1. **What is Reflection in C#?**

C being the procedural language while C# is a more object-oriented language. The biggest difference is that C# supports automatic garbage collection by Common Language Runtime (CLR) while C does not. C# primarily needs a[**.NET framework**](https://www.interviewbit.com/dot-net-interview-questions/) to execute while C is a platform-agnostic language.

### What are the different ways in which a method can be Overloaded in C#?

### Overloading means when a method has the same name but carries different values to use in a different context. Only the main() method cannot be overloaded.

### Difference between the Equality Operator (==) and Equals() Method in C#?

### **Equality operator (==**) is a reference type which means that if equality operator is used, it will return true only if both the references point to the same object.

### **Equality operator:** Compares by reference

### **Equals() method:** Equals method is used to compare the values carried by the objects. int x=10, int y=10. If x==y is compared then, the values carried by x and y are compared which is equal and therefore they return true.

### **Equals():** Compares by value

### What are Indexers in C#?

### C# indexers are usually known as smart arrays. A C# indexer is a class property that allows you to access a member variable of a class or struct using the features of an array. In C#, indexers are created using this keyword. Indexers in C# are applicable on both classes and structs.

### What are the Arrays in C#?

### An array is used to store a collection of data. Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

### What is the difference between late binding and early binding in C#?

Early Binding and Late Binding concepts belong to polymorphism in C#. Polymorphism is the feature of object-oriented programming that allows a language to use the same name in different forms. For example, a method named Add can add integers, doubles, and decimals.

Polymorphism we have 2 different types to achieve that:

* Compile Time also known as Early Binding or Overloading.
* Run Time is also known as Late Binding or Overriding.

**Compile Time Polymorphism or Early Binding**

In Compile time polymorphism or Early Binding, we will use multiple methods with the same name but different types of parameters, or maybe the number of parameters. Because of this, we can perform different-different tasks with the same method name in the same class which is also known as Method overloading.

**Run Time Polymorphism or Late Binding**

 Run time polymorphism is also known as late binding. In Run Time Polymorphism or Late Binding, we can use the same method names with the same signatures, which means the same type or the same number of parameters, but not in the same class because the compiler doesn’t allow for that at compile time. Therefore, we can use that bind at run time in the derived class when a child class or derived class object will be instantiated. That’s why we call it Late Binding. We have to create my parent class functions as partial and in driver or child class as override functions with the override keyword.

### What are partial classes in C#?

### Partial classes implement the functionality of a single class into multiple files. These multiple files are combined into one during compile time. The partial class can be created using the partial keyword.

### What are Properties in C#?

### Properties are the special type of class members that provides a flexible mechanism to read, write, or compute the value of a private field. Properties can be used as if they are public data members, but they are actually special methods called accessors. It uses pre-defined methods which are “get” and “set” methods which help to access and modify the properties.

**Accessors:**The block of “set” and “get” is known as “Accessors”. It is very essential to restrict the accessibility of the property. There are two types of accessors i.e. get accessors and set accessors. There are different types of properties based on the “get” and set accessors:

* Read and Write Properties: When property contains both get and set methods.
* Read-Only Properties: When property contains only the get method.
* Write Only Properties: When property contains only set method.

### What is Boxing and Unboxing in C#?

### Boxing is process of converting from value type to reference type.

### ex.,

int a= 23; // 23 will assigned to num

### Object Obj = a; // Boxing

### Unboxing is process of converting from reference type to value type.

### Ex.,

int a= 23; // value type is int

Object Obj =a; // Boxing

int i = (int)Obj; // Unboxing

### What is inheritance? Does C# support multiple inheritance?

### **Inheritance** is an important concept of C#. Inheritance is a concept in which you define parent classes and child classes. The child classes inherit methods and properties of the parent class, but at the same time, they can also modify the behavior of the methods if required. The child class can also define methods of its own if required.

### C# doesn’t support multiple inheritances.

### Instead, you can use interfaces to inherit the properties using the class name in the signature.

### What is the difference between an Array and ArrayList in C#?

* + - Array is strongly typed. This means that an array can store only specific type of items\elements
    - In arrays we can store only one datatype either int, string, char etc…
    - Arrays belong to System.Array namespace  
      using System;
    - Example -  
      int[] intArray=new int[]{2};  
      intArray[0] = 1;  
      intArray[2] = 2;
    - ArrayList can store any type of items\elements.
    - In arraylist we can store all the datatype values
    - Arraylist belongs to System.Collection namespaces  
      using System.Collections;
    - Example -  
      ArrayList Arrlst = new ArrayList();  
      Arrlst.Add("Sagar");  
      Arrlst.Add(1);  
      Arrlst.Add(null);

### .

### What are Generics in C#?

C# allows you to define generic classes, interfaces, abstract classes, fields, methods, static methods, properties, events, delegates, and operators using the [type parameter](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/generic-type-parameters) and without the specific data type. A type parameter is a placeholder for a particular type specified when creating an instance of the generic type.

