Part 02

**What is copy constructor?**

A **copy constructor** is a special type of constructor that creates a new object as a copy of an existing one, often used when passing or returning objects by value. In C#, it takes a reference to an object of the same class and initializes the new object with the same data.

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

An **indexer** in C# allows objects to be accessed like arrays, providing a way to access elements within a class using an index. It's useful when you need to create collections or data structures that can be accessed by a custom index, such as in an inventory system or matrix. For example, you can use an indexer to retrieve products by ID in a product inventory or access specific elements in a matrix.

**Summarize keywords we have learnt last lecture**

* **Encapsulation:**
  + Private attributes with getter/setter or properties.
  + Encapsulation separates data definition and access/control.
* **Access Modifiers (AM):**
  + Struct: private, internal, public.
  + Class: private, private protected, protected, internal, internal protected, public.
* **Object-Oriented Programming (OOP)**:
  + **Pillars**: Encapsulation, Inheritance, Polymorphism, Abstraction.
  + Structs support encapsulation and partial polymorphism but not inheritance.
  + Classes support all pillars.
* **Namespaces**: Define a scope for organizing types (struct, Enum, class, interface).
* **Struct Memory:** Value types (e.g. Point) are allocated on the stack. Default values are assigned by less constructors.
* **Inheritance:** 
  + Access modifiers (public, internal, internal protected) demonstrated in TypeC inheriting TypeA.
  + Structs do not support inheritance.
  + TypeA used to demonstrate visibility (protected, internal, etc.).
* **Main Method Examples:**
  + Struct (Point): Shows default initialization and parameterized constructor.
  + Class (Employee): Highlights encapsulation, object initialization, property validations, and method usage.
* **Constructor**: Default and parameterized constructors in struct (Point struct examples).
* **Constructor Overloading:** Different parameter lists for flexibility.