C# LinkedIn Assessment Q&A

Q1. In which of these situations are interfaces better than abstract classes?

- [ ] When you need to define an object type's characteristics, use an interface. When you need to define an object type's capabilities, use an abstract class.
- [ ] Interfaces are a legacy of older versions of C#, and are interchangeable with the newer abstract class feature.
- [x] When you need a list of capabilities and data that are classes-agnostic, use an interface. When you need a certain object type to share characteristics, use an abstract class.
- [ ] You should use both an interface and an abstract class when defining any complex object.

Q2. Which statement is true of delegates?

- [ ] Delegates are not supported in the current version of C#
- [ ] They cannot be used as callbacks.
- [ ] Only variables can be passed to delegates as parameters.
- [x] They can be chained together.

Q3. Which choice best defines C#'s asynchronous programming model?

- [ ] reactive
- [ ] inherited callback
- [x] task-based
- [ ] callback-based

Q4. How would you determine if a class has a particular attribute?

- [ ] var type = typeof(SomeType); var attribute = type.GetCustomAttribute<SomeAttribute>();
- [ ] var typeof(MyPresentationModel).Should().BeDecoratedWith<SomeAttribute>();
- [ ] Attribute.GetCustomAttribute, typeof(SubControllerActionToViewDataAttribute)

- [x] Attribute.GetCustomAttribute(typeof(ExampleController), typeof(SubControllerActionToViewDataAttribute))

Q5. What is the difference between the ref and out keywords?

- [ ] Variables passed to out specify that the parameter is an output parameter, while ref specifies that a variable may be passed to a function without being initialized.
- [ ] Variables passed to ref can be passed to a function without being initialized, while out specifies that the value is a reference value that can be changed inside the calling method.
- [x] Variables passed to out can be passed to a function without being initialized, while ref specifies that the value is a reference value that can be changed inside the calling method.
- [ ] Variables passed to ref specify that the parameter is an output parameter, while out specifies that a variable may be passed to a function without being initialized.

Q6. How could you retrieve information about a class, as well as create an instance at runtime?

- [x] reflection
- [ ] serialization
- [ ] abstraction
- [ ] dependency injection

Q7. What is this code an example of?

```
    private static object objA;
    private static object objB;
    private static void performTaskA()
    {
        lock (objB)
        {
            Thread.Sleep(1000);
            lock (objA) { }
        }
    }

    private static void PerformTaskB()
    {
        lock (objA)
        {
            lock (objB) { }
        }
    }
```

- [ ] a private class that uses multithreading
- [ ] multithread coding
- [ ] thread mismanagement
- [x] a potential deadlock

Q8. What is the difference between an anonymous type and a regular data type?

- [x] Anonymous types don't have type names
- [ ] Anonymous types can only be static
- [ ] Anonymous types can be used only in struts
- [ ] Anonymous types don't work with LINQ.

Q9. When would you use a Dictionary rather that an Array type in your application?

- [ ] when you need a jagged collection structure
- [ ] when you need to store values of the same type
- [x] when you need to store key-value pairs rather than single values
- [ ] when you need an ordered, searchable list

Q10. What is the difference between a.Equals(b) and a == b?

- [ ] The .Equals method compares reference identities while the `==` compares contents.

- [ ] The .Equals method compares primitive values while `==` compares all values.
- [x] The .Equals method compares contents while `==` compares reference identity.
- [ ] The .Equals method compares reference type while `==` compares primitive value types

Q11. Which choice best describes a deadlock situation?
- [ ] when you try to instantiate two objects at the same time in the same class or struct
- [ ] when you are trying to execute an action after a user event is registered
- [x] when simultaneous instructions are waiting on each other to finish before executing
- [ ] when you try to execute a series of events simultaneously on multiple threads

Q12. How does the async keyword work?
- [ ] It allows access to asynchronous methods in the C# API
- [ ] It allows thread pooling and synchronous processes in static classes.
- [x] It allows the await keyword to be used in a method
- [ ] It allows access to synchronous methods in the C# API

Q13. What is an object in C#?
- [ ] a class or struct, including its variables and functions
- [ ] a primitive data type that can be created only at compile time
- [ ] a value type that can be used only with an abstract class
- [x] an instance of a class or struct that includes fields, properties, and/or methods

Q14. Which code snippet declares an anonymous type named userData?
- [ ] var<<!---->T> userData = new <<!---->T> { name = "John", age = 32 };
- [x] var userData = new { name = "John", age = 32 };
- [ ] AType userData = new AType { name = "John", age = 32 };
- [ ] Anonymous<T> userData = new Anonymous<T> { name = "John", age = 32 };

