Lab 2
Object Oriented Programming in C#

1. Vector

Objective:

- Create a derived class.
- Add methods to a derived class.
- Redefine methods in a derived class

Define a class named **Vectors2D** characterized by the abscissa X and the ordinate Y, as well as an attribute which provides information on the number of vectors created during the execution of the program.

Define the constructors (by default, parametrized and by copy), and the accessors to the various attributes.

Define the ToString and Equals methods, the first returns a string representing the abscissa and the ordinate of the vector as follows: X = 1.5 - Y = 2, the second returns True when the abscissa and the ordinate of the two vectors are equal.

Define a method that returns the norm of a vector, this method can be redefined in derived classes.

Define a **3DVectors** class derived from the **Vector2D** class which contains, in addition to the properties of the base class, the Z property symbolizing the third dimension.

Define the constructors (by default, parametrized and by copy) of this class, as well as the property accessors.

Redefine the ToString and Equals methods to integrate the third property.

Redefine the Norm method so that it returns the norm of a vector in space.

Define a test program allowing the creation of two Vectors2D type objects and two others of3D Vectors type. Display information for all objects, and test the Equals and Norm methods. Display the number of objects created.

2. Person

Objective:

- The concept of polymorphism.

Consider the following classes:

The class **Person** which has three private fields, first name, last name and date of birth. This class has a constructor to allow data to be initialized. It also has a polymorphic method named **Display** to display information for each person.

The class **Employe** which derives from the class **Person**, with in addition a Salary field accompanied by its property, a constructor and the redefinition of the Display method.

The class **Manager** which derives from the class **Employee**, with in addition a **Department** field accompanied by its property, a constructor and the redefinition of the Display method.

The class **Director** which derives from the **Manager** class, with in addition a **Company** field accompanied by its property, a constructor and the redefinition of the Display method.

Work to be done:

1. Write the class **Person**, **Employee**, **Manager** and **Director**.
2.create a test program that has a list of eight people with five employees, two managers and a director (8 references of the class **Person** in which to store 5 instances of the class **Employee**, 2 of the class **Manager** and 1 of the class **Director**).
Display all the elements of the list.
.

3. Employee

Objective:

- Create an abstract class.
- Derive an abstract class
- Implement an abstract method

The abstract class **Employee** is characterized by the following attributes: Matricule, First name, Last name and Date of birth.

The class **Employee** must have the following methods:

1. The constructors

2. Properties for the different attributes
3. To string method
4. An abstract method GetSalary.

A worker is an employee who is characterized by his date of entry into the company.
All workers have a common value called SMIG = 2500 DH
The worker has a monthly salary which is:
Salary = SMIG + (Seniority in years) * 100.
In addition, the salary must not exceed SMIG * 2.

An executive(cadre) is an employee who is characterized by the index.

The executive has a salary which depends on his index:
1: monthly salary 13,000 DH
2: monthly salary 15,000 DH
3: monthly salary 17,000 DH
4: monthly salary 20,000 DH

A boss is an employee who is characterized by a turnover and a percentage. The turnover is common between the bosses.

The boss has an annual salary which is equal to x% of the turnover:
Salary = CA(turnover) * percentage / 100

Work to be done:

1. Create the abstract class **Employee**.
2. Create the class **Worker**, the **Executive** class and the **Boss** class which inherit from the **Employee** class, and provide the Constructors and the ToString method of each of the 3 classes.
3. Implement the GetSalary () method which calculates the salary for each of the classes
4. Define the test program.

4.Notification System

Objective:

- Create a delegate

We want to design a notification system to inform the **registrar responsible** and **prof**, the new modified average obtained by adding each new grade of the students

Work to be done:

Create the class **Student** with the properties as following: Full name, The average of grades and the number of courses

Create the constructors for the class Student.

Declare a delegate which returns void named **AvgChanged** and one double parameter named average.

Create an instance of delegate .

Create the method **RecordGrade** which calculate the new average as below:

Average of grades = (Average of grade * number of courses + new grade) / (number of courses + 1);

Invoke the method RecordGrade through the delegate.

Create the class **Registrar** with the method void named **report** with the same signature as delegate. It will show the message as below:

My name is …….. as a registrar, I received the new average of Student named…… successfully.

Create the class **Prof** with the method void named **report** with the same signature as delegate. It will show the message as below:

My name is …….. as a prof, I received the new average of Student named …… successfully.

In class main create the objects of class Student, Registrar and Prof

Add the methods to delegate object by “+=”

The console should take the new grade and show the notifications by message passing between all classes at the same time.

Now, remove one of the methods from the delegate by “-=” m) This Time, instead of “+=” or”-=” use only “=” for one of the methods.

5.Managing a stock

Objective:

- Handle the List type collections.

Develop an application for managing a stock. An item is characterized by its reference number, name, selling price and a quantity in stock. The stock is represented by a collection of items.

Work to be done:

Create the article class containing the following elements:
• The attributes / properties.
• The constructors.
• ToString () method.

In the class **Program** create:
The stock in the form of a collection of items of your choice. A menu with the following functions:
1.Search for an article by reference.
2.Add an item to the stock by checking the uniqueness of the reference.
3.Delete an article by reference.
4.Modify an article by reference.
5.Search for an article by name.
6.Search for an item by sale price range.
7.View all articles.
8.Quit