Part02:

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

Ans :

1. Memory Allocation: Heap vs. Stack

- **Class (Reference Type):** When you create a class instance, the actual data is stored on the **Heap**. The variable itself only holds a "pointer" or reference to that memory location.
- **Struct (Value Type):** The data is stored exactly where the variable is declared, typically on the **Stack** (unless it's part of a class).

2. Copying Behavior

This is where bugs often hide. Because of how they are stored, they behave differently during assignment:

- **Classes:** If you set classB = classA, both variables point to the **same object** in memory. Changing one changes the other.
- **Structs:** If you set structB = structA, C# creates a **full independent copy** of the data. Changing structB does not affect structA.

Feature	**Class**	**Struct**
Type	Reference Type	Value Type
Inheritance	Supports full inheritance	Cannot inherit from other structs/classes

Feature	**Class**	**Struct**
Nullability	Can be null	Cannot be null (unless using Nullable<T>)
Constructor	Default is optional	Requires all fields to be assigned (C# 10+)
Performance	Overhead from Garbage Collection	Faster for small, short-lived data

If inheritance is relation between classes clarify other relations between classes?

Ans:

1. Association

Association is the most basic relationship. It is a "link" where two classes know about each other but their lifecycles are independent. They can exist without one another.

- **Relationship:** "Uses-A" or "Knows-A"
- **Example:** A **Doctor** and a **Patient**. A doctor treats many patients, and a patient may see many doctors. If the doctor retires, the patient still exists.

2. Aggregation

Aggregation is a specialized form of Association. It represents a "Whole-Part" relationship where the "Part" can exist outside of the "Whole."

- **Relationship:** "Has-A" (Weak)
- **Example:** A **Library** and its **Books**. If the library building is demolished, the books still exist and can be moved elsewhere.

3. Composition

Composition is a strong form of Aggregation. The "Part" is physically part of the "Whole" and cannot exist without it. If the parent is destroyed, the child is destroyed too.

- **Relationship:** "Has-A" (Strong)
- **Example:** A **House** and its **Rooms**. A room cannot exist without a house. If the house is deleted, the rooms are deleted with it.

4. Dependency

Dependency is the weakest relationship. It occurs when one class uses another class temporarily, usually as a parameter in a method. The class doesn't "hold" the other as a property.

- **Relationship:** "Depends-On"
- **Example:** A **Printer** and a **Document**. The printer needs a document to perform the Print(Document doc) action, but it doesn't "own" the document permanently.