**.NET Framework:**

**1. what is .net framework?**

.NET Framework is a software development framework for building and running applications on Windows..NET Framework is part of the .NET platform, a collection of technologies for building apps for Linux, macOS, Windows, iOS, Android, and more.

**2.components of .net framework?**

The two major components of .NET Framework are the Common Language Runtime and the .NET Framework Class Library.

The Common Language Runtime (CLR) is the execution engine that handles running applications. It provides services like thread management, garbage collection, type-safety, exception handling, and more.

The Class Library provides a set of APIs and types for common functionality. It provides types for strings, dates, numbers, etc. The Class Library includes APIs for reading and writing files, connecting to databases, drawing, and more.

**3.what is CLS and CTS?**

CTS and CLS are parts of .NET CLR and are responsible for type safety within the code. Both allow cross-language communication and type safety.

CLS stands for Common Language Specification and it is a subset of CTS. It defines a set of rules and restrictions that every language must follow which runs under the .NET framework. In simple words, CLS enables cross-language integration or Interoperability.

Common Type System (CTS) describes the datatypes that can be used by managed code. CTS defines how these types are declared, used and managed in the runtime. It facilitates cross-language integration, type safety, and high-performance code execution. The rules defined in CTS can be used to define your own classes and values.

Example- https://www.c-sharpcorner.com/blogs/what-are-cts-and-cls-in-net

**4.what is managed code?**

 managed code is just that: code whose execution is managed by a runtime. In this case, the runtime in question is called the **Common Language Runtime** or CLR, regardless of the implementation (for example, .NET Framework, or .NET Core/.NET 5+). CLR is in charge of taking the managed code

**5.what is assembly?**

An assembly is **a collection of types and resources that are built to work together and form a logical unit of functionality**. Assemblies take the form of executable (.exe) or dynamic link library (. dll) files, and are the building blocks of .NET applications.

**6. which are types of assembly?**

There are three types of assemblies:

* Private assemblies
* Shared assemblies
* Satellite assemblies

More details- https://tutorialslink.com/Articles/Types-of-assemblies-in-Net/1546#:~:text=There%20are%20three%20types%20of,Satellite%20assemblies

**7. what is manifest?**

In the .NET Framework, an assembly manifest is **a text file containing metadata about the code within a CLI assembly**. It describes the relationship and dependencies of the components in the assembly, versioning information, scope information and the security permissions required by the assembly.

**8.what is DLL hell?**

DLL Hell is **a term for the complications that arise when one works with dynamic-link libraries (DLLs) used with Microsoft Windows operating systems**, particularly legacy 16-bit editions, which all run in a single memory space.

**9.what is GAC?**

The Global Assembly Cache (GAC) is a folder in Windows directory to store the .NET assemblies that are specifically designated to be shared by all applications executed on a system. Assemblies can be shared among multiple applications on the machine by registering them in global Assembly cache (GAC). GAC is a machine wide a local cache of assemblies maintained by the .NET Framework.

**10. what is reflection?**

.NET Framework's Reflection API allows you to fetch Type (Assembly) information at runtime or programmatically. We can also implement late binding using .NET Reflection. At runtime, Reflection uses the PE file to read the metadata about an assembly. Reflection enables you to use code that was not available at compile time. .NET Reflection allows application to collect information about itself and also manipulate on itself. It can be used effectively to find all the types in an assembly and/or dynamically invoke methods in an assembly. This includes information about the type, properties, methods and events of an object. With reflection, we can dynamically create an instance of a type, bind the type to an existing object, or get the type from an existing object and invoke its methods or access its fields and properties. We can also access attributes using Reflection.

**11.what is JIT? Types of JIT**

The Just-In-Time (JIT) compiler is a component of the runtime environment that improves the performance of Java™ applications by compiling bytecodes to native machine code at run time.

