INHERITANCE IN C#

MEANING, BENEFITS AND APPLICATIONS

Inheritance in C# can be defined as the ability to create classes which will inherit attributes and behaviors from an existing class. To be more clear, when a class is created and after its name it is putted a colon [:] followed by the name of other class, it will inherit the attributes and behaviors of the class putted after the colon.

To be clearer yet, I will give an example of the code I have wrote this week:

```
public abstract class Exercise
{
    public string _exerciseName;
    public string _exerciseResume;
    protected int _initialTime;
    protected int _finalTime;

    public string ExerciseName
    {
        get { return _exerciseName; }
        protected set { _exerciseName = value; }
    }

    public string ExerciseResume
    {
        get { return _exerciseResume; }
        protected set { _exerciseResume = value; }
    }

    protected int InitialTime
    {
        get { return _initialTime; }
        set { _initialTime = value; }
    }

    protected int FinalTime
    {
        get { return _finalTime; }
        set { _finalTime = value; }
    }

    protected void InitialNotice()
    {
```

```
        Console.Clear();
        Console.WriteLine($"Get ready for the {_exerciseName}
exercise!");
        Console.WriteLine($"{_exerciseResume}\n");
    }

    protected void FinalNotice()
    {
        Console.WriteLine($"\nGreat job! You completed the
{_exerciseName} exercise!");
    }

    protected void SetExerciseTime()
    {
        Console.WriteLine("How many seconds would you like to spend on
this exercise?");
        int time = int.Parse(Console.ReadLine());
        _initialTime = 0;
        _finalTime = time;
    }

    protected void SetClock()
    {
        Console.WriteLine("\nStarting in...");
        for (int counter = 3; counter >= 1; counter--)
        {
            Console.WriteLine($"{counter}...");
            Thread.Sleep(1000);
        }
        Console.WriteLine("Go!");
    }

    protected void UpdateClock()
    {
        _initialTime += 10;
        Console.WriteLine($"Time left: {_finalTime - _initialTime}
seconds");
        Thread.Sleep(2000);
    }
}
```

```
public class Breath : Exercise
{
    private string _breatheIn;
    private string _breatheOut;

    public Breath()
    {
        _exerciseName = "Breathing";
        _exerciseResume = "This activity will help you to control your
chakra by guiding you through breathing in and out slowly. Clear your
mind and focus on your breathing";
        _breatheIn = "Breathe in...";
        _breatheOut = "Now breathe out...";
    }

    public void DoBreathing()
    {
        InitialNotice();
        SetExerciseTime();
        SetClock();

        GetReady();

        while (_initialTime < _finalTime)
        {
            Console.CursorVisible = false;

            Console.WriteLine($"{_breatheIn}");
            Thread.Sleep(4000);

            Console.WriteLine($"{_breatheOut}");
            Thread.Sleep(4000);

            Console.WriteLine();
            UpdateClock();
        }

        FinalNotice();
    }

    private void GetReady()
    {
        Console.WriteLine("Get ready to start breathing...");
        Thread.Sleep(2000);
    }
}
```

As it can be seen above, there are two classes: exercise and breath. The class exercise is responsible for displaying the countdown and the initial message when starting an exercise and displaying the final message after the end of the exercise. And to reuse it in other classes and economize time, it's very useful to use inheritance. That's why there is a colon after the name of the class breath.

Also, there are so many benefits when using inheritance:

- Code reuse, which reduce time and effort in coding and testing;
- Enables the construction of a hierarchy of related class that can be reused;
- Helps to improve modularity and performs divisions of large pieces of code into smaller parts;
- And form ways to achieve polymorphism, which helps an object to represent more than one type;

Is it possible to manage multiple classes using inheritance?

The answer for it is: Y-E-S

It is very possible to use inheritance with multiple classes, doesn't matter how many classes is been used, with inheritance in C#, it's possible to fix the problems and update the code faster than if it was without inheritance.