C# Garbage Collection & Memory Management Cheat Sheet

Garbage Collector Overview

- Automatically reclaims memory for unused objects.
- Managed by CLR, uses generational and compacting algorithms.
- Non-deterministic: you can't predict when it will run.

Generations in GC

- Gen 0: Short-lived objects. Collected frequently.
- Gen 1: Medium-lived objects. Acts as buffer between Gen 0 and Gen 2.
- Gen 2: Long-lived objects. Collected less often.
- Large Object Heap (LOH): Objects > 85KB. Promoted directly to Gen 2. Not compacted by default.

Stack vs Heap

- Stack: Stores value types and method call frames. Fast allocation and deallocation.
- Heap: Stores reference types. Managed by GC.
- Stack is thread-specific, Heap is shared across threads.

Finalizer (~Destructor)

- Declared with ~ClassName(). Used to clean up unmanaged resources.
- Called by GC before object memory is reclaimed.
- Slows down collection; use IDisposable + Dispose pattern instead.

IDisposable and Dispose Pattern

- Implement IDisposable to release unmanaged resources deterministically.
- Use 'using' statement to ensure Dispose() is called.
- Prefer Dispose over relying on finalizer for cleanup.

Forcing GC

- GC.Collect(): Forces a collection (not recommended for general use).
- GC.GetTotalMemory(): Returns memory usage estimate.
- GC.SuppressFinalize(): Prevents finalizer from running.

Best Practices

- Minimize allocations in hot paths.
- Use structs for small, short-lived objects.
- Pool large objects when possible.
- Avoid finalizers unless necessary. Prefer Dispose.