C# Abstract

In c#, **abstract** is a [keyword](https://www.tutlane.com/tutorial/csharp/csharp-keywords-reserved-contextual) and it is useful to define classes and class members that are needed to be implemented or overridden in a derived [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples).

In c#, you can use abstract modifiers with [classes](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples), [methods](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples), [properties](https://www.tutlane.com/tutorial/csharp/csharp-properties-get-set), [events](https://www.tutlane.com/tutorial/csharp/csharp-events), and [indexers](https://www.tutlane.com/tutorial/csharp/csharp-indexer) based on our requirements. The members which we defined as abstract or included in an abstract class must be implemented by classes that derive from an abstract class.

Now we will see how to use abstract modifier in [classes](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) and [methods](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples) with examples.

## C# Abstract Class

In c#, **abstract class** is a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) that is declared with an abstract modifier. If we define a class with abstract modifier, then that [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) is intended only to be used as a [base](https://www.tutlane.com/tutorial/csharp/csharp-base-keyword) class for other classes.

The **abstract class** cannot be instantiated and it can contain both abstract and non-abstract members. The class that is derived from the abstract class must implement all the inherited abstract methods and accessors.

In c#, you can define an abstract class by using abstract keyword. Following is the example of defining an abstract class using abstract keyword.

abstract class Info

{

abstract public void GetDetails();

}

If you observe above code snippet, we defined an abstract class (**Info**) using abstract [keyword](https://www.tutlane.com/tutorial/csharp/csharp-keywords-reserved-contextual) with **GetDetails** method signature.

If we define a [method](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples) with abstract modifier, then that [method](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples) implementation must be done in a derived class.

Following is the example of implementing a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) by deriving from the abstract class.

class User : Info

{

    public override void GetDetails()

    {

        // Method Implementation

    }

}

If you observe the code snippet, we inherited an abstract class (**Info**) in the **User** class and implemented a defined abstract method in the **User** class using [override](https://www.tutlane.com/tutorial/csharp/csharp-override-keyword) keyword. In c#, abstract methods are internally treated as [virtual](https://www.tutlane.com/tutorial/csharp/csharp-virtual-keyword) methods so those methods need to be [overridden](https://www.tutlane.com/tutorial/csharp/csharp-method-overriding) by the derived class.

In c#, we should not use a [sealed](https://www.tutlane.com/tutorial/csharp/csharp-sealed-keyword) keyword with abstract class because the [sealed](https://www.tutlane.com/tutorial/csharp/csharp-sealed-keyword) keyword will make a class as not inheritable but abstract modifier requires a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) to be inherited.

## C# Abstract Class Example

Following is the example of defining an abstract class using abstract modifier in c# programming language.

using System;

namespace YourApp

{

    abstract class Info

    {

        abstract public void GetDetails(string x, string y, int z);

    }

    class User : Info

    {

        public override void GetDetails(string a, string b, int c)

        {

            Console.WriteLine("Name: {0}", a);

            Console.WriteLine("Location: {0}", b);

            Console.WriteLine("Age: {0}", b);

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            User u = new User();

            Console.WriteLine("\*\*\*\*Abstract Class Example\*\*\*\*");

            u.GetDetails("Suresh Dasari", "Hyderabad", 32);

            Console.ReadLine();

        }

    }

}

If you observe above example, we defined an abstract class called “**Info**” with required [methods](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples) and the derived class “**User**” has implemented all the [inherited](https://www.tutlane.com/tutorial/csharp/csharp-inheritance) abstract methods and accessors.

Try executing the above program.

This is how we can use abstract classes in our applications based on our requirements.

## C# Abstract Class Features

The following are important features of abstract class in c# programming language.

* In c#, abstract classes cannot be instantiated.
* The abstract classes can contain both abstract and non-abstract methods and accessors.
* In c#, we should not use a [sealed](https://www.tutlane.com/tutorial/csharp/csharp-sealed-keyword) keyword with abstract class because the [sealed](https://www.tutlane.com/tutorial/csharp/csharp-sealed-keyword) keyword will make a class as not inheritable but abstract modifier requires a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) to be [inherited](https://www.tutlane.com/tutorial/csharp/csharp-inheritance).
* A class that is derived from an abstract class must include all the implementations of inherited abstract methods and accessors.

## C# Abstract Method

In c#, the **abstract method** is a [method](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples) that is declared with an abstract modifier. If we define a method with abstract modifier, then that method doesn’t contain any implementation and method declaration simply ends with a semicolon.

Following is the example of defining an abstract method in c# programming language.

public abstract void GetDetails();

The abstract methods in c# are permitted to declare only in abstract classes and the [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) that is derived from an abstract class must provide an implementation for defined abstract methods.

