Large File Reading Efficiency

Objective
To compare the performance of StreamReader and FileStream when reading large files and understand how file access methods affect efficiency.

Overview
Reading large files efficiently is critical in data processing, logging systems, and file-based applications. The choice of API directly impacts execution time and resource usage. StreamReader and FileStream serve different purposes and are optimized for different workloads.

StreamReader
StreamReader is designed for reading text data. It reads characters instead of raw bytes and performs encoding conversions internally. While this simplifies text handling, it introduces additional overhead, making it slower for very large files or binary data.

FileStream
FileStream reads raw bytes directly from disk. It avoids character-level processing and encoding overhead, resulting in faster sequential reads. It is well-suited for large files, binary data, and performance-critical scenarios.

Performance Comparison
In practical benchmarks, FileStream consistently outperforms StreamReader as file size increases. The performance gap widens significantly for files larger than 100 MB due to reduced processing overhead.

Conclusion
FileStream is the preferred choice for large or binary files due to its speed and efficiency. StreamReader remains useful for structured text processing where readability and encoding support are more important than raw performance.