  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Part01**

**Question: Why is it better to code against an interface rather than a concrete class?**

علشان الكود يبقى مرن وقابل للتبديل وإعادة الاستخدام وسهل في الاختبار

**Question: When should you prefer an abstract class over an interface?**

لما تحتاج تشارك كود أو بيانات أساسية بين الكلاسات، أو لما تكون فيه علاقة واضحة (is-a) بين الكلاسات.

**Question: How does implementing IComparable improve flexibility in sorting?**

لأنه يسمح بتعريف طريقة مقارنة للكائنات، وبالتالي أي list أو array يقدر يتعمله sort بسهولة من غير ما تكتب منطق الفرز كل مرة.

**Question: What is the primary purpose of a copy constructor in C#?**

الغرض الأساسي هو إنشاء كائن جديد بنفس قيم كائن موجود بالفعل.

**Question: How does explicit interface implementation help in resolving naming conflicts?**

بيخلي تنفيذ الدالة مرتبط بالـ interface نفسه، فلو فيه كذا interface عندهم نفس اسم الدالة تقدر تميز بينهم وتنفذ كل واحدة بشكل منفصل.

**Question: What is the key difference between encapsulation in structs and classes?**

في \*\*class\*\* الكابسلة بتتعامل مع \*\*reference type\*\*، وفي \*\*struct\*\* الكابسلة بتتعامل مع \*\*value type\*\*، لكن المبدأ واحد إنك تخفي البيانات وتتحكم في الوصول ليها.

**Question: what is abstraction as a guideline, what’s its relation with encapsulation?**

الـ \*\*Abstraction\*\* معناه تركز على \*إيه\* الكائن بيعمل وتخفي \*إزاي\* بيتعمل.

الـ \*\*Encapsulation\*\* معناه تخفي تفاصيل البيانات جوه الكائن وتتحكم في الوصول ليها.

العلاقة: الكابسلة بتساعد تحقق الـ abstraction.

**Question: How do default interface implementations affect backward compatibility in C#?**

بتسمح تضيف دوال جديدة في الـ interface من غير ما تكسر الكود القديم، لأن أي كلاس قديم مش محتاج يطبق الدالة الجديدة إلا لو حب يغير التنفيذ الافتراضي.

**Question: How does constructor overloading improve class usability?**

بيخلي عندك أكتر من طريقة لتهيئة الكائن حسب البيانات المتاحة، وده بيسهل الاستخدام ويقلل التكرار.

**Part02**

**2- 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 ?**

يعني تكتب الكود يعتمد على الشكل العام زي interfaceأو abstract class،

مش على كلاس محدد، عشان يبقى مرن وسهل التبديل.

**3- What is abstraction as a guideline and how we can implement this through what we**

**have studied ?**

الـ abstraction كقاعدة معناها نركز على الوظيفة الأساسية ونخفي التفاصيل.

نطبق ده في C# باستخدام interfaces أو abstract classes اللي بتحدد إيه لازم يتعمل من غير ما تقول إزاي يتعمل.

**Part03 Bonus**

**1- self study report**

**1) What Access Modifiers are allowed for the interface?**

public أو internal فقط.

الأعضاء جوه الإنترفيس بتكون public بشكل افتراضي.

**2) Memory Allocation**

الـ Interface نفسها ملهاش Memory.

اللي بياخد Memory هو الكلاس اللي بيعمل implement للـ Interface.

**3) Dependency Inversion**

الكود يعتمد على Abstraction (Interface) مش على الكلاس نفسه.

كده يبقى مرن وسهل تغير الكلاسات من غير ما نغيّر الكود الأساسي.

**4) Specification design pattern : interface**

نستخدم Interface عشان نكتب شروط أو Specifications.

نقدر نركبها مع بعض (AND / OR).

بيسهل الإضافة أو التغيير من غير ما نعدل الكود الأساسي.

**2- what is operator overloading**

Operator Overloading يعني ندي للـ operator (+, -, …) وظيفة جديدة يشتغل بيها مع الـ objects بدل ما يشتغل بس مع الأرقام العادية.