Q15. What will be returned when this method is executed?
public void userInput(string charParameters) { }
- [x] nothing
- [ ] a Boolean
- [ ] a string variable
- [ ] an integer

Q16. In what order would the employee names in this example be printed to the console?

```
string[] employees = { "Joe", "Bob", "Carol", "Alice", "Will" };
IEnumerable<string> employeeQuery = from person in employees orderby person select person;
foreach(string employee in employeeQuery)
{
    Console.WriteLine(employee);
}
```

- [x] ascending
- [ ] unordered
- [ ] descending
- [ ] first in, first out

Q17. Lambda expressions are often used in tandem with which of the following?
- [ ] Namespaces
- [x] LINQ
- [ ] Type Aliasing
- [ ] Assemblies

Q18. What is the correct formatting for single line and multiline comments?
- [ ] /_/ - Single Line   /_ - Multiline
- [ ] // Multiline   /_ Single Line _/
- [ ] //* Multiline   / Single Line
- [x] // Single Line   /* Multiline */

Q19. How do you make a method in an abstract class overridable?
- [ ] Make it public
- [ ] Make it static
- [ ] Make it private
- [x] Make it virtual

Q20. How would you write code for an integer property called Age with a getter and setter?
- [ ] public int Age { get - set }
- [ ] public int Age: get set;
- [ ] public int Age (get, set);
- [x] public int Age { get; set; }

Q21. What is an abstract class?
- [ ] a class that is denoted by the class keyword (can be seen and used by any other class in the system--thus it is by default public)
- [ ] something denoted by the abstract keyword and used system wide; if you want any program to create an object of a class you use the abstract class
- [ ] a class that is denoted by the virtual keyword
- [x] a class that can be used only as base class

Q22. When using a thread pool what happens to a given thread after it finishes its task?
- [ ] The thread is destroyed and memory is freed up.
- [ ] The thread runs in loop until the next assignment.
- [ ] The thread goes inactive in the background and waits for garbage collection.
- [x] The thread returns to the pool for reuse.

Q23. Which choice represents a class that inherits behavior from a base class?
- [ ] a second base class
- [ ] a revised class
- [x] a derived class

- [ ] a parent class

Q24. What does operator overloading allow you to do?
- [ ] hide built-in operators when necessary
- [ ] add methods to be interpreted by the compiler at runtime
- [ ] define how enums and other primitive value types work within the rest of the application
- [x] define custom functionality for common operators like addition and equality

Q25. What is the main purpose of LINQ?
- [ ] to delete duplicate data
- [ ] to bind namespaces and assemblies
- [x] to query and transform data
- [ ] to connect assemblies

Q26. What is the correct syntax for a new generic list of strings named contacts?
- [ ] public List<string names> contacts = new List<string names>();
- [ ] public List(string names) contacts = new List(string names)();
- [x] var contacts = new List<string>();
- [ ] var contacts = new List(string);

Q27. What is the difference between throw exceptions and throw clauses?
- [ ] Throw clauses fire only at runtime, while throw exceptions can fire at any time.
- [x] Throw exceptions overwrite the stack trace, while throw clauses retain the stack information.
- [ ] Throw clauses overwrite the stack trace, while throw exceptions retain the stack information.
- [ ] Throw exceptions fire only at runtime, while throw clauses can fire during compile time.

Q28. When an asynchronous method is executed, the code runs but nothing happens other than a compiler warning. What is most likely causing the method to not return anything?
- [ ] The return yield statement is missing at the end of the method.
- [x] The method is missing an await keyword in its body.
- [ ] The wait keyword is missing from the end of the method.
- [ ] The yield keyword is missing from the method.

