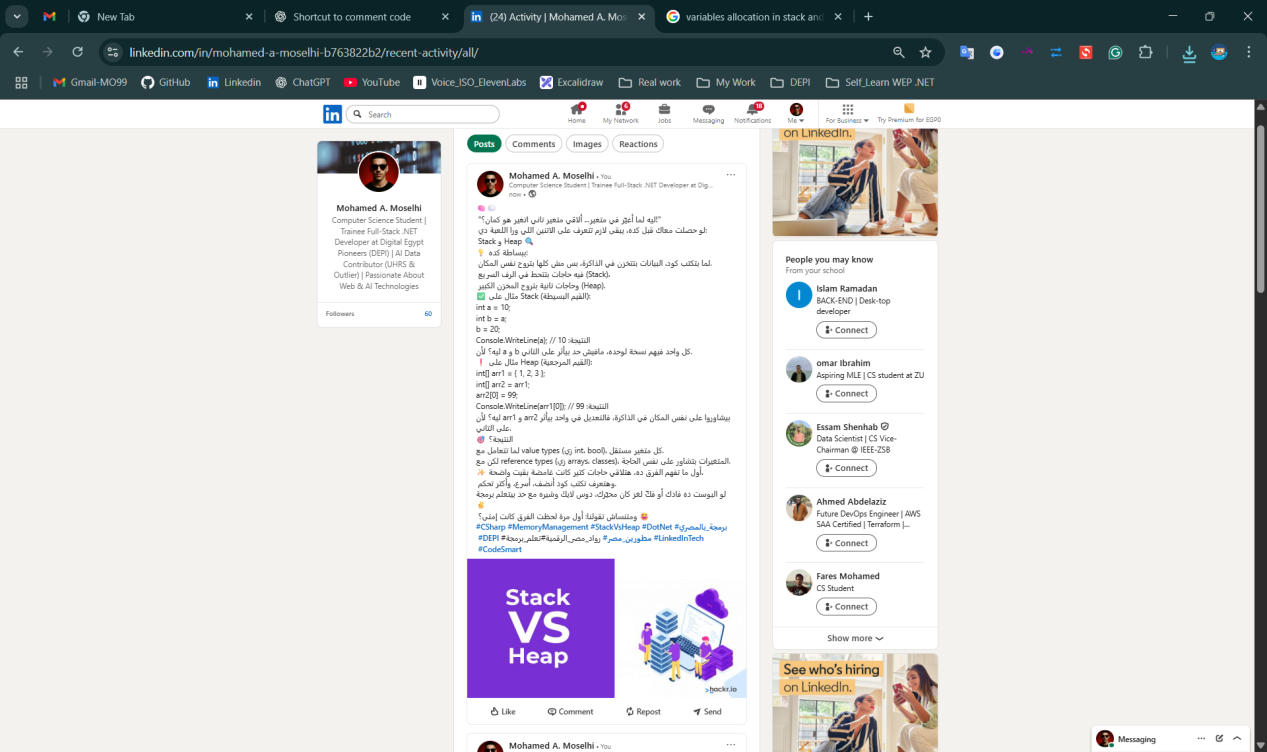


LinkendIn



Question: What is the shortcut to comment and uncomment a selected block of code in Visual Studio?

لو حابب تعلّق جزء من الكود في Visual Studio (يعني تخليه ما يشتغل مؤقتًا)، ببساطة ظلّله واضغط:

**Ctrl + K وبعدها على طول Ctrl + C**

ولو عايز تشيل التعليق وترجع الكود يشتغل تاني، ظلّله واضغط:

**Ctrl + K وبعدها Ctrl + U**

يعني أول واحدة للتعليق، والتانية لإلغاء التعليق.

Question: Explain the difference between a runtime error and a logical error with examples.

الـ Runtime Error

يعني البرنامج بيبدأ يشتغل، وبعدين يحصل خطأ فجأة ويقف.

