Part02

. ليه الـ OOP فيه أنواع Classes مختلفة؟
الكلاسات أنواع، وكل نوع ليه "شخصية" وصلاحيات بتخلي كودك منظم ومحمي:

1- الـ Concrete Class (الكلاس العملي)
ده النوع "الطيب" اللي بنستخدمه 90% من الوقت. كلاس كامل المواصفات، تقدر تاخد منه نسخة (Object) وتستخدمها فوراً.

وظيفته: بيحتوي على الـ Data (الخصائص) والـ Logic (الأفعال) اللي هتتنفذ فعلياً.

2- الـ Abstract Class (الكلاس المايسترو)
ده كلاس "نظري" شوية، مينفعش تعمل منه Object أبداً. وظيفته إنه يكون مرجع لباقي الكلاسات.

وظيفته: بيحط (Contract) اللي أي كلاس هيورث منه لازم يلتزم بيها. لو عندك 10 كلاسات (قطة، كلب، أسد)، الـ Abstract Class بيكبرهم كلهم إن يكون عندهم دالة اسمها MakeSound بس كل واحد بطريقته.

قوته: بيمنع تكرار الكود (DRY) وبيوحد طريقة التعامل مع الأنواع المختلفة.

3- الـ Static Class (الكلاس الخدمي)
ده كلاس "منعزل"، لا تقدر تورث منه ولا تعمل منه Object. هو موجود عشان يقدم خدمات سريعة.

وظيفته: مخزن للدوال والأدوات اللي مش محتاجة حالة (State). تناديه باسمه (Class.Method) وتخلص مصلحتك.

مثال: كلاس العمليات الحسابية أو التحويل بين العملات.

1)

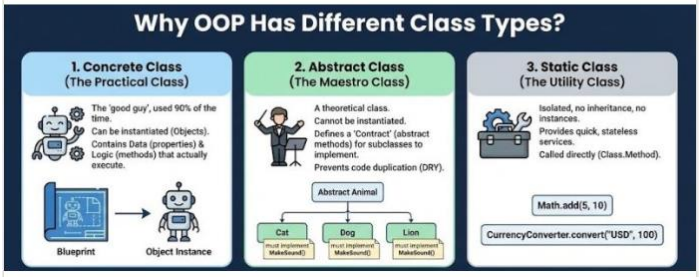
ليه التقسيمة دي مفيدة؟

التقسيمة دي مش بس عشان التنظيم، هي اللي بتخلي كودك يحترم مبادئ الـ SOLID بشكل تلقائي:

لما تستخدم Abstract Classes، إنت كدة بتطبق حرف الـ O (مبدأ الـ Open-Closed)؛ كودك "مفتوح" للإضافة (تقدر تضيف أنواع جديدة بتورث منه) لكنه "مغلق" ضد التعديل (مش هتغير في الكود القديم).

وبتحقق حرف الـ D (الـ Dependency Inversion)؛ لأنك بتخلي كودك يعتمد على "الفكرة العامة" (Abstract) مش على "النوع المحدد" (Concrete).
SoftwareEngineering #OOP #SOLIDPrinciples #DesignPatterns #Coding# #CleanCode

Show translation



2) Generalization using Generics?

Generalization means creating reusable and flexible code that works with different data types instead of rewriting the same logic multiple times.

Generics allow us to define classes, methods, or interfaces with type parameters (like <T>), making the code type-safe and reusable.

3) Hierarchy Design in Real Business?

Hierarchy design means organizing system components into parent-child relationships that reflect real-world structures.

In business systems, we create a base class (e.g., Employee) and extend it into specialized subclasses (e.g., Manager, Developer).

Part03

---

1) Generalization using Generics?

Event-Driven Programming is a programming paradigm where the flow of the program is determined by events such as user actions, messages, or system signals.

Instead of executing code sequentially, the program waits for specific events to occur. When an event happens, a predefined function called an event handler is executed.