**There are 3 Types of JIT**  
**1. Pre-JIT:-** Pre-JIT complies complete source code into native code in a single compilation cycle. This is done at the time of deployment of the application.  
**2. Econo-JIT:-** Econo-JIT complies only those methods that are called at runtime. However, these complied methods are removed when they are not required.  
**3. Normal-JIT:-** Normal-JIT complies only those methods that are called at runtime. These methods are compiled the first time they are called, and then they are stored in cache. When the same methods are called again, the complied code from cache is used for execution.

**C# :**

**1.diff between value types and ref types?**

**Variables of reference types store references to their data (objects), while variables of value types directly contain their data**.

With reference types, two variables can reference the same object; therefore, operations on one variable can affect the object referenced by the other variable. With value types, each variable has its own copy of the data, and it is not possible for operations on one variable to affect the other.

**All fundamental data types, Boolean, Date, structs, and enums are examples of value types.** **Examples of reference types include: strings, arrays, objects of classes, etc**.

**2. C# ref types**

The reference data type in C# does not have the actual data stored in a variable, but they contain a reference to the variables.

In C#, the following are the built-in reference types −

**Object Type**

The Object Type is the ultimate base class for all data types in C# Common Type System (CTS). The object types can be assigned values of any other types, value types, reference types, predefined or user-defined types.

Example-

object ob;

ob = 250; // boxing

**Dynamic Type**

Store any type of value in the dynamic data type variable. Type checking for these types of variables takes place at run-time.

Example-

dynamic d = 100;

**String Type**

The String Type allows you to assign any string values to a variable. The string type is an alias for the System. String class. It is derived from object type.

Example-

String val = "Welcome!";

**3. diff between out and ref**

The **out**is a keyword in C# which is used for the passing the arguments to methods as a reference type. It is generally used when a method returns multiple values. The out parameter does not pass the property.

The ref is a keyword in C# which is used for the passing the arguments by a reference. Or we can say that if any changes made in this argument in the method will reflect in that variable when the control return to the calling method. The ref parameter does not pass the property.

**4.what do you know about params?**

Params is an important keyword in C#. It is used as a parameter which can take the **variable number of arguments**.

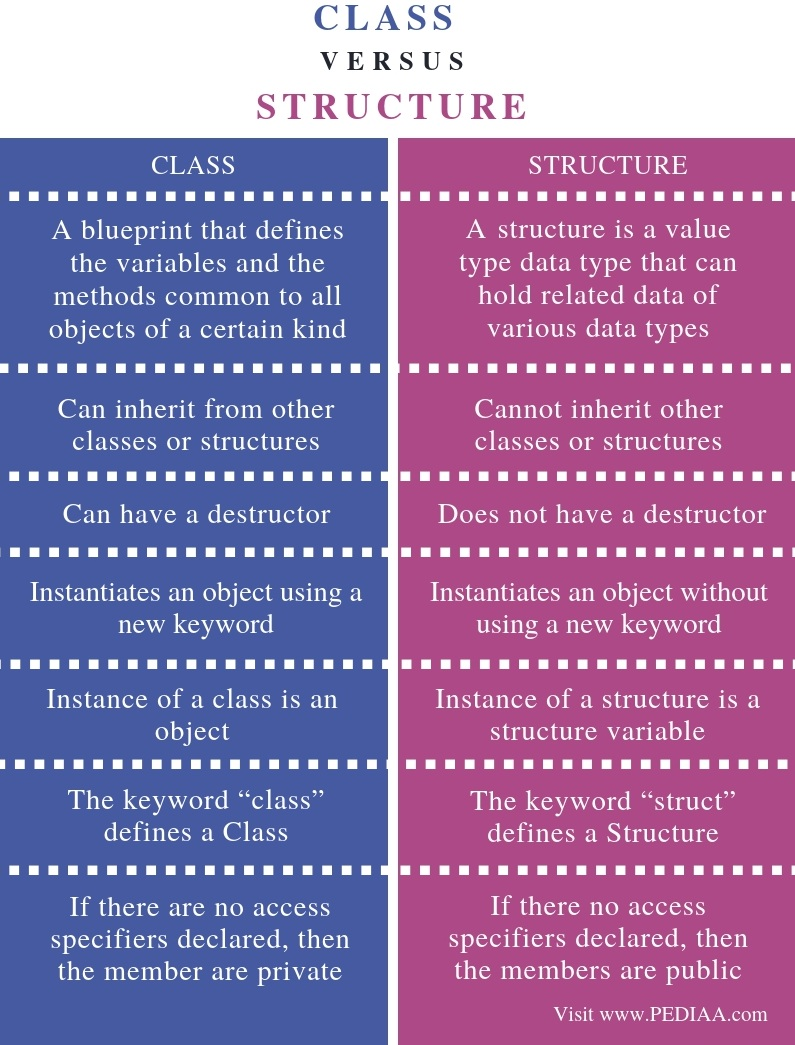
**Important Point About Params Keyword :**