Q29. What are C# events?
- [ ] system actions that communicate directly with the compiler at runtime
- [ ] actions that execute when the code compiles, generating logs and test output
- [x] actions that generate notifications, which are sent to their registered listeners
- [ ] user-only methods that send data to the application's back end

Q30. What kind of values can arrays store?
- [ ] unordered collections of numeric values
- [ ] key-value pairs of any C# supported type
- [ ] class and struct instances
- [x] multiple variables, or collections, of the same type

Q31. Given this enumeration, how would you access the integer-type value of 'AppState.Loading'?
```
enum AppState { OffLine, Loading, Ready }
```
- [ ] string currentState = (string)AppState.Loading;
- [ ] string currentState = AppState.Loading.integralVal;
- [ ] int currentState = AppState.Loading.rawValue;
- [x] int currentState = (int)AppState.Loading;

Q32. What character would you use to start a regular expression pattern at a word boundary?
- [ ] d
- [ ] \a
- [x] \b
- [ ] \w

Q33. To conform to the following interface, which of its members need to be implemented?
```
public interface INameable
{
    string FirstName { get; set; }
    string LastName { get; }
}
```
- [x] Both the FirstName and LastName properties need to be implemented.
- [ ] Neither, they are both optional.
- [ ] Only the LastName property needs to be implemented.
- [ ] Only the FirstName property needs to be implemented.

Q34. You're dealing with multiple assemblies in your program, but are worried about memory allocation. At what point in the program life cycle are assemblies loaded into memory?
- [ ] at runtime
- [ ] at compile time
- [x] only when required
- [ ] only when programmatically loaded

Q35. What is most accurate description of a regular expression?
- [ ] A regular expression is a C# tool used to parse HTML
- [x] A regular expression is a special text string for describing a search patters.
- [ ] A regular expression allows a variable to be passed by reference.
- [ ] A regular expression allows a class to conform to the Equatable protocol.

Q36. Why would you use a class field in C#
- [ ] To define behaviours of the class
- [x] To hold information and data contained in the class object
- [ ] To communicate between classes and object
- [ ] To store the class definition value

Q37. When would you use generics in your code?
- [ ] to increase code performance
- [x] all of these answers
- [ ] when code reuse is a priority
- [ ] when type safety is important

Q38. What prints to the console when this code is executed?
```
public delegate void AuthCallback(bool validUser);
public static AuthCallback loginCallback = Login;
public static void Login()
{
   Console.WriteLine("Valid user!");
}

public static void Main(string[] args)
{
   loginCallback(true);
}
```
- [ ] Login successful...
- [ ] Valid user!
- [x] an error, because the method signature of Login doesn't match the delegate
- [ ] Login successful... Valid user!

Q39. How would you declare a sealed class named User?
- [ ] public class User {}
- [ ] abstract User {}
- [x] sealed class User {}
- [ ] private sealed class User {}

Q40. What is the difference between non-static and static classes?
- [x] non-static classes need to be initialized before use, while static classes do not
- [ ] non-static classes are accessible only from an interface while static classes are accessible from anywhere
- [ ] non-static classes need to initialize all class members at runtime, while static classes do not
- [ ] non-static classes do not need to be initialized while static classes do

Q41. Which characteristic prevents this code from compiling?
```
public int age="28"
```
- [x] type safety
- [ ] single inheritance
- [ ] dependency injection
- [ ] multiple inheritance

Q42. How would you serialize this class?
```
public class User {}
```
- [ ] Mark the User class with the `DeserializableAttribute`.
- [ ] Declare the class as `public serializable class User {}`.
- [x] Mark the User class with the `SerializableAttribute` attribute.
- [ ] Declare the class as `private serializable class User {}`.

Q43. How would you write a delegate named ResultCallback with an int parameter named responseCode?
- [ ] public delegate ResultCallback(int responseCode);
- [ ] public delegate void ResultCallback<(int) responseCode>;
- [ ] public void delegate ResultCallback\<int responseCode\>;
- [x] public delegate void ResultCallback(int responseCode);

Q44. What is the difference between a static and non-static method?
- [ ] non-static methods always need to have a void return type
- [ ] non-static methods do not have access to static member variables
- [x] static methods do not have to instantiate an instance of the class to call the method
- [ ] static methods always have to be public

Q45. What is the correct way to write an event named apiResult based on a delegate named ResultCallback?
- [ ] public void event ResultCallback apiResult;
- [ ] public event ResultCallback(() -> apiResult);
- [ ] public event void ResultCallback
- [x] public event ResultCallback apiResult;

Q46. When will the code inside finally block be executed in a try-catch statement?
- [ ] if there is an error, it won't execute at all
- [ ] between the try and catch blocks
- [x] after the try and catch blocks
- [ ] when the finally block overrides the catch block and executes in its place

Q47. What method correctly extends the string class?
- [x] public static string IsvalidName(this string i, string value) {}
- [ ] public static void IsvalidName(this string i, string value) {}
- [ ] public string IsvalidName(this string i, string value) {}
- [ ] public void IsvalidName(this string i, string value) {}

Q48. How are C# classes limited?
- [x] They do not support multiple inheritance.

