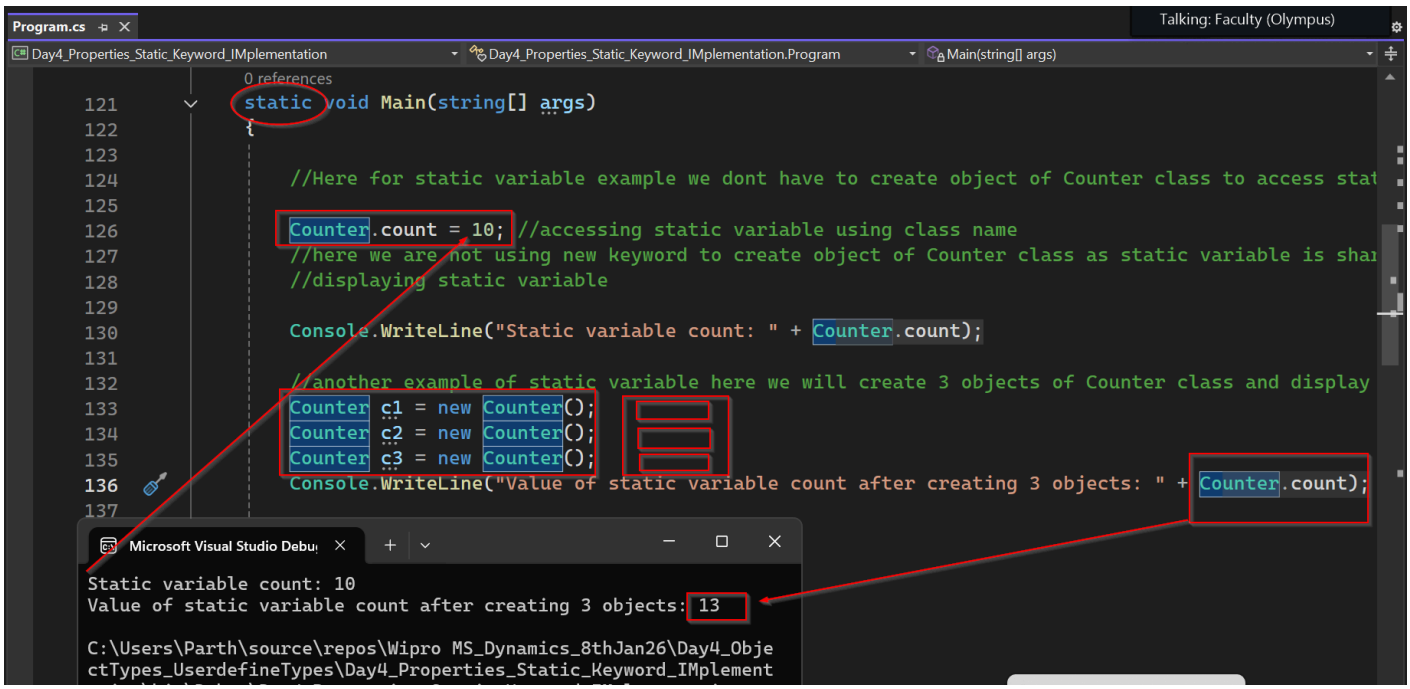
Getting started with Inheritance for code reusability	
Inheritance in C#.NET	Object Oriented Concept
	Inheritance
	Single
	Multilevel
	Multiple (using Interface)
	Hierarchical
	Hybrid
OOPs principles ie Polymorphism in Action	
Polymorphism in C#.NET	Polymorphism
	Method override
	Method overload
	Encapsulation
	Abstraction
Object types in C#.NET	Properties- Why do we need properties ?
	Static - Different use of static key word
	Static class
	Static method
	Static constructor
User define types in C#.NET	Structs
	Interface
	Abstract
	Enumeration
	Difference between Interface and Abstract
	Problem with multiple class inheritance
	Solution for multiple class inheritance using interface.
	Exception handling - Why handling exception is very crucial in any application ? How do we handle them ? diff between finally, finalize and final in C# .NET ?

What are different features of C# released with different versions ?
5 features of C# added in C# version /release 8.0
Searching and sorting

Solid Principles
Design patterns



case study based on static keyword and properties :

An **online learning platform** wants to track **platform-wide settings and usage metrics** that must be **shared across all users and sessions**.

Examples:

- Total number of registered users
- Platform name
- Maximum login attempts allowed
- Global discount rate for courses
    - Must be **common for all users**
    - Should **not belong to a specific object**
      Must be accessible **without creating an instance**

User Story 1	**User Story 2**	**User Story 3**
As a system administrator, **I want to store the platform name globally** **So that it remains consistent across the application.**	**As a system,** **I want to keep track of the total number of registered users** **So that analytics can be generated easily.**	**As a security administrator,** **I want to define maximum login attempts globally** **So that all users follow the same security rule.**

Step 1: Create a Static Configuration Class

- **static properties like platform name, maxLoginAttempts, TotalUsers**
- **static method to increment user ie IncrementUser()**

Step 2: Create a User Class (Instance-based)

- **Username(get,set)**
- **User()**

Step 3: Use Static Properties in Program (main Class)

- **showing maxlogin attempts and platforms (no objects are requires)**
- **Creating user objects and displaying details**

Case study based on User define types - Class, Struct, Enum, Interface,

An **e-commerce platform** wants to build a simple **Order Management System** to:

- Handle different payment methods
- Track order status
- Store lightweight delivery location data
- Maintain clean, extensible architecture

To achieve this, the system uses **user-defined types** in C#.

As a system, I want to represent each order with properties and behavior So that order details can be created and managed easily. Implemented using: `Class`	**As a business user**, I want predefined order statuses So that invalid states are avoided. Implemented using: `Enum`	**As a system**, I want to store delivery coordinates efficiently So that performance is optimized. Implemented using: `Struct`

As a system architect, I want to support multiple payment modes So that new payment methods can be added without changing existing code. Implemented using: `Interface`