Part01

- **Question: Why is it better to code against an interface rather than a concrete class?**
  Writing code against an interface makes the program more flexible, extensible, and easier to maintain and test compared to writing against a concrete class directly.

- **Question: When should you prefer an abstract class over an interface?**
  You should prefer an abstract class over an interface when you want to share common behavior (implementation) among classes, not just define method signatures. An abstract class allows you to provide some ready-made methods and shared properties, whereas an interface only defines method signatures without implementation**.**

- **Question: How does implementing IComparable improve flexibility in sorting?**
  Implementing IComparable allows objects to define their natural order, so collections can be sorted consistently without hardcoding comparison logic. This makes sorting flexible and reusable for different types of objects.

- **Question: What is the primary purpose of a copy constructor in C#?**
  The primary purpose of a Copy Constructor in C# is to create a new, independent copy of an existing object, so that the new object has the same property values without affecting the original object.

- **Question: How does explicit interface implementation help in resolving naming conflicts?**
  Explicit Interface Implementation helps resolve naming conflicts when a class implements multiple interfaces that have methods with the same name, because it allows each method to be accessed only through its specific interface, preventing conflicts**.**

- **Question: What is the key difference between encapsulation in structs and classes?**
  The key difference is that structs are value types, so copying them creates independent objects, while classes are reference types. Therefore, encapsulation in a struct operates on a separate copy, whereas in a class, encapsulation controls the same shared reference.

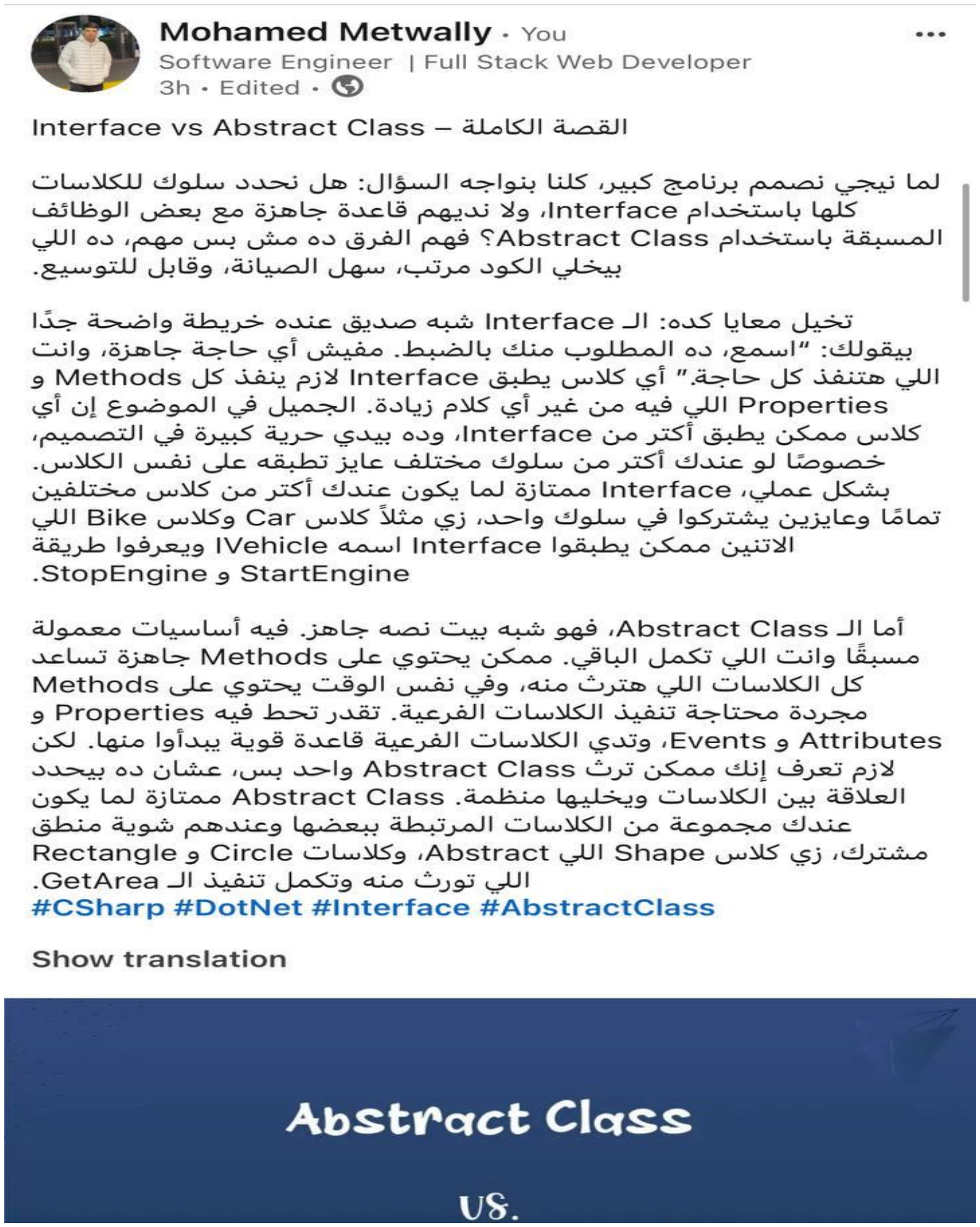
- **Question: what is abstraction as a guideline, what’s its relation with encapsulation?**
  Abstraction is a programming guideline that focuses on hiding internal details and exposing only what is necessary to the user. Its relation to encapsulation is that encapsulation hides data inside the object and controls access to it, making abstraction possible.

