Part02

1-What is the difference between class and struct in C#?

-Class is a reference type stored in heap, supports inheritance, and multiple variables can reference the same object.
-Struct is a value type stored in stack, cannot inherit, and each variable holds its own copy.

2-What is the difference between class and struct in C#?

1. Inheritance (is-a)

   - Child class inherits from Parent class

   -Example: class Dog : Animal → Dog is-a Animal

2. Association (has-a / uses)

   -One class uses or has a reference to another class

   -Example: class Car { public Engine engine; } → Car has-a Engine

3. Aggregation (part-of, weak ownership)

   -Whole has parts that can exist independently

   -Example: University → Departments → Departments exist even if University is deleted

4. Composition (part-of, strong ownership)

   -Whole owns the parts; if whole is destroyed, parts are destroyed too

   -Example: Car → Engine → Engine cannot exist without Car

5. Dependency (uses temporarily)

   -One class uses another only temporarily, e.g., in a method call

   -Example: Computer.Print(Printer) → Computer depends on Printer

---

Part03

1-chaining to Base?

-It’s when a derived class calls a constructor or method of its base class.

-Ensures that base class initialization happens before derived class logic.

-Useful for code reuse and proper inheritance.

2-why the struct don't support inheritance ? (memory allocation)

-Structs are value types stored on stack or embedded in other objects.

-Inheritance requires heap allocation and flexible object layout to support polymorphism.

-Allowing inheritance would break the lightweight and fast nature of structs.

-Structs can still implement interfaces to get some polymorphic behavior without full inheritance.

3-Overload / Versions for Functions & Performance (Memory & Method Table)?

-Overloads = same method name, different parameters.

-Compiler chooses the correct version at compile time, no runtime lookup needed.

-Virtual/override methods use method table (vtable) for dynamic dispatch.

-Overloads use memory for each version but improve runtime performance because calls are direct.

-Excessive overloads can slightly increase executable size, but runtime speed is better than polymorphic calls.

4- Modular Monolothic?

Chaining to base lets a derived class call the base constructor. Structs can’t inherit because they’re value types on the stack. Overloaded methods are chosen at compile time, fast; virtual methods use vtable at runtime. Modular monolith is one app split into modules—organized, fast, simple deployment, limited scalability.

5-Early binding Vs Late binding (Static binding Vs Dynmic binding)?

Early Binding (Static Binding):

-Method call is resolved at compile time.

-Used for non-virtual methods.

-Fast, memory known, no runtime lookup.

Late Binding (Dynamic Binding):

-Method call is resolved at runtime.

-Used for virtual/overridden methods.

-Supports polymorphism, slightly slower due to vtable lookup.