In c#, abstract methods are internally treated as [virtual](https://www.tutlane.com/tutorial/csharp/csharp-virtual-keyword) methods so those methods need to be [overridden](https://www.tutlane.com/tutorial/csharp/csharp-method-overriding) in the derived class and we should not [static](https://www.tutlane.com/tutorial/csharp/csharp-static-keyword) or [virtual](https://www.tutlane.com/tutorial/csharp/csharp-virtual-keyword) modifiers during abstract method declaration.

## C# Abstract Method Example

Following is the example of declaring an abstraction method in an abstract class in c# programming language.

using System;

namespace YourApp

{

    abstract class Info

    {

        public void Welcome()

        {

            Console.WriteLine("Welcome to YourApp");

        }

        public int age = 32;

        public abstract void GetDetails(string x, string y);

    }

    class User : Info

    {

        public override void GetDetails(string a, string b)

        {

            Welcome();

            Console.WriteLine("Name: {0}", a);

            Console.WriteLine("Location: {0}", b);

            Console.WriteLine("Age: {0}", age);

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            User u = new User();

            Console.WriteLine("\*\*\*\*Abstract Class Example\*\*\*\*");

            u.GetDetails("Suresh Dasari", "Hyderabad");

            Console.ReadLine();

        }

    }

}

If you observe the above example, we defined an abstract class called “**Info**” with required abstract and non-abstract methods and the derived class “**User**” has implemented all the [inherited](https://www.tutlane.com/tutorial/csharp/csharp-inheritance) abstract methods and accessors.

Try executing the above program.

This is how we can use abstract methods in c# abstract classes based on our requirements.

## C# Abstract Method Features

The following are the important features of the abstract method in c# programming language.

* In c#, abstract methods are permitted to declare only within abstract classes.
* The abstract method declaration will not contain any implementation, only the derived classes will provide an actual implementation for abstract methods.
* In c#, abstract methods are internally treated as [virtual](https://www.tutlane.com/tutorial/csharp/csharp-virtual-keyword) methods so those methods need to be [overridden](https://www.tutlane.com/tutorial/csharp/csharp-method-overriding) in the derived class.
* We should not use [static](https://www.tutlane.com/tutorial/csharp/csharp-static-keyword) or [virtual](https://www.tutlane.com/tutorial/csharp/csharp-virtual-keyword) modifiers during the abstract method declaration.
* In c#, abstract properties will act the same as abstract methods but the only difference is declaration and invocation syntax.

# C# Interface

In c#, the **interface** is same as a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) but the only difference is [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) can contain both declarations and implementation of [methods](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples), [properties](https://www.tutlane.com/tutorial/csharp/csharp-properties-get-set) and events but **interface** will contain only the declarations of [methods](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples), [properties](https://www.tutlane.com/tutorial/csharp/csharp-properties-get-set), and events that a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) or [struct](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs" \o "Structures in C# with Examples" \t "_blank) can implement.

An **interface** in c# is more like a contract and the [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) or [struct](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs" \o "Structures in C# with Examples" \t "_blank) that implements an interface must provide an implementation for all the members that are specified in the interface definition.

Generally, c# will not support multiple [inheritance](https://www.tutlane.com/tutorial/csharp/csharp-inheritance) of [classes](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) but that can be achieved by using an interface. In addition, a [structure](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs) in c# cannot be [inherited](https://www.tutlane.com/tutorial/csharp/csharp-inheritance) from another [structure](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs) or [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) but that can be [inherited](https://www.tutlane.com/tutorial/csharp/csharp-inheritance) by using interfaces.

In c#, we can define an interface by using interface keyword. Following is the example of defining an interface using interface keyword.

interface IUser

{

    void InsertDetails();

}

If you observe the above code snippet, we defined an interface (**IUser**) using interface keyword with the **InsertDetails** method signature. Now, the **IUser** interface can be implemented by any [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) or [struct](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs" \o "Structures in C# with Examples" \t "_blank) by providing a definition for the **InsertDetails** method.

To implement an interface in a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) or [structure](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs) the syntax will be like class ClassName : Interface Name. Following is the example of implementing an interface in a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples).

class User : IUser

{

    void InserDetails()

    {

        // Method Implementation

    }

}

If you observe the code snippet, we inherited an interface (**IUser**) in a class (**User**) and implemented a defined interface method in a class.

In c#, an interface cannot be instantiated directly, but it can be instantiated by a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) or [struct](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs" \o "Structures in C# with Examples" \t "_blank) that implements an interface. Following is the example of creating an instance for the interface in c# programming language.

IUser u = new User();

In c#, a class can inherit only from one class but we can implement multiple interfaces in a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) or [struct](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs" \o "Structures in C# with Examples" \t "_blank) by using interfaces.

