.NET - Practice Sheet

10 Q&A on CLR, GC, async/await, LINQ, generics, and testing

Q1. What is the CLR and how does JIT compilation work in .NET?

A: The Common Language Runtime is the virtual machine that executes .NET code. Source is compiled to CIL (Common Intermediate Language). At runtime the JIT compiler translates CIL to native machine code on first execution. The result is cached for subsequent calls.

Q2. What is the difference between .NET Framework, .NET Core, and .NET 5+?

A: .NET Framework (Windows only, ships with Windows). .NET Core (cross-platform, performant, open-source, side-by-side). .NET 5+ unified both into a single cross-platform platform with yearly releases. .NET Framework still exists for legacy Windows apps but receives only

Q3. How does the .NET Garbage Collector manage object generations?

A: Objects start in Gen0 (collected frequently, cheaply). Objects that survive Gen0 are promoted to Gen1, then Gen2. Large objects (>85 KB) go directly to the Large Object Heap and are collected with Gen2. Shorter-lived Gen0 collection pauses are the fastest.

Q4. How does async/await work in .NET and what does ConfigureAwait(false) do?

A: await suspends the method and returns a Task. When the awaited operation completes, execution resumes on the original SynchronizationContext by default (e.g., UI thread). ConfigureAwait(false) skips context capture, avoiding deadlocks in library code.

Q5. What is LINQ and what is deferred execution?

A: LINQ provides a query syntax over any IEnumerable<T>. Deferred execution means the query is not evaluated until you enumerate it (foreach, .ToList(), .Count()). This enables composing queries without materializing intermediate results.

Q6. What is the difference between a value type and a reference type in .NET?

A: Value types (struct, int, bool, DateTime) are stored inline in the stack or in the object containing them. Reference types (class) store a pointer on the stack; the object lives on the heap. Assigning a value type copies data; assigning a reference type copies the pointer.

Q7. Why are generics preferred over object-based collections in .NET?

A: List<int> stores ints directly without boxing. List<object> boxes each int into a heap-allocated object, adding GC pressure and casting overhead. Generics are also type-safe: the compiler rejects List<string>.Add(42) at compile time without a runtime cast.

Q8. What is Span<T> and when should you use it?

A: Span<T> is a stack-allocated view of a contiguous memory region (array, string, native memory). It enables sub-slice operations without allocation: span.Slice(5, 10) does not copy. Use it in hot parsing code (HTTP headers, CSV rows) to eliminate allocations.

Q9. How does the .NET dependency injection container work?

A: Register services with builder.Services.AddSingleton<IFoo, Foo>(). The container builds a ServiceProvider that resolves constructor parameters recursively. Call provider.GetRequiredService<IFoo>() or let the framework inject via constructor injection in controllers and services.

Q10. How do you write unit tests in .NET with xUnit and Moq?

A: Annotate test methods with [Fact] or [Theory] with [InlineData]. Use Mock<IService>() to create a mock. Setup behavior with mock.Setup(s => s.GetData()).Returns(fakeData). Pass mock.Object to the class under test. Assert with Assert.Equal, Assert.Throws.