Copy Constructor

هو Constructor بياخد Object من نفس النوع ويعمل منه نسخة جديدة.

✏️ ملحوظة مهمة #C مفيهوش Copy Constructor built-in زي C++، بس ممكن نعمله احنا عادي!

```
public class Employee
{
    public string Name;
    public int ID;
    public string Department;
    public double Salary;
    public DateTime HireDate;

    // Regular Constructor
    public Employee(string name, int id, string department, double salary)
    {
        Name = name;
        ID = id;
        Department = department;
        Salary = salary;
        HireDate = DateTime.Now;

        Console.WriteLine($"New employee hired: {Name} (ID: {ID})");
    }

    // Copy Constructor - بياخد Employee تاني ويعمل منه نسخة
    public Employee(Employee originalEmployee)
    {
        Name = originalEmployee.Name + " (Copy)"; // عشان نفرق بينهم
        ID = originalEmployee.ID + 1000; // ID مختلف
        Department = originalEmployee.Department;
        Salary = originalEmployee.Salary;
        HireDate = originalEmployee.HireDate;

        Console.WriteLine($"Employee copied: {Name} based on {originalEmployee.Name}");
    }

    public void DisplayEmployee()
    {
        Console.WriteLine($"Employee: {Name} | ID: {ID}");
        Console.WriteLine($"Department: {Department} | Salary: ${Salary}");
        Console.WriteLine($"Hire Date: {HireDate.ToShortDateString()}");
        Console.WriteLine("---");
    }
}
```

Constructor and its types

الـ Constructor هو method خاص بيتنده **تلقائياً** لما تعمل object جديد. وظيفته إنه يجهز الـ object ويدي initial values للمتغيرات.

- ✧ نفس اسم الكلاس بالظبط
- ✧ مالوش return type (حتى void)
- ✧ بيتنده مع keyword `new`
- ✧ لو معملتوش، #C تعملك default constructor فاضي

أنواع الـ Constructors:

1. Default Constructor

```
class Point {
    int x, y;

    public Point() {  // Default constructor
        x = 0;
        y = 0;
    }
}

Point p1 = new Point();  // x=0, y=0
```

2. Parameterized Constructor

```
class Point {
    int x, y;

    public Point(int x, int y) {  // Parameterized constructor
        this.x = x;
        this.y = y;
    }
}

Point p1 = new Point(10, 20);  // x=10, y=20
```

3. Copy Constructor

```
class Point {
    int x, y;

    public Point(Point other) {  // Copy constructor
        this.x = other.x;
        this.y = other.y;
    }
}

Point p1 = new Point(5, 15);
Point p2 = new Point(p1);  // نسخ p1
```

4. Static Constructor

```
class Point {
    static int count = 0;

    static Point() {  // Static constructor
        Console.WriteLine("تم تحميل الكلاس لأول مرة");
        count = 100;
    }
}
// بيتنده مرة واحدة بس عند أول استخدام للكلاس
```

5. Private Constructor

```
class Singleton {
    private static Singleton instance;

    private Singleton() {  // Private constructor
        // منع عمل objects من بره الكلاس
    }

    public static Singleton GetInstance() {
        if (instance == null)
            instance = new Singleton();
        return instance;
    }
}
```

Indexer

الـ **Indexer** ده حاجة تخليك تتعامل مع الأوبجكت بتاعك زي الأرراي كده، تحط رقم أو كلمة جوه أقواس مربعة `[]` وتجيب البيانات.

```
public class MyClass
{
    private string[] data = new string[10];

    // This is an indexer
    public string this[int index]
    {
        get { return data[index]; }
        set { data[index] = value; }
    }
}

// Usage
MyClass obj = new MyClass();
```

```
obj[0] = "Hello";         // Uses indexer setter
string value = obj[0];    // Uses indexer getter
```

When to Use Indexers?

1. Collections & Data Structures

When your class represents a collection or container of items.

```
public class StudentGrades
{
    private Dictionary<string, int> grades = new Dictionary<string, int>();

    public int this[string subject]
    {
        get { return grades[subject]; }
        set { grades[subject] = value; }
    }
}

// Usage
StudentGrades ahmed = new StudentGrades();
ahmed["Math"] = 85;
ahmed["Science"] = 92;
```

2. Matrix/Grid Operations

For 2D arrays or matrix-like structures.

```
public class Matrix
{
    private int[,] data;

    public int this[int row, int col]
    {
        get { return data[row, col]; }
        set { data[row, col] = value; }
    }
}
```