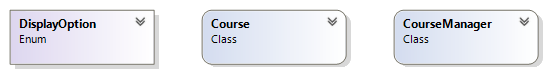
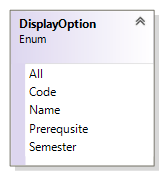
### Read the class and member specification before coding. The diagrams should be a reminder/verifier for the type.

#### The data for this project is in the file “course.txt” on the p drive

This exercise you will be practicing your file reading skills. There are 3 main types in this exercise: an enum and two classes.

# The DisplayOption enum

This enum is used to specify the filter in displaying the course. The method **Display(DisplayOption option, string toMatch="")** uses this enum.

#### Description of class members

There are five members as shown in the diagram above

# The Course class

We are going to model a course type. There are 7 members as shown in the class diagram below.

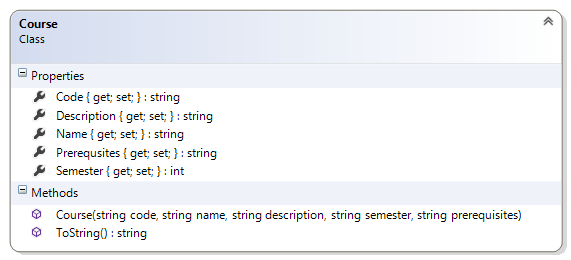
#### Description of class members

##### Fields:

There are no fields in this type

##### Properties:

**Code** – this string represents course code of this object such as "COMP100". This is an auto-implemented property, the getter is public and the setter is private.

**Description** – this string represents description of this course object. This is an auto-implemented property, the getter is public and the setter is private.

**Name** – this string represents course name of this object such as "Programing I". This is an auto-implemented property, the getter is public and the setter is private.

**Prerequsite** – this string represents prerequisites of this course object. All the prerequisites are contained in a single string. This is an auto-implemented property, the getter is public and the setter is private.

**Semester** – this int represents the semester that this course is being offered. This is an auto-implemented property, the getter is public and the setter is private.

##### Constructor:

**Course(string code, string name, string description, string semester, string prerequisites)** – This is constructor assigns the arguments to the appropriate properties. The fourth parameter has to be converted to an int

##### Methods

**ToString()** – This is a public method overrides the corresponding method in the object class to return a stringify form of the object. You decide on the format of the returned string the only constraint is that the length of the description should be capped at 60 letters. [Use the **Substring()** method in the string class]

# The CourseManager class

This class will manage the course objects; it provides methods to load the course information from a local text file and to display selected courses. This class is declared static so all the members are also static

#### Description of class members

##### Fields:

**courses** – this static field is a list of course. This is initialized by the **LoadCourse()** method.

##### Properties:

There are no properties in this type.

##### Constructor:

There is no user-defined constructor for this class

##### Methods

**public static void Display(DisplayOption option, string toMatch="")** – This is a public method displays all or some of the objects in the list courses. It takes two arguments: the first argument stipulates what field the filter is on and the second argument is the value. For most of the options you check for exact match, however if the option is DisplayOption.Prerequsite then you will check if the property contains the second argument. [Use the **Contains()** of the string class]

**public static void LoadCourses(string filename)** – This is a public method reads the text in the file and creates courses and adds it to the field courses. You must examine the text file to figure out the order of the data and the format of the records. You should process all the data in a single method invocation.

Steps in reading a text file

* Add the necessary using statement
* Declare and initialize the reader
* Use ReadLine() to read one line at a time
* Close the reader

### Test Harness

Insert the following code statements in your Program.cs file:

CourseManager.LoadCourses("course.txt");

Console.WriteLine("\nAll courses");

CourseManager.Display(DisplayOption.All);

string toSearch = "COMP100";

Console.WriteLine("\nWith prerequsite " + toSearch);

CourseManager.Display(DisplayOption.Prerequsite, toSearch);

Console.WriteLine("\nWith title " + toSearch);

CourseManager.Display(DisplayOption.Code, toSearch);

toSearch = "Programing I";

Console.WriteLine("\nWith name " + toSearch);

CourseManager.Display(DisplayOption.Name, toSearch);

toSearch = "2";

Console.WriteLine("\nIn semester " + toSearch);

CourseManager.Display(DisplayOption.Semester, toSearch);