C++ Inheritance

By:
Dr. Shweta Jain
Head Coordinator, Data Science

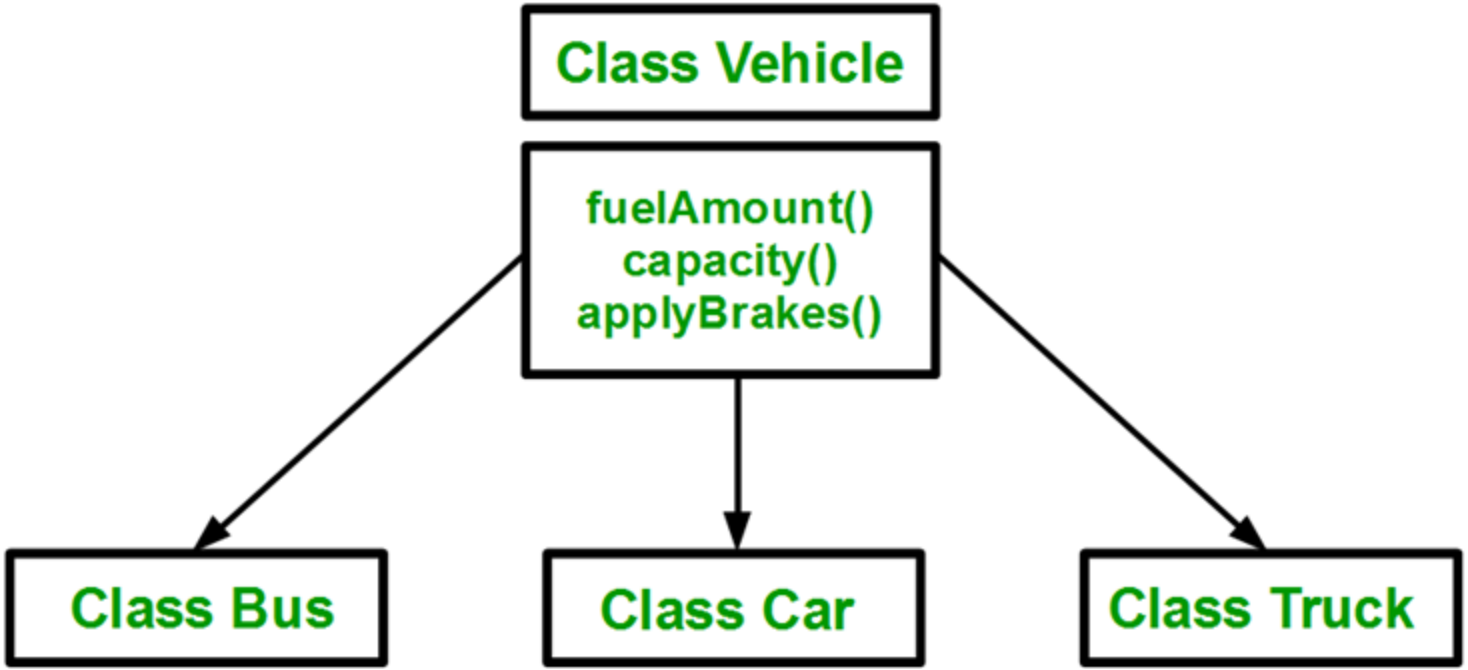
C++ Inheritance

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important feature of Object Oriented Programming.

Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.

Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.

Why and when to use inheritance?



Syntax for Inheritance and Access Modes:

In C++, there are three access specifiers:

- public - members are accessible from outside the class
- private - members cannot be accessed (or viewed) from outside the class
- protected - members cannot be accessed from outside the class, however, they can be accessed in inherited classes. You will learn more about Inheritance later.

