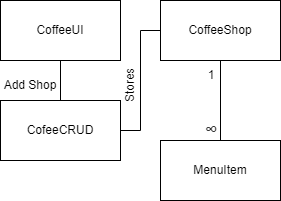
**5.2- Domain Model with only Class Name**

****

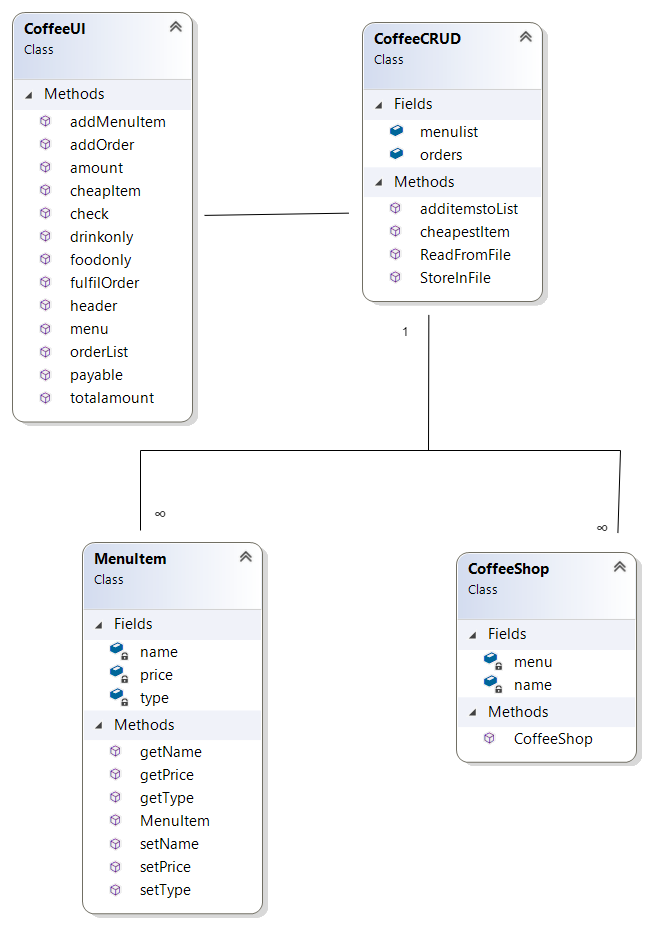
**5.3- Domain Model with Relation and Constraints**

****

**5.4- Domain Model with Multiplicity**

****

**5.5- Class Diagram**

****

**5.7- Complete Code**

**5.7.1-Business Logic**

**5.7.1.1- CoffeeShop Class**

class CoffeeShop

{

private string name;

List<MenuItem> menu = new List<MenuItem>();

public CoffeeShop(string name,List<MenuItem> menu)

{

this.name = name;

this.menu = menu;

}

**5.7.1.2- MenuItem Class**

class MenuItem

{

private string name;

private string type;

private int price;

public MenuItem(string name,string type,int price)

{

this.name = name;

this.type = type;

this.price = price;

}

public void setName(string name)

{

this.name = name;

}

public string getName()

{

return name;

}

public void setType(string type)

{

this.type = type;

}

public string getType()

{

return type;

}

public void setPrice(int price)

{

this.price=price;

}

public int getPrice()

{

return price;}

**5.7.2-Data Layer**

**5.7.2.1- CofeeCRUD**

class CoffeeCRUD

{

public static List<MenuItem> menulist = new List<MenuItem>();

public static List<string> orders = new List<string>();

static public void additemstoList(MenuItem item)

{

menulist.Add(item);

}

static public string cheapestItem()

{

int cheapest= menulist[0].getPrice();

string cheapestItem = menulist[0].getName();

for (int i = 0; i < menulist.Count - 1; i++)

{

if(menulist[i+1].getPrice()<cheapest)

{

cheapest = menulist[i + 1].getPrice();

cheapestItem = menulist[i + 1].getName();

}

}

return cheapestItem;

}

public static void StoreInFile(MenuItem M)

{

string path = "menuItems.txt";

StreamWriter file = new StreamWriter(path,true);

file.WriteLine(M.getName() + "," + M.getType() + "," + M.getPrice());

file.Flush();

file.Close();

}

public static bool ReadFromFile()

{

string path = "menuItems.txt";

StreamReader file = new StreamReader(path);

string record = "";

if(File.Exists(path))

{

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

{

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

string Name = splittedRecord[0];

string type = splittedRecord[1];

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

MenuItem M = new MenuItem(Name, type, price);

CoffeeCRUD.additemstoList(M);

}

file.Close();

return true;

}

return false;

**5.7.3- User Interface**

**5.7.3.1- CofeeUI**

static public void header()

