

CsharpDay09Assignment

PART01

Name: Mohamed Waleed Elkady

- **Questions**

1) Why explicitly assign enum values?

To control exact numeric values for database storage, API communication, versioning, and avoiding accidental value shifts if new members are inserted.

2) What if enum value exceeds underlying type range?

You get a compile-time error because the value cannot fit into the specified underlying type.

3) Purpose of `virtual` with properties?

It allows derived classes to override and provide a new implementation.

4) Why can't you override a sealed member?

Because `sealed` prevents further modification in derived classes to protect the implementation.

5) Static vs object members?

Static members belong to the class itself.
Object members belong to instances.

6) Can you overload all operators?

No. Some operators like `.`, `:`, `sizeof`, `typeof` cannot be overloaded because they are language-defined behavior.

7) When change enum underlying type?

When optimizing memory or matching external systems (e.g., network protocol using byte).

8) Why can't static class have instance constructors?

Because static classes cannot be instantiated.

9) Advantages of `Enum.TryParse`?

It prevents exceptions and safely handles invalid input.

10) Equals vs == in struct and class?

Struct: == not defined by default; Equals compares values.
Class: == compares references unless overloaded.

11) Why override `ToString`?

To provide meaningful readable output instead of type name.

12) Can generics be constrained?

Yes. Example:

```
where T : IComparable<T>
```

13) Generic method vs generic class?

Generic method applies type parameter only to method.
Generic class applies type parameter to entire class.

14)Why prefer generic swap?

Reduces duplication and works with any type.

15)Why override Equals in Department?

Ensures logical comparison (by Id) instead of reference comparison.

16)Why == not implemented by default for structs?

Because structs are value types and equality logic depends on developer intent, so C# avoids automatic operator generation.

PART02

1-Linkedin article:



Mohamed Waleed Elkady · You

Undergraduate Electronics and Communication Engineer / Full stack .Net dev...
now • Edited • ④

...

أنواع الـ **Classes** في C# وأimenti تستخدم كل نوع
في C#. الـ Class مثل بس وسيلة لتخزين البيانات، ده كمان أداة لتنظيم الكود وتطبيق مبادئ
OOP بشكل صح، لما نفهم أنواع الكلاسات، هنقدر تختار النوع المناسب لكل جزء في مشروعك،
وتحل الكود مرتب وسهل الصياغة.

١. الكلاس العادي (Regular Class)

ده النوع الأساسي والأكثر استخداماً، قادر تعمل منه Objects.
مناسب لأنّي حاجة محتاج تعاملها Object-based.
سهل في الاستخدام ومدرب.
مثال على استخدامه: موديل لموظف، سيارة، منتج... أي حاجة عندها خصائص وسلوكيات.

٢. الكلاس مجرد (Abstract Class)

ده كلاس مش ممكن تعمل منه Object مباشرة.
بنستخدمه لما يكون عندنا فكرة عامة أو قاعدة مشتركة، وعابرين الكلاسات الثانية تطبق
التفاصيل.
ممكن يحتوي على Abstract Methods عامة أو Methods لازم الكلاسات الموروثة تطبقها.
مهم لما تحب تفصل بين الفكرة العامة والتطبيق التفصيلي.

٣. الكلاس المغلق (Sealed Class)

ده كلاس مش ممكن تعمل منه وراثة (inherit).
مقييد لو عندك كود مكتمل ومش عايز أي حد يغيره أو يورثه.
قدّر تعمل منه Objects عادي.
مثال: كلاس Utilities أو Helper لو حاب تمنع أي تعديل بالوراثة.

٤. الكلاس الثابت (Static Class)

ده كلاس كله Static:
مش ممكن تعمل منه Object.
كل الوظائف والبيانات فيه ثابتة.
مناسب للـ Constants أو Helper Methods اللي هستخدمها في كل المشروع.
لما تحب توفر كود مش محتاج Objects، ده النوع الصح.

٥. الكلاس الجزئي (Partial Class)

ممكن تقسيم نفس الكلاس على أكثر من ملف.
يسهل إداره الكود في المشاريع الكبيرة.
كل ملف ممكن يحتوي جزء من الخصائص أو الوظائف.
بعد التجميع، الكلاس الكامل يشتمل كأنه ملف واحد.
مثال: لو عندك Employee Class ضخم، ممكن تقسيمه بين ملف للبيانات وملف للوظائف.

لله أنواع الكلاسات دي مهمّة؟
لأنّك لما تعرف النوع الصح لكل جزء من مشروعك:
الكود ه يكون مرتب ومنظم.
سهل تعديل فيه أو توسيع عليه بعدين.
هتطبق OOP principles صح: الوراثة، التجريد، التعديل المحدود...
هتقلل مشاكل الأداء أو الأخطاء الناتجة عن سوء اختيار نوع الكلاس.

الخلاصة:

أي كلاس عادي يتعامله Objects.
قدرة عامة بتطبقها في كلاس ثان.

مش مسموح بالوراثة: Sealed Class.

كل حاجة فيه ثابتة ومش ممكن تعامل منه Object: Static Class.
مقسم على أكثر من ملف لإدارة الكود الكبير: Partial Class.

CSharp #Programming #OOP #ObjectOrientedProgramming #Coding#
#SoftwareDevelopment #DotNet #CodeSmart #LearnToCode #DeveloperLife
#TechTips #CleanCode #SoftwareEngineering #CodeBetter #ProgrammingTips

Show translation

Like

Comment

Repost

Send

2- Generalization concept using Generics

Generalization is the idea of writing code that can work with many types instead of being limited to one specific type. In C#, **generics** are used to achieve generalization. They allow classes, methods, or interfaces to operate on any data type while maintaining **type safety**. This means you can write reusable and flexible code that works for integers, strings, or even custom classes, without duplicating logic for each type.

Key points:

- Reusable code for multiple types.
 - Type-safe operations.
 - Reduces duplication and increases flexibility.
-

3- Hierarchy design in real business

Hierarchy design is the practice of organizing classes and objects in a **tree-like structure** that represents real-world business relationships and roles. In C#, this is implemented through **inheritance, abstract classes, and interfaces**. The idea is to model general concepts as base classes (e.g., Employee) and specific roles as derived classes (e.g., Manager, Developer).

Key points:

- Base classes represent general concepts; derived classes represent specific roles.
- Reflects real business structures and responsibilities.
- Improves code clarity, maintainability, and reusability.