CODER FOUNDRY

C# Interview Questions

ADAPTED FROM "C# INTERVIEW QUESTIONS," TECHKATAK.COM, 23 JULY 2015

1. **What is C#?**
   C# is an object-oriented, strongly typed and managed language compiled by the .NET framework to generate the Microsoft Intermediate Language for program execution.

2. **What are the types of comments used in C#?**
   Single line, Multi-line, XML document

3. **Can multiple catch blocks be executed?**
   No. Once the proper catch code is executed, control is transferred to the finally block, after which the code that follows the finally block is executed. This does not mean you can't USE multiple catch blocks. You may catch as many specific exceptions as you like, but only one will be executed on a given pass through the code.

4. **What is the difference between public, static and void?**
   Public declared variables or methods are accessible anywhere in the application. Static declared variables or methods are globally accessible without creating an instance of the class. The compiler stores the address of the method as the entry point and uses this information to begin execution before any objects are created. Void is a type modifier that identifies a method as having no return value.

5. **What is an Object?**
   An object is an instance of a class through which we access the methods of that class. The "new" keyword is used to create an object. A class that creates an object in memory will contain the information about the methods, variables and behavior of that class.

6. **What is a constructor?**
   A constructor is a member function in a class that has the same name as its class. The constructor is automatically invoked whenever an object (instance) of the class is created. It constructs the values of data members while initializing the class.

7. **What are Jagged Arrays?**
   An array which has elements of type Array is called a jagged array. The elements can be of different dimensions and sizes. A jagged array may also be called an Array of arrays.

8. **What is the difference between ref & out parameters?**
   An argument passed as a *ref* must be initialized before passing it to the method, whereas the *out* parameter need not be initialized before passing to a method.

9. **What the using statement in C#?**
   The using statement is used to obtain a resource, use it in a block of code, and then automatically dispose of it when the execution of the block is complete. At compile time, any using statements are converted to try + finally blocks.

   Using statements may be at the beginning of a code file, or as a statement block with open and close braces.

10. **What is serialization?**
    When we want to transport an object through a network we have to convert the object into a stream of bytes. The process of converting an object to a stream of bytes is called Serialization. For an object to be serializable, it should inherit the *ISerialize* interface. De-serialization is the reverse process of creating an object from a stream of bytes.

11. **Can "this" be used within a static method?**
    No. The "this" keyword represents the current object, or class instance. As such, it is not static and therefore cannot be used in a static method.

12. **What is difference between constant and read-only variables?**
    Constant variables are declared and initialized at compile time. The value cannot be changed after initialization. Read-only variables are initialized only from the constructor of the class. Therefore, read-only is used when we want to assign the value at run time rather than compile time.

13. **What is an interface in C#?**
    An interface is contract that can contain methods, properties, events, indexers, or any combination of those four member types. The contract specifies that any class implementing the interface must have those include implementations of the specified members.

    An interface cannot contain constants, fields, operators, instance constructors, destructors, or types. Interface members are automatically public, and they cannot include any access modifiers. Members also cannot be static.

    There are no method or property accessor bodies in an interface. C# classes may implement multiple interfaces, and must implement all methods and properties declared in those interfaces.

14. **What are value types and reference types?**
    Value types are stored in the Stack and contain the actual value of the data represented. Examples of value types in C# are: int, enum, byte, decimal, double, float, long, bool.

    Reference types are stored on the Heap and contain references (memory addresses) that point to the location of the data represented by the variable. Examples of reference types in C# are: string, class, interface, object.

15. **What are sealed classes in C#?**
    The sealed modifier is used to prevent derivation from a class, that is, you cannot inherit from a sealed class.

16. **What is method overloading?**
    Method overloading is the creation of multiple methods with the same name but unique signatures in

the same class. When we compile, the compiler uses overload resolution to determine the specific method to be invoked.

17. **What is the difference between an Array and an ArrayList?**
    An array contains items of the same type only and has a fixed size. An ArrayList is similar to an array, but does not have a fixed size.

18. **Can a private virtual method be overridden?**
    No. Such a method is would not be accessible outside of the class in which it is declared. In fact, a private virtual method will result in a compile-time error.