A generic type is declared by specifying a type parameter in an angle brackets after a type name, e.g. TypeName<T> where T is a type parameter.

### Generic classes are defined using a type parameter in an angle brackets after the class name.

### The following defines a generic class.

class DataStore<T>

{

public T Data { get; set; }

}

### What are extension methods in C#?

C# allows us to create and **add new methods to existing class** without creating a new child class. The existing class does not require recompiling the code. C# extension methods are the special type of the static methods that can be called as instance methods.

We can add extension methods in both C# predefined classes and user created custom classes. We need to consider the following points to define an extension method.

* An extension method should be a **static method.**
* It must have **this keyword** associate with class name.
* The class name should be the first parameter in the parameter list.

### What are the differences between ref and out keywords?

### ref keyword:

### ref keyword is used when a called method has to update the passed parameter.

### ref keyword is used to pass data in bi-directional way.

### Before passing a variable as ref, it is required to be initialized otherwise compiler will throw error.

### In called method, it is not required to initialize the parameter passed as ref.

### out keyword :

### out keyword is used when a called method has to update multiple parameter passed.

### out keyword is used to get data in uni-directional way.

### No need to initialize variable if out keyword is used.

### In called method, it is required to initialize the parameter passed as out.

### What is the difference between an abstract class and an interface?

An abstract class can define and declare methods which a class can implement or override whereas an interface can only define functionalities

A class can implement only one abstract class but a class can implement multiple interfaces and so multiple inheritance can be achieved by interface but not by abstract class

An abstract class can have a constructor but an interface does not have a constructor

### What is a managed and unmanaged code?

### Managed Code:

### Executed by CLR, Common Language Runtime, also named as Managed Runtime Environment.

### CLR handles security concerns and provides inbuilt security to code written in .NET.

### Memory buffer overflow never happens, as CLR handles memory allocation and deallocation automatically.

### CLR provides automatic garbage collection, exception handling to managed code.

### Managed Code is converted to IL, Intermiddiate Language also termed as CIL of MSIL.

### Programmer has no low level access using Managed Code.

### Unmanaged Code:

### Executed by Operating System directly on the underlying hardware.

### No inbuilt security present. It is developer's responsibility to write safe and secure code.

### Memory buffer overflow can occur and can hamper the program execution badly.

### No automatic garbage collection and other services are provided to unmanaged code.

### Unmanaged Code is converted to native language code.

### Programmer can write low level access code using unmanaged code.

### What are the types of classes in C#?

Class is an entity that encapsulates all the properties of its objects and instances as a single unit. C# has four types of such classes:

**Static class:** Static class, defined by the keyword ‘static’ does not allow inheritance. Therefore, you cannot create an object for a static class.

**Partial class:** Partial class, defined by the keyword ‘partial’ allows its members to partially divide or share source (.cs) files.

**Abstract class:** Abstract classes are classes that cannot be instantiated where you cannot create objects. Abstract classes work on the OOPS concept of abstraction. Abstraction helps to extract essential details and hide the unessential ones.

**Sealed class:** Sealed classes are classes that cannot be inherited. Use the keyword sealed to restrict access to users to inherit that class.

### What is garbage collection in C#?

### What is Common Language Runtime (CLR)?

### How is C# different from C?

### C being the procedural language while C# is a more object-oriented language. The biggest difference is that C# supports automatic garbage collection by Common Language Runtime (CLR) while C does not. C# primarily needs a [****.NET framework****](https://www.interviewbit.com/dot-net-interview-questions/) to execute while C is a platform-agnostic language.

### What is C#?

### C# is a computer programming language. C# was developed by Microsoft.

C# supports concepts of classes and objects. Classes have members such as fields, properties, events, and methods.

### Can multiple catch blocks be executed?

### No, multiple catch blocks cannot be executed. Once first catch block is catched, it will not read the next block.

### What is the difference between public, static, and void?

**public** − This is the access specifier that states that the method can be accesses publically.

**static** −  Here, the object is not required to access static members.

**void** − This states that the method doesn’t return any value.

**main** − is As stated above, it s the entry point of a C# program i.e. this method is the method that executes first.

### What is an object?

### Object, in C#, is an instance of a class that is created dynamically. Object is also a keyword that is an alias for the predefined type System. Object in the . NET framework. The unified type system of C# allows objects to be defined.

### Define Constructors

### A constructor in C# is a member of a class. It is a method in the class which gets executed when a class object is created. Usually we put the initialization code in the constructor. The name of the constructor is always is the same name as the class. A C# constructor can be public or private. A class can have multiple overloaded constructors.

## Constructor Overloading :

C# supports overloading of constructors, that means we can have constructors with different set of parameters.

