UNIT-3

Inheritance

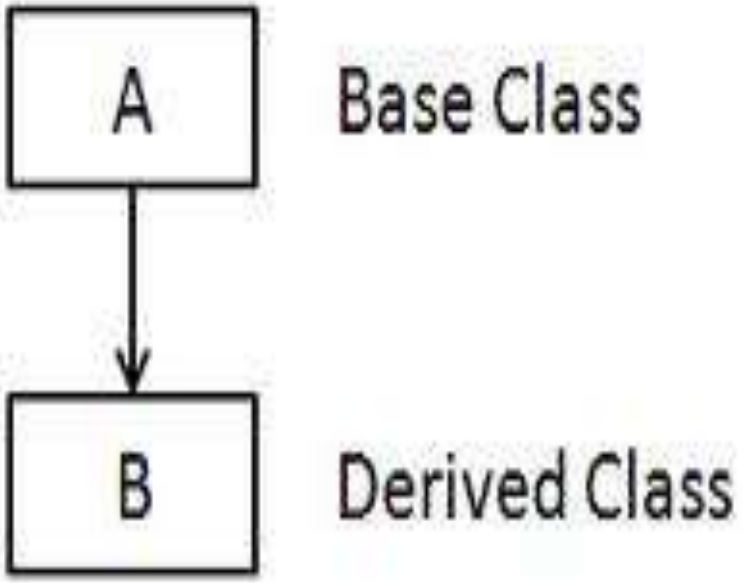
BCA-2nd Sem

Inheritance

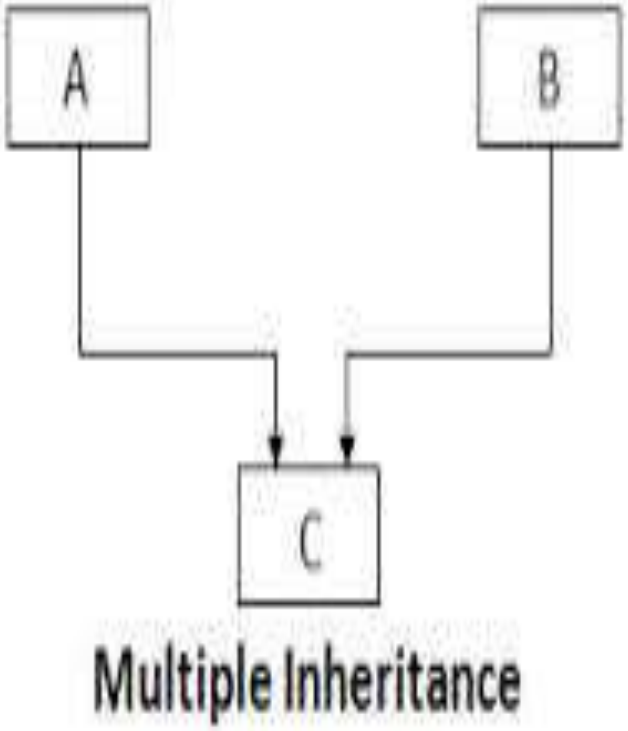
- Inheritance is the process of creating new classes from the existing class or classes.
- Using inheritance, one can create general class that defines traits common to a set of related items.
- This class can then be inherited (reused) by the other classes by using the properties of the existing ones with the addition of its own unique properties.

Forms of Inheritance

1.Single Inheritance: It is the inheritance hierarchy wherein one derived class inherits from one base

