JYOTHISHMATHI INSTITUTE OF TECHNOLOGY & SCIENCE
Nustulapur, Karimnagar – 505481.

Data Structures through C++

II Year B.Tech. I Sem.

2018-19

B.Sankeerthana
Asst. Professor
CSE DEPT.

Inheritance

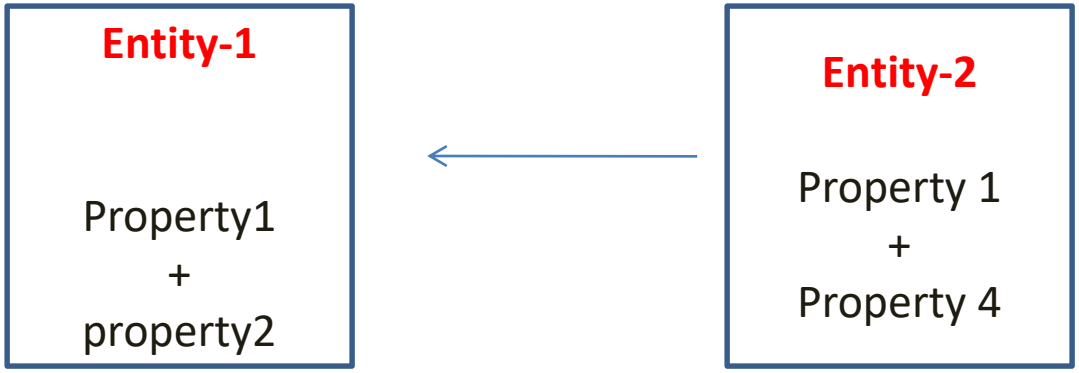
INHERITANCE

- Inheritance in Object Oriented Programming can be described as a process of creating new classes from existing classes.

- New classes inherit some of the properties and behavior of the existing classes. An existing class that is "parent" of a new class is called a **base class**.

- New class that inherits properties of the base class is called a **derived class**.

- Acquiring properties of one entity into another entity.



- This provides Reusability
  - Save time and money
  - Increases a program reliability
  - The ease of distributing class libraries

Inheritance Syntax

- The basic syntax of inheritance is:

  class derivedclass : accessspecifier baseclass

- Access specifier can be public, protected and private. The default access specifier is **private.**
- Access specifiers affect accessibility of data members of base class from the derived class.

Example:

```
class  computer :  public  maths
  {
       ----
  };
```

Inheritance Access Specifiers

- Access rules for base class members

Access specifier	Accessible from base class	Accessible from derived class	Accessible from Obj's outside the class
Public	Yes	Yes	Yes
Protected	Yes	Yes	No
Private	Yes	No	No

Modes of Inheritance

- **Public mode**: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class. Private members of the base class will never get inherited in sub class.

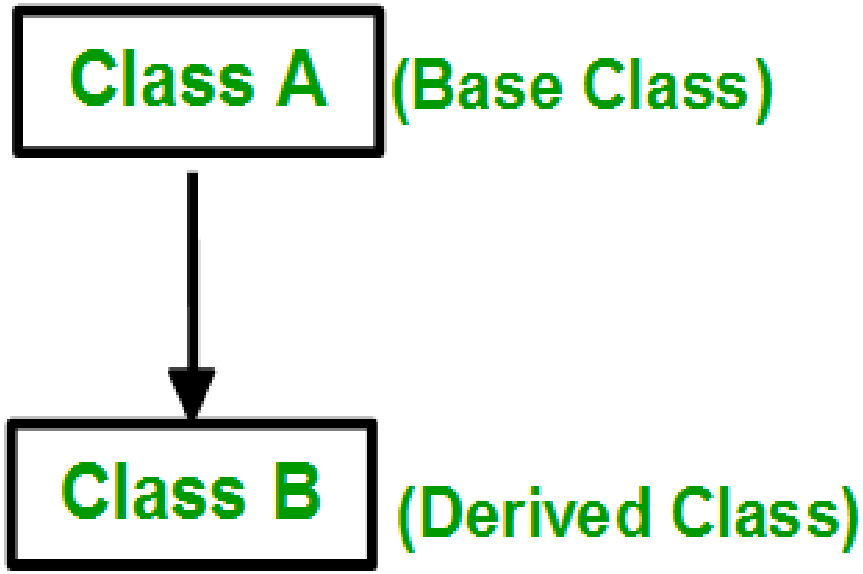
- **Protected mode**: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class. Private members of the base class will never get inherited in sub class.

- **Private mode**: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class. Private members of the base class will never get inherited in sub class.

