***Advance c#***

**Debugging in Visual Studio:**

**Introduction**

Debugging is a critical part of the development process. It allows you to identify, diagnose, and fix issues in your code

**Debug points**

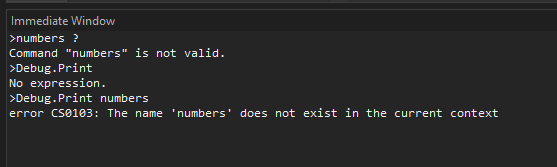
Breakpoints are the most common debugging tool. You set a breakpoint in the code where you want the program to pause during execution, allowing you to inspect the application's state at that point.

**Different Debug windows**

**Locals Window :** The Autos and Locals windows show variable values while you are debugging

**Watch Window**While you're debugging, you can use Watch windows and QuickWatch to watch variables and expressions

**Immediate Window** **Immediate** window to debug and evaluate expressions, execute statements, and print variable values.



**Call Stack**: Shows the sequence of method calls that led to the current point in the program. This helps you trace how the code arrived at the breakpoint.

**Editing**

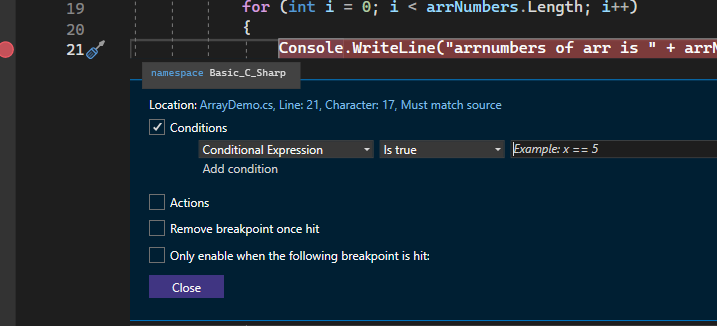
When the program is paused at a breakpoint, you can modify your code, such as fixing bugs or changing logic, without restarting the debugging session.

If you modify a value then you will out of the dub point.

**Conditional break points**

You can control when and where a breakpoint executes by setting conditions.

You have to add a hit count and filter condition.



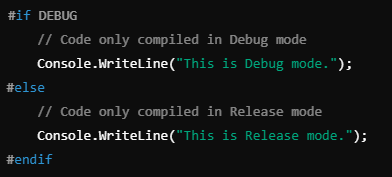
**Data inspector**

The Data Inspector window helps you view and manipulate complex data structures while debugging

It allows you to inspect objects, collections, and any other data types in a more readable format.

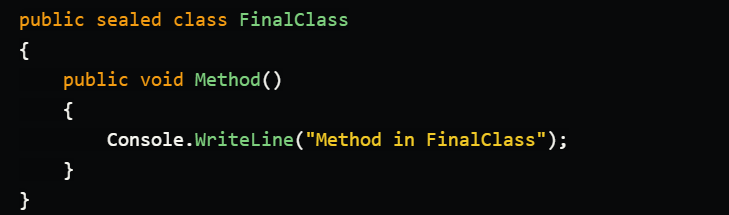
**Conditional compilation**

Include or exclude code based on build configurations (e.g., Debug vs Release).

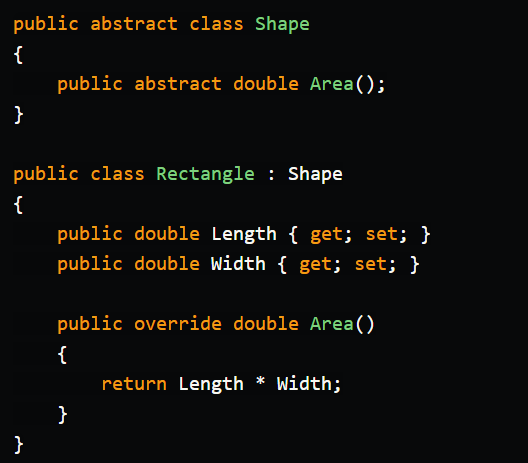
****

**Type of classes:**

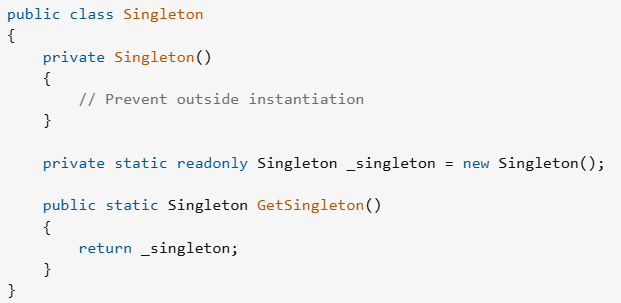
### **1. Sealed Class:** A sealed class is a class that cannot be inherited by other classes. This means no other class can derive from it.



