**1. What is C#?**

C# is an object oriented, type safe and managed language that is compiled by .Net framework to generate Microsoft Intermediate Language.

**2. Can multiple catch blocks be executed?**

No, Multiple catch blocks can't be executed. Once the proper catch code executed, the control is transferred to the finally block and then the code that follows the finally block gets executed.

**3. What is Jagged Arrays?**

The array which has elements of type array is called jagged array. The elements can be of different dimensions and sizes. We can also call jagged array as Array of arrays.

**4. What is the difference between ref & out parameters?**

An argument passed as ref must be initialized before passing to the method whereas out parameter needs not to be initialized before passing to a method.

**5. What is serialization?**

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

**6. What is difference between constants and read-only?**

Constant variables are declared and initialized at compile time. The value can't be changed afterwards. Read only is used only when we want to assign the value at run time.

**7. What is an interface class?**

Interface is an abstract class which has only public abstract methods and the methods only have the declaration and not the definition. These abstract methods must be implemented in the inherited classes.

**8. What are value types and reference types?**

Value types are stored in the Stack whereas reference types stored on heap.Value types:

int, enum , byte, decimal, double, float, long

Reference Types:

string , class, interface, object

**9. Can a private virtual method be overridden?**

No, because they are not accessible outside the class.

**10. Describe the accessibility modifier "protected internal".**

Protected Internal variables/methods are accessible within the same assembly and also from the classes that are derived from this parent class.

**11. What are the differences between System.String and System.Text.StringBuilder classes?**

System.String is immutable. When we modify the value of a string variable then a new memory is allocated to the new value and the previous memory allocation released. System.StringBuilder was designed to have concept of a mutable string where a variety of operations can be performed without allocation separate memory location for the modified string.

**12. What's the difference between the System.Array.CopyTo() and System.Array.Clone() ?**

Using Clone() method, we creates a new array object containing all the elements in the original array and using CopyTo() method, all the elements of existing array copies into another existing array. Both the methods perform a shallow copy.

**13. What's the difference between an interface and abstract class?**

Interfaces have all the methods having only declaration but no definition. In an abstract class, we can have some concrete methods. In an interface class, all the methods are public. An abstract class may have private methods.

**14. What is the difference between Finalize() and Dispose() methods?**

Dispose() is called when we want for an object to release any unmanaged resources with them. On the other hand Finalize() is used for the same purpose but it doesn't assure the garbage collection of an object.

**28. What are generics in C#.NET?**

Generics are used to make reusable code classes to decrease the code redundancy, increase type safety and performance. Using generics, we can create collection classes. To create generic collection, System.Collections.Generic namespace should be used instead of classes such as ArrayList in the System.Collections namespace. Generics promotes the usage of parameterized types.

**29. What is an object pool in .NET?**

An object pool is a container having objects ready to be used. It tracks the object that is currently in use, total number of objects in the pool. This reduces the overhead of creating and re-creating objects.

**30. List down the commonly used types of exceptions in .Net?**

ArgumentException, ArgumentNullException , ArgumentOutOfRangeException, ArithmeticException, DivideByZeroException ,OverflowException , IndexOutOfRangeException ,InvalidCastException ,InvalidOperationException , IOEndOfStreamException , NullReferenceException , OutOfMemoryException , StackOverflowException etc.

**32. What are delegates?**

Delegates are same are function pointers in C++ but the only difference is that they are type safe unlike function pointers. Delegates are required because they can be used to write much more generic type safe functions.

**38. How can we set class to be inherited, but prevent the method from being over-ridden?**

Declare the class as public and make the method sealed to prevent it from being overridden.

**40. What is the difference between a Struct and a Class?**

Structs are value-type variables and classes are reference types. Structs stored on the stack, causes additional overhead but faster retrieval. Structs cannot be inherited.

**43. What is difference between is and as operators in c#?**

"is" operator is used to check the compatibility of an object with a given type and it returns the result as Boolean.

"as" operator is used for casting of object to a type or a class.

**44. What's a multicast delegate?**

A delegate having multiple handlers assigned to it is called multicast delegate. Each handler is assigned to a method.