Syntax:

```
class subclass_name : access_mode base_class_name
{
  //body of subclass
};
```

Code 1: Demonstration of Inheritance

```
// C++ program to demonstrate
// implementation of Inheritance

#include <iostream.h>
using namespace std;

//Base class
class Parent
{
    public:
      int id_p;
};

// Sub class inheriting from
// Base Class(Parent)
class Child : public Parent
{
    public:
      int id_c;
};
```

```
//main function
int main()
   {

        Child obj1;

        // An object of class child
         // has all data members
        // and member functions of class parent
        obj1.id_c = 7;
        obj1.id_p = 91;
        cout << "Child id is " <<  obj1.id_c << endl;
        cout << "Parent id is " <<  obj1.id_p << endl;

        return 0;
   }
```

Output

```
Child id is 7
Parent id is 91
```

Code 2: Demonstration of Modes of Inheritance

```
// C++ Implementation to show that a derived class
// doesn't inherit access to private data members.
// However, it does inherit a full parent object
class A
{
public:
    int x;
protected:
    int y;
private:
    int z;
};

class B : public A
{
    // x is public
    // y is protected
    // z is not accessible from B
};
```

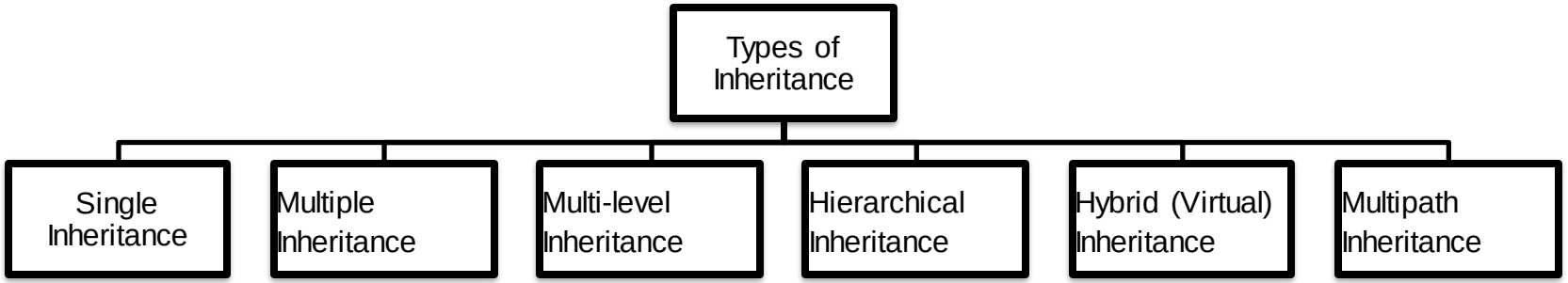
```
class C : protected A
{
    // x is protected
    // y is protected
    // z is not accessible from C
};

class D : private A   // 'private' is default for
                      // classes
{
    // x is private
    // y is private
    // z is not accessible from D
};
```

Type of Inheritance based on Access Modes/Specifiers

Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

Types of Inheritance

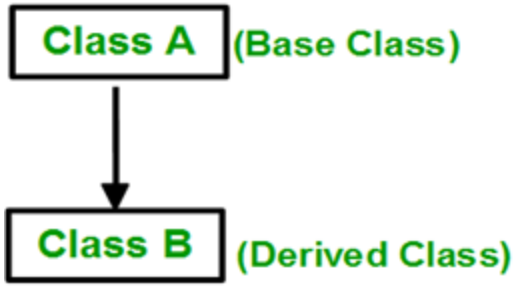


1. Single Inheritance

In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.

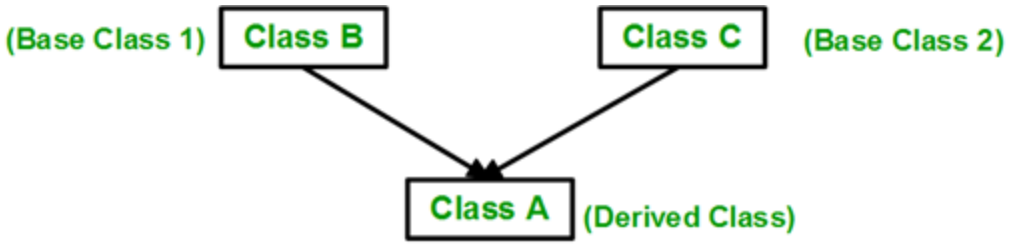
Syntax:

```
class subclass_name : access_mode base_class
{
  //body of subclass
};
```



2. Multiple Inheritance

Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one sub class is inherited from more than one base classes.

