**Snippet 11 — Custom Get/Set Logic**

* **Code Recap**

public class MyClass

{

    public string MyProperty { get; set; }

    private int \_otherProperty;

    public int OtherProperty

    {

        get { return \_otherProperty; }

        set { \_otherProperty = value; }

    }

}

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

snippet **11** is about writing **custom getters and setters** instead of using auto-properties. it’s less common in day-to-day code (since auto-properties cover most needs), but it’s **essential knowledge** for when you need extra logic.

**private int \_otherProperty;**

* This is a **backing field** — a private variable that stores the actual data.
* Naming convention: often starts with \_ or lowercase.

**public int OtherProperty**

* This is the **property** exposed to outside code.
* It controls access to \_otherProperty.

**get { return \_otherProperty; }**

* Defines what happens when you read the property.
* Here, it just returns \_otherProperty.
* But you could add logic, e.g., formatting, logging, calculations.

**set { \_otherProperty = value; }**

* Defines what happens when you assign to the property.
* **value** is a special keyword that represents the right-hand side of the assignment.

obj.OtherProperty = 10;

// inside setter → value = 10

**Practical Example:**

**Example with validation (“Program.cs”):**

public class Person

{

    private int \_age;

    public int Age

    {

        get { return \_age; }

        set

        {

            if (value < 0) throw new ArgumentException("Age cannot be negative");

            \_age = value;

        }

    }

}

class Program

{

    static void Main()

    {

        var p = new Person();

        p.Age = 25;  // ✅ works

        Console.WriteLine(p.Age); // 25

        p.Age = -5;  // ❌ throws exception

    }

}

**✅ Why/When to Use Custom Get/Set**

* **Validation** → ensure only valid values are set.
* **Transformation** → e.g., trimming strings before saving.
* **Lazy loading** → load data on first access.
* **Read-only with logic** → return calculated values.

💡 Example:

public string FullName => $"{FirstName} {LastName}";

**🔧 Extras**

* **Auto-properties** are shorthand for this pattern:

public int MyProperty { get; set; }

* compiler secretly creates a backing field for you.
* **Best practice in real-world code:**
  + Start with auto-properties.
  + Switch to full property with custom get/set **only when you need extra logic**.

✅ snippet 11 explained: you now know how to **write custom property accessors** and why they matter (validation, transformations, read-only values).