

Part01

1. Question: Why can't a struct inherit from another struct or class in C#?

structs are value-typed that means coping by value and stored in stack unlike classes that copied by ref and stored in the heap

struct-class: So, allowing class to inherit from a struct would break consistency in how values are represented and copied

struct-struct: structs in general built to be simple, only data carriers, light weighted so if you doubt a class can be inherited to more than one then you in wrong way and you must use classes instead

2. Question: How do access modifiers impact the scope and visibility of a class member?

	same class	same project (.dll)	entire solution (other projs)	only children (inheritance)
public	✓	✓	✓	✓
internal	✓	✓	✗	✓ only if in the same project
private	✓	✗	✗	✗
protected	✓	✗	✗	✓
internal protected	✓	✓	✓ (only via inheritance)	✓
private protected	✓	✓ (only via inheritance)	✗	✓ only if in the same proj

3. Question: Why is encapsulation critical in software design?

- Info hidden: hide necessary details from the user like sensitive attrs (attributes) or methods

- Security: by restricting direct access to fields -> enforce validation rules (positive salaries) which is leading to data integrity and prevent invalid states
 - Complexity management: in mega systems like google, amazon encapsulation enforces us to break large methods into smaller ones
4. Question: what is constructors in structs?

Special methods with the same name of the struct with no return type that specializes attributes of this class.
 5. Question: How does overriding methods like ToString() improve code readability?

To specialize and customize some function as I want not as it built
 6. Question: How does memory allocation differ for structs and classes in C#?
Structs allocated in the stack as value typed while Classes allocated in the heap as a reference type.
-

Part02

1. What is copy constructor?
It's a special type of constructors to copy data only (not a ref) from an obj to a new obj
2. LinkedIn article about constructor and its types?
https://www.linkedin.com/posts/marwan-hussein-568373314_fullstack-backend-dotnet-activity-7428506663051223040-55uz?utm_source=share&utm_medium=member_desktop&rcm=ACoAAE_JsmABvTFVwhoN_Ti9LOJQCQO2JKvt0pM



3. What is Indexer, when used, as business mention cases u have to utilize it?
to suppose an object as an array (usually an attr is an arr) to set a value or get it
using the indexer (

```
public dtype this[dtype index]{
    set { // body }
    get { // body }
}
```

Business cases:

Company[Emp_ID]: to return employee data with id = Emp_ID easily

Calender[data]: reuturns all events for this date

4. Summarize keywords we have learnt last lecture
Public, private, internal, protected, private protected, internal protected,
Struct, class, set, get, value, new, this, override,
Encapsulation, inheritance, polymorphism, constructor
-

Part03 (Bonus)

- 1- when to make the constructor Private in which design pattern under use case demonstrating clear point of why??

When using singleton design pattern (only 1 obj in the project) -> like building a vole machine project, then we need only one RAM to store info

- 2- why in .Net 7.0 the constructor not obligated to initialize every class attribute (business wise)

firstly, when an obj is created all attrs initialized with default values (numeric: 0, bool: false, ref: null)

in .net 7 (C#11) the 'required' keyword is added to define if an attribute must be enforced to initialize in the constructor or not.

Business wise:

Table in db may= class in C#

Maybe a field in the table or entity have no values (null) or it's mandatory to have (not null)=(required)

- 3- why allowing developers to override the default constructor from .Net6 (business wise)

suppose we are talking about banking system and there is a class Account that has an attr (recurrence). Then it's not logical to make it as default value (null) in default ctor. But we may need to assign it to egp or \$.etc