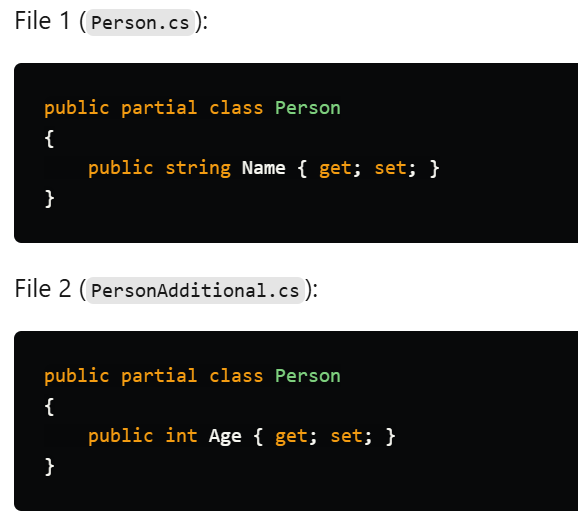
### **2. Abstract Class** An abstract class is a class that cannot be instantiated and may contain abstract methods (methods without implementation). Child classes must override these methods.



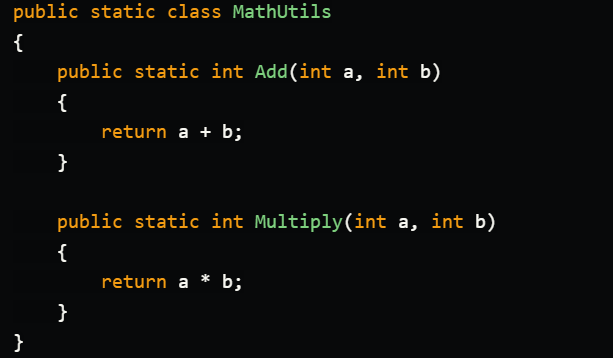
### **3. Singleton Class** A singleton class is a class that allows only one instance of itself to be created during the program's lifetime.



### **4. Partial Class** A partial class allows a single class to be split across multiple files. All parts are combined at compile time.



### **5. Static Class** A static class is a class that cannot be instantiated and can contain only static members.



**Generic class**

Generics in C# allow you to create reusable classes, interfaces, and methods that can work with any data type.

This provides type safety and code reusability without specifying a fixed data type.

## **1. Generic Class** A generic class is a class that takes a type parameter (T), allowing it to work with different data types without code duplication.

## **2. Generic Interface** A generic interface allows an interface to work with any data type, making it flexible and reusable.

## **3. Generic Method** A generic method is a method that takes a type parameter (T), allowing it to process different types of data in a single method.

**Data Serialization (json,xml)**

Serialization is the process of converting an object into a format (like JSON or XML) that can be stored or transmitted and later deserialized back into an object.

It is mainly used to save data (e.g., to a file or database) or transfer data (e.g., via API).

JSON (JavaScript Object Notation) is a lightweight, human-readable format used for storing and exchanging data between applications. In C#, we use newtonsoft.json to serialize/deserialize JSON.

XML (Extensible Markup Language) is a structured format used for storing and exchanging data in a hierarchical way. In C#, we use system.serialization.xmlto serialize/deserialize XML.

**Base library features**

Base libraries in C# refer to the core set of classes, methods, and data types that provide fundamental functionalities required for application development.

**Use Cases of Base Library Features in C#**

Data Types and Collections

File Handling

Networking Support

Exception Handling

**Lambda Expression**

A Lambda Expression in C# is an anonymous function (a function without a name) that can contain expressions or statements.

It is used to making code shorter and more readable.

**syntax:**

(parameters) => expression

=> is the lambda operator

The left side contains parameters.

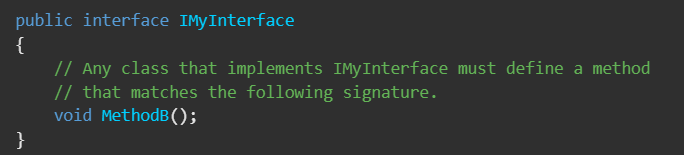
The right side contains the expression or function body