* It is useful when programmer don’t have any prior knowledge about the number of parameters to be used.
* Only one Params keyword is allowed and no additional Params will be allowed in function declaration after a params keyword.
* The length of params will be zero if no arguments will be passed.

**5. about static class in C#**

A static class can only contain static data members, static methods, and a static constructor. It is not allowed to create objects of the static class. Static classes are sealed, means you cannot inherit a static class from another class.

**6. what is diff between struct and class**

****

**7. methods of system.object class**

**1. Equals():**  
**Syntax:**

*Public virtual bool Equals(){}*

This method is used to compare the state of object not the reference of object.  
  
**2. Finalize()**  
**Syntax:**

*Protected virtual void finalize(){}*

This method allows an object to attempt to free resources and perform other clean-up operation  
Before the object is reclaimed by garbage collector  
  
**3. GetHashcode():**  
**Syntax:**

*Public virtual int Gethashcode(){}*

Hash code is calculated by looking towards the state of the object  
  
**4. GetType():**  
**Syntax:**

*Public Type gettype(0{]*

We can say it is the same as GetClass() in JAVA.  
We can Get the type of current object.  
  
**5. MemberwiseClone ():**  
**Syntax:**

*Protected object MemberwiseClone(){]*

If you Remember shallow Copy, it creates the Shallow copy of the Current Object.  
  
**6. ReferenceEquals ():  
  
Syntax:**

*Public static bool ReferenceEquals(object obj1,object obj2){}*

It determines Whether the References of two Object IIS Same or Not.  
  
**7. ToString ():**  
**Syntax:**  
It is the Most Familiar Method.

*Public virtual string ToString(){}*

It represents the current Object in String format.  
Object Class Always on sist of parameter less Constructor.

**8. diff between == referenceequals and equals**

Equals() can return True for different instances of the same object, and this is the most commonly overridden method. . ReferenceEquals() tests whether or not two objects are the same instance and cannot be overridden. == is the same as the ReferenceEquals() by default, but this CAN be overridden.

**9. diff between sealed override and abstract override**

The abstract keyword enables you to create classes and class members that are incomplete and must be implemented in a derived class. The sealed keyword enables you to prevent the inheritance of a class or certain class members that were previously marked virtual.

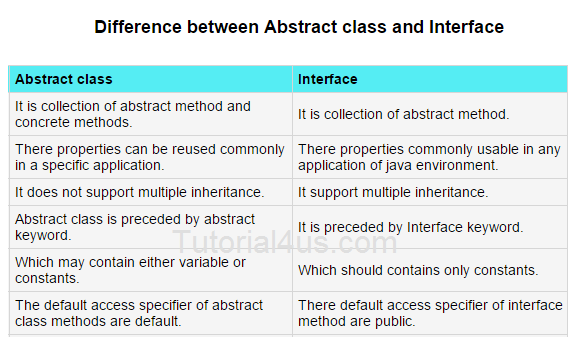
**10. why base class method cannot be sealed?**

**Sealed class cannot be inherited and sealed method in C# programming cannot be overridden**. If we need to stop a method to be overridden or further extension of a class in inheritance hierarchy, we need to use Sealed method and Sealed class respectively in C# object oriented programming

**11.diff between is and as operator**

The is operator is used to check if the run-time type of an object is compatible with the given type or not, whereas the as operator is used to perform conversion between compatible reference types or nullable types. The is operator is of Boolean type, whereas the as operator is not.

