Inheritance

Inheritance is one of the three foundational principles of object-oriented programming

because it allows the creation of hierarchical classifications. Using inheritance, you can

create a general class that defines traits common to a set of related items. This class can

then be inherited by other, more specific classes, each adding those things that are unique

to it.

In the language of C#, a class that is inherited is called a *base class.* The class that does

the inheriting is called a *derived class.* Therefore, a derived class is a specialized version of a

base class. It inherits all of the variables, methods, properties, and indexers defined by the

base class and adds its own unique elements.

**Inheritance Basics**

C# supports inheritance by allowing one class to incorporate another class into its declaration.

This is done by specifying a base class when a derived class is declared. Let’s begin with

an example. The following class called **TwoDShape** stores the width and height of a twodimensional

object, such as a square, rectangle, triangle, and so on.

// A class for two-dimensional objects.

class TwoDShape {

public double Width;

public double Height;

public void ShowDim() {

Console.WriteLine("Width and height are " +

Width + " and " + Height);

}

}

**TwoDShape** can be used as a base class (that is, as a starting point) for classes that

describe specific types of two-dimensional objects. For example, the following program uses

**TwoDShape** to derive a class called **Triangle**. Pay close attention to the way that **Triangle** is

declared.

// A simple class hierarchy.

using System;

// A class for two-dimensional objects.

class TwoDShape {

public double Width;

public double Height;

public void ShowDim() {

Console.WriteLine("Width and height are " +

Width + " and " + Height);

}

}

// Triangle is derived from TwoDShape.

class Triangle : TwoDShape {

public string Style; // style of triangle

// Return area of triangle.

public double Area() {

return Width \* Height / 2;

}

// Display a triangle's style.

public void ShowStyle() {

Console.WriteLine("Triangle is " + Style);

}

}

class Shapes {

static void Main() {

Triangle t1 = new Triangle();

Triangle t2 = new Triangle();

t1.Width = 4.0;

t1.Height = 4.0;

t1.Style = "isosceles";

t2.Width = 8.0;

t2.Height = 12.0;

t2.Style = "right";

Console.WriteLine("Info for t1: ");

t1.ShowStyle();

t1.ShowDim();

Console.WriteLine("Area is " + t1.Area());

Console.WriteLine();

Console.WriteLine("Info for t2: ");

t2.ShowStyle();

t2.ShowDim();

Console.WriteLine("Area is " + t2.Area());

}

}

**Multi Level Inheritance**

Generally, c# supports only **single inheritance** that means a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) can only inherit from one base [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples). However, in c# the inheritance is transitive and it allows you to define a hierarchical inheritance for a set of types and it is called a multi-level inheritance.

For example, suppose if class **C** is derived from class **B**, and class **B** is derived from class **A**, then the class **C** inherits the members declared in both class **B** and class **A**.

public class A

{

// Implementation

}

public class B : A

{

// Implementation

}

public class C : B

{

// Implementation

}

If you observe above code snippet, the class **C** is derived from class **B**, and class **B** is derived from class **A**, then the class **C**inherits the members declared in both class **B** and class **A**. This is how we can implement **multi-level** inheritance in our applications.

C# Multi Level Inheritance Example

Following is the example of implementing **multi-level inheritance** in c# programming language.

using System;

namespace Tutlane

{

    public class A

    {

        public string Name;

        public void GetName()

        {

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

        }

    }

    public class B : A

    {

        public string Location;

        public void GetLocation()

        {

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

        }

    }

    public class C : B

    {

        public int Age;

        public void GetAge()

        {

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

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            C c = new C();

            c.Name = "Suresh Dasari";

            c.Location = "Hyderabad";

            c.Age = 32;

            c.GetName();

            c.GetLocation();

            c.GetAge();

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

            Console.ReadLine();

        }

    }

}

If you observe above example, we implemented a three classes (**A**, **B**, **C**) and the class **C** is derived from class **B**, and class **B** is derived from class **A**.

By implementing a multi-level inheritance, the class **C** can inherits the members declared in both class **B** and class **A**.

When we execute above c# program, we will get the result like as shown below.

**Multiple Inheritance**

As discussed, c# supports only **single inheritance** that means a [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples) can only inherit from one base [class](https://www.tutlane.com/tutorial/csharp/csharp-classes-and-objects-with-examples). In case, if we try to inherit a class from multiple base classes, then we will get a compile time errors.

For example, if class **C** is trying to inherit from Class **A** and **B** at the same time, then we will get a compile time error because multiple inheritance is not allowed in c#.

public class A

{

// Implementation

}

public class B

{

// Implementation

}

public class C : A, B

{

// Implementation

}

If you observe above code snippet, the class **C** is trying to inherit a [properties](https://www.tutlane.com/tutorial/csharp/csharp-properties-get-set) from both class **A** and **B** at the same time which will lead to compile time error like “**Class C cannot have a multiple classes: A and B**”.

As discussed, **multi-level** inheritance is supported in c# but **multiple** inheritance is not supported. In case, if you want to implement **multiple** inheritance in c#, that we can achieve by using **interfaces**. In next chapters, we will learn how to use **interfaces** to achieve multiple inheritance in detailed manner.