Armen Melkumyan • 1st
Technical / Solutions Architect
11mo •

🚨 Understanding Checked and Unchecked in C# 🚨

Today, I want to break down a super important but often overlooked aspect of C# programming: the checked and unchecked contexts.

In C#, operations on integral types (like int and long) can overflow if the result exceeds the type's capacity. By default, these overflows are silent in most project settings (thanks to unchecked), which can lead to unexpected results. But don’t worry, C# gives us a way to handle these scenarios more safely!

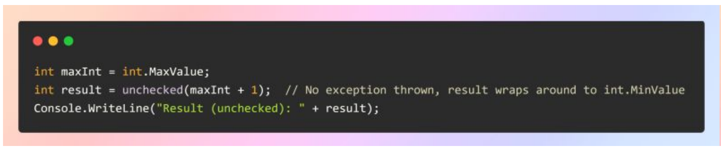
The checked keyword forces the runtime to throw an OverflowException if an overflow occurs. This is super useful when the accuracy of numerical operations is critical, like in financial calculations. 🔍💸

unchecked lets us explicitly ignore overflow, which can be useful for performance optimization when the risk is manageable. 🏎️💨

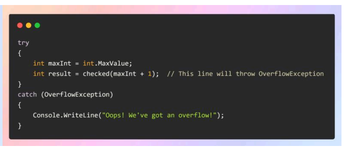
Knowing when to use checked and unchecked can help you write more robust and reliable applications, especially in areas sensitive to data accuracy.

Please look at attached screenshots for examples.

#CSharp #DotNet #Programming #CodeQuality #SoftwareDevelopment #TechTalk



```
int maxInt = int.MaxValue;
int result = unchecked(maxInt + 1);  // No exception thrown, result wraps around to int.MinValue
Console.WriteLine("Result (unchecked): " + result);
```



```
try
{
    int maxInt = int.MaxValue;
    int result = checked(maxInt + 1);  // This line will throw OverflowException
}
catch (OverflowException)
{
    Console.WriteLine("Oops! We've got an overflow!");
}
```

35 4 reposts