### What is Jagged Arrays?

### A jagged array in C# is an array whose elements are arrays. The elements of a jagged array can be of different dimensions and sizes. A jagged array is sometimes called an "array of arrays." A special type of array is introduced in C#. A Jagged Array is an ****array of an array**** in which the length of each array index can differ.

### What is the use of ‘using’ statement in C#?

### The using statement simplifies the code that you have to write to create and then finally clean up the object. The using statement obtains the resource specified, executes the statements and finally calls the Dispose method of the object to clean up the object.

### What is serialization?

Serialization in C# is the process of converting an object into a stream of bytes to store the object to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called deserialization.

There are three types of serialization,

1. Binary serialization (Save your object data into binary format).
2. Soap Serialization (Save your object data into binary format; mainly used in network-related communication).
3. XmlSerialization (Save your object data into an XML file).

### Can we use “this” command within a static method?

### No, we can't use “this” keyword inside a static method. “this” refers to current instance of the class. But if we define a method as static , class instance will not have access to it, only CLR executes that block of code. Hence we can't use “this” keyword inside static method

### What is the difference between constants and read-only?

### readonly keyword:

readonly keyword is used to create a readonly fields.

### readonly is a constant defined at runtime.

### readonly field value can be changed after declaration.

### readonly fields cannot be defined within a method.

### readonly variables are declared as instance variable and assigned values in constructor.

### const keyword :

### const keyword is used to create constant fields.

### const is used to create a constant at compile time.

### const field value cannot be changed after declaration.

### const fields can be declared within a method.

### const fields are to be assigned at the time of declaration.

### What are Custom Control and User Control?

### What are sealed classes in C#?

### What is method overloading?

### Can a private virtual method can be overridden?

### You can't declare private virtual methods because there's no point (since there'd be no way to override them)... But you can override protected virtual methods.

### Describe the accessibility modifier “protected internal”.

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

The major difference between String and StringBuilder is that String objects are immutable while StringBuilder creates a mutable string of characters. StringBuilder will make the changes to the existing object rather than creating a new object.

StringBuilder simplifies the entire process of making changes to the existing string object. Since the String class is immutable, it is costlier to create a new object every time we need to make a change. So, the StringBuilder class comes into picture which can be evoked using the System.Text namespace.

### What’s the difference between the System.Array.CopyTo() and System.Array.Clone() ?

### How can we sort the elements of the Array in descending order?

### What is the difference between Finalize() and Dispose() methods?

### Dispose()

### The method dispose( ) is defined in the interface IDisposable interface.

### public void Dispose( ){ // Dispose code here }

### The method dispose( ) is invoked by the user.

### Method dispose( ) is used to free unmanaged resources whenever it is invoked.

### The method dispose( ) is to be implemented whenever there is a close( ) method.

### The method dispose( ) is declared as public.

### The method dispose( ) is faster and instantly disposes an object.

### Finalize()

### The method finalize( ) id defined in java.lang.object class.

### protected void finalize( ){ // finalization code here }

### The method finalize( ) is invoked by the garbage collector.

### Method finalize( ) is used to free unmanaged resources before the object is destroyed.

### The method finalize( ) is to be implemented for unmanaged resources.

### The method finalize( ) is declared as private.

### The method finalize is slower as compared to dispose

### The method finalize( ) being slower affects the performance of the websites.

### What are delegates?

* A delegate is a type safe function pointer
* A delegate is used when we want to pass a function as a parameter or when we have callback functions

### What’s a multicast delegate?

### A Multicast Delegate in C# is a delegate that holds the references of more than one function. When we invoke the multicast delegate, then all the functions which are referenced by the delegate are going to be invoked.

### What is Console application?

### What is fluent validation?

### Fluent Validation contains .NET libraries and the validation is performed using the Lambda expression. Use Fluent Validation when you want to create some advanced and complex validation for the user data.

### What is the difference between a struct and a class in C#?

**Struct**

* The struct is a value type in C# and it inherits from System.Value Type.
* Struct is usually used for smaller amounts of data.
* Struct can’t be inherited from other types.
* A structure can't be abstract.
* No need to create an object with a new keyword.
* Do not have permission to create any default constructor.

**Class**

* The class is a reference type in C# and it inherits from the System.Object Type.
* Classes are usually used for large amounts of data.
* Classes can be inherited from other classes.
* A class can be an abstract type.
* We can create a default constructor.

### What is enum in C#?

* An enum is a value type with a set of related named constants often referred to as an enumerator list. The enum keyword is used to declare an enumeration. It is a primitive data type that is user-defined.
* An enum type can be an integer (float, int, byte, double, etc.). But if you use it beside int it has to be cast.
* An enum is used to create numeric constants in the .NET framework. All the members of enum are enum type. There must be a numeric value for each enum type.
* The default underlying type of the enumeration element is int. By default, the first enumerator has the value 0, and the value of each successive enumerator is increased by 1.