- **Question: How do default interface implementations affect backward compatibility in C#?**
  Default Interface Implementations in C# allow adding new methods to an existing interface without breaking existing code that depends on the interface. This preserves backward compatibility.

- **Question: How does constructor overloading improve class usability?**
  Constructor overloading improves class usability by allowing objects to be created in different ways with different parameters, depending on the need, without writing extra methods.

Part02

- **LinkedIn article about abstract class ?**



Mohamed Metwally • You ...
Software Engineer | Full Stack Web Developer
3h • Edited •

Interface vs Abstract Class – القصة الكاملة

لما نيجي نصمم برنامج كبير، كلنا بنواجه السؤال: هل نحدد سلوك للكلاسات كلها باستخدام Interface، ولا نديهم قاعدة جاهزة مع بعض الوظائف المسبقة باستخدام Abstract Class؟ فهم الفرق ده مش بس مهم، ده اللي بيخلي الكود مرتب، سهل الصيانة، وقابل للتوسيع.

تخيل معايا كده: الـ Interface شبه صديق عنده خريطة واضحة جدًا بيقولك: "اسمع، ده المطلوب منك بالضبط. مفيش أي حاجة جاهزة، وانت اللي هتنفذ كل حاجة." أي كلاس يطبق Interface لازم ينفذ كل Methods و Properties اللي فيه من غير أي كلام زيادة. الجميل في الموضوع إن أي كلاس ممكن يطبق أكتر من Interface، وده بيدي حرية كبيرة في التصميم، خصوصًا لو عندك أكتر من سلوك مختلف عايز تطبقه على نفس الكلاس. بشكل عملي، Interface ممتازة لما يكون عندك أكتر من كلاس مختلفين تمامًا وعايزين يشتركوا في سلوك واحد، زي مثلاً كلاس Car وكلاس Bike اللي الاتنين ممكن يطبقوا Interface اسمه IVehicle ويعرفوا طريقة StartEngine و StopEngine.

أما الـ Abstract Class، فهو شبه بيت نصه جاهز. فيه أساسيات معمولة مسبقًا وانت اللي تكمل الباقي. ممكن يحتوي على Methods جاهزة تساعد كل الكلاسات اللي هترث منه، وفي نفس الوقت يحتوي على Methods مجردة محتاجة تنفيذ الكلاسات الفرعية. تقدر تحط فيه Properties و Attributes و Events، وتدي الكلاسات الفرعية قاعدة قوية يبدأوا منها. لكن لازم تعرف إنك ممكن ترث Abstract Class واحد بس، عشان ده بيحدد العلاقة بين الكلاسات ويخليها منظمة. Abstract Class ممتازة لما يكون عندك مجموعة من الكلاسات المرتبطة ببعضها وعندهم شوية منطق مشترك، زي كلاس Shape اللي Abstract، وكلاسات Circle و Rectangle اللي تورث منه وتكمل تنفيذ الـ GetArea.
#CSharp #DotNet #Interface #AbstractClass

Show translation

- **What we mean by coding against interface rather than class ? and if u get it so What we mean by code against abstraction not concreteness ?**
  The code becomes flexible, interchangeable, and easier to test, because any class implementing the interface can be used without changing the code

  Reduce coupling between different parts of the program, making the code easier to extend and modify.

- **What is abstraction as a guideline and how we can implement this through what we have studied ?**
  Abstraction is a programming guideline that focuses on hiding internal details of an object and exposing only what is necessary to the user.
  We can implement it using Abstract Classes and Interfaces, by defining essential methods and properties without specifying the implementation, making the code flexible and extensi

Part03 Bonus

- **what is operator overloading**
  Operator Overloading means *redefining or customizing the behavior of standard operators (like +, -, , ==) for objects of a specific class.
  This allows you to use operators naturally with objects instead of calling special methods.