By default, the members of an interface are [public](https://www.tutlane.com/tutorial/csharp/csharp-access-modifiers-public-private-protected-internal#divcpblm) and we are not allowed to include any other [access modifiers](https://www.tutlane.com/tutorial/csharp/csharp-access-modifiers-public-private-protected-internal). In c#, an interface can contain [methods](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples), [properties](https://www.tutlane.com/tutorial/csharp/csharp-properties-get-set), events, indexers but it can’t contain [constants](https://www.tutlane.com/tutorial/csharp/csharp-const-constant-keyword), [fields](https://www.tutlane.com/tutorial/csharp/csharp-variables-with-examples), [operators](https://www.tutlane.com/tutorial/csharp/csharp-operators-arithmetic-relational-logical-assignment-precedence), instance [constructors](https://www.tutlane.com/tutorial/csharp/csharp-constructors-with-examples), finalizers or types.

## C# Interface Example

Following is the example of creating and implementing an instance using a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) in c# programming language.

using System;

namespace YourApp

{

    interface IUser

    {

        void GetDetails(string x);

    }

    class User : IUser

    {

        public void GetDetails(string a)

        {

            Console.WriteLine("Name: {0}", a);

        }

    }

    class User1 : IUser

    {

        public void GetDetails(string a)

        {

            Console.WriteLine("Location: {0}", a);

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            IUser u = new User();

            u.GetDetails("Suresh Dasari");

            IUser u1 = new User1();

            u1.GetDetails("Hyderabad");

            Console.WriteLine("\nPress Enter Key to Exit..");

            Console.ReadLine();

        }

    }

}

If you observe above example, we created an interface **IUser** and the two classes “**User & User1**” implemented an interface **IUser** by providing an implementation for **GetDetails()** method and we created an instance for interface “**IUser**” using **User & User1** classes.

Try executing the above program.

## C# Multiple Inheritance with Interface

As discussed, c# will not support multiple [inheritance](https://www.tutlane.com/tutorial/csharp/csharp-inheritance) of [classes](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) but that can be achieved by using the interface.

Following is the example of implementing a multiple [inheritance](https://www.tutlane.com/tutorial/csharp/csharp-inheritance) using interfaces in c# programming language.

using System;

namespace YourApp

{

    interface IName

    {

        void GetName(string x);

    }

    interface ILocation

    {

        void GetLocation(string x);

    }

    interface IAge

    {

        void GetAge(int x);

    }

    class User : IName, ILocation, IAge

    {

        public void GetName(string a)

        {

            Console.WriteLine("Name: {0}", a);

        }

        public void GetLocation(string a)

        {

            Console.WriteLine("Location: {0}", a);

        }

        public void GetAge(int a)

        {

            Console.WriteLine("Age: {0}", a);

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            User u = new User();

            u.GetName("Suresh Dasari");

            u.GetLocation("Hyderabad");

            u.GetAge(32);

            Console.WriteLine("\nPress Enter Key to Exit..");

            Console.ReadLine();

        }

    }

}

If you observe example, we created a multiple interfaces and implementing those interfaces using User class to achieve multiple [inheritance](https://www.tutlane.com/tutorial/csharp/csharp-inheritance).

Try executing the above program.

This is how you can achieve multiple [inheritance](https://www.tutlane.com/tutorial/csharp/csharp-inheritance) using the interface in the c# programming language.

## C# Interface Overview

The following are the important properties of the interface in the c# programming language.

* In c#, the interface is like an [abstract class](https://www.tutlane.com/tutorial/csharp/csharp-abstract) and it can contain only declarations of members such as [methods](https://www.tutlane.com/tutorial/csharp/csharp-methods-functions-with-examples), [properties](https://www.tutlane.com/tutorial/csharp/csharp-properties-get-set), indexers and events.
* By default, the members of an interface are [public](https://www.tutlane.com/tutorial/csharp/csharp-access-modifiers-public-private-protected-internal#divcpblm) and we are not allowed to include any other [access modifiers](https://www.tutlane.com/tutorial/csharp/csharp-access-modifiers-public-private-protected-internal).
* In c#, an interface cannot be instantiated directly, but it can be instantiated by a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) or [struct](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs" \o "Structures in C# with Examples" \t "_blank) that implements an interface.
* The [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) or [struct](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs" \o "Structures in C# with Examples" \t "_blank) that implements an interface must provide an implementation for all the members that are specified in the interface definition.
* The [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) or [struct](https://www.tutlane.com/tutorial/csharp/csharp-structures-structs" \o "Structures in C# with Examples" \t "_blank) can implement multiple interfaces.

## Difference between Abstract Class and Interface

The following are the differences between [abstract class](https://www.tutlane.com/tutorial/csharp/csharp-abstract) and interface in c# programming language.

