1-Why can't a struct inherit from another struct or class in C#?

Structs in C# can’t inherit from other structs or classes because:

1. **They’re value types** – copied by value, not by reference, which conflicts with polymorphism.
2. **They already inherit from System.ValueType** – and C# doesn’t allow multiple inheritance.
3. **Performance/memory layout** – inheritance would require extra metadata, removing the lightweight nature of structs.
4. **Copy semantics vs. polymorphism** – reference-based inheritance doesn’t mix well with value-based copying.

2-How do access modifiers impact the scope and visibility of a class member?

- **private** → Only inside the same class.
- **internal** → Anywhere in the same project/assembly.
- **public** → Accessible from anywhere.
- **protected** → Inside the class and its derived classes.
- **protected internal** → Same project OR derived classes in other projects.
- **private protected** → Same class OR derived classes in the same project.

3-Why is encapsulation critical in software design?

- **Protects data integrity** – controls how fields are accessed and modified.

- **Hides internal implementation** – allows changes without breaking other code.
- **Improves maintainability** – reduces bugs and makes code easier to update.
- **Supports abstraction & modularity** – keeps components independent.
- **Enables validation & security** – enforces rules and protects sensitive data.

4-Why is encapsulation critical in software design?

- **Protects data integrity** – controls how fields are accessed and modified.
- **Hides internal implementation** – allows changes without breaking other code.
- **Improves maintainability** – reduces bugs and makes code easier to update.
- **Supports abstraction & modularity** – keeps components independent.
- **Enables validation & security** – enforces rules and protects sensitive data

5-what is constructors in structs?

Constructors in structs initialize fields when creating a struct:

- A **default constructor** exists automatically, setting fields to default values.
- You can define **parameterized constructors** to set specific values, and all fields must be assigned in them.

- Before C# 10, you can't define your own parameterless constructor; from C# 10 onward, you can.
- Structs don't have inheritance-based constructor chaining but can call other constructors in the same struct.

6-How do overriding methods like ToString() improve code readability?
Overriding ToString() improves readability by:

1. Providing clear, human-readable object descriptions.
2. Eliminating repetitive manual formatting in code.
3. Making logs, debugging, and output more informative.
4. Centralizing format changes in one place for easier maintenance.

7-How does memory allocation differ for structs and classes in C#?

Structs (value types):

Stored **directly** on the stack or inline in containing objects.

Fast allocation, no garbage collection overhead.

Assignment makes a **full copy** of the data.

Classes (reference types):

Variable holds a **reference** on the stack, object stored on the **heap**.

Requires garbage collection.

Assignment copies the **reference**, so both variables point to the same object.

8-What is copy constructor?

A **copy constructor** creates a new object by copying data from another object of the same type.

Takes one parameter: an object of the same class/struct.

Useful for duplicating objects, making deep copies, and controlling the copy process.

In C#, it must be **manually implemented** (unlike C++ where it's built-in).

9-What is Indexer, when used, as business mention cases u have to utilize it?

Indexer: Allows accessing an object's elements like an array using [].

When used: In classes/structs representing collections (e.g., employee directory, inventory, settings).

Last lecture keywords:

struct / class – Value vs. reference types.

Access modifiers – private, public, internal, protected.

Encapsulation – Hide fields, control access via methods/properties.

Constructors & Overloading – Initialize objects in different ways.

ToString() override – Custom object display.

Copy constructor – Duplicate an object's data