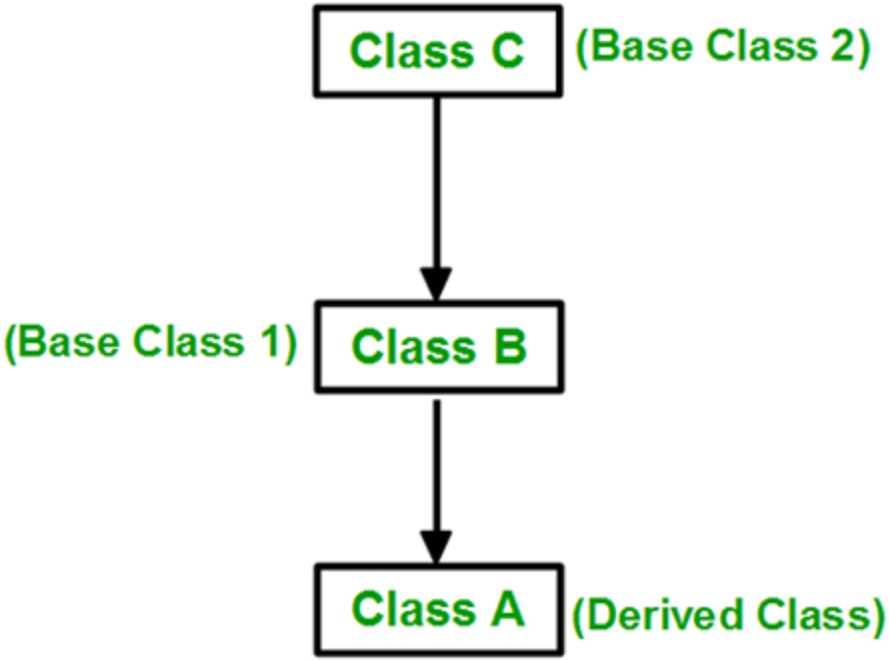


Syntax:

```
class subclass_name : access_mode base_class1, access_mode base_class2, ....
{
  //body of subclass
};
```