{

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

Console.WriteLine(" ");

Console.WriteLine(" Welcome to the Tesha's Coffee ");

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

Console.WriteLine(" ");

}

static public string menu()

{

header();

Console.WriteLine("[1]- Add a Menu Item");

Console.WriteLine("[2]- View the Cheapest Item in the Menu");

Console.WriteLine("[3]- View the Drink Menu");

Console.WriteLine("[4]- View the Food Menu");

Console.WriteLine("[5]- Add Order");

Console.WriteLine("[6]- Fulfil the Order");

Console.WriteLine("[7]- View the Order List");

Console.WriteLine("[8]- Total Payable Amount");

Console.WriteLine("[9]- Exit");

Console.Write("Your Option....");

string o = Console.ReadLine();

return o;

}

static public MenuItem addMenuItem()

{

Console.Write("Enter Name : ");

string name = Console.ReadLine();

Console.Write("Enter Type(Drink or Food) : ");

string type = Console.ReadLine();

Console.Write("Enter price : ");

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

MenuItem I = new MenuItem(name, type, price);

Console.Clear();

return I;

}

public static void cheapItem()

{

if (CoffeeCRUD.menulist.Count > 0)

{

string cheapItem = CoffeeCRUD.cheapestItem();

Console.WriteLine("Cheapest Item Is : " + cheapItem);

Console.ReadKey();

}

else

{

Console.WriteLine("No Item Availabe");

Console.ReadKey();

}

Console.Clear();

}

public static void drinkonly()

{

if (CoffeeCRUD.menulist.Count > 0)

{

Console.WriteLine("Drinks Items are : ");

foreach (MenuItem I in CoffeeCRUD.menulist)

{

if (I.getType() == "Drink")

{

Console.Write(I.getName() + " , ");

}

}

}

else

{

Console.WriteLine("No Drink Item Availabe");

}

Console.ReadKey();

Console.Clear();

}

public static void foodonly()

{

if (CoffeeCRUD.menulist.Count > 0)

{

Console.WriteLine("Food Items are : ");

foreach (MenuItem I in CoffeeCRUD.menulist)

{

if (I.getType() == "Food")

{

Console.Write(I.getName() + " , ");

}

}

}

else

{

Console.WriteLine("No Food Item Availabe");

}

Console.ReadKey();

Console.Clear();

}

public static void addOrder()

{

if (CoffeeCRUD.menulist.Count > 0)

{

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

string order = Console.ReadLine();

check(order);

}

else

{

Console.WriteLine("No Such Item Availabe");

}

Console.ReadKey();

Console.Clear();

}

public static void check(string order)

{

foreach(MenuItem I in CoffeeCRUD.menulist)

{

if(order==I.getName())

{

Console.WriteLine("Order Placed Succesfully");

CoffeeCRUD.orders.Add(order);

}

}

}

public static void orderList()

{

if (CoffeeCRUD.orders.Count > 0)

{

foreach (var o in CoffeeCRUD.orders)

{

Console.Write(o + " , ");

}

}

else

{

Console.WriteLine("No Pending Order");

}

Console.ReadKey();

Console.Clear();

}

public static int totalamount()

{

int sum = 0;

bool flag = false;

if (CoffeeCRUD.orders.Count > 0)

{

foreach (var o in CoffeeCRUD.orders)

{

flag = true;

sum = sum + amount(o);

}

}

return sum;

}

static public int amount(string order)

{

foreach (MenuItem M in CoffeeCRUD.menulist)

{

if (order == M.getName())

{

return M.getPrice();

}

}

return 0;

}

public static void fulfilOrder()

{

if (CoffeeCRUD.orders.Count > 0)

{

foreach (var o in CoffeeCRUD.orders)

{

Console.WriteLine("The " + o + " Is Ready");

}

CoffeeCRUD.orders.RemoveAt(0);

}

else

{

Console.WriteLine("All Orders have been Fulfiled");

}

Console.ReadKey();

Console.Clear();

}

public static void payable()

{

int total = CoffeeUI.totalamount();

Console.WriteLine("Total Payable Amount Is : " + total);

Console.ReadKey();

Console.Clear();

}

**5.7.4- Main Driver Program**

namespace Coffee

{

class Program

{

static void Main(string[] args)

{

if(CoffeeCRUD.ReadFromFile())

{

Console.WriteLine("Data Read Succesfully");

}

while (true)

{

string op = CoffeeUI.menu();

if (op == "1")

{

MenuItem M = CoffeeUI.addMenuItem();

CoffeeCRUD.additemstoList(M);

CoffeeCRUD.StoreInFile(M);

}

else if (op == "2")

