C# Interview Questions and Answers

In this article I will cover all the possible interview questions of the C# programming language and explain in as simple language as possible so it becomes understandable by every fellow programmer.

C# is an enormous programming language with many concepts which is being constantly updated by Microsoft and hence the Questions and Answer list cannot be constant.

Thus this article will be periodically updated with the new questions that fit best into this list of C# Interview Questions and Answers.

C# Interview Questions and Answers

1. What is C#?

C# (pronounced as C-Sharp) is an Object Oriented programming language developed by Microsoft. This language is very similar to Java and was developed to bring together C++ features and the easy programming of Visual Basic (VB).

2. Difference between C++ and C#?

C#

- C# is a higher level programming language built over C++ programming language.
- It is similar to Java.
- It can be considered as a hybrid programming language consisting of features of C++ and ease of use of VB.Net.

C++

- C++ is a low level programming language built by adding object oriented concepts to the C programming language
- It is NOT similar to Java.
- C++ itself is a language and does not inherit any features of other languages.

3. What is Struct and Class in C#?

Struct

- Value Type i.e. whenever a variable is created, it holds the value.
- Copying a Struct variable, copies the value.
- A new variable is created in Stack memory.
- Cannot be inherited.
- Cannot be of Abstract type.
- Cannot contain Private parameterless constructors.
- Doesn’t support Destructor.

Class

- Reference Type i.e. whenever a variable is created, it holds the reference (address) while the value is stored some location.
- Copying a Class variable, copies the reference while the value remains on same location.
- A new variable is created in Heap memory.
- Can be inherited.
- Can be of Abstract type.
- Can contain Private parameterless constructors.
- Supports Destructor.

4. Difference between String and StringBuilder in C#?

String

- Belongs to the System namespace.
- Immutable i.e. whenever you append a String object it creates a new object with new memory allocation.
- When changed, a new object is created and old object is destroyed.
- Avoid using inside loops, as if the loop executes N times. N memory allocation operations will be performed.
- Concat function is used for String concatenation.

Example:

```
string s = "Mudassar";
s = string.Concat(s, "Khan");
```

StringBuilder

- Belongs to the System.Text namespace.
- Mutable i.e. whenever you append a String object it does not creates a new object with new memory allocation.
- When changed, no new object is created or old object is destroyed.
- Recommended for loops, as only the size of object will change.
- Append function is used for String concatenation.

Example:

```
StringBuilder sb = new StringBuilder();
sb.Append("Mudassar");
sb.Append("Khan");
```

5. What is Static function and example of in-built Static function in .Net?

- Has the keyword static.
- A Static function is one that can be accessed without creating an object of the class.
- It can be accessed using the name of the class.
- It can belong to Static as well as Non-Static classes.

- One cannot use external Non-Static objects and functions inside a Static method.
- One can use Static function in Static as well as Non-Static classes.
- Keyword this is not supported.
- Example of one inbuilt Static function in C# is Convert.ToString().

Example:

```
public class A
{
    public static void C()
    {
        //Invalid
        this.D();
    }
    public void D()
    {
        //Valid
        C();
    }
}
```

6. What is Static class and its difference with Non-Static class?

A Static class is a class that cannot be instantiated i.e. its object cannot be created.

Static class

- Has the keyword static.
- Cannot be instantiated and no object can be created.
- Keyword new is not supported.
- Objects and Functions can be accessed by class name.
- Mandatory for functions and methods to be marked static i.e. can contain only Static members.
- Members cannot be accessed using this keyword.
- Only one instance can be created.

Example:

```
public static class B
{
    public static object E;

    public static void F()
    {
    }

    public static void G()
    {
        //Valid
        var q = E;

        //Valid
        F();
    }
}
```

Non-Static class

- Does not have the keyword static.
- Can be instantiated and object can be created.
- Keyword new is supported.
- Objects and Functions cannot be accessed by class name unless marked static.
- Not mandatory for Functions and Methods to be marked Static i.e. can contain Static as well as Non-Static members.
- Non-Static members can be accessed using this keyword. But Static members cannot be accessed using this keyword.
- Multiple instances can be created.

Example:

```
public class A
{
    public object C;
    public static object G;

    public static void D()
    {
        //Invalid
        var p = C;

        //Valid
        var q = G;

        //Invalid
        E();
    }
    public void E()
    {
        //Valid.
        var p = this.C;

        //Valid
        var q = G;

        //Valid
        D();
    }
}
```

7. What is Namespace in C#? Give example of in-built Namespace in .Net?

A Namespace is used to organize classes. Example of in-built .Net Namespace is the System Namespace.

