Survey on Types of Inheritance Using Object Oriented Programming with C++

Manoj Kumar Sah [1], **Vishal Garg** [2]

[1] Assistant Professor, Dept. of Computer Science and Engineering
Shobhit University, Gangoh, UP, India.

[2] *Student of B.Sc. Computer Science*
Shobhit University, Gangoh, UP, India.

Abstract— Object oriented programming has become a very important programming paradigm. Object oriented programming languages supports the classes, Inheritance, Encapsulation and polymorphism. This takes a detailed look at different types of Inheritance. OOPs come into existence in 1960s through the Simula [1] language. Inheritance is one of the cornerstones of OOP because it allows the creation of hierarchical classifications. Using inheritance, we can create general class that defines traits common to a set of related items. This class may then be inherited by other, more specific classes, each adding only those things that are unique to the inheriting class. When a class inherits another, the members of the base class become members of the derived class.

***Index Terms*—About Base class, Derived class, Inheritance, Reusability, Private, Public, and Protected.**

I. INTRODUCTION

Inheritance is the basic mechanism by which a derived class can inherit the properties of base class. The derived class inherits all behavior of base class. Inheritance provides the reusability of code. Object-oriented programming allows classes to inherit commonly used state and behavior from other classes. Inheritance is the mechanism which allows a class A to inherit properties of a class B. We say ``A inherits from B".The class B is referred to as the Base class and the class A is called the Derived class or Subclass. The relationships of objects or classes through inheritance give rise to a hierarchy. Inheritance was invented in 1967 for Simula.

Advantages of Inheritance:

1. It provides code reusability.
2. It reduces code redundancy
3. It reduces source code size and improves code readability.
4. By using inheritance, code becomes easy to manage and divided into parent and child classes
5. It supports code extensibility by overriding the base class functionality within child class.

Manuscript received Dec, 2014.

***Manoj Kumar Sah**, Asst. Prof. Dept. of Computer Science and Engineering, Shobhit University, Gangoh, UP, India*
(E-mail: eindiacse@gamil.com).
Vishal Gar, *He is a student of B.Sc. Computer Science in Shobhit University, Gangoh, UP, India (e-mail: vishalsiet022@gmail.com).*

Disadvantages of Inheritance:

1. In inheritance, super class and sub class are tightly coupled, If we change the code of super class it will get affects to the all the sub classes.

2. In multiple inheritances, if both the super and sub class have same function with same signature, Ambiguity arises in sub class.

For Example, point and circle, we can define a circle which inherits from point:

```
Class Circle inherits from Point
    {
   Attributes:
    int radius;
  methods:
    set Radius (int new Radius);
    get Radius ();
    }
```

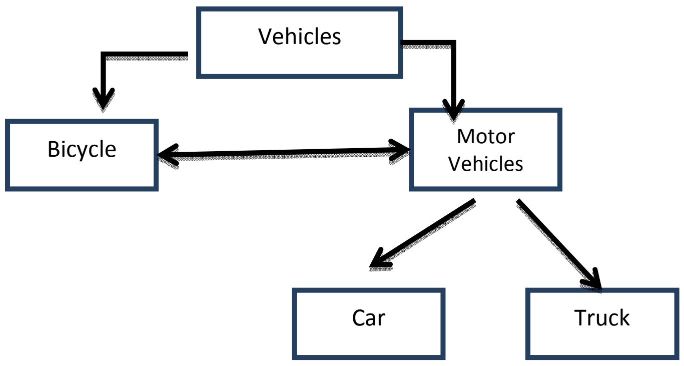


Fig 1: Diagram Illustration

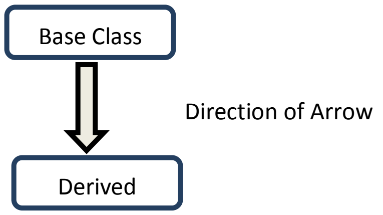
Diagram illustrates the Vehicle as a Base Class or Super Class and the Motorized vehicle and Bicycle are two Derived Class. In this diagram there are two more Derived Class that are; Car and truck. These are the Derived Class of Motorized vehicle as well as Vehicle but the Motorized vehicle is referred as Base Class for the Derived Class Car and truck.