19. **Describe the accessibility of a variable declared as "protected internal"?**
    Such a variable is accessible within the same assembly and also from any classes that are derived from the class (including those in other assemblies) in which the variable is declared.

20. **What are the differences between System.String and System.Text.StringBuilder classes?**
    System.String is immutable, that is, an object of this type cannot be changed. When we modify the value of a string variable, a new memory space is allocated to store the new value and the previous memory allocation flagged for release. System.StringBuilder provides the creation of a mutable string, where a variety of operations may be performed without allocating additional memory spaces for the modified string.

21. **What's the difference between the System.Array.CopyTo() and System.Array.Clone()?**
    The Clone() method creates a new array object containing all of the elements in the original array. The CopyTo() method copies all of the elements of the existing array into another existing array. Both methods perform a shallow copy.

22. **How can we sort the elements of the array in descending order?**
    Use the Sort() method followed by the Reverse() method.

23. **How do you catch an exception?**
    To catch an exception, we use try catch blocks. Place the block of code you want to execute inside a try block. Then create the catch block with a parameter of type system.Exception. The parameter can be omitted from catch statement if desired.

24. **What's the difference between an interface and abstract class?**
    An interface is not a class, and contains member declarations only. Abstract classes can have method declarations as well as implementations. In an interface, member accessors are always public, while an abstract class may have private methods.

25. **What is the difference between the Finalize() and Dispose() methods?**
    Dispose() is called when we want an object to release any unmanaged resources associated with it. Finalize() is used for the same purpose, but does not assure the garbage collection of an object.

26. **What are circular references?**
    A circular reference is a condition in which two or more resources are dependent on each other, causing a lock condition and making both resources unusable.

27. **What are generics in C#?**
    Generics decrease code redundancy while increasing type safety and performance through reusable code classes with parameterized types. Using generics, we can create a wide variety of collection classes. To

create a generic collection, the System.Collections.Generic namespace should be used instead of classes such as ArrayList in the System.Collections namespace.

28. **What is an object pool in .NET?**
    An object pool is a container that holds objects that are ready to be used. It tracks the object that is currently in use and maintains a count of the total number of objects in the pool. This reduces the overhead of creating and re-creating objects.

29. **What are some of the most common types of exceptions in .NET?**
    ArgumentException, ArgumentNullException, ArgumentOutOfRangeException, ArithmeticException, DivideByZeroException,OverflowException, IndexOutOfRangeException, InvalidCastException, InvalidOperationException, IOEndOfStreamException, NullReferenceException, OutOfMemoryException, StackOverflowException, etc.

30. **What are Custom Exceptions?**
    Occasionally it is necessary to handle certain errors based upon specific, per user, requirements. Custom exceptions may be written, extending the Exception class, to define such error conditions. They may be thrown and caught just as predefined exceptions.

31. **What are delegates?**
    Delegates are type-safe function pointers. We can use delegates to pass generic type-safe functions as arguments to other functions.

32. **How do you inherit a class into other class in C#**
    The colon is the inheritance operator in C#. It is placed after the name of the child class and is followed by the name of the parent class.

33. **What is the base class in .NET from which all other classes are derived?**
    System.Object

34. **What is the difference between method overriding and method overloading?**
    In method overriding, we change the method definition in the derived class that changes the method behavior. Method overloading is the creating of a method with the same name but different signatures within the same class.

35. **What are the different ways a method can be overloaded?**
    Methods can be overloaded using different data types for parameters, different order of parameters, and different number of parameters.

36. **Why can't you specify the accessibility modifier for methods inside an interface?**
    All method declarations in an interface are methods that must be implemented in the derived class. Consequently, they are all public.

37. **How can we set up a class to be inherited, but prevent a class method from being overridden?**
    Declare the class as public and declare the method as sealed to prevent it from being overridden.

38. **What happens if inherited interfaces have conflicting method names?**
    Interfaces only specify a contract for method signatures, and as such, are not concerned with implementation details. If a class implements two interfaces share the same method signature declaration, the compiler will simply ensure that the shared method signature must be implemented in the class.

39. **What is the difference between a Struct and a Class?**
    Structs are value-type variables, while classes are reference types. Structs are therefore stored on the stack, requiring additional overhead but offering faster retrieval. Unlike classes, structs cannot be inherited.