	Derived Class	Derived Class	Derived Class
Base class	**Public Mode**	**Private Mode**	**Protected Mode**
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected

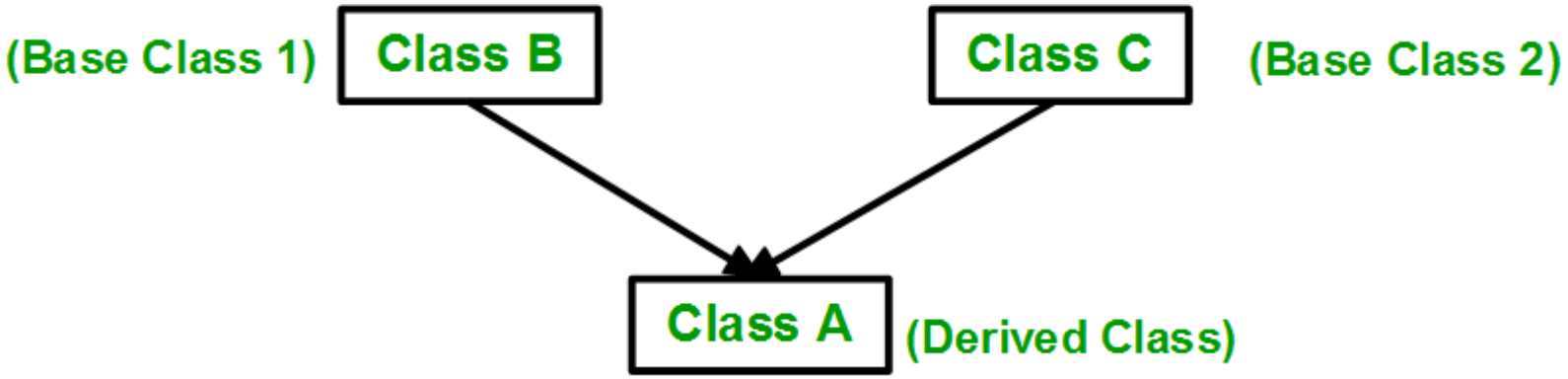
Single Inheritance

- 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.



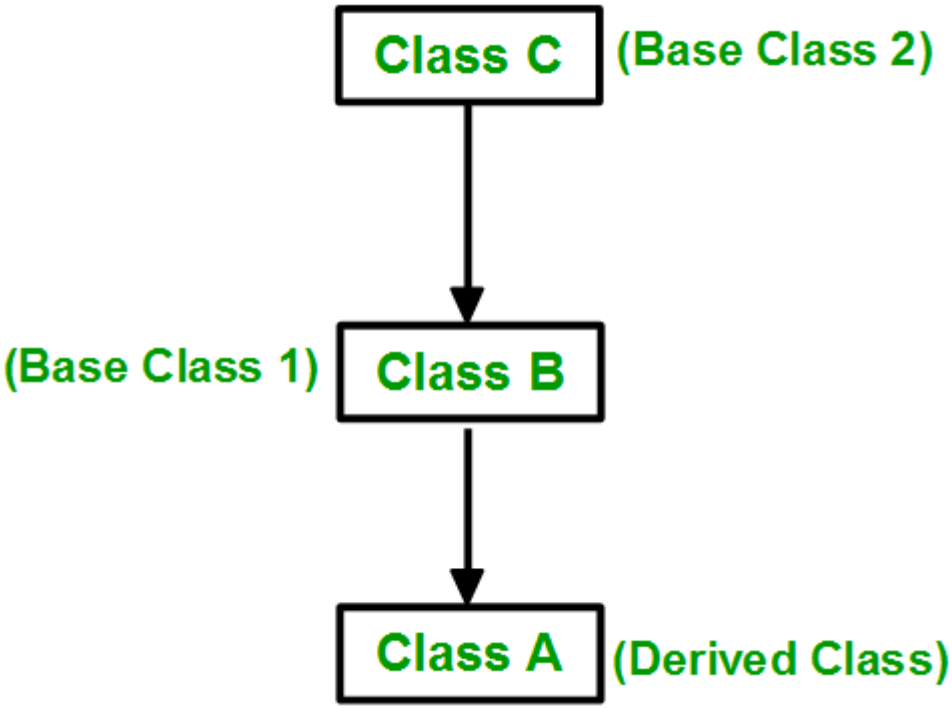
Multiple Inheritance

- 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**.



