

**Part 01**

**Question: Why does defining a custom constructor suppress the default constructor in C#?**

لأن في ‎C#‎ لو إنت ما كتبتش أي constructor، الكومبايلر بيضيف لك واحد افتراضي من غير بارامترات.

لكن أول ما تكتب constructor خاص بيك، الكومبايلر بيفترض إنك مسئول عن الإنشاء، وبالتالي مش بيولّد الـ default constructor تلقائيًا.

**Question: How does method overloading improve code readability and reusability**?

لـ method overloading بيخلي عندك نفس اسم الدالة يشتغل مع مدخلات مختلفة.

ده بيسهّل القراءة عشان الاسم واحد، وبيقلل التكرار عشان تستخدم نفس الدالة بدل ما تعمل دوال كتير بأسماء مختلفة.

**Question: What is the purpose of constructor chaining in inheritance?**

الـ constructor chaining في الوراثة فكرته إن:

* أولًا يتنفذ constructor الأب يجهّز الأساس،
* وبعده يتنفذ constructor الابن يجهّز الباقي.

**Question: How does new differ from override in method overriding?**

override:   
بيغيّر تنفيذ دالة موجودة في الـ base class ويستبدلها بتطبيق جديد في الـ child class.

new:  
 بيخفي دالة الـ base class ويعمل نسخة جديدة مختلفة، من غير ما يغيّر القديمة.

**Question: Why is ToString() often overridden in custom classes?**

علشان بشكل افتراضي ‎ToString()‎ بيرجع اسم الـ Class بس،

لكن لما نعمله override نقدر نخليه يدي وصف أو بيانات أوضح عن الكائن نفسه،

وده بيسهّل عرض المعلومات بدل ما يبان مجرد اسم الكلاس.

**Question: Why can't you create an instance of an interface directly?**

لأن الـ interface ما فيهاش تنفيذ، فيها بس تعريف للوظائف.

فالكومبايلر مش يعرف يجهّز كائن منها مباشرة.

لازم Class يطبّق الـ interface الأول، وبعدها تنشئ كائن من الـ Class ده.

**Question: What are the benefits of default implementations in interfaces introduced in C# 8.0?**

الفايدة من الـ default implementations في C# 8.0:

* تقدر تضيف دوال جديدة في الـ interface من غير ما تعطل الكود القديم.
* الكلاسات يا تستخدم التنفيذ الجاهز، يا تعمل تنفيذ خاص بيها.

**Question: Why is it useful to use an interface reference to access implementing class methods?**

لأنك لما تستخدم interface reference تقدر تتعامل مع أي كلاس مطبّق للـ interface بنفس الطريقة،

وده بيديك مرونة وبيخلي الكود أسهل في التوسيع وإعادة الاستخدام.

**Question: How does C# overcome the limitation of single inheritance with interfaces?**

في C# الكلاس ما يقدرش يرث غير من أب واحد،

لكن يقدر يطبّق كذا interface مع بعض،

وبكده ياخد مميزات الوراثة المتعددة بشكل أبسط وآمن.

**Question: What is the difference between a virtual method and an abstract method in C#?**

virtual method:   
ليها تنفيذ افتراضي، وممكن الكلاس الابن يغيّره أو يسيبه زي ما هو.

abstract method:   
ما ليهاش أي تنفيذ، والكلاس الابن لازم يكتب لها تنفيذ.

**Part02**

**What is the difference between class and struct in C#?**

class:   
نوع مرجعي (reference type)، بيتخزن في الـ heap، وبيسمح بالوراثة.

struct:   
نوع قيمي (value type)، بيتخزن في الـ stack أو جوه الكائن نفسه)، ومش بيدعم الوراثة (غير من (interfaces).

**If inheritance is relation between classes clarify other relations between classes**

العلاقات بين الكلاسات غير الوراثة:

Association:   
كلاس يعرف كلاس تاني ويستخدمه.

Aggregation:   
كلاس جواه كائن من كلاس تاني، لكن كل واحد ممكن يعيش لوحده.

Composition:   
كلاس جواه كائن من كلاس تاني، ومصيره مربوط بيه لو الأساسي راح، التاني يروح معاه.

**Part03 Bonus**

**1- self study report**

1. General constructor:  
   
بيتنفذ أول ما نعمل كائن ويهيئ المتغيرات ولو ما كتبناش بيعمله الكومبايلر افتراضي

2. Memory model and struct:   
  
الكلاس reference type في heap والـ struct value type في stack والهدف إنه يكون خفيف للبيانات الصغيرة

3. Overloading and memory:   
  
كل دالة overload ليها توقيع مختلف والكومبايلر بيحددها وقت الترجمة

4. Early vs Late binding: early (new)   
  
يتحدد وقت الترجمة وبيخفي دالة الأب late (override) يتحدد وقت التشغيل وبيغير دالة الأب

**2- what is static and dynamic binding**

Static binding:   
الكومبايلر يحدد أي دالة هتتندَه وقت الترجمةزي ( method overloading).

Dynamic binding:   
يعني البرنامج يحدد وقت التشغيل أي دالة تتنفذ،  
ولو الكائن الحقيقي من الـ Child هتتنفذ دالته حتى لو استدعيتها من متغير نوعه Parent.