Self-Study Report – Day 8

1) Access Modifiers allowed for an Interface:

In C#, an interface can have public or internal access modifiers.

public → accessible from any class or assembly.

internal → accessible only within the same assembly.

Other modifiers like private, protected, or protected internal are not allowed at the top level.

2) Memory Allocation:

Memory allocation refers to how data is stored and managed in memory.

Value types (like int, double, struct) are stored in the stack.

Reference types (like class objects, arrays) are stored in the heap.

The stack is fast and temporary, the heap is larger and used for objects that need to live longer.

3) Dependency Inversion Principle (DIP):

Part of the SOLID principles.

States that high-level modules should not depend on low-level modules, both should depend on abstraction (like interface or abstract class).

This reduces coupling, increases flexibility, and makes the code easier to maintain.

4) Specification Design Pattern – Interface:

The Specification Pattern is used to define rules or criteria for objects in a reusable way.

Usually implemented using interfaces to allow combining different rules (like AND, OR).

Helps in keeping business rules flexible, clear, and easy to extend without changing existing code.