**Snippet 03 — Constructors & Deconstructors**

* **Code Recap**

public class MyClass

{

    public string ValueA { get; }

    public MyClass(string valueA)

    {

        ValueA = valueA;

    }

    public void Deconstruct(out string valueA)

    {

        valueA = this.ValueA;

    }

}

1. **Working Theory (keywords & concepts)**

**constructor**

* Special method with the **same name as the class**.
* Runs automatically when you use new MyClass(...).
* Used to **initialize object state**.

Example from snippet:

public MyClass(string valueA)

{

   ValueA = valueA;

}

When you do:

var obj = new MyClass("Hello");

→ the constructor sets ValueA = "Hello".

**deconstruct**

* Opposite of constructor.
* Allows you to **break an object into parts**.
* Declared with Deconstruct(out ...).
* Lets you use **deconstruction syntax** like with tuples.

Example from snippet:

public void Deconstruct(out string valueA)

    {

        valueA = this.ValueA;

    }

Usage:

var obj = new MyClass("World");

var (val) = obj;   // calls Deconstruct

Console.WriteLine(val); // World

**this**

* Refers to the **current instance** of the class.
* In snippet: this.ValueA ensures we use the property of the current object.

**Practical Example:**

public class Person

{

    public string Name { get; }

    public int Age { get; }

    // Constructor

    public Person(string name, int age)

    {

        Name = name;

        Age = age;

    }

    // Deconstructor

    public void Deconstruct(out string name, out int age)

    {

        name = this.Name;

        age = this.Age;

    }

}

class Program

{

    static void Main()

    {

        var p = new Person("Anas", 22);

        // normal access

        Console.WriteLine($"{p.Name}, {p.Age}");

        // deconstruction

        var (n, a) = p;

        Console.WriteLine($"{n}, {a}");

    }

}

**Output**

Anas, 22

Anas, 22

**🔧 Extras**

* **Records** in C# support deconstruction automatically:

public record User(string Name, int Age);

var u = new User("Ali", 30);

var (n, a) = u;  // works out of the box

* **When to use deconstructors**:
  + When you want your class to behave like a **tuple**.
  + For convenience in LINQ queries or pattern matching.

✅ snippet 14 explained: now you know **constructors (for building)** and **deconstructors (for breaking down)** objects in C#.