**12.diff between abstract class and interface**

****

**13.which members can we declare jn interface**

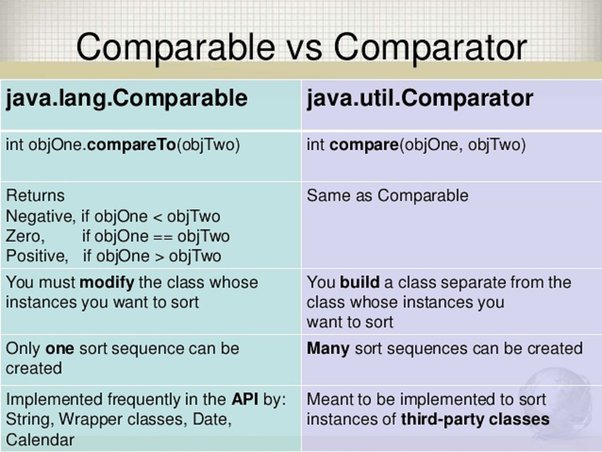
 methods, properties, events, and indexers

**14.diff between string clone and string copy**

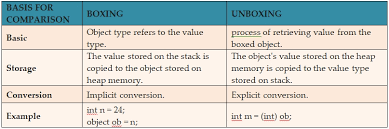
The String.Copy() method creates a new instance of String. This is same as the specified String.

The String.Clone() method returns a reference to the instance of String.

**15. diff between compare and compareto**



**16.diff between boxing and unboxing**

****

**17. what is Serialization and deserialization**

Serialization in C# is the process of bringing an object into a form that it can be written on stream. It's the process of converting the object into a form so that it can be stored on a file, database, or memory; or, it can be transferred across the network. Its main purpose is to save the state of the object so that it can be recreated when needed.

As the name suggests, deserialization in C# is the reverse process of serialization. It is the process of getting back the serialized object so that it can be loaded into memory. It resurrects the state of the object by setting properties, fields etc.

**Types**

* Binary Serialization
* XML Serialization
* JSON Serialization

**18.delegates and types of delegates**

Delegate is one of the base types in .NET. Delegate is a class, which is used to create and invoke delegates at runtime. A delegate in C# is similar to a function pointer in C or C++. It's a new type of object in C#. Delegate is very special type of object as earlier the entire the object we used to defined contained data but delegate just contains the details of a method.

**Singlecast delegate** - Singlecast delegate point to single method at a time. In this the delegate is assigned to a single method at a time. They are derived from System.Delegate class.

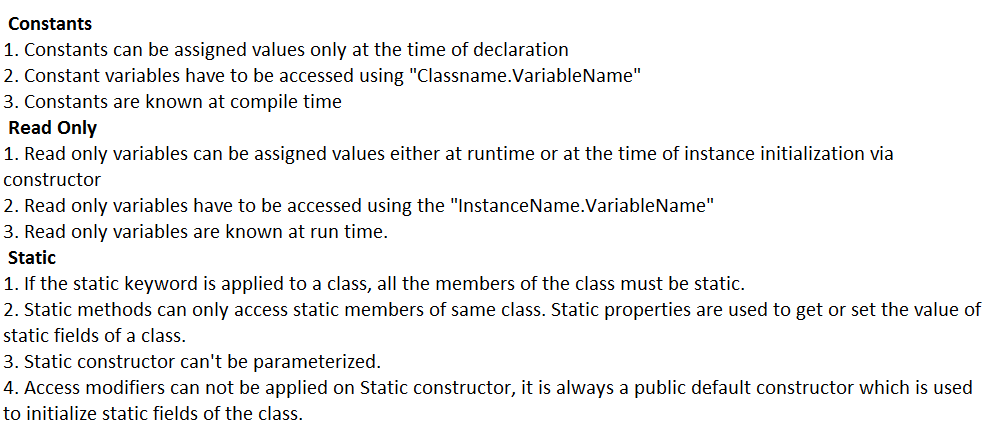
**Multicast Delegate** -When a delegate is wrapped with more than one method that is known as a multicast delegate.