40. **How do you use nullable types in .NET?**
    Any value type can be made nullable by declaring the type with a question mark after the data type (i.e. int? x = null;). The nullable keyword can also be used. Making a type nullable allows the variable to contain a null value in addition to the values associated with its base type. All reference types are automatically nullable.

41. **How can we create an array with non-default values?**
    The syntax for instantiating a new array with specified values is new T[] { new T_1(), new T_2(), new T_3()…, new T_n() }; Alternatively, we can create the array using Enumerable.Repeat.

42. **What is the difference between the *is* and *as* operators in c#?**
    The "is" operator is used to check the compatibility of an object with a given type and return the result as a Boolean value. The "as" operator is used to cast an object to a different type or class.

43. **What's a multicast delegate?**
    A delegate having multiple handlers assigned to it is called a multicast delegate. Each handler is assigned to a specific method.

44. **What are indexers in C#?**
    Indexers are known as smart arrays in C#. They allow the instances of a class to be indexed in the same way as array. (i.e. public int this[int index] // indexer declaration)

45. **What is the difference between "throw" and "throw ex" in C#?**
    The "throw" statement preserves the original error stack, while the "throw ex" statement contains only the stack trace from the throw point. Developers should always use "throw", as it provides more accurate error information.

46. **What are C# attributes and what is their significance?**
    C# provides developers a way to define declarative tags (attributes) on certain entities, such as classes, methods, etc. An attribute's information can be retrieved at runtime using Reflection.

47. **How do you implement a singleton design pattern in C#?**
    In singleton pattern, a class can only have one instance and has a global access point.

```
public sealed class Singleton
{
    private static readonly Singleton _instance = new Singleton();
}
```

48. **What is the difference between directcast and ctype?**
    Directcast is used to convert the type of an object that requires the run-time type to be the same as the specified type in DirectCast. Ctype is used for conversion where the conversion is defined between the expression and the type.

49. **Is C# code managed or unmanaged?**
    C# is managed code because Common language runtime can compile C# code to Intermediate language.

50. **What is a static constructor**?
    A static constructor is used to initialize static data members as soon as the class is referenced the first time, while an instance constructor is used to create an instance of the class. A static constructor does not take access modifiers or have parameters, and it cannot access any non-static data member of any class.

51. **What is the use of the Monitor class in C#?**
    The Monitor class controls access to objects by granting a lock for an object to a single thread. Object locks provide the ability to restrict access to a block of code, commonly called a critical section. While a thread owns the lock for an object, no other thread can acquire that lock. You can also use the Monitor class to ensure that no other thread is allowed to access a section of application code being executed by the lock owner, unless the other thread is executing the code using a different locked object.

    For more visit: http://msdn2.microsoft.com/en-us/library/system.threading.monitor.aspx

52. **What is the lock statement in C# use for?**
    The lock statement ensures that one thread does not enter a critical section of code while another thread is in the critical section. If another thread attempts to enter a section of locked code, it will wait in a blocked state until the object is released.

53. **How do you loop through all rows of a DataTable?**
    Use a foreach loop.

54. **What is an Array?**
    An array is a collection of related instances of either value or reference types. The number of dimensions and size of the array are fixed at instantiation. C# supports single-dimensional, multi-dimensional, and jagged arrays.

    Single Dimensional Array: sometimes called a vector array consisting of a single row.
    Multi-Dimensional Array: rectangular, consisting of rows and columns.
    Jagged Array: Consisting of rows and columns, but irregular in shape (i.e. row 1 has 3 columns, row 2 has 5 columns, etc).

55. **What is an ArrayList?**
    An ArrayList is a dynamic array. Elements may be added and removed at runtime. Elements are not automatically sorted.

56. **What is BitArray?**
    A BitArray is a collection of bit values, 1 or 0 where 1 is true and 0 is false.

57. **What is a HashTable?**
    A Hashtable is a collection of key-value pairs. HashTable entires are instances of the DictionaryEntry type. It implements IDictionary, ISerilizable, IDeserializable and callback interfaces.

58. **What is a Queue?**
    A collection that abstracts a FIFO (First In First Out) data structure. Initial capacity is 32 elements. It is ideal for messaging components.

59. **What is a Stack?**
    A collection that abstracts a LIFO (Last In First Out) data structure, Initial capacity is 32 elements.