مثال:  
عايز تقسم رقم على صفر:

int x = 5;

int y = x / 0; // هنا البرنامج هيقف

الـ Logical Error

يعني البرنامج يشتغل عادي ومافيش أي خطأ ظاهر، بس النتيجة تطلع غلط لأنك فكرت غلط.

مثال:  
كنت عايز تقسم الرقم على 2، بس بالغلط ضربته:

int x = 10;

int y = x \* 2; // المفروض x / 2

Question: Why is it important to follow naming conventions such as PascalCase in C#?

لما بنكتب كود في C#، مهم نلتزم بطريقة تسمية معينة زي PascalCase علشان:

أولًا: الكود يبقى شكله واضح ونضيف

يعني لما حد غيرك ييجي يقرأ الكود، يقدر يفهم بسرعة إن دي دالة أو كلاس من شكل الاسم.  
مثلاً:

GetUserData() // دالة

StudentProfile // كلاس

لو كل واحد سمّى على مزاجه، الكود هيبقى ملخبط ومش مفهوم.

ثانيًا: الشغل الجماعي يكون منظم

لو بتشتغل مع فريق، كل واحد لازم يكتب بنفس الشكل علشان الكود يبان كأنه مكتوب بإيد واحدة، وده يسهل التعاون والمراجعة.

ثالثًا: يخلي شغلك شكله احترافي

الناس اللي بتشتغل شغل محترف دايمًا بتهتم بتفاصيل زي دي، مش بس الكود يشتغل، كمان يكون شكله مرتب.

يعني إيه PascalCase؟

يعني كل كلمة في الاسم تبدأ بحرف كبير، زي:  
PrintReport, CalculateTotal, UserProfile

Question: Explain the difference between value types and reference types in terms of memory allocation.

Value Types أنواع القيمة:

بيتخزنوا في جزء من الذاكرة اسمه Stack، ولما تعمل نسخة منهم، كل نسخة بتبقى مستقلة لوحدها.

Reference Types أنواع المرجع:  
بيتخزنوا في جزء تاني من الذاكرة اسمه Heap، لكن المتغير نفسه بيكون في Stack وبيحتوي على "رابط" للمكان في الـ Heap. ولما تنسخهم، بتنسخ العنوان مش البيانات، يعني النسخ بيشاوروا على نفس الحاجة.

Question: What will be the output of the following code? Explain why:

int a = 2, b = 7;   
Console.WriteLine(a % b);

لما تستخدم العلامة % في C#، دي معناها "باقي القسمة".

يعني لو قسمت رقم على رقم تاني، مش مهم الناتج كام، إحنا بنشوف الباقي اللي فاضل بعد القسمة.

في الحالة دي، الرقم الأول أصغر من التاني، فمش هيقدر يتقسم عليه، والباقي هيفضل هو نفس الرقم.

النتيجة هتكون : 2

لأن 2 ما بتتقسمش على 7، فـ الباقي هو 2 نفسه.

Question: How does the && (logical AND) operator differ from the & (bitwise AND) operator?

=&& بتشتغل في الشروط، ولو أول شرط غلط مش بتكمل.

=& بتشتغل على الأرقام أو الشروط، بس بتكمل دايمًا حتى لو أول حاجة غلط.

يعني && أذكى شوية، وبتعرف إمتى توقف، لكن & ماعندهاش تفاهم، بتنفذ كله.

Question: Why is explicit casting required when converting a double to an int?

لما تيجي تحوّل رقم فيه فاصلة زي 5.9) إلى رقم صحيح (زي5)،)  
الكمبيوتر بيقولك:  
"أنت متأكد؟ هتشيل الجزء العشري وهتخسر جزء من الرقم".

علشان كده، ما بيعملش التحويل لوحده، ولازم تقوله صراحة إنك موافق، وده اسمه explicit casting.

Question: What exception might occur if the input is invalid and how can you handle it

لما المستخدم يدخل حاجة غلط، إيه اللي ممكن يحصل؟

البرنامج ممكن يوقف فجأة ويطلعلك خطأ (Exception).

