C# Assignment Day07 Part02

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

In C#, the main difference between a class and a struct is how they behave in memory and how they are used in design.

A **class** is a reference type, which means its instances are stored on the heap, and variables hold a reference (memory address) to the object. When you assign one class variable to another, both variables point to the same object. Classes support inheritance, can be null, and are typically used for complex business models and objects that represent entities like User, Order, or Product.

A struct is a value type, which means it usually stores its data directly (commonly on the stack). When you assign a struct variable to another, a full copy of the data is created. Structs do not support inheritance from other classes (except implicitly from (System.ValueType), but they can implement interfaces. Structs cannot be null unless declared as nullable (e.g., int?). They are best used for small, lightweight objects that represent a single value, such as coordinates, points, or simple data containers.

In short, classes are better for large, complex, and shared objects, while structs are better for small, simple, and immutable data structures.

2. If inheritance is a` relation between classes, clarify other relations between classes.

1. Association (Uses-a Relationship)

-Association means one class uses another class.

-Weak relationship
-Both objects can exist independently

2.Aggregation (Has-a, Weak Ownership)

-Aggregation is a special type of association.

-One class **has** another class
-But both can exist independently

3.Composition (Has-a, Strong Ownership)

-Composition is a stronger form of aggregation.

-One class fully owns another
-If the parent is destroyed, the child is destroyed

4.Dependency (Temporary Usage)

-Dependency means one class depends on another **temporarily**, usually through method parameters.