Object-Oriented Programming Notes

Abstract Class

- A **partial implementation** for other classes.
- A **container** for common code (**implemented members, abstract members**) among many classes.
- **Cannot instantiate** an abstract class (it is not fully implemented).
- Abstract classes do not actually exist in the business model.
- If an abstract class is implemented, properties must match exactly (cannot add `get` or `set` if not originally defined).
- Unlike interfaces, abstract classes **can have constructors**.

Struct vs Class in C#

Key Differences:

1. **Structs are VALUE types** (stored on the stack), while **Classes are REFERENCE types** (stored on the heap).
2. **Structs are passed by VALUE** (copied), while **Classes are passed by REFERENCE** (pointers).
3. **Structs cannot have explicit parameterless constructors**, while **Classes can**.
4. **Structs do NOT support inheritance** (only interfaces), while **Classes do**.
5. **Structs are more memory-efficient for small data**, while **Classes are better for large, complex objects**.
6. **Structs cannot be null** (unless `Nullable<T>` is used), but **Classes can**.

When to Use Struct:

☑ Use when the object is **small, short-lived, and immutable**. ☑ Use when **you don't need inheritance**. ☑ Use when **value-type behavior (copying) is required**. ☑ Use for **performance-sensitive operations to reduce heap allocation**.

When to Use Class:

☑ Use when **you need reference-type behavior (shared instances)**. ☑ Use for **large, complex objects that require modification**. ☑ Use when **inheritance or polymorphism is needed**. ☑ Use when **the object should be shared across multiple parts of an application**.

Rule of Thumb:

Prefer **structs** for small, immutable, frequently used objects. Use **classes** otherwise.

Static Class and Static Members

- A **static class** is a container for **static members** (attributes, properties, constructors, methods, and constants).
- **Cannot create an object** from a static class (**Helper Class**).
- **No inheritance** is allowed for a static class.

- Static methods or properties **should not vary per object**.

Static Class Examples:

- `Math`
- `Console`
- `Convert`
- `Guid` (static struct)

Static Constructor:

- **Called once per class lifetime** before the first usage of the class.

Const vs Readonly

`const`:

- **Must be initialized at declaration**.
- **Cannot be changed after that**.
- **Cannot be static**.
- **Cannot be readonly**.

`readonly`:

- **Can be initialized at declaration or in the constructor**.
- **Can be changed after that**.
- **Can be static**.

☑ **You can initialize `readonly` in the constructor, but you cannot initialize `const` in the constructor**.

Sealed Class

Benefits:

1. **Prevents inheritance**.
2. **Prevents overriding**.
3. **More efficient** since CLR does not search for overridden methods in child classes.

☑ `sealed` can be used with:

- **Classes** (prevents inheritance).
- **Methods** (prevents overriding).
- **Properties** (prevents overriding).

Partial Class

- The `partial` keyword allows **splitting a class, struct, or interface across multiple files**.

Why Use Partial Classes?

1. **Improves code organization and readability**.

2. **Allows multiple developers to work on the same class**.
3. **Separates auto-generated code from developer code** (e.g., generated UI code in frameworks).

☑ **Partial can be used with:**

- **Classes**
- **Structs**
- **Interfaces**
- **Methods**

Partial Methods:

- **No access modifier if not implemented**.
- **No parameters if not implemented**.
- **No return type if not implemented**.

Object-Relational Mapping (ORM)

- ORM is used to **link applications with databases**.
- **Entity Framework Core** is a popular ORM in C#.

ORM Approaches:

1. **Code First**: Classes are created first, and the database is generated from them.
2. **Database First**: The database is created first, and classes are generated from it.

Database Table	ORM Mapping	C# Class
Employees	--->	Employee

Naming Convention Note:

☑ **Class names should be singular (e.g., `Employee`).** ☑ **Database table names should be plural (e.g., `Employees`).**

Summary of Class Types

1. **Concrete Class** - Fully implemented.
2. **Static Class** - Only contains static members, no instantiation.
3. **Abstract Class** - Partial implementation, cannot instantiate.
4. **Sealed Class** - Prevents inheritance.
5. **Partial Class** - Splits implementation across multiple files.

Additional Notes

- **New OOP feature: `record` (research further)**.