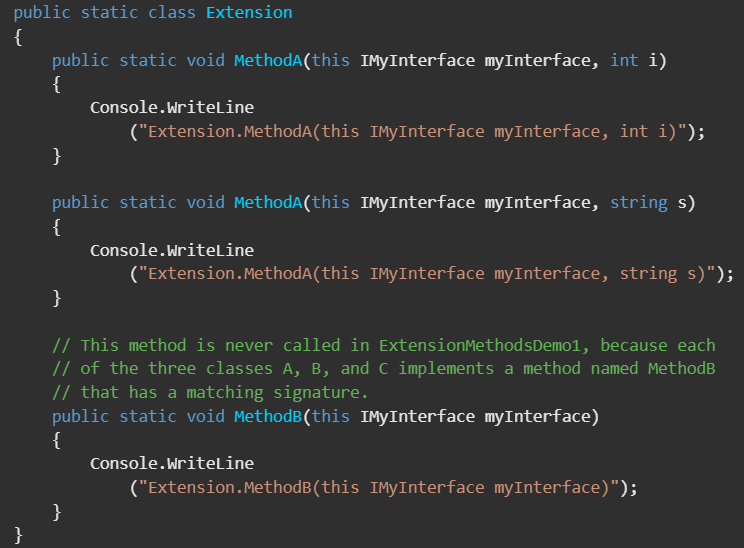
**Extension Methods**

An Extension Method in C# is a static method that allows you to add new methods to an existing class without modifying its original code

It is used to extend the functionality of built-in or custom classes without modifying the source code.

Extension methods are defined in static classes and use the this keyword before the first parameter, which specifies the type being extended



****

public static class → The extension method must be inside a static class.

public static return type method → The method must be static

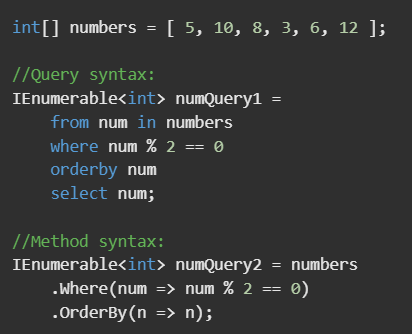
this classname obj , parameters→ The this keyword before the first parameter indicates that this is an extension method for that type

**LINQ (with DataTable, List, etc.):**

LINQ (Language Integrated Query) is a feature in C# that provides a consistent way to query data from different sources like collections (List, Array), databases, XML, DataTable, etc. using a SQL-like syntax.

It makes data manipulation easier, faster, and more readable by using methods and lambda expressions.

include method and query syntax



Both do the same thing, but method syntax is more commonly used in C#.

**var result = students.Where(s => s.Marks > 80 && s.Name.StartsWith("A"));**

**ORM (Object-Relational Mapping)**

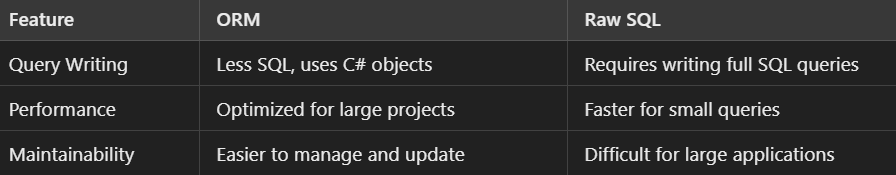
ORM (Object-Relational Mapping) is a technique that allows developers to interact with a relational database (like MySQL, SQL Server) using C# objects instead of writing raw SQL queries.

An ORM tool automatically maps database tables to C# classes, making it easier to perform CRUD (Create, Read, Update, Delete) operations using objects instead of SQL queries.

## **Why Use ORM?**

**Reduces SQL Code**

**Improves Code Maintainability**

****

**Security & Cryptography**

Cryptography is a technique used to encrypt and decrypt data to enhance security.

### **Authentication & Authorization**

Authentication: Verifying who the user is (e.g., login with a username & password).

Authorization: Defining what an authenticated user can do (e.g., Admin vs. Normal User access).

### **Secure Data Storage**

Use hashed and salted passwords instead of storing plain text passwords.

Use SecureString for sensitive data.

Encrypt database connections.

AES (Advanced Encryption Standard) is a strong and commonly used encryption algorithm.

Same key is used for encryption & decryption.

### Asymmetric Encryption (RSA)

Asymmetric encryption uses two keys:  
Public Key → Used for encryption  
Private Key → Used for decryption

SHA-256 is a strong hashing algorithm that helps secure passwords.

Hashes cannot be decrypted, making them useful for password storage.

Security & Cryptography are essential to protect data from unauthorized access.

Authentication & Authorization control user access.

Hashing is used for password storage.

AES is used for secure data storage.

RSA is used for secure communication.

Digital Signatures ensure data integrity and authenticity.

**Dynamic Type**

the dynamic type allows you to store any type of data without specifying its type at compile time. Unlike var which gets its type at compile-time, dynamic is resolved at runtime.

Declared using: dynamic var = name;

Type determined at runtime

Bypasses compile-time type checking

Mainly used for interoperability (e.g., working with COM objects, reflection, JSON, etc.)

