1️⃣ Can a struct inherit a struct?

❌ **No.**

- In C#, a struct **cannot inherit** another struct or class.
- But: all structs **implicitly inherit** from System.ValueType (which itself inherits from object).
- Structs **can implement interfaces**.

2️⃣ Save in memory (struct vs class)?

- **Structs** → stored in **stack** (value types).
- **Classes** → stored in **heap** (reference types).
- If struct has fields like string (reference type), that part goes to the heap, but struct itself stays in stack.

3️⃣ Dependency Inversion Principle (DIP):

👉 High-level modules should not depend on low-level modules.
👉 Both should depend on **abstractions** (interface/abstract class).
✅ Example: Use ILogger instead of hardcoding ConsoleLogger in your business logic.

4️⃣ Specification Design Pattern (interface):

- Used to represent **business rules** as reusable, combinable specifications.
- Typically defined with **interfaces** like ISpecification<T>.
- Example: IsAdultSpecification : ISpecification<Person> checks if person.Age >= 18.
- You can chain them (And, Or, Not).

5️⃣ What other allowed Access Modifiers (AM) in C#?

For **classes, methods, properties**:

- public
- private
- protected
- internal
- protected internal
- private protected

For **struct members**:

- Same as classes, but **structs cannot be protected** (since they don't support inheritance).

6️⃣ Business case: private constructor

Why use a **private constructor**?

- To **prevent creating objects directly**.
- Common in **Singleton Pattern** or **Factory Pattern**.

Example (Singleton):

```
public class Singleton
{
   private static Singleton _instance;

   private Singleton() { }  // private ctor

   public static Singleton Instance
   {
      get
```

```
    {
        if (_instance == null)
            _instance = new Singleton();
        return _instance;
    }
  }
}
```

📌 الخلاصة:

- Structs don’t inherit other structs (only interfaces).
- Stored in stack (mostly).
- DIP = depend on abstraction, not concreteness.
- Specification Pattern uses **interfaces** to define business rules.
- Access Modifiers = public, private, protected, internal, protected internal, private protected.
- Private constructor used in special cases (Singleton, Factory).