زي إيه؟

لو كنت طالب منه يكتب رقم، وكتب حروف، يحصل خطأ اسمه FormatException.

إزاي نحل المشكلة؟

بنستخدم حاجة اسمها try-catch، ودي بتخلّي البرنامج ما ينهارش، وبدل ما يوقف، يقدر يطبعلك رسالة أو يتصرف.

Question: Given the code below, what is the value of x after execution? Explain why

int x = 5;

int y = ++x + x++;

إيه اللي بيحصل؟

الأول زوّد x فبقت 6

بعدين استخدمها وجمعها، وبعدين زوّدها تاني فبقت 7

النتيجة:

=X بعد التنفيذ 7

2- what's the difference between compiled and interpreted languages and in this way what about Csharp?

اللغة المترجمة (Compiled):  
الكود كله بيتحوّل مرة واحدة لبرنامج جاهز قبل ما يشتغل.  
ده بيخلي التشغيل سريع.

اللغة المفسّرة (Interpreted):  
الكود بيتنفذ سطر بسطر وقت التشغيل.  
ده بيخليها أبطأ شوية.

طيب C# تبع أنهي نوع؟

C# مش مترجمة بس، ومش مفسّرة بس.

الأول الكود بيتحوّل لمرحلة وسط اسمها Intermediate Language (IL).

بعد كده، وقت التشغيل، الجهاز بيشغّله باستخدام حاجة اسمها CLR (Common Language Runtime).

3- Compare between implicit, explicit, Convert and parse casting

Implicit

الكمبيوتر بيحوّل لوحده من نوع صغير لكبير.  
زي من رقم صحيح (int) لرقم عشري (double).  
مفيش خطر، فمش محتاج تكتب حاجة.

Explicit

أنت اللي لازم تكتب التحويل بنفسك.  
لما تحوّل من نوع كبير لصغير، ممكن يحصل فقد في البيانات.  
علشان كده لازم توضحله.

Convert

طريقة جاهزة في C#، بتحوّل أي نوع لأي نوع.  
آمن شوية، وبيفهم لو القيمة فاضية.

Parse

بيحوّل من نص (string) لرقم.  
بس لو النص فيه حاجة غلط، البرنامج هيقف.

4- self study report

1) How to customize the garbage collector

الـ Garbage Collector في C# بيشتغل لوحده، لكن ممكن تتحكم فيه شوية:

لو عايز تشغّله يدوي:  
GC.Collect();

لو عايز تمنع كائن إنه يتجمع بدري:  
GC.KeepAlive(object);

ممكن تحدد تجمع جيل معين من الذاكرة:  
GC.Collect(0); يعني الجيل 0 فقط.

بس الأفضل تسيبه يشتغل لوحده، لأنه ذكي، والتدخل ممكن يبطّأ البرنامج.

2) Bitwise Operators

دي عمليات بتشتغل على مستوى البت (bit) في الأرقام:

& → AND: بيرجع 1 لو الاتنين 1

| → OR: بيرجع 1 لو واحد منهم 1

^ → XOR: بيرجع 1 لو واحد بس 1

~ → NOT: بيقلب البتات

<< → Left shift: يزق البتات للشمال (يضرب ×2)

>> → Right shift: يزق البتات لليمين (يقسم ÷2)

تستخدم في الحسابات السريعة أو التعامل مع البيانات على مستوى البت.

5- what meant by Csharp is managed code

يعني كود C# بيشتغل تحت إشراف نظام اسمه CLR،  
واللي بيتكفّل بحاجات زي:

* تنظيم الذاكرة
* تنظيف البيانات اللي ملهاش لازمة
* التعامل مع الأخطاء
* يعني ببساطة: الـ CLR بيهتم بكل حاجة تقيلة، وانت تركز في الكود بس.

6- what meant by struct is considered like class before

يعني الـ struct في C# شبه class في الشكل،  
الاتنين فيهم بيانات ودوال، وتقدر تستخدمهم بنفس الطريقة تقريبًا.  
بس الفرق بينهم في طريقة التخزين في الذاكرة.