## How does memory allocation differ for structs and classes in C#?

### Struct:

**Stored on the stack** (or inline inside another object).

Passed **by value** → a **copy** is made when assigned or passed to a method.

Changes inside a method **don’t affect** the original.

### Classes (Reference Types)

* **Stored on the heap**, with the variable holding a **reference (pointer)** to the object.
* Passed **by reference** → multiple variables can point to the same object.
* Changes inside a method **affect the original**.
* Suitable for large, complex objects with behavior.

## What is copy constructor?

A **copy constructor** is a special type of constructor used to create a new object as a copy of an existing object.

## What is Indexer, when used, as business mention cases u have to utilize it?

An **indexer** in C# allows objects to be **indexed like arrays**, using the [] brackets.

* It’s like giving your class array-like behavior.
  + Instead of writing a method like GetEmployee(1), you can simply use employee[1].  
      
    Using   
    Imports namespaces so you can access classes/methods without fully qualifying their names.  
    **public** → Accessible from anywhere.  
    Business case: Exposing APIs or service methods.  
    **private** → Accessible only inside the class.  
    Business case: Hide sensitive data like passwords, credit card validation logic.

## **protected** → Accessible within the class + derived (inherited) classes.

## Business case: Base class (e.g., Employee) sharing core logic with child classes (Manager, Intern).

## **internal** → Accessible only within the same assembly/project.

## Business case: HR system internal utilities not meant for external apps.

## **protected internal** → Combination of protected + internal.

## Business case: Sharing functionality across projects in same solution but only for inherited classes.

## **private protected** → Accessible only within the containing class or derived classes in the same assembly.

## Business case: Advanced encapsulation in enterprise apps when you want tight control.

Self Study

 Before .NET 6, C# team wanted structs to always be fully initialized (because structs are value types, and partial initialization could cause undefined/unexpected states).

In .NET 6+, runtime safety improved, so they relaxed this requirement for **developer convenience**.

في OOP تقدر تخلي الـ constructor **private**.

ده بيتعمل في الـ **Design Patterns** زي Singleton أو Factory:

* **Singleton**: تمنع إنشاء object من بره وتخلي instance واحدة بس.
* **Factory**: تمنع الـ new المباشر وتخلي creation يتحكم فيه factory method.
* أي struct أو class في النهاية وارث من System.Object.
* و System.Object عنده ToString() virtual method.
* بالتالي انت **ممكن تعمل override** للـ ToString() في الـ struct علشان تعرض معلومات مفيدة بدل اسم الـ type.
* ده practice مهم في الــ **Business Logic Layer (BCL)** أو حتى في التطبيقات العادية علشان debugging + readability.