Day06 Part02

1. **What is a Copy Constructor?**

   It's a constructor that creates a new object by copying the data from another object of the same class.

   We use it when we want to make a duplicate object with the same values as an existing object. So instead of assigning values one by one, we copy them from another object.

   C# does not have a built-in copy constructor like C++, but we can create one manually, so we just write a constructor that takes an object of the same class as a parameter and copy its values.

   For example, this is a class for Student

```
class Student
{
  public string Name;
  public int Age;

  // Normal constructor
  public Student(string name, int age)
  {
    Name = name;
    Age = age;
  }

  // Copy constructor  // and this is what we are gonna use
  public Student(Student other)
  {
    Name = other.Name;
    Age = other.Age;
  }
}

// here is the MAIN to test it
Student s1 = new Student("Mariam", 19);

    // Creating a copy of s1
    Student s2 = new Student(s1);
```

```
Console.WriteLine(s2.Name);  // Mariam
Console.WriteLine(s2.Age);   // 19
```

And now as we can see S2 has the same values in S1.

---

2. **LinkedIn article about constructor and its types?**
   Link to the Post in LinkedIn

3.1. What is an Indexer in C#?

An Indexer is a special feature in C# that allows an object to be accessed like an array using square brackets [like these brackets].
It allows us to access class data using an index without exposing internal data structures directly.
So an indexer is a special member of a class that enables objects of that class to be accessed like arrays using the square bracket operator. It is defined using the “this” keyword and is commonly used when a class represents a collection of values.

For example, lets get back to the student class

```
class Students
{
   private string[] names = new string[3];

   // Indexer
   public string this[int index]
   {
      get { return names[index]; }
      set { names[index] = value; }
   }
}

// and here is the MAIN
Students s = new Students();

      s[0] = "Mariam";
      s[1] = "Aya";

Console.WriteLine(s[0]);  // Mariam
```

As we can see, instead of writing s.SetName(0, "Mariam"); we just write s[0] = "Mariam"; which is simpler

3.2. When used

We use it when:

- The class represents a collection of objects
- We want to access data using an index
- We want cleaner and more readable code
- We wanna hide internal implementation details

3.3. As business mention cases u have to utilize it?

as we mentioned before like in student Management System
If we have a class StudentsCollection, we use indexer to access students by index

or in Inventory System and we wanna store products in a warehouse system

even in Banking Systems if we wanna access the transactions by number and so on.

---

4. Summarize keywords we have learnt last lecture

- Access Modifiers: private, private protected, protected, internal, internal protected, public
- Struct vs Class: Value type vs reference type, memory allocation, inheritance limitations.
- Encapsulation: Using private fields with getters/setters and properties.
- Constructor Overloading: Multiple constructors with different parameters.
- ToString() Override: Custom string representation.
- this keyword: Referring to current instance, especially in constructors.
- Properties: Full properties (with get/set) and auto-implemented properties.
- new keyword: For object creation and constructor selection (in structs, just calls constructor).