II. TYPES OF INHERITANCE

In this paper we have considered the following types of inheritance:

i. Single Inheritance
ii. Multiple Inheritance
iii. Hierarchical Inheritance
iv. Multilevel Inheritance
v. Hybrid Inheritance

1 Single Level Inheritance:
A derived class with only one base class is called single inheritance. In single level inheritance the subclass inherits variables and methods that are declared by the super class, although some of these variables and methods may not be accessible by the subclass.



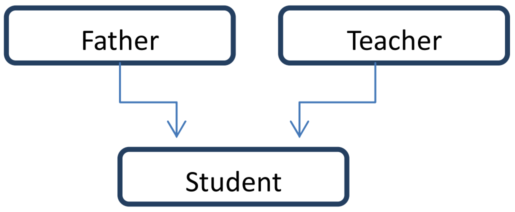
Syntax:

```
Class A
      {
     Statements;
      };
Class B: Public A
      {
      Statements;
      };
```

Example

```
#include<iostream.h>
class A
{
public:
inta,b;
void student()
{
cout<<"Enter the value a";
cin>>a;
cout<<"Enter the value b";
cin>>b;
}
void print()
{
cout<<"a="<<a;
cout<<"b="<<b;
}
};
class B: Public A
{
public:
void sum()
{
cout<<"Sum:"<<(a+b);
}
};
void main()
{
B obj;
obj.student();
obj.print();
obj.sum();
 }
```

2 Multiple Inheritances: One important object-oriented mechanism is multiple inheritances. Multiple inheritances mean that one subclass can have more than one superclass. Multiple inheritances allows us to combine the features of several existing classes as a starting point for defining new classes.Multiple inheritance in languages with C++ style constructors impairs the inheritance problem of constructors and constructor chaining, thereby creating maintenance and extensibility problems in these languages.



Syntax:
Class A: Access specifierBAccess specifier C,……….,B_n

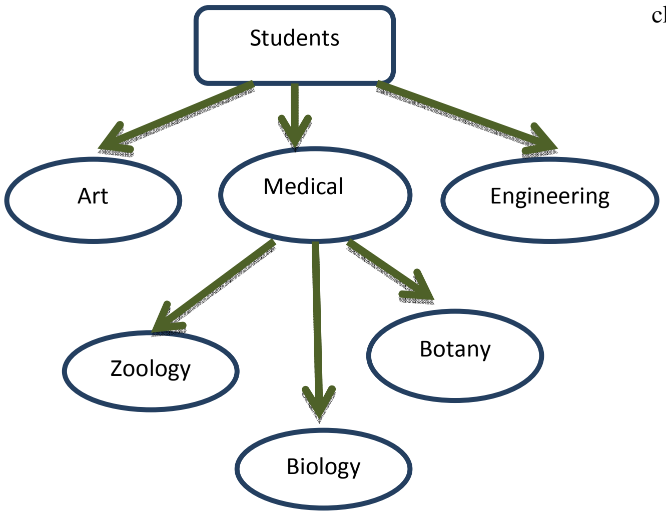
```
{
Statements;
};
```

Example:

```
#include<iostream.h>
#include<conio.h>
class student
{
   protected:
     int rno,m1,m2;
   public:
             void get()
            {
                        cout<<"Enter the Roll no :";
                        cin>>rno;
                        cout<<"Enter the two marks   :";
                        cin>>m1>>m2;
            }
};
class sports
{
   protected:
     int sm;
   public:
             void getsm()
            {
              cout<<"\nEnter the sports mark :";
              cin>>sm;
             }
};
class statement : public student, public sports
{
   int tot, avg;
   public:
   void display()
          {
            tot=(m1+m2+sm);
            avg=tot/3;
 cout<<"\n\n\tRoll No    : "<<rno<<"\n\tTotal      : "<<tot;
 cout<<"\n\tAverage    : "<<avg;
          }
};
void main()
{
  statement obj;
  obj.get();
  obj.getsm();
  obj.display();
  getch();
}
```

3 Hierarchical Inheritances:

When the properties of one class are inherited by more than one class, it is called hierarchical inheritance. In the hierarchical inheritance the Base class will include a the features that are common to the Subclasses. A Subclass can be constructed by inheriting the properties of the Base class.