**49. 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.

**Q #5) What are the different types of classes in C#?**

**Ans: The different types of class in C# are:**

* **Partial class** – Allows its members to be divided or shared with multiple .cs files. It is denoted by the keyword *Partial.*
* **Sealed class** – It is a class which cannot be inherited. To access the members of a sealed class, we need to create the object of the class.  It is denoted by the keyword *Sealed*.
* **Abstract class** – It is a class whose object cannot be instantiated. The class can only be inherited. It should contain at least one method.  It is denoted by the keyword *abstract.*
* **Static class** – It is a class which does not allow inheritance. The members of the class are also static.  It is denoted by the keyword *static*. This keyword tells the compiler to check for any accidental instances of the static class.

**Q #6) Explain Code compilation in C#.**

**Ans: There are four steps in code compilation which include:**

* Compiling the source code into Managed code by C# compiler.
* Combining the newly created code into assemblies.
* Loading the Common Language Runtime(CLR).
* Executing the assembly by CLR.

**Q #7) What are the differences between a Class and a Struct?**

**Ans: Given below are the differences between a Class and a Struct:**

| **Class** | **Struct** |
| --- | --- |
| Supports Inheritance | Does not support Inheritance |
| Class is Pass by reference (reference type) | Struct is Pass by Copy (Value type) |
| Members are private by default | Members are public by default |
| Good for larger complex objects | Good for Small isolated models |
| Can use waste collector for memory management | Cannot use Garbage collector and hence no Memory management |

**Q #17) What is an Abstract Class?**

**Ans:** An **Abstract class** is a class which is denoted by abstract keyword and can be used only as a Base class. An Abstract class should always be inherited. An instance of the class itself cannot be created. If we do not want any program to create an object of a class, then such classes can be made abstract.

Any method in the abstract class does not have implementations in the same class. But they must be implemented in the child class.

**Q #20) What is the difference between finally and finalize block?**

**Ans:*finally* block** is called after the execution of try and catch block. It is used for exception handling. Regardless of whether an exception is caught or not, this block of code will be executed. Usually, this block will have clean-up code.

finalize method is called just before garbage collection. It is used to perform clean up operations of Unmanaged code. It is automatically called when a given instance is not subsequently called.

**Q #27) What are Regular expressions? Search a string using regular expressions?**

**Ans: Regular expression** is a template to match a set of input. The pattern can consist of operators, constructs or character literals. Regex is used for string parsing and replacing the character string.

**Q #36) What are Synchronous and Asynchronous operations?**

**Ans:** Synchronization is a way to create a thread-safe code where only one thread can access the resource at any given time.

Asynchronous call waits for the method to complete before continuing with the program flow. Synchronous programming badly affects the UI operations, when the user tries to perform time-consuming operations since only one thread will be used.

In Asynchronous operation, the method call will immediately return so that the program can perform other operations while the called method completes its work in certain situations.

In C#, Async and Await keywords are used to achieve asynchronous programming. Look at **Question 43** for more details on synchronous programming.

**Q #37) What is Reflection in C#?**

**Ans:** Reflection is the ability of a code to access the metadata of the assembly during runtime. A program reflects upon itself and uses the metadata to inform the user or modify its behavior. Metadata refers to information about objects, methods.

**Q #43) What are Async and Await?**

**Ans:**Async and Await keywords are used to create asynchronous methods in C.

Asynchronous programming means that the process runs independently of main or other processes.

**What is Use of Private Constructor in C# ?**

* Stop object creation of a class
* Use in Singleton class
* Stop a class to be inherited

Create a Thread Safe Singleton class?

