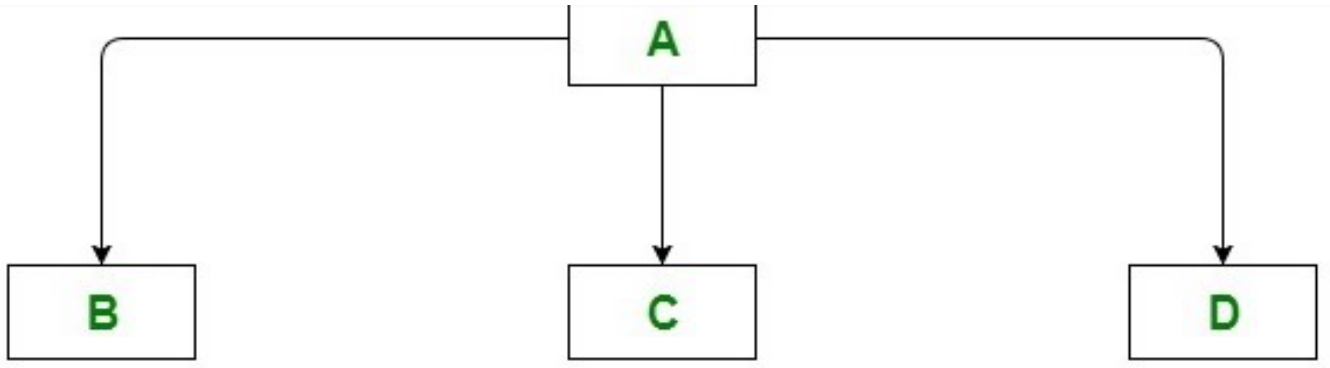
C# Program For Hierarchical Inheritance

Last Updated : 30 Sep, 2021

Inheritance is a basic aspect of object-oriented programming. A superclass, also known as a base class, is a class whose members are inherited, whereas a subclass, also known as a derived class, is a class that inherits from a superclass. They are also known as the parent and child classes, respectively. In the same way that a child inherits the traits of his or her parents, and parents inherit the characteristics of their predecessors, inheritance in programming languages works in the same way.

Hierarchical Inheritance

It is a way of transmitting features from a parent class to a base, child, or subclass in terms of technical terms and the object-oriented aspect. The parent class or superclass is the class from which the properties are taken, i.e. the features are inherited. Hierarchical inheritance describes a situation in which a parent class is inherited by multiple subclasses. A type of inheritance in which more than one class is inherited from a single parent or base class is known as hierarchical inheritance. The base class shares many of the same properties as the parent class, especially those that are common in the parent class. A single base class gives rise to many classes. It's like having several children, each with their own set of characteristics acquired from their parents. For example, In the diagram below, class A acts as the base class(parent class) for the child classes B, C, and D.



Example 1:

The base class in the following example is Father, and the derived classes are ChildFirst and ChildSecond. We've created objects from both derived classes and are calling the same base class function.

C#

```
// C# program to illustrate the above concept
using System;

// Base Class
public class Father
{
    public string FatherName()
    {
        return "Ravi";
    }
}

// Derived Class
public class ChildFirst : Father
{
    public string ChildDName()
    {
        return "Rohan";
    }
}

// Derived Class
```

```
    {
        return "Nikhil";
    }
}

class GFG{

static public void Main()
{
    ChildFirst first = new ChildFirst();

    // Displaying Child Name and Father Name for
    // ChildFirst
    Console.WriteLine("My name is " + first.ChildDName() +
                      ". My father name is " +
                      first.FatherName() + ".");
    ChildSecond second = new ChildSecond();

    // Displaying Child Name and Father Name for
    // ChildSecond
    Console.WriteLine("My name is " + second.ChildDName() +
                      ". My father name is " +
                      second.FatherName() + ".");
}
}
```

Output

```
My name is Rohan. My father name is Ravi.
My name is Nikhil. My father name is Ravi.
```

Example 2:

In the following code, we created three classes: Person, Teacher, and Doctor. In this example, the Person class was inherited by both the Teacher and Doctor classes. A constructor in every class is used to initialize data members. Then we created Teacher and Doctor objects and used TeacherDetails() and DoctorDetails() to produce information for the Teacher and Doctor respectively.

```
// Base Class
class Person
{
    public string name;
    public int aadhar_id;
    public int age;

    public Person(int aadhar_id, int age, string name)
    {
        this.aadhar_id = aadhar_id;
        this.name = name;
        this.age = age;
    }
}

// Derived Class
class Teacher : Person
{
    public int teacher_salary;

    public Teacher(int aadhar_id, int salary,
                   string name, int age) : base(aadhar_id,
                                                age, name)
    {
        teacher_salary = salary;
    }
    public void TeacherDetails()
    {
        Console.WriteLine("teacher ID:      " + aadhar_id);
        Console.WriteLine("teacher Name:    " + name);
        Console.WriteLine("teacher Salary:  " + teacher_salary);
        Console.WriteLine("teacher Age:     " + age);
    }
}

// Derived Class
class Doctor : Person
{
    public int doctor_fees;

    public Doctor(int aadhar_id, int fees,
                  string name, int age) : base(aadhar_id,
                                               age, name)
    {
        doctor_fees = fees;
    }
    public void DoctorDetails()
```

```
        Console.WriteLine("Doctor Fees:    " + doctor_fees);
        Console.WriteLine("Doctor Age:     " + age);
    }
}

class GFG{

static public void Main()
{

    // Creating objects
    Teacher t = new Teacher(25054, 50000, "Sanjay", 28);
    Doctor d = new Doctor(25045, 750, "Rohit", 32);

    t.TeacherDetails();
    Console.WriteLine(
        "-----------------------------------");
    d.DoctorDetails();
}
}
```

Output

```
teacher ID:      25054
teacher Name:    Sanjay
teacher Salary:  50000
teacher Age:     28
-----------------------------------
Doctor ID:      25045
Doctor Name:    Rohit
Doctor Fees:    750
Doctor Age:     32
```

RECOMMENDED ARTICLES

01 **C# Program to Demonstrate Abstract Class with Multiple-level Inheritance**
28, Oct 21

02 **C# Program to Demonstrate Interface Implementation with Multi-level Inheritance**
28, Oct 21

03 **C# Program to Demonstrate the Inheritance of Abstract Classes**
20, Nov 21

04 **C# Program to Implement Multiple-Inheritance using Abstract Class and Interface**
22, Jan 22

05 **C# | Inheritance**
19, Jul 18

06 **C# | Multilevel Inheritance**
27, Nov 18

07 **C# | Inheritance in Constructors**
27, Nov 18

08 **C# | Inheritance in interfaces**
18, Dec 18

Article Contributed By :

priyavermaa1198
@priyavermaa1198

Easy | Normal | Medium | Hard | Expert

Article Tags : CSharp-Inheritance, CSharp-programs, Picked, C#

Improve Article | Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks

A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Top News

Technology

Work & Career

Business

Finance

Lifestyle

Knowledge

Python

Java

CPP

Golang

C#

SQL

Kotlin

Web Development

Web Tutorials

Django Tutorial

HTML

JavaScript

Bootstrap

ReactJS

NodeJS

Contribute

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship