Day 48/60 - Understanding the lock Keyword in C#

When multiple threads access shared resources, race conditions can occur, leading to unpredictable behavior. The lock keyword helps ensure thread safety by allowing only one thread to access a critical section at a time.

Why Use lock?

Prevents multiple threads from modifying shared data at the same time.
Ensures data consistency in multithreaded applications.
Simple and easy-to-use synchronization mechanism.

Example: Using lock to Prevent Race Conditions

```
class Counter
{
 private int _count = 0;
 private readonly object _lockObj = new object();

 public void Increment()
 {
 lock (_lockObj) // Ensures only one thread modifies _count at a time
 {
 _count++;
 Console.WriteLine($"Count: {_count}");
 }
 }
}
```

When to Use lock?

When modifying shared variables in multithreaded applications.
When working with critical sections that should not be accessed simultaneously.
When ensuring data integrity in parallel execution.

Using lock properly avoids race conditions and ensures thread safety in .NET applications.

#dotnet #csharp #multithreading #threadsafe #lock