{

CoffeeUI.cheapItem();

}

else if (op == "3")

{

CoffeeUI.drinkonly();

}

else if (op == "4")

{

CoffeeUI.foodonly();

}

else if (op == "5")

{

CoffeeUI.addOrder();

}

else if (op == "6")

{

CoffeeUI.fulfilOrder();

}

else if (op == "7")

{

CoffeeUI.orderList();

}

else if (op == "8")

{

CoffeeUI.payable();

}

else if (op == "9")

{

break;

}

else

{

Console.WriteLine("Wrong Input.....");

Console.ReadLine();

Console.Clear();

}

}

}

**6- GameObject**

**6.1-Case Study Scenario**

The requirement of this program is to create a full-fledged GameObject with the concepts of OOP. There are several options in this program, like moving game object from left to right,right to left,patrolling, diagonally and in the parabola direction etc. By Constructing three classes.

**6.2-Domain Model with only Class Name**

**withclassname.png**

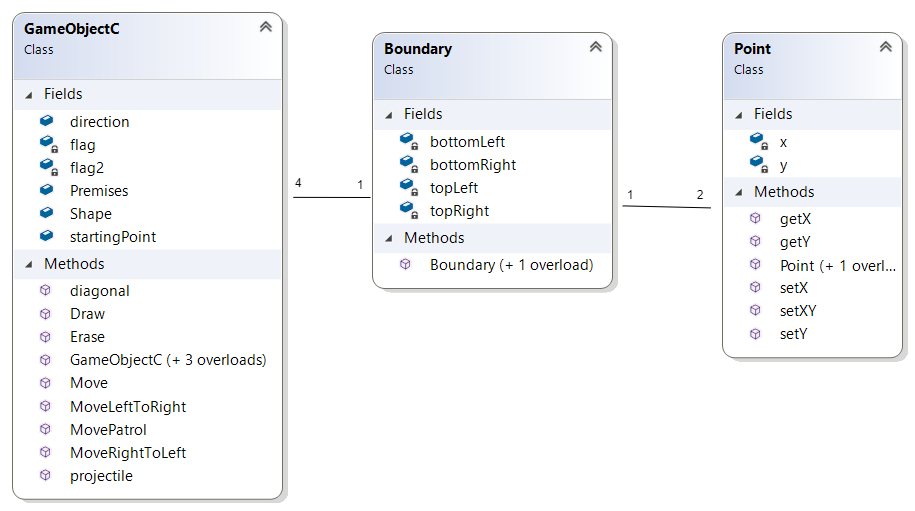
**6.3-Domain Model with Relation and Constraint**

**withclassname (1).png**

**6.4-Domain Model with Multiplicity**

withclassname (3).png

**6.5- Class Diagram**



6.7- Complete Code

6.7.1-Business Logic

6.7.1.1- Point

class Point

{

private int x;

private int y;

public Point()

{

}

public Point(int x,int y)

{

this.x = x;

this.y = y;

}

public int getX()

{

return x;

}

public void setX(int x)

{

this.x = x;

}

public int getY()

{

return y;

}

public void setY(int y)

{

this.y=y;

}

public void setXY(int x,int y)

{

this.x = x;

this.y = y;

}

}

6.7.1.2- Boundary

class Boundary

{

private Point topLeft;

private Point topRight;

private Point bottomLeft;

private Point bottomRight;

public Boundary()

{

this.topLeft.setX(0);

this.topLeft.setY(0);

topRight.setX(0);

topRight.setY(90) ;

bottomRight.setX(90);

bottomRight.setY(90);

bottomLeft.setX(90);

bottomLeft.setY(0);

}

public Boundary(Point topLeft,Point topRight,Point bottomRight,Point bottomLeft)

{

this.topLeft = topLeft;

this.topRight = topRight;

this.bottomLeft = bottomLeft;

this.bottomRight = bottomRight;

}

}

6.7.1.3- GameObjectC

class GameObjectC

{

public char[,] Shape = new char[7, 13];

public Point startingPoint;

public Boundary Premises;

public string direction = "LeftToRight";

bool flag = true;

bool flag2 = false;

public GameObjectC(char[,] Shape, Point startingPoint, Boundary Premises)

{

this.startingPoint = new Point();

direction = "LeftToRight";

}

public GameObjectC()

{

char[,] shape = new char[7, 13];

this.startingPoint = new Point();

this.Premises = new Boundary();

direction = "LeftToRight";

}

public GameObjectC(char[,] Shape, Point startingPoint)

{

this.Shape = Shape;

this.startingPoint = startingPoint;

this.Premises = new Boundary();

direction = "LeftToRight";

}

public GameObjectC(char[,] Shape, Point startingPoint, Boundary Premises, string direction)