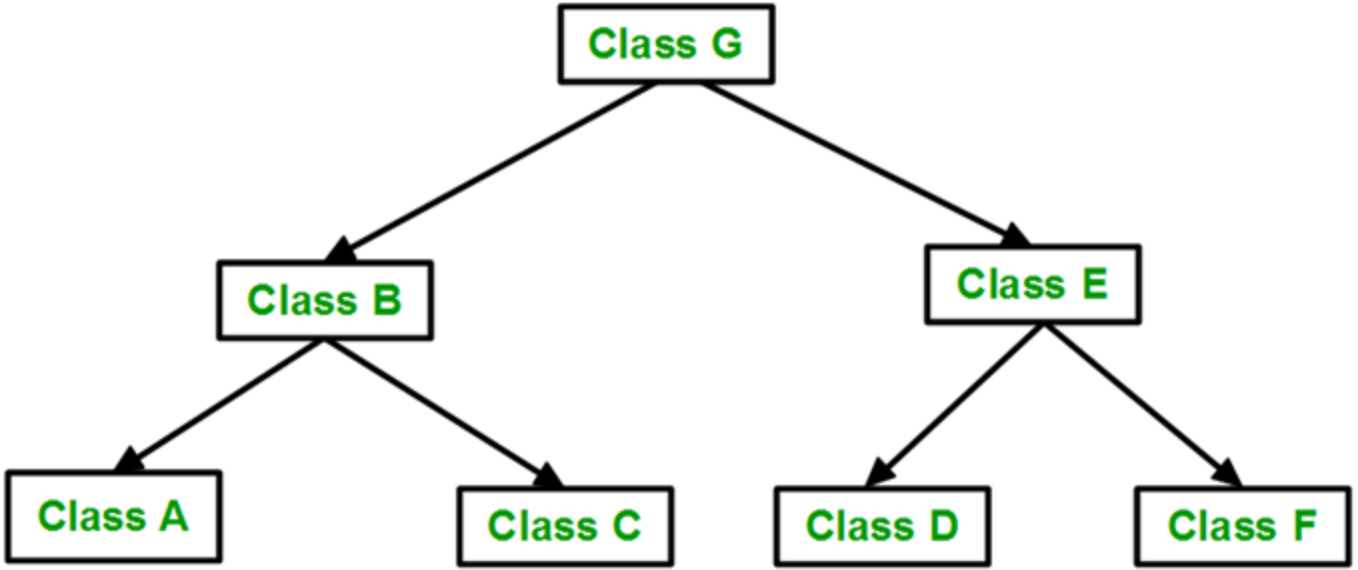
3. Multilevel Inheritance

In this type of inheritance, a derived class is created from another derived class.



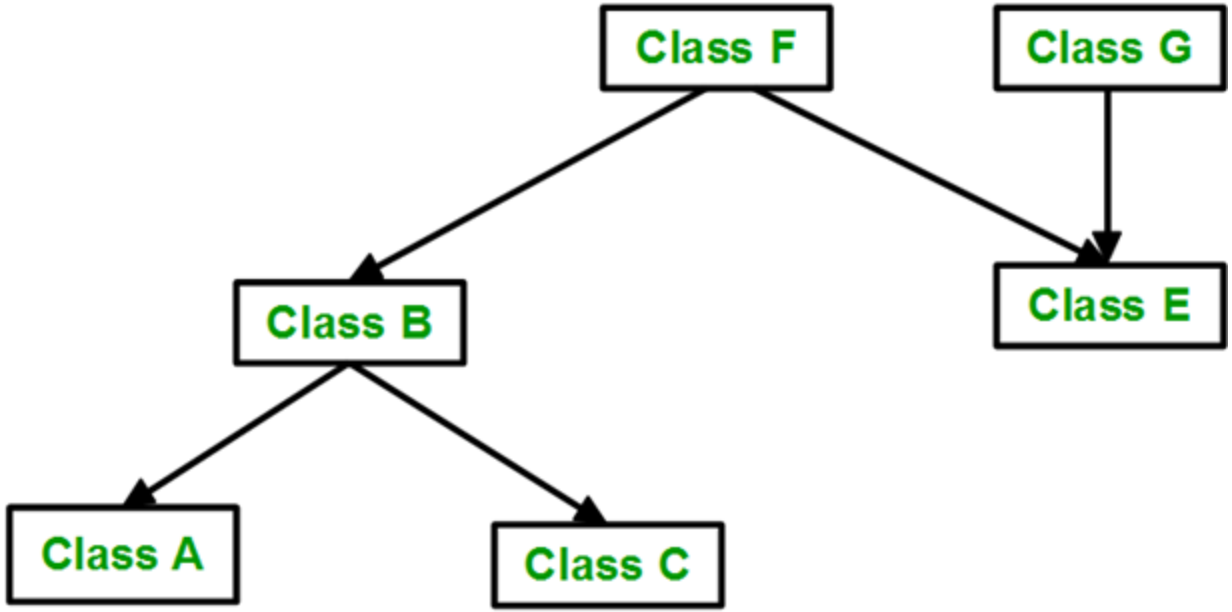
4. Hierarchical Inheritance

In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.



5. Hybrid (Virtual) Inheritance

Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.



6. Special case of hybrid inheritance : Multipath inheritance

A derived class with two base classes and these two base classes have one common base class is called multipath inheritance. An ambiguity can arise in this type of inheritance.