📌 C# 7.0 → Enum

- Enum gives you named constants instead of using “magic numbers.”
- Example:

```
enum OrderStatus { Pending, Shipped, Delivered, Cancelled }
```

- Instead of status = 2, you write:

```
status = OrderStatus.Shipped;
```

- Makes code more readable and maintainable.

📌 Constructor → One Common Use Case

- A constructor is a special method called automatically when creating an object.
- Used to initialize fields and ensure the object starts in a valid state.

Example:

```
class Student
{
    public string Name { get; }
    public Student(string name)
    {
        Name = name ?? throw new ArgumentNullException(nameof(name));
    }
}
```

📌 Code Performance → Approach

- Often there are multiple ways to achieve the same task.
- Some approaches perform better than others.

🔷 Example: String concatenation performance:

```
// Slower for large loops
string text = "";
for (int i = 0; i < 1000; i++)
    text += i;

// Better approach
var sb = new StringBuilder();
for (int i = 0; i < 1000; i++)
    sb.Append(i);
string result = sb.ToString();
```

📌 Alternative Approach

- Always ask yourself:
    - "Is there a better alternative for the same result?"
    - "Which option is more efficient or easier to maintain?"

🔷 Example:
Instead of using a fixed array:

```
int[] numbers = new int[5];
```

Use a List<T> which is dynamic:

```
List<int> numbers = new List<int>();
```