In the above diagram there is a Base Class named Students and 3 Derived Class as we can in the diagram. There are3 another derived class also derived from the Base Class Medical.

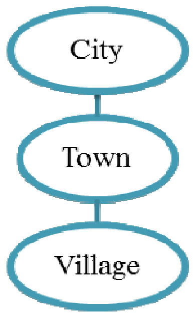
Syntax:

```
class A
    {
               -------
    };

   class B : public A
    {
               -------
    };

 class C : public A
    {
               -------
    };

class D:public B
    {
               -------
    };

   class E: public B
    {
               -------
    };

   class F: public C
    {
               -------
    };

   class G: public C
    {
               -------
    };
```

Example:

```
#include <iostream.h>
#include<conio.h>
class Side
        {
        protected:
        int i;
        public:
                void set_values (int x)
                        { i=x;}
        };
                class Square: public Side
                        {
                        public:
                                int sq()
                                {
                                return (i*i);
                                }
                        };
                class Cube:public Side
                        {
                        public:
                                int cub()
                                {
                               return (i*i*i);
                                }
                        };
  void main ()
        {
        Square  s;
        s.set_values (10);
        cout << "The square value is    " << s.sq();
        Cube c;
        c.set_values (20) ;
        cout << "The cube value is   " << c.cub() ;
        getch();
        }
```

4 Multilevel Inheritances:
It is a type of inheritance in which a class is derived from another derived class is called multilevel inheritance.
In multilevel inheritance, the members of Base Class are inherited to the Derived Class and the member of Derived Class is inherited to the grand Derived Class.



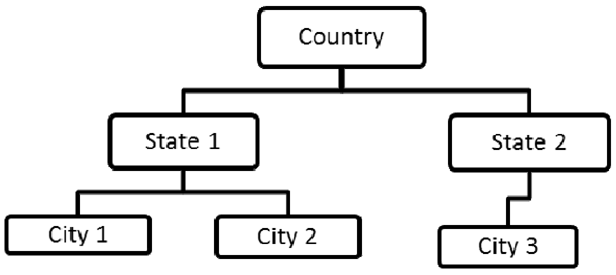
Syntax:

```
Class A
        {
        Statement1;
        Statement2;
        };
Class B
  {
  Statement3;
  Statement4;
  };
Class C: Public B
  {
  Statement5;
  Statement6;
  };
```

Example:

```
#include<iostream.h>
class A
{
protected:
int x;
public:
void showA()
{
cout<<"enter a value for x:"<<endl;
cin>>x;
}
};
class B:public A
{
protected:
int y;
public:
void showB()
{
cout<<"enter value for y:";
cin>>y;
}
};
class C:public B
{
public:
void showC()
{
showA();
showB();
cout<<"x*y ="<<x*y;
}
};
void main()
{
C ob1;
ob1.showc();
}
```

5 Hybrid Inheritance: Hybrid Inheritance is a method where one or more types of inheritance are combined together and used. Since .NET Languages like C#, F# etc. does not support multiple and multipath inheritance. Hence hybrid inheritance with a combination of multiple or multipath inheritance is not supported by .NET Languages.



