Understanding static in C#

What does 'static' mean?

- The keyword 'static' means belonging to the class itself, not to any specific object (instance).

- A static member (variable, method, property, class) is shared across all instances - or even without creating any instance.

Static Members

1. Static Variables

- Shared memory for the class.

- Only one copy exists, regardless of how many objects you create.

- Used for values that should be common to all objects, like configuration, counters, etc.

2. Static Methods

- Belong to the class, not any object.

- Can be called using the class name directly.

- Cannot access non-static members (because those belong to an object).

3. Static Properties

- Same behavior as static variables but accessed like properties (with get/set).

- Can be used for shared state or computed values related to the whole class.

Static Class

- A class declared with the 'static' keyword.

- Cannot be instantiated - no object can be created.

- All its members must be static.

- It is implicitly sealed, so it cannot be inherited.

- Mostly used for utility/helper classes (e.g., Math class in .NET).

Key Rules and Behavior

- Static members are initialized once when the class is first accessed.

- You don't need an object to access static members - the class name is enough.

- Static methods can only call other static members.

- Instance methods can access both static and non-static members.

- Static constructors are used to initialize static members and are called automatically.

Example Code:

```
public static class Utility
{
public static int counter = 0;

public static void ShowMessage()
{
Console.WriteLine("Hello from static method");
}
}

public class Program
{
static void Main()
{
```

```
Utility.counter = 10;

Utility.ShowMessage();

}

}
```

This shows how static class and members work in C#.