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace SingletonDesignPattern

{

//Singleton class that returns only one object.

public class SingleInstanceClass

{

//create a mutex object to lock shared statement in GetInstance

//method

private static readonly object mutex = new object();

private static SingleInstanceClass instance = null;

private SingleInstanceClass()

{

}

public static SingleInstanceClass GetInstance()

{

if (instance == null)

{

lock (mutex)

{

if (instance == null)

{

instance = new SingleInstanceClass();

}

}

}

return instance;

}

public void Display(){

Console.WriteLine("Singleton class");

}

}

//CLIENT CLODE

class Program

{

static void Main(string[] args)

{

//Create 2 objects of the singleton class

SingleInstanceClass obj1 = SingleInstanceClass.GetInstance();

obj1.Display();

SingleInstanceClass obj2 = SingleInstanceClass.GetInstance();

obj2.Display();

}

}

}

Write a custom exception example

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

namespace CustomException

{

//custom exception class or user defined Exception class.

class UnwantedCharException : Exception

{

//create custruction and pass a error message to

//base class i.e.System exception class

public UnwantedCharException(String msg) : base(msg) { }

}

//Test Program

class Program

{

static void Main(string[] args)

{

try

{

string s = GetInputString();

//process further if no exception.

Console.WriteLine(s);

}

catch (UnwantedCharException ex)

{

//Get custom exception

Console.WriteLine(ex.Message);

}

}

static string GetInputString()

{

string s = Console.ReadLine();

//User String is not alphnumeric, throw Exception.

if (!IsAlphaNumeric(s))

{

throw new UnwantedCharException("Only Alphanumeric string is allowed");

}

return s;

}

public static bool IsAlphaNumeric(String strToCheck)

{

Regex objAlphaNumericPattern = new Regex("[^a-zA-Z0-9]");

return !objAlphaNumericPattern.IsMatch(strToCheck);

}

}

}

**What is method hiding in C# inheritance?**

Method hiding occurs in inheritance relationship when base class and derived class both have a method with same name. When we create the object of derived class it will hide the base class method and will call its own method and this is **called method hiding or name hiding in C# inheritance.**

**How to implement two interface with same method in C#?**

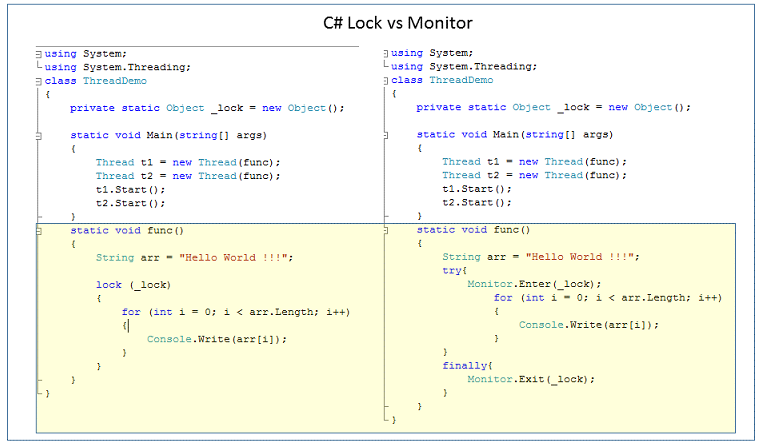
If we have two interface with same method name then a class need to implement interface explicitly in a program.

**Lock Vs Monitor?**

**Answer:** Lock Vs Monitor in C# multithreading: Difference between monitor and lock in C# is that **lock** internally wraps the Enter and Exit methods in a **try**…**finally** block with exception handling. Whereas for [Monitor class in C#](https://www.interviewsansar.com/2016/06/19/what-is-monitor-class-object-in-csharp-threading/), we use try and finally block explicitly to release lock properly.

Lock = Monitor + try finally.

Below is the source code of lock vs Monitor class used in below C# thread example.



**Secondly,** **Monitor class has extra option that is signalling option**, that is used to communicate/signal to other threads using wait(), pulse() and pulseAll() methods.

C# Monitor.wait(): A thread wait for other threads to notify.

Monitor.pulse(): A thread notify to another thread.

Monitor.pulseAll(): A thread notifies all other threads within a process.

Another important C# thread interview question is

**if lock handles try and finally block internally and gives clean and readable code, then why to use Monitor class**?

In fact, this is clear that lock gives clean code as mentioned in above example and we should not use monitor in above case. But, if in any program, we are using signalling options i.e. Monitor wait pulse C# methods and need to synchronize the critical section, we should go for Monitor class in C# multithreading.