### What is the difference between “continue” and “break” statements in C#?

**Break (breaks the loop/switch)**  
Break statement is used to terminate the current loop iteration or terminate the switch statement in which it appears  
  
Break statement can be used in the following scenarios:

* for loop (For loop & nested for loop and **Parallel.for**)
* foreach loop (foreach loop & nested foreach loop and **Parallel. foreach**)
* While (while loop & nested while loop)
* Do while (do while loop and nested while loop)
* Switch case (Switch cases and nested switch cases)

**Continue (skip the execution of current iteration)**  
The continue statement is not same as break statement. Break statement breaks the loop/switch whereas continue skip the execution of current iteration only and it does not break the loop/switch i.e. it passes the control to the next iteration of the enclosing while loop, do while loop, for loop or for each statement in which it appears.

### What is IEnumerable<> in C#?

### IEnumerable in C# is an interface that defines one method, GetEnumerator which returns an IEnumerator interface. This allows readonly access to a collection then a collection that implements IEnumerable can be used with a for-each statement.

1. IEnumerable interface contains the System.Collections.Generic namespace.
2. IEnumerable interface is a generic interface which allows looping over generic or non-generic lists.
3. IEnumerable interface also works with linq query expression.
4. IEnumerable interface Returns an enumerator that iterates through the collection.

### What are the differences between IEnumerable and IQueryable?

**IEnumerable**

1. IEnumerable exists in the System.Collections namespace.
2. IEnumerable is suitable for querying data from in-memory collections like List, Array and so on.
3. While querying data from the database, IEnumerable executes "select query" on the server-side, loads data in-memory on the client-side and then filters the data.
4. IEnumerable is beneficial for LINQ to Object and LINQ to XML queries.

**IQueryable**

1. IQueryable exists in the System.Linq Namespace.
2. IQueryable is suitable for querying data from out-memory (like remote database, service) collections.
3. While querying data from a database, IQueryable executes a "select query" on server-side with all filters.
4. IQueryable is beneficial for LINQ to SQL queries.

### Difference between Throw Exception and Throw Clause

### What is an Object Pool in .Net?

### ObjectPool is part of the ASP.NET Core infrastructure that supports keeping a group of objects in memory for reuse rather than allowing the objects to be garbage collected.

### What is Stack in oop?

### Stack follows **LIFO (last in, first out)** order or approach in which the operations are performed. This means that the element which was added last to the stack will be the first element to be removed from the stack.

Following are the basic operations that are supported by the stack.

* **push –** Adds or pushes an element into the stack.
* **pop –**Removes or pops an element out of the stack.
* **peek –** Gets the top element of the stack but doesn’t remove it.
* **isFull –**Tests if the stack is full.
* **isEmpty –** Tests if the stack is empty.

### Difference between value type and reference type.

### If variable store value known as a value type and If variable store address or reference known as a reference type.

### Value type will be created during Complie time and reference type will be created during runtime.

### Value type memory will be allocated with in a Stack and reference type memory will be allocated with in a heap.

### Garbage Collector can’t access stack memory they can only heap memory.

### Value type datatype is predefined,struct,enums and reference type datatype is classes,object,interface,delegate.

### Value type ex,

### int a=10;

### Reference type ex,

### int \*p, a;

### p= &a;

### Singleton Design Patten

Singleton design pattern in C# is one of the most popular design patterns. In this pattern, a class has only one instance in the program that provides a global point of access to it. In other words, a singleton is a class that allows only a single instance of itself to be created and usually gives simple access to that instance.

There are various ways to implement a singleton pattern in C#. The following are the common characteristics of a singleton pattern.

* Private and parameterless single constructor
* Sealed class.
* Static variable to hold a reference to the single created instance
* A public and static way of getting the reference to the created instance.

The advantages of a Singleton Pattern are,

1. Singleton pattern can implement interfaces.
2. Can be lazy-loaded and has Static Initialization.
3. It helps to hide dependencies.
4. It provides a single point of access to a particular instance, so it is easy to maintain.

## Disadvantages of Singleton Design Pattern

The disadvantages of a Singleton Pattern are,

1. Unit testing is a bit difficult as it introduces a global state into an application
2. Reduces the potential for parallelism within a program by locking.

## Singleton class vs. Static methods

The following compares Singleton class vs. Static methods,

1. A Static Class cannot be extended whereas a singleton class can be extended.
2. A Static Class cannot be initialized whereas a singleton class can be.
3. A Static class is loaded automatically by the CLR when the program containing the class is loaded.

# Why is singleton class sealed

Marking the class sealed prevents someone from trivially working around your carefully-constructed singleton class because it keeps someone from inheriting from the class.