60. **What is a SortedList?**
    A collection that is a combination of key/value entries and an ArrayList. The collection is sorted by key.

61. **What is a collection?**
    A collection is a container for instances of other classes. All collection classes implement the ICollection interface, which implements the IEnumerable interface.

62. **What is reflection?**
    Reflection is the ability to find the information about type values contained in an assembly at runtime.

63. **What class is underneath the SortedList class?**
    A sorted HashTable.

64. **What is the use of CommandBehavior.CloseConnection?**
    Pass it with the ExecuteReader method of the Command object to define additional behavior. For example:

    ```
    reader = cmd.ExecuteReader(CommandBehavior.CloseConnection);
    ```

    will make sure that when reader.Close() is called, the associated connection object will also be closed.

65. **What does the term immutable mean?**
    The data value may not be changed. The variable value may be changed, but the original immutable data value would have been discarded and a new data value created in a new memory location.

66. **Are private class-level variables inherited?**
    Yes, but they are not accessible. Although they are not visible or accessible via the class interface, they are inherited.

67. **What are the various types of errors in C#?**
    Configuration Errors, Parser Errors, Compilation Errors, Run-Time Errors

68. **How would you write a single line of code to read the entire content of a text file.**
    ```
    string strContent = System.IO.File.ReadAllText(Server.MapPath("~/MyTextFile.txt"));
    ```

69. **Write a single line of code to create a text file and write contents into it.**
    ```
    System.IO.File.WriteAllText(@"c:\MyTextFile.txt", "MyContents");
    ```

70. **What is garbage collection?**
    Garbage collection allows the computer to detect when an object can no longer be accessed. It then releases the memory used by that object (as well as calling a clean-up routine, called a “finalizer,” which is written by the user). Some garbage collectors, like the one used by .NET, compact memory and therefore decrease your program’s working set.

71. **What is a nested class?**
    A class defined within another class.

72. **What is the difference between a constructor and a destructor?**
    The constructor is the first method that is run when an instance of a type is created. It is used to initialize class and structure data before use. Constructors never return a value and may be overridden to provide

custom initialization functionality. Constructors provides a way to set default values for data or perform other necessary functions before the object becomes available for use.

Destructors are called just before an object is destroyed and can be used to run clean-up code. You cannot control when a destructor is called.

73. When do you absolutely have to declare a class as abstract?

a. When the class itself is inherited from an abstract class, but not all base abstract methods have been overridden.

b. When at least one of the methods in the class is abstract.

74. What is UDP and how does it work?

The User Datagram Protocol (UDP) provides an unreliable datagram-oriented protocol on top of IP. The delivery and order of datagrams are not guaranteed. A UDP application does not have to connect explicitly to another. Datagrams are simply sent or received.

75. What are the various access modifiers in C# ?

a. *Public* – can be accessed from any code in the project.

b. *Private* – can only be accessed by code within the containing class.

c. *Protected* – can be accessed by any method in the inherited classes and any method within the same class. The protected access modifier cannot be applied to classes and interfaces. Methods and fields in an interface cannot be declared protected.

d. *Internal* – access is restricted to classes within the current project assembly.

e. *Protected Internal* – access is restricted to classes within the current project assembly and to types derived from the containing class in any assembly.

76. What is Constructor Overloading in C#?

Any number of constructors can be created for the same class, provided the signatures for each constructor are different.

77. What is Data Encapsulation?

Data encapsulation, also known as data hiding, is the mechanism whereby the implementation details of a class are kept hidden from the user. The user can only perform a restricted set of operations on the hidden members of the class by executing special functions commonly called *methods*

78. What is Inheritance?

In object-oriented programming (OOP), inheritance is a way to reuse the base code of existing objects. In inheritance, there will always be at least two classes: a base (parent) class and derived (child) classes. A derived (child) class inherits attributes and methods from a base (parent) class.

79. What is Polymorphism in C#?

From the Greek root meaning *having many forms*. The ability of a programming language to process objects in different ways depending on their data type or class. There are two types of polymorphism.

a. *Compile time polymorphism*. One example is Overloading

b. *Runtime polymorphism*. One example is Overriding

80. Explain the use of the virtual keyword in C#?

To give permission to a derived class to override a method in a base class.