- [ ] They support multiple inheritance.
- [ ] They can have only a set number of properties.
- [ ] They can have only a set number of methods.

Q49. What function do namespaces perform?
- [ ] Namespaces calculate code coverage at runtime.
- [ ] Namespaces compile application code together at compile time.
- [ ] Namespaces group code together into a single repository.
- [x] Namespaces separate code into groupings, control access, and avoid naming collisions.

Q50. What is the correct way to write a public property with a private backing field?
- [ ] A
private int _password;
pubic int Password = { get; set; }

- [ ] B
private int _password;
public int Password = _password;

- [ ] C
private int _password;
public int Password
{
  get -> _password;
  set-> _password = value;
}

- [x] D
private int _password;
public int Password
{
  get { return _password; }
  set { _password = value; }
}

Q51. What is a thread pool?
- [ ] a collection of synchronous methods created during initialization that cannot be reused
- [x] a collection of threads created during initialization that can be reused
- [ ] a collection of threads only recognized at compile time that can be reused
- [ ] a collection of asynchronous methods created at compile time that cannot be reused

Q52. When an object in C# is serialized, what is it converted to?
- [ ] XML
- [ ] JSON
- [x] byte stream
- [ ] value stream

Q53. What is a delegate
- [ ] a variable that holds a reference to a value type and its content
- [ ] a specific value type that can be used only in callback methods
- [x] a type that holds a reference to a method with a particular parameter list and return type
- [ ] a custom variable type that can be used in abstract classes

Q54. What are the four keywords associated with exception handling in C#?
- [ ] try, catch, valid, invalid
- [ ] try, valid, finally, throw
- [x] try, catch, finally, throw
- [ ] finally, throw, valid, invalid

Q55. What is the main difference between the is and as operators?
- [ ] The is operator checks instance types, while the as operator checks the inherited type.
- [ ] The is operator checks primitive data types, while the as operator checks the object type.
- [ ] The as operator checks object type, while the is operator attempts to cast an object to a specific type.
- [x] The is operator checks object type, while the as operator attempts to cast an object to a specific type.

Q56. What is the difference between finally and finalize blocks?
- [ ] The finally block is called during the execution of a try and catch block, while the finalize method is called after garbage collection.
- [x] The finally block is called after the execution of a try and catch block, while the finalize method is called just before garbage collection.
- [ ] The finalize block is called before the execution of a try and catch block, while the finally method is called just before garbage collection.
- [ ] The finalize block is called during the execution of a try and catch block, while the finally method is called after garbage collection.

Q57. Your application has a value type called username that needs to be able to accept null values, but this is generating compile-time errors. How would you fix this in code?
- [ ] Null<string> username = null;
- [x] string? username = null;
- [ ] Type<string>? username = null;
- [ ] Optional<string> username = null;

Q58. Which code snippet correctly declares a custom exception named InvalidResponse?
- [ ] struct InvalidResponse: Exception {}
- [x] class InvalidResponse: Exception {}
- [ ] public Exception InvalidResponse = new Exception ();
- [ ] public Exception InvalidResponse () -> Exception;

Q59. How would you write an enum variable called AppState with values for Offline, Loading, and Ready?
- [ ] enum AppState = [Offline, Loading, Ready]
- [ ] enum AppState {"Offline", "Loading", "Ready"}
- [ ] enum AppState = {Offline, Loading, Ready}
- [x] enum AppState {Offline, Loading, Ready}

Q60. What is the main difference between a value type and a reference type?
- [ ] A value type can be any primitive type, while reference types must be type-agnostic.
- [ ] A value type refers to another value, while a reference type refers to a value in memory.
- [x] A value type stores an actual value, while a reference type is a pointer to a value.
- [ ] A value type is available only at runtime, while a reference type is available only at compile time.

Q61. What is the difference between the `break` and `continue` keywords?
- [ ] The `break` keyword is used to break out of multiple iteration statements, while `continue` can only break out of code blocks that have single iterations.
- [x] The `break` keyword literally breaks out of a control flow statement, while `continue` ignores the rest of the control statement or iteration and starts the next one.
- [ ] The `break` keyword literally breaks out of the current control flow code and stops it dead, while `continue` keeps executing the code after an exception is thrown.
- [ ] The `break` keyword jumps out of an iteration and then proceeds with the rest of the control flow code, while `continue` stops the executing code dead.