In C#, delegates are multicast, which means that they can point to more than one function at a time. They are derived from System.MulticastDelegate class.

**19. what is event**

An event **is a notification sent by an object to signal the occurrence of an action**.

**20.diff between const and Read-only**



**21. foreground and background thread diff**

Background threads are identical to foreground threads with one exception: **a background thread does not keep the managed execution environment running**. Once all foreground threads have been stopped in a managed process (where the .exe file is a managed assembly), the system stops all background threads and shuts down.

**22. how to achieve Synchronization between threads?**

We can use **lock keyword** to execute program synchronously. It is used to get lock for the current thread, execute the task and then release the lock. It ensures that other thread does not interrupt the execution until the execution finish.

**23. inter thread communication between threads**

Interthread Communication in C# is **a mechanism of communication between two or more threads that act on the shared resource**. To perform the multiple actions at a time we need Inter-thread communication. In order to have smooth interthread communication in C#, we can use the Monitor Class in C#.

**24. what is thread local storage**

**Threads share the data of the process to which it belongs to**. This data sharing provides one of the benefits of multithreaded programming. However, in some circumstances, each thread might need its own copy of certain data. Such data is called thread-local storage (or TLS)

**25. thread life cycle**

In C#, each thread has a life cycle. The life cycle of a thread is started when instance of *System.Threading.Thread class* is created. When the task execution of the thread is completed, its life cycle is ended.

There are following states in the life cycle of a Thread in C#.

* Unstarted
* Runnable (Ready to run)
* Running
* Not Runnable
* Dead (Terminated)

Unstarted State

When the instance of Thread class is created, it is in unstarted state by default.

Runnable State

When start() method on the thread is called, it is in runnable or ready to run state

Running State

Only one thread within a process can be executed at a time. At the time of execution, thread is in running state.

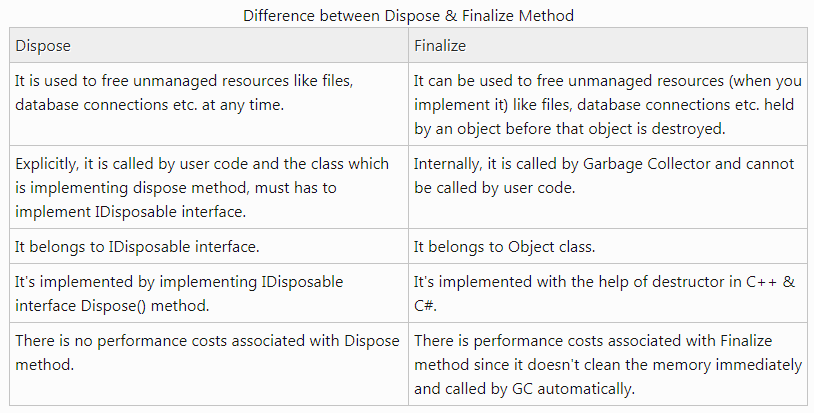
Not Runnable State

The thread is in not runnable state, if sleep() or wait() method is called on the thread, or input/output operation is blocked.

Dead State

After completing the task, thread enters into dead or terminated state.

**26. diff between finalize and dispose method**

****

**27. what is thread safe singleton class**

 A thread safe singleton is **created so that singleton property is maintained even in multithreaded environment**. To make a singleton class thread safe, getInstance() method is made synchronized so that multiple threads can't access it simultaneously.

**28. what is indexer**

Indexers **allow instances of a class or struct to be indexed just like arrays**. The indexed value can be set or retrieved without explicitly specifying a type or instance member. Indexers resemble properties except that their accessors take parameters.

**29. LINQ in C#**

Language-Integrated Query (LINQ) is **the name for a set of technologies based on the integration of query capabilities directly into the C# language**.

**LINQ offers a common syntax for querying any type of data sources**. Secondly, it binds the gap between relational and object-oriented approaches. LINQ expedites development time by catching errors at compile time and includes IntelliSense & Debugging support.