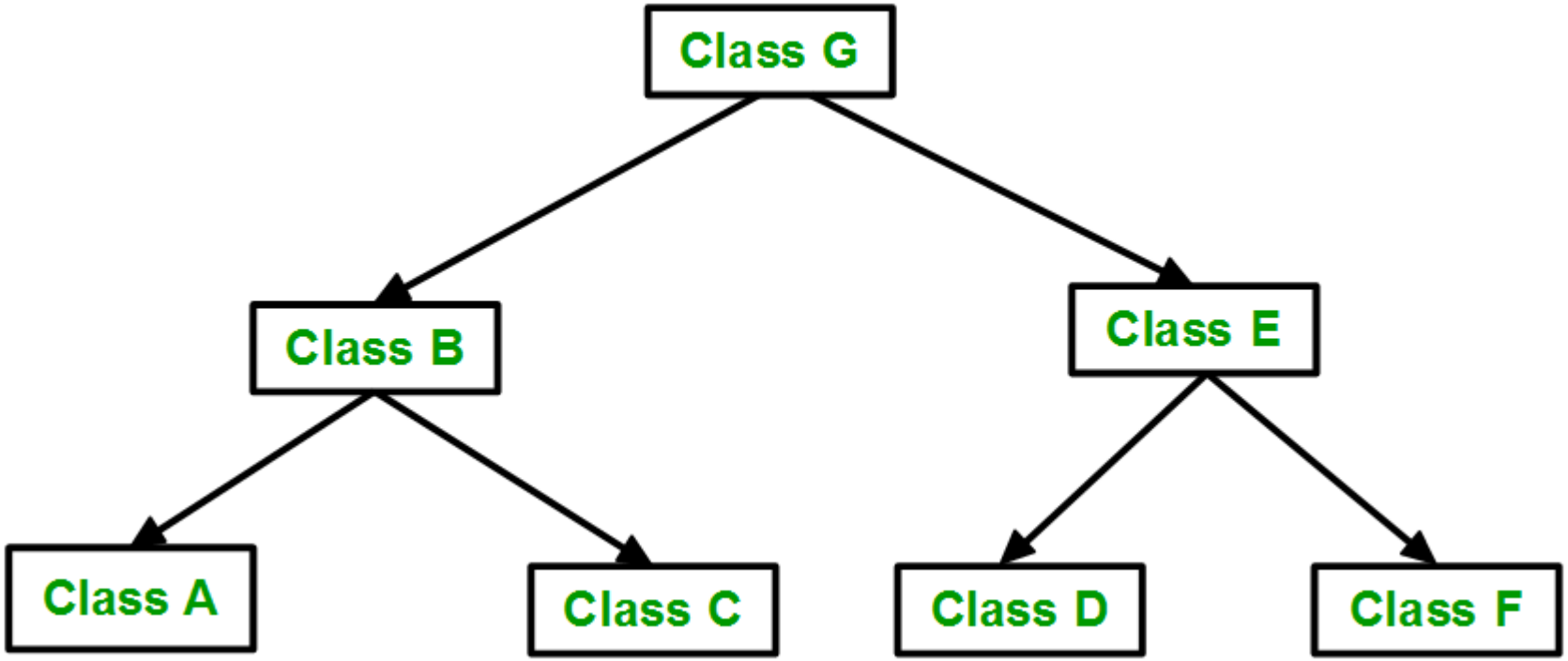
Multilevel Inheritance

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



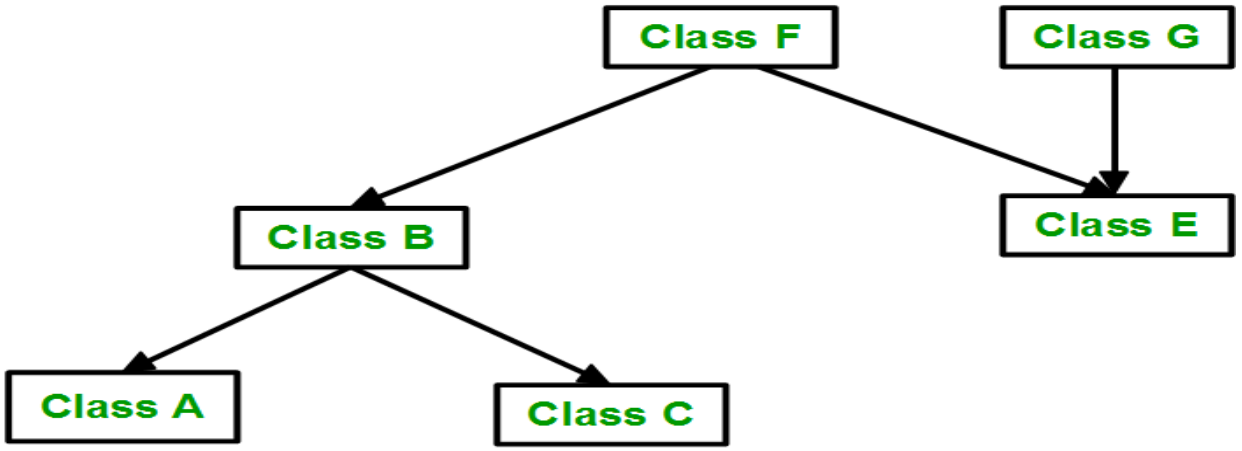
Hierarchical Inheritance

- 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.



Hybrid (Virtual) Inheritance

- Hybrid Inheritance is implemented by combining more than one type of inheritance.
- For example: Combining Hierarchical inheritance and Multiple Inheritance.
  Below image shows the combination of hierarchical and multiple inheritance:



- // C++ ogm to implement single inheritance

```
#include<iostream>
Using namespace std;

class base
{
 public:
      int a;
      void get_a();
 };

 class derived : public base
 {
  public:
     void print(void);
  };
  void base::get_a()
  {
   a=10;
   }
```

```
void derived::print()
  {
  cout<<"the value of a is ="<<a;
  }
  int main()
  {

  derived d;
  d.get_a();
  d.print();

  return 0;
  }
```

Output: the value of a is = 10

Ex: inheritence

```
#include<iostream>
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
```