Advanced C# programming interview questions with detailed answers, tailored for a candidate with 8+ years of experience:

**### 1. Explain the concept of async/await in C# and how it improves performance.**

\*\*Answer:\*\*

The `async` and `await` keywords in C# are used for asynchronous programming. They help in executing tasks without blocking the main thread, which is especially useful in applications with a user interface or I/O-bound operations. The `async` keyword marks a method as asynchronous, and the `await` keyword is used to pause the execution of the method until the awaited task completes. This approach improves the responsiveness of applications by freeing up the main thread to continue processing other tasks.

public async Task<string> GetDataAsync()

{

using (var httpClient = new HttpClient())

{

string data = await httpClient.GetStringAsync("https://example.com");

return data;

}

}

```

**### 2. What are generics in C# and how do they enhance code reusability and type safety?**

\*\*Answer:\*\*

Generics allow you to define classes, methods, and interfaces with a placeholder for the type of data they store or use. This means you can write a method or class that can operate on any data type, providing greater code reusability and type safety. Generics reduce the need for casting and boxing/unboxing, thereby improving performance.

public class GenericRepository<T> where T : class

{

private readonly List<T> \_entities = new List<T>();

public void Add(T entity)

{

\_entities.Add(entity);

}

public IEnumerable<T> GetAll()

{

return \_entities;

}

}

```

**### 3. What is Dependency Injection and how is it implemented in .NET?**

\*\*Answer:\*\*

Dependency Injection (DI) is a design pattern used to implement Inversion of Control (IoC), allowing the creation of dependent objects outside of a class and providing those objects to a class in different ways. DI in .NET is typically implemented using constructor injection, property injection, or method injection.

In .NET Core, you can configure DI in the `Startup.cs` file:

public void ConfigureServices(IServiceCollection services)

{

services.AddScoped<IMyService, MyService>();

}

```

**### 4. Can you explain the concept of LINQ and provide an example of its use?**

\*\*Answer:\*\*

Language Integrated Query (LINQ) is a set of methods and syntax in C# that provides a consistent way to query various data sources such as collections, databases, XML, etc. LINQ makes it easier to query and manipulate data using a unified syntax.

List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6 };

var evenNumbers = from num in numbers

where num % 2 == 0

select num;

foreach (var num in evenNumbers)

{

Console.WriteLine(num);

}

```

**### 5. How do you handle exceptions in C#? Provide an example with custom exceptions.**

\*\*Answer:\*\*

Exceptions in C# are handled using try-catch-finally blocks. You can also create custom exceptions by inheriting from the `Exception` class.

public class InvalidUserInputException : Exception

{

public InvalidUserInputException(string message) : base(message) { }

}

public void ProcessUserInput(string input)

{

try

{

if (string.IsNullOrEmpty(input))

{

throw new InvalidUserInputException("Input cannot be empty.");

}

Console.WriteLine("Input processed successfully.");

}

catch (InvalidUserInputException ex)

{

Console.WriteLine($"Error: {ex.Message}");

}

}

```

**### 6. What is the purpose of the `yield` keyword in C#? Provide an example.**

\*\*Answer:\*\*

The `yield` keyword is used to return each element one at a time from a method that returns `IEnumerable` or `IEnumerable<T>`. It allows for deferred execution and can improve performance when working with large collections.

```csharp

public IEnumerable<int> GetEvenNumbers(int max)

{

for (int i = 0; i <= max; i++)

{

if (i % 2 == 0)

{

yield return i;

}

}

}

foreach (var number in GetEvenNumbers(10))

{

Console.WriteLine(number);

}

```

**### 7. Explain the difference between value types and reference types in C#.**

\*\*Answer:\*\*

Value types are stored in the stack and hold the actual data. Examples include `int`, `float`, `bool`, and `struct`. Reference types are stored in the heap and hold a reference to the data. Examples include `class`, `string`, `array`, and `delegate`.

```csharp

int valueType = 10; // Value type stored in the stack

string referenceType = "Hello"; // Reference type stored in the heap

```

**### 8. What is the purpose of the `lock` statement in C#?**

\*\*Answer:\*\*

The `lock` statement in C# is used to ensure that a block of code runs to completion without interruption by other threads, which helps in maintaining thread safety when accessing shared resources.

```csharp

private readonly object \_lockObject = new object();

public void ThreadSafeMethod()

{

lock (\_lockObject)

{

// Critical section of code

}

}

```

**### 9. Describe the role of garbage collection in .NET and how you can manage it.**

\*\*Answer:\*\*

Garbage collection (GC) in .NET automatically manages the allocation and release of memory for your applications. It frees developers from manual memory management, reducing memory leaks and other related issues. You can manage GC by implementing the `IDisposable` interface and using the `Dispose` method to free unmanaged resources.

```csharp

public class ResourceHolder : IDisposable

{

private bool \_disposed = false;

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

protected virtual void Dispose(bool disposing)

{

if (!\_disposed)

{

if (disposing)

{

// Free managed resources

}

// Free unmanaged resources

\_disposed = true;

}

}

~ResourceHolder()

{

Dispose(false);

}

}

```

**### 10. What are extension methods in C# and how do you create them?**

\*\*Answer:\*\*

Extension methods allow you to add new methods to existing types without modifying the original type. They are static methods defined in static classes and the first parameter specifies the type they extend, preceded by the `this` keyword.

```csharp

public static class StringExtensions

{

public static bool IsNullOrEmpty(this string str)

{

return string.IsNullOrEmpty(str);

}

}

public void TestExtensionMethod()

{

string test = null;

bool result = test.IsNullOrEmpty(); // True

}

```

These questions and answers cover a range of advanced topics in C# and .NET, providing a solid foundation for a candidate with extensive experience to demonstrate their knowledge and expertise during an interview.