Syntax:

```
Class A
{
Statements;
};
Class B: Public A
{
Statements;
};
Class C: Public B
{
Statements;
};
Class D: Public B
{
Statement;
};
```

Example:

```
#include<iostream.h>
#include<conio.h>
class base
        {
```

```
        int a;
        public:
        base()
                {
                cout<<"Enter a: ";
                cin>>a;
                }
        int show()
                {
                cout<<"a = "<<a<<endl;
                return(a);
                }
        ~base()
                {
        cout<<"Destructor of base class is executed";
                }
        };
        class derived1:public base
                {
                int b;
                        public:
                        derived1():base()
                                {
                                cout<<"Enter b: ";
                                cin>>b;
                                }
                int show1()
                        {
                        cout<<"b = "<<b;
                        return(b);
                        }
                ~derived1()
                        {
    cout<<"Destructor of derived1 class is executed";
                        }
                };
                class derived2:public derived1
                {
                int a,b,c,sum;
                public:
                        derived2():derived1()
                        {
                        cout<<"Enter c: ";
                        cin>>c;
                        }
        void show2()
                {
                a=show();
                b=show1();
                cout<<"c = "<<c<<endl;
                sum=a+b+c;
        cout<<"Sum of given numbers: "<<sum;
                }
        ~derived2()
                {
       cout<<"Destructor of derived2 class is executed";
                }
        };
        class derived3:public derived1
                {
                int a,b,c,sum;
                public:
                        derived3():derived1()
                        {
                        cout<<"Enter c: ";
                        cin>>c;
                        };
                void show3()
                        {
                        cout<<"c = "<<c;
                        a=show();
                        b=show1();
                        sum=a+b+c;
                cout<<"Sum of given numbers: "<<sum;
                        }
                ~derived3()
                        {
        cout<<"Destructor of derived3 class is executed";
                        }
                };
        void main()
        {
        cout<<"Getting data for first object";
        {
        derived3 d3;
        d3.show3();
        }
        cout<<"Enter data for second object";
        {
        derived2 d2;
        d2.show2();
        }
        getch();
        }
```

III. RESULTS:

C++	**Supports**
Object Oriented	Hybrid
Static / Dynamic Typing	Static
Inheritance	Multiple
Method Overloading	Yes
Operator Overloading	Yes
Generic Class	Yes
Dynamic Binding	Yes (Static by Default)

Base Class Member Access Specifier	Types of Inheritance		
	public	protected	private
public	public	protected	private
protected	protected	protected	private
private	Not accessible -HIDDEN	Not accessible -HIDDEN	Not accessible -HIDDEN

IV. Conclusion:

Here, in this paper we have to study the above five types of inheritance. We have to find that inheritance is central concepts in C++ that allows deriving a class from multiple classes at a time. Inheritance helps the programmer to reuse the code in many situations if it needed. Using this concept the programmer can create as many derived class from the base class. In inheritance we can extend the base class logic as per our business requirement. If we have mouse programming and we want to extend some functionality in mouse programming as mouse pointer should be red, then by use of inheritance we can easily implement this mouse programming in such a way that its pointer or cursor color becomes red. Inheritance helps in company to extend business Logic.

A. *References*

[1] Bjarne Stroustrup, the C++ Programming Language, Person Publication.

[2] Kim B. Bruce. Foundations of object-oriented languages: types and semantics. MIT Press, Cambridge, MA, USA, 2002.

[3] Kyle Loudon, C++ Pocket Reference 1st Edition, O'reilly Publication.

[4] E Balagurusamy, Object oriented Programming with C++, 6th Edition, New Delhi: Tata McGraw-Hill

[5] The C++ Standard Library: A Tutorial and Reference (Jousts, 2000)

[6] Advanced C++ Programming Styles and Idioms (Coplien, 1992)

[7] Yashavant Kanetkar, Test your C++ Skills, 1st Edition, BPB Publication.

Author Profile :

. [1] Mr**. Manoj Kumar Sah**, He received his M. Tech. in year 2014 (Computer Science and Engineering) from Indian School of Mines (I.S.M. Dhanbad), Jharkhand, India. His interests are Wireless Sensor Network, Data Base Management System and Algorithm. Now he is working as Asst. Prof in Dept. of Computer Science and Engineering, Shobhit University, Gangoh, UP, India.

[2] Mr. **Vishal Garg**, *He is a student of B.Sc. Computer Science in Shobhit University, Gangoh, UP, India (e-mail: vishalsiet022@gmail.com).*