DAY 6(QUESTION)

Part1

1. Why can't a struct inherit from another struct or class in C#?
    - Struct it type value type, stored directly on stack. (Memory allocation mechanism >> can't support inheritance)

2. How do access modifiers impact the scope and visibility of a class member?
    - Private: Can be accessed only within the same class.
    - Private protected: Can be accessed within the same class or derived classes that exist in the same project (**inheritance**).
    - Protected: Can be accessed within the same class or derived classes (**inheritance**).
    - Internal : Can be accessed within the same project.
    - Internal protected: Can be accessed within the same project or in derived classes (**inheritance**).
    - Public: Can be accessed both in the same project and from outside the project.

3. Why is encapsulation critical in software design?
    - Control Get & Set Method , so data protected
    - More Readable & Reusability

4. what is constructors in structs?
    - Special method used to initial properties of the struct when an object of the struct is created.
    - Type constructors
        - Default Constructor
        - Parameterized Constructors
        - Copy Constructors
        - Private Constructors
        - Static Constructors

5. How does overriding methods like ToString() improve code readability?
    - Can easily understand code and what the object represents just by looking at its string representation.

6. How does memory allocation differ for structs and classes in C#?
    - Structs is value type ClR allocate 4 bytes at stack
    - Classes is reference type

      declare ref of type object, refer to null

      ClR allocate 4 bytes at stack

      CLR allocate 0 bytes at heap

Part2

What is copy constructor?

By copying variables from another object, this constructor generates an object. Its primary purpose is to set the values of a new instance to those of an existing one.

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

An **indexer** in C# allows objects of a class, struct, or interface to be accessed as if they were arrays. It provides a way to define a custom mechanism for retrieving and setting elements within an object using array-like syntax.

LinkedIn article about constructor and its types?

Basmala Said • You
CS Student | Front-end Developer | React intern @ITI | Trainee @DEPI | Full Sta...
2m • Edited •

الـ Constructor

الـ **Constructor** هو عبارة عن (**Method**)داخل الـ**class**، والاختلاف الوحيد أنه يتم استدعاؤه تلقائيًا بمجرد إنشاء **object** من الـ **class**. لكي يعمل بشكل صحيح، يجب أن يكون اسم الـ **constructor** مطابقًا تمامًا لاسم الـ **class** نفسه.

أنواع الـ **Constructor**:

1.Default Constructor:
هذا النوع من الـ **constructor** يتم استدعاؤه تلقائيًا عند إنشاء **object** من الـ **class**. إذا لم تقم بتعريفه، يقوم الـ **compiler** بإنشائه تلقائيًا.

Syntax:

```
public ClassName() { }
```

2.Parameterized Constructor:
الـ **constructor** الذي يحتوي على parameter واحد على الأقل. يتم استخدامه لتخصيص قيم معينة عند إنشاء الـ **object**.

Syntax:

```
public ClassName(int age, string name)
{
    Age = age;
    Name = name;
}
```

---

3. Private Constructor:

- يتم استخدامه لمنع الآخرين من إنشاء object من هذا الـ **class**. هذا النوع من الـ **constructor**مفيد في حال كنت تريد استخدام الـ **class** فقط في سياقات معينة، مثل الـ **Singleton** أو الكلاسات التي تحتوي على طرق ثابتة فقط.
- مثال على ذلك هو الـ **Math class** في .**NET** الذي يحتوي على **constructors** خاصة.

Syntax:

```
private ClassName() { }
```

---

4. Copy Constructor:

يتم استخدامه لنسخ القيم من الـ **object** القديم إلى الـ **object** الجديد.

Syntax:

```
public ClassName(ClassName value)
{
    Name = value.Name;
}
```

---

5. Static Constructor:

- يتم استخدام الـ **static constructor** لتهيئة (**static members**) في الـ **class**. يتم استدعاؤه مرة واحدة فقط، وذلك قبل أي شيء آخر في الـ **class** يتم تنفيذه. لا يمكنك تمرير معلمات له.

Syntax:

```
static ClassName() { }
```