Session 8&9:
Abstract classes, Interfaces, Properties, Indexers

A. OBJECTIVES:

- Abstract classes
- Interfaces
- Properties
- Indexers

B. EXERCISES

1. Create interfaces and implement these interfaces

Abstract classes, Interfaces, Properties, Indexers

```
interface Animal
{
    void eat();
    void run();
}
interface Bird
{
    void eat();
    void fly();
}
class Bear : Animal
{
    string name;
    public Bear(string name)
    {
        this.name = name;
        Console.WriteLine();
        Console.WriteLine("Bear named {0} was born", this.name);
    }
    public void eat()
    {
        Console.WriteLine("Bear eat fish");
    }
    public void run()
    {
        Console.WriteLine("{0} is running", name);
    }
}
class Eagle : Bird
{
    string name;
    public Eagle(string name)
    {
        this.name = name;
        Console.WriteLine();
        Console.WriteLine("Egale named {0} was born", this.name);
    }
    public void eat()
    {
        Console.WriteLine("Eagle eat dove");
    }
```

```
        public void fly()
        {
            Console.WriteLine("{0} is flying", name);
        }
    }
    class Bat :Animal, Bird
    {
        string name;
        public Bat(string name)
        {
            this.name = name;
            Console.WriteLine();
            Console.WriteLine("Bat named {0} was born", this.name);
        }
        void Animal.eat()
        {
            Console.WriteLine("Bat eat mosquito");
        }
        void Bird.eat()
        {
            Console.WriteLine("Bat eat fruit");
        }
        public void run()
        {
            Console.WriteLine("{0} can't run", name);
        }
        public void fly()
        {
            Console.WriteLine("{0} is flying", name);
        }
    }
    class ImplementInterfacesDemo
    {
        public void run()
        {
            Bat Sarah = new Bat("Sarah");
            Sarah.run();
            Sarah.fly();
            ((Animal)Sarah).eat();
            ((Bird)Sarah).eat();

            Bird Jack = new Eagle("Jack");
            Jack.eat();
            Jack.fly();

            Animal Joe = new Bear("Joe");
            Joe.eat();
            Joe.run();

        }
    }
```

2. Using properties and indexers

```
class MyList
{
    int[] a;

    private int _capacity;
    public int Capacity
    {
        get { return _capacity; }
        set
        {
            if (value > 0)
            {
                _capacity = value;
            }
            else
            {
                _capacity = 50;
                Console.WriteLine("The capacity of the list is set to 50");
            }
            a = new int[_capacity];
            _count = 0;
        }
    }

    private int _count;
    public int Count
    {
        get { return _count; }
    }

    public MyList(int n)
    {
        Capacity = n;
    }

    public void Add(int value)
    {
        a[_count] = value;
        _count++;
    }

    public int this[int index]
    {
        //set { } //set accessor is disabled
        get { return a[index]; }
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        MyList a = new MyList(20);
        a.Add(10);
        a.Add(2);
        a.Add(-9);
        a.Add(1);
        a.Add(7);
        a.Add(15);

        Console.WriteLine("Array's members:");
        for (int i = 0; i < a.Count; i++)
        {
            Console.Write("{0,5}", a[i]);
        }

        Console.ReadLine();
    }
}
```

3. Using properties and indexers

3.1. Create Worker class has properties:

- Code: id of worker
- Name: fullname of worker
- Level: level of worker
- and TimeKeeping indexer: timekeeping of workers in a week. Everyday is checked the total hours that worker woking.

3.2. Write display() method to to print the informations of worker on screen

3.3. Write checkTime() method has 2 param (the day be checked from Monday to Sunday and the hours that worker work in that day) to check for worker. Worker is checked many times in a day. The time is cheked is: The time in system + the hours that timekeeper input.

3.4. Write displayTimeKeeping() method to show the working hours of worker from Monday to Sunday.

3.5. Write getHours() method to return the total hours that worker work in a week.

3.6. Write getSalary() method to return the salary of worker in a week.

- A normal day (from Monday to Friday): Standard time is 8 hours. The more time is OverShift time
- Saturday and Sunday: time is overshift time

Salary = Standard Time * 15000 + Overshift *20000

3.7. Create WorkerDemo class. Write Main() method in WorkerDemo class can do:

- Create instance of Worker.
- Input informations of worker (code, name, level)
- Perform display() method to print the informations of worker on screen

- Input in turn the working time of worker: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
- Use checkTime() method to add 2 hours to Friday
- Dislay the salary that worker can received in a week

4. Extra exercises 1

Writing a program to store information of a pupil and display the details of this pupil.

Hint: create a class named PupilInfo includes some details showed as below:

1. This class has 5 properties:
    a. PupilCode: this is a string use for storing pupil code.
    b. FullName: this is a string use for storing pupil full name.
    c. Math: this is a real number from 0 to 10. When user update the value of this field, this value must be valid and the value of "rate" must be updated too.
    d. Literature: this is a real number from 0 to 10. When user update the value of this field, this value must be valid and the value of "rate" must be updated too.
    e. Rate: this is a read-only field and accept only one of 5 string values includes:
        - Fail: mark from 0 to under 4.
        - Pass: mark from 4 to under 6.
        - Distinction: mark from 6 to under 8.
        - Excellent: mark from 8 to under 9.
        - Outstanding: mark from 9 to 10.
2. This class has 2 constructor:
    a. The first constructor has two parameters: PupilCode and FullName.
    b. The second constructor has four parameters: PupilCode, FullName, Math, and Literature.
3. This class has a method named ToString that returns a string describe information of pupil.

5. Extra exercises 2

a. Create a class named Food that uses properties to store information includes food name and price.

b. Create a class named FoodList that store a list of foods, this class have methods to add new food and uses indexer to find a food price by its name.

c. Create a program to demo how to use FoodList class.

Abstract classes, Interfaces, Properties, Indexers