- One can create his own Namespace and define the scope of the classes belonging to the Namespace.
- Keyword using is used in order to access a class belonging to a particular Namespace.
- A class belonging to a Namespace can be accessed only when the Namespace is made accessible with the help of using directive.
- A namespace does not have any access modifiers i.e. namespace cannot be public, private, etc.

- A namespace can belong to multiple assemblies and also a single assembly can have multiple namespaces.

Example:

```
namespace MySpace
{
    public class A
    {
        public static void Fun()
        {
        }
    }
}

namespace YourSpace
{
    //Accesing the Namespace
    using MySpace;
    public class B
    {
        public void Fun()
        {
            //Accessing the class of external namespace
            A.Fun();
        }
    }
}
```

8. What is a Sealed class in C#?

A Sealed class is a class which cannot be inherited. The sealed keyword is used to prohibit inheritance of a particular class in C#.

A sealed class can be public as well as private.

Example:

```
public sealed class A
{
    public void Fun()
    {
    }
}
//Compiler Error: 'B': cannot derive from sealed type 'A'
public class B : A
{
    public static void Fun()
    {
    }
}
public class C
{
    public static void Fun()
    {
        //Valid
        A a = new A();
```

```
            a.Fun();
        }
}
```

8. What is an Internal class in C#?

An Internal class is a class which cannot be used outside its Assembly. The internal keyword is used to mark a particular class Internal i.e. it restrict its access outside the Assembly.

- An Assembly could be a Project, a DLL or an EXE.
- Inside the Assembly, the internal class is like public class.
- An internal class can be inherited within the Assembly.

Example:

```
internal class A
{
    public void Fun()
    {
    }
}
//Valid
public class B : A
{
    public static void Fun()
    {
    }
}
public class C
{
    public static void Fun()
    {
        //Valid
        A a = new A();
        a.Fun();
    }
}
```

9. What is an Abstract class in C#?

An Abstract class is a special class which is majorly used for inheritance and it cannot be instantiated.

- Cannot be instantiated i.e. object cannot be created using the new keyword.
- Can contain both Abstract and Non-Abstract members.
- Abstract members are simply declared and are defined in the classes deriving the Abstract class.
- Abstract members are defined by using the override keyword.
- Non-Abstract members are defined within the Abstract class.
- Non-Abstract members can be accessed within the derived classes only if marked public or protected.
- Private Non-Abstract members are not accessible outside the Abstract class.
- Abstract and Non-Abstract members can be accessed using the derived classes.
- Does not support Multiple Inheritance.

Example:

```
public abstract class A
{
    public abstract void Fun1();
    public void Fun4()
    {
    }
    protected void Fun5()
    {
    }
    private void Fun6()
    {
    }
}
public class B : A
{
    public override void Fun1()
    {
    }
    public  void Fun2()
    {
        //Valid
        Fun4();
        Fun5();

        //Compiler Error: Cannot create an instance of the abstract class or
interface
        Fun6();
    }
}
public class C
{
    public static void Fun3()
    {
        //Compiler Error: Cannot create an instance of the abstract class or
interface 'A'
        A a = new A();

        //Valid
        B b = new B();
        b.Fun1();
        b.Fun2();
        b.Fun4();
    }
}
```

10. What is an Interface in C#?

An Interface is similar to an Abstract class and it also cannot be instantiated.

- Cannot be instantiated i.e. object cannot be created using the new keyword.
- Can contain only declaration of Members and not definition.
- Members are simply declared and are defined in the classes deriving the Interface.
- Members are defined without using the override keyword.
- Cannot contain Access modifiers i.e. Members cannot be public, private, etc.

- The derived member can only be public.
- Does support Multiple Inheritance.

Example:

```
public interface A
{
    void Fun1();
    void Fun2();
}
public class B : A
{
    public void Fun1()
    {
    }
    //Error: 'B' does not implement interface member 'A.Fun2()'
    //'B.Fun2()' cannot implement an interface member because it is not public
    private void Fun2()
    {
    }
}
public class C
{
    public static void Fun3()
    {
        //Compiler Error: Cannot create an instance of the abstract class or
interface 'A'
        A a = new A();

        //Valid
        B b = new B();
        b.Fun1();

        //Invalid
        b.Fun2();
    }
}
```