Week 4 - Workshop

Class Object Method Constructor

Task 1: Class, Object & Fields [10 Minutes]

1. Inside a new class named ***Student***, perform following tasks:
   - Create **3 instance fields** for storing any of the student details.
   - Create **1 static field**
2. In ***Program.cs*** class, create **two objects** of ***Student***.
3. **Assign different values** for instance fields for both object.
4. Display the field values of both objects and **print the static field**.

Task 2: Methods & Return Types [15 Minutes]

1. Create a new class ***Calculator*** and perform following tasks:
   - Create a void method named ***PrintWelcome()*** that prints ***"Welcome to the Calculator"***.
   - Create a method named ***Add(int num1, int num2)*** with suitable return type and **return the addition.**
   - Create another method named ***Multiply(int num1, int num2)*** make **num2 an optional parameter** and keep its **default value as 1** and return the multiplication of two numbers.
2. In ***Program.cs***, call the methods by creating an object and then **print the values.**

Task 3 : Parameter Types [15 Minutes]

1. Create a new class ***ParameterDemo***.
    - In this class, create a method ***Increase(ref int number)*** that **increases the number by 10.**
    - Create a method name ***GetFullName(out string fullname)*** which will **assign your full name** into the out parameter.
    - Create a method ***SumAll(params int[] numbers)*** which will **return the sum of all numbers** passed into the method.
2. Now in ***Program.cs***:
    - Create an object of the ***ParameterDemo*** class and **call all three methods** following their respective procedures.

Task 4 : Constructors [10 Minutes]

1. Create a new class ***Player*** and perform following tasks:
   - Create instance fields ***playerName, level, health***.
   - Create a **default constructor** which will only print ***"Default constructor has been called"***.
   - Create a **parameterized constructor** which will **set the value** of all three fields. **(⚠ Reminder: you set the value while creating constructor)**
2. Now in ***Program.cs***:
   - Create **one object** using **default constructor**.
   - Create **another object** using **parameterized constructor**.
   - **Print the values** of the fields using both objects one after another.

Task 5 : Enums and Records[15 Minutes]

1. Create and **enum** named ***DayType*** with values **Weekday** and **Weekend.**
2. In ***Program.cs***, ask the **user to input** the day (Example: "Sunday").
   - If the day entered is either Friday or Saturday, **print "It is: Weekend"** otherwise print **"It is: Weekday"**.
3. Create a **record *Book(string title, string author, double price)*.**
4. In ***Program.cs***:
   - Create an **object for book** and **assign the values.**
   - Now **create another object** and by using ***with*** expression, **change the title and price.**
   - Print the value of first object.

Task 6: Debugging [15 Minutes]

1. Write the following program and use **debugging to analyze what went wrong.**
   - Ask the user to **input two integers : *marks* and *total*.**
   - **Important:** For taking inputs try utilizing the ***TryParse*** function provided by C# by reading the documentation here. (*Hint: Checkout the example)*
   - Calculate the value using **double percentage = marks / total * 100.**
   - Print the **percentage.**
2. Set **two breakpoints:**
   - One **before the calculation** process takes place.
   - One **after calculation** process.
3. **Run the program and observe:**
   - Why is the output incorrect?

Task 7: Research activity[10 Minutes]

- Research on how **constructors** help in software development. Explain how constructors contribute to **object initialization, code reliability, and maintainability** in object-oriented programming. Provide at least **three examples of the real-world use** cases where constructors are essential in solving practical software problems.. Also mention the resources used for research.

AND

- Research on any **one of the OOP principles** (Encapsulation, Inheritance, Abstraction or Polymorphism). Also **explain about Classes and objects** as well. Give at least **two examples** how you can use the OOP principles.