Q62. Which code snippet correctly declares a variable named userId with a public `get` and private `set`?
- [ ] public int userID <get, set>;
- [ ] public int userID [get, private set];
- [x] public int userID { get; private set; }
- [ ] public int userID = { public get, private set };

Q63. What is true about virtual methods?
- [ ] Overriding virtual methods in a derived class is mandatory.
- [ ] Overriding virtual methods in a derived class is not possible.
- [x] Virtual methods always need a default implementation.
- [ ] Virtual methods cannot have default implementation.

Q64. What is likely to happen if you have multiple threads accessing the same resource in your program?
- [ ] resource overload
- [ ] thread jumping
- [x] deadlock and race conditions
- [ ] nothing, since this is what threading is for

Q65. How do you indicate that a string might be null?
- [ ] A string cannot be nullable.
- [x] string? myVariable
- [ ] string myVariable = null
- [ ] string(null) myVariable

Q66. Do you need to declare an out variable before you use it?
- [x] No, you can declare an out in the parameter list.
- [ ] Out variables are no longer part of C#.
- [ ] You must declare it if it is a primitive type.
- [ ] Yes.

Q67. How would you access the last two people in an array named People?
- [x] People[..^2]
- [ ] You cannot do this in C#.
- [ ] People[..^3]
- [ ] People[^2]

Q68. When can anonymous types be created?
- [x] at compile time
- [ ] after runtime
- [ ] at runtime
- [ ] after compile time

Q69. What is true about thread multitasking?
- [x] Thread multitasking allows code to be executed concurrently
- [ ] Thread multitasking allows code to be executed only when handling a user event.
- [ ] Thread multitasking blocks code from being executed simultaneously to guard memory.
- [ ] Thread multitasking adds single-threaded code blocks together.

Q70. What accessibility level does this class field have ?
private string LastName;
- [x] It can be used by other code only in the same class or struct.
- [ ] It can be used by other code in a referenced assembly.
- [ ] It can be used only by code contained in a derived class.
- [ ] It can be used by other code in the same assembly.

Q71. How would you correctly declare a jagged array called 'partyInvites' with 10 empty elements?
- [ ] string[] partyInvites = new string[10];
- [x] string[][] partyInvites = new string[10][];
- [ ] string[][] partyInvites = new string[10]();
- [ ] string <[]> partyInvites = new string <[10]>;

Q72. How could you pause a thread for three seconds?
- [ ] Thread.Pause(3000);
- [ ] Thread.Resume(-3000);
- [ ] Thread.Suspend(3000);
- [x] Thread.Sleep(3000);

Q73. What is wrong with this code?

```
void MyFunction()
{
    {
        int a = 10;
        int b = 20;
        int c = a + b;
    }
    Console.WriteLine(c);
}
```

- [ ] Variable c is never used; displaying it on the console does not count as usage.
- [ ] Variables a and b are never used.
- [ ] You cannot place code inside brackets inside another block.
- [x] Variable c no longer exists outside the block.

Q74. Which statement is True?
- [ ] All are true.
- [ ] None are true.
- [ ] string is a value type.
- [x] string is an alias for String

Q75. How would you return more than one value from a method?
- [x] Use either a tuple or an out variable.
- [ ] The only way is to use an out variable.
- [ ] The only way is to use a tuple.
- [ ] This cannot be done

Q76. Which is a valid example of a derived class?
- [ ] public class PremiumUser sub User {}
- [x] public class PremiumUser: User {}
- [ ] public class PremiumUser -> sub User {}
- [ ] public class User: PremiumUser {}

Q77. What is the correct way to call a static method named DebugString from a static class called InputManager?
- [ ] static InputManager.DebugString();
- [ ] InputManager().DebugString;
- [ ] new InputManager().DebugString();
- [x] InputManager.DebugString();

Q78. What values can be assigned to this variable?
public string? nickname
- [ ] null
- [ ] String values
- [x] String values or null
- [ ] String values with more than one character

Q79. What is a destructor?
- [ ] a special called automatically whenever an object is created or updated
- [ ] an implicit method called automatically when thread pools are processed concurrently
- [ ] an explicit method called automatically when the compiler starts running
- [x] a special method called automatically whenever an object is deleted or destroyed

Q80. Which code snippet correctly declares a CustomInt type alias of type Int32?
- [ ] typealias CustomInt = System.Int32;
- [ ] var<T> CustomInt = Int32;
- [x] using CustomInt = System.Int32;
- [ ] type CustomInt = System<Int32>;