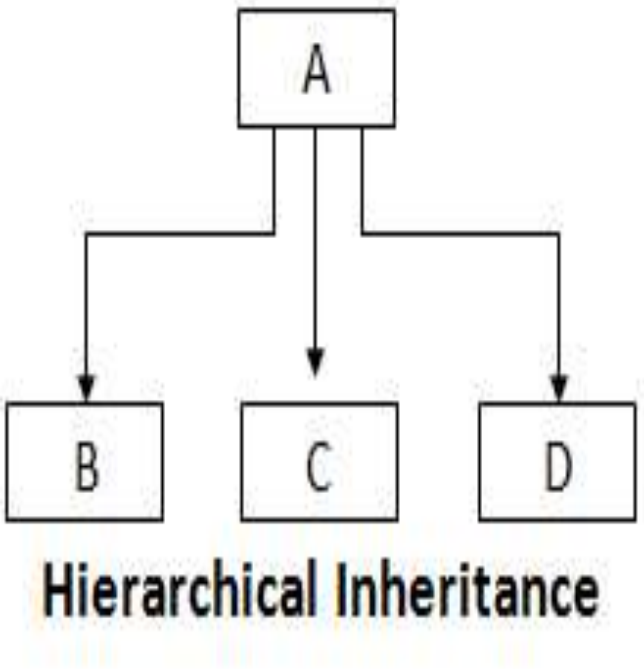


2. Multiple Inheritance:



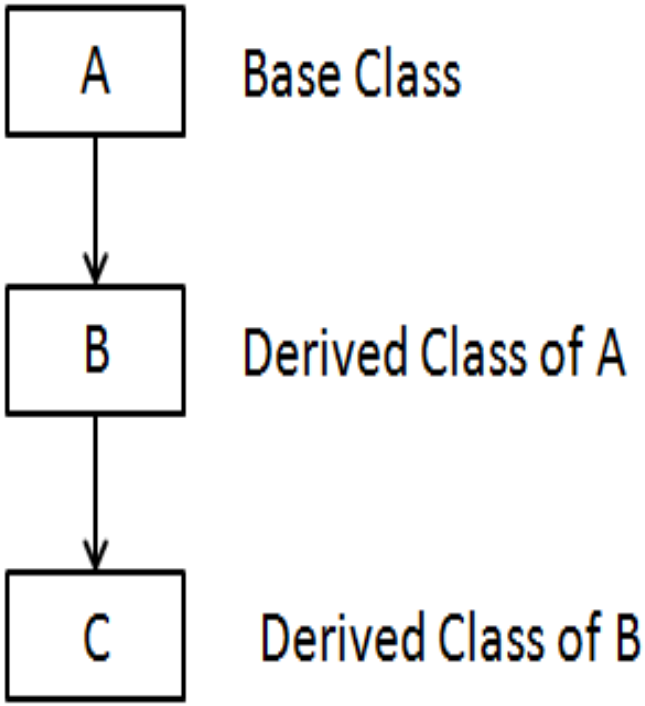
It is the inheritance hierarchy wherein one derived class inherits from multiple base class(es)

3.Hierarchical Inheritance:



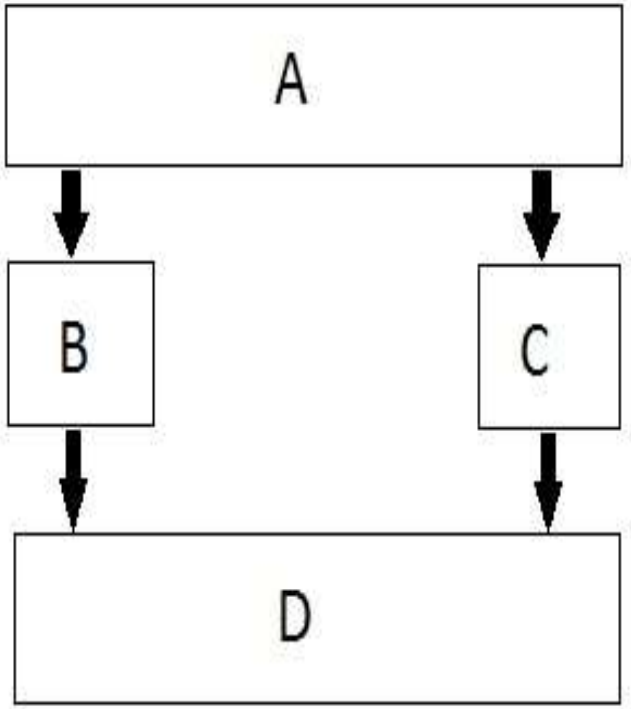
It is the inheritance hierarchy wherein multiple subclasses inherit from one base class.

4.Multilevel Inheritance:



It is the inheritance hierarchy wherein subclass acts as a base class for other classes.

5.Hybrid Inheritance



The inheritance hierarchy that reflects any legal combination of other four types of inheritance.

Advantage Of Inheritance:

1. **Reusability --** facility to use public methods of base class without rewriting the same.

2. **Extensibility --** extending the base class logic as per business logic of the derived class.

3. **Data hiding --** base class can decide to keep some data private so that it cannot be altered by the derived class.

Advantage Of Inheritance:

4. **Overriding--**With inheritance, we will be able to override the methods of the base class so that meaningful implementation of the base class method can be designed in the derived class.

Protected' Access Specifier

- There are 3 access specifiers for a class/struct/Union in C++. These access specifiers define how the members of the class can be accessed.

- Any member of a class is accessible within that class(Inside any member function of that same class). type of access specifiers, they are:

Protected' Access Specifier

- **Public -** The members declared as Public are accessible from outside the Class through an object of the class.

- **Protected -** The members declared as Protected are accessible from outside the class **BUT** only in a class derived from it.

- **Private -** These members are only accessible from within the class. No outside Access is allowed.

Public Inheritance:

1. All Public members of the Base Class become Public Members of the derived class.

2. All Protected members of the Base Class become Protected Members of the Derived Class.

Public Inheritance:

```
Class Base{
 public:
int a;
protected:
int b;
private:
int c;};
class Derived:public Base
{
void doSomething()
```

Public Inheritance:

```
{ a = 10;
 b = 20;
 c = 30;}};
 int main()
 {
 Derived obj;
 obj.a = 10;
obj.b = 20;
 obj.c = 30;
}
```

Function Overriding.

- If we inherit a class into the derived class and provide a definition for one of the base class's function again inside the derived class, then that function is said to be **overridden**, and this mechanism is called **Function Overriding**

Requirements for Overriding

1.Inheritance should be there. Function overriding cannot be done within a class. For this we require a derived class and a base class.

2. Function that is redefined must have exactly the same declaration in both base and derived class, that means same name, same return type and same parameter list.

Example of Function Overriding

```
class Base
 {
 public:
void shaow()
 {
cout << "Base class";
 }};
 class Derived:
```

Example of Function Overriding

```
public Base
 {
 public:
void show()
{
cout << "Derived Class";
 }}
```

REFRENCES

- Learn Programming in C++ By Anshuman Sharma, Anurag Gupta, Dr.Hardeep Singh, Vikram Sharma