{

this.Shape = Shape;

this.startingPoint = startingPoint;

this.Premises = Premises;

this.direction = direction;

}

public void Move()

{

if (direction == "LeftToRight")

{

MoveLeftToRight();

}

if (direction == "RightToLeft")

{

MoveRightToLeft();

}

if (direction == "Patrolling")

{

MovePatrol();

}

if (direction == "Diagonal")

{

diagonal();

}

if (direction == "Projectile")

{

projectile();

}

}

public void MoveLeftToRight()

{

if (startingPoint.getY() < 90)

{

if (startingPoint.getX() < 90)

{

startingPoint.getX();

startingPoint.setY(startingPoint.getY() + 1);

}

}

}

public void MoveRightToLeft()

{

if (startingPoint.getY() > 2)

{

if (startingPoint.getX() > 1)

{

startingPoint.getX();

startingPoint.setY(startingPoint.getY() - 1);

}

}

}

public void MovePatrol()

{

if (flag == true)

{

if (startingPoint.getY() < 90)

{

startingPoint.getX();

startingPoint.setY(startingPoint.getY() + 1);

}

if (startingPoint.getY() == 90)

{

flag = false;

}

}

else if (flag == false)

{

if (startingPoint.getY() > 2)

{

startingPoint.getX();

startingPoint.setY(startingPoint.getY() - 1);

}

else if (startingPoint.getY() == 2)

{

flag = true;

}

}

}

public void diagonal()

{

if (startingPoint.getX() <= 90 && startingPoint.getY() <= 90)

{

startingPoint.setX(startingPoint.getX() + 1);

startingPoint.setY(startingPoint.getY() + 3);

}

}

public void Erase()

{

for (int i = startingPoint.getX(); i < startingPoint.getX() + 8; i++)

{

for (int j = startingPoint.getY(); j < startingPoint.getY() + 14; j++)

{

Console.SetCursorPosition(j, i);

Console.Write(" ");

}

}

}

public void projectile()

{

if (startingPoint.getX() <= 90 && startingPoint.getY() <= 90)

{

if (flag2 == false)

{

if (startingPoint.getY() < 40)

{

startingPoint.setX(startingPoint.getX() - 1);

startingPoint.setY(startingPoint.getY() + 1);

}

else if (startingPoint.getY() == 40)

{

flag2 = true;

}

}

else if (flag2 == true)

{

if (startingPoint.getX() < 89)

{

if (startingPoint.getY() > 2)

{

startingPoint.setX(startingPoint.getX() + 1);

startingPoint.setY(startingPoint.getY() + 2);

}

}

}

}

}

public void Draw()

{

for (int i = 0; i <7; i++)

{

for (int j = 0; j < 13; j++)

{

Console.SetCursorPosition(startingPoint.getY() + j, startingPoint.getX() + i);

Console.Write(Shape[i, j]);

}

}

}

6.7.2- Main Driver Program

class Program

{

static void Main(string[] args)

{

char[,] A = new char[7,13] { { ' ', ' ', ' ', ' ', ' ', ' ', '\*', ' ', ' ', ' ', ' ', ' ', ' ' }, { ' ', ' ', ' ', ' ', ' ', '\*', ' ', '\*', ' ', ' ', ' ', ' ', ' ' }, { ' ', ' ', ' ', ' ', '\*', ' ', ' ', ' ', '\*', ' ', ' ', ' ', ' ' },

{ ' ', ' ', ' ', '\*', '\*', '\*', '\*', '\*', '\*', '\*', ' ', ' ', ' ' },{ ' ', ' ', '\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*', ' ', ' ' }, { ' ', '\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*', ' ' },{ '\*', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\*' } };

Boundary b = new Boundary(new Point(0, 0), new Point(0, 90), new Point(90, 0), new Point(90, 90));

GameObjectC G1 = new GameObjectC(A, new Point(1, 1), b, "LeftToRight");

GameObjectC G2 = new GameObjectC(A, new Point(20, 90), b, "RightToLeft");

GameObjectC G3 = new GameObjectC(A, new Point(50, 1), b, "Projectile");

GameObjectC G4 = new GameObjectC(A, new Point(1, 1), b, "Diagonal");

GameObjectC G5 = new GameObjectC(A, new Point(10, 1), b, "Patrolling");

List<GameObjectC> Objects = new List<GameObjectC>();

Objects.Add(G1);

Objects.Add(G2);

Objects.Add(G3);

Objects.Add(G4);

Objects.Add(G5);

while (true)

{

Thread.Sleep(100);

foreach (GameObjectC G in Objects)

{

G.Erase();

G.Move();

G.Draw();

}