Inheritance and its type in Object Oriented Programming using C++

Mehul Patel, R. D. Modi, S. M. Pillai

Abstract*: Reusability is one in every of most vital advantage of C++ programming language. C++ categories are often reused in many ways in which. Once the parent (Base) category has been written it are often changed by another technologist to suit their needs. the most plan of inheritance is making new categories, reusing the properties of the present base category. The mechanism of etymologizing a replacement category (Child/Derived Class) from associate Existing category (Base/Parent Class) is termed inheritance. The previous category is stated because the base (Parent) category and therefore the new category is termed the derived category (Child) or taxonomic group. A derived category includes all options of the generic base category so adds qualities specific to the derived category. This paper reflects the learning of the Inheritance conception and its varieties victimization C++ (oops)

Keywords: Base (Parent) class, Reusability, – Sub (Derived/Child) Class, Visibility Modes and Types of Inheritance

I. INTRODUCTION

Inheritance is that the method by that objects of 1 category acquires the properties of objects of another category within the hierarchy. The capability of a class to derive properties and characteristics from another class is termed Inheritance. Inheritance is one in all the foremost vital feature of Object familiarized Programming.

A. Sub Class:

The category that inherits properties from another category is termed Sub category or Derived class.

B. Super Class:

The category whose properties square measure transmissible by sub category is termed Base category or Super class. Sub categories will be created from the prevailing categories. It implies that we will add further options to a Base category while not modifying it. The new category is referred as derived category or taxonomic group and also the original category is understood as base categories or super category.

II. LITERATURE REVIEW

Suvarnalata Hiremath & C M Tavade (May2016), in keeping with Author "Review Paper on Inheritance and problems in Object homeward Languages", the target of this analysis paper is to review idea of inheritance in object homeward languages.

Revised Manuscript Received on November 15, 2019.

Dr. Mehul Patel, Assistant Professor, C. P. Patel & F. H. Shah Commerce College, Anand. E-mail: mehul29978@gmail.com

Dr. R.D.Modi, Principal, C. P. Patel & F. H. Shah Commerce College, Anand. E-mail:drrdmodi@gmail.com

Dr.S.M.Pillai, IQAC Co-ordinator, C. P. Patel & F. H. Shah Commerce College, Anand. E-mail: Sajenapillai.sp@gmail.com

The review paper begins upon the survey of inheritance and reusability of object homeward language. Inheritance plays a very important role for code reusability. Since object homeward has been wide acclaimed because the technology that may support creation of reusable code, significantly owing to the inheritance feature. Then discuss a brand new approach of inheritance mechanism, that overcomes the encapsulation problems and alternative problems derived from inheritance, that compromises severely reusability in object homeward language and conjointly we have a tendency to explore the affiliation between inheritance and code reusability.

Shyamapriya Chowdhury & Sudip Chatterjee (February 2016). during this analysis paper "A Thorough Learning of various styles of Inheritance Victimization Object homeward Programming with JAVA" the authors demonstrate the idea of inheritance in our way of life. Creation of a brand new category from associate degree existing one is named inheritance. while not modifying the previous knowledge new options are often further in an exceedingly category. The recently created category is named kid category and from that it's created is thought as parent category. a bit like soul, kid category will mechanically access all the properties of Parent category and new strategies may be entered in kid category. idea of reusability is directly supported by JAVA victimization this inheritance.

Bjarne Stroustrup, he wrote analysis paper titled Multiple Inheritance for C++, during this analysis paper the author describe that Multiple Inheritance is that the ability of a class to possess quite one base class (super class). in an exceedingly language wherever multiple inheritance is supported a program are often structured as a collection of inheritance lattices rather than (just) as a collection of inheritance trees. this can be wide believed to be a very important structuring tool. it's conjointly wide believed that multiple inheritance complicates a programing language considerably, is difficult to implement, and is pricey to run. i'll demonstrate that none of those last 3 conjectures square measure true

Generalized Syntax for inheritance

```
class parent class
{
        visibility mode:
        data member declaration;
        member function declaration;
        .       .       .
        .       .       .
        .       .       .
};
class derived class : visibility mode base class name
{
```

```
        visibility mode:
                data member declaration;
                member function declaration;

        .       .       .
        .       .       .
        .   .   .
};
```

Visibility mode is employed within the inheritance of C++ to point out or relate however base categories are viewed with reference to derived category. once one category gets transmitted from another, visibility mode is employed to inherit all the general public and guarded members of the bottom category. personal members ne'er get transmitted and therefore don't participate in visibility. By default, visibility mode remains "private".

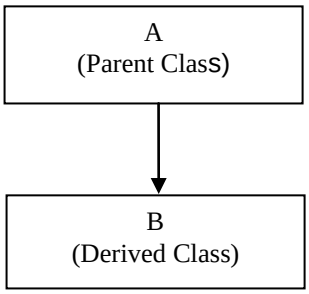
Types of Inheritance

1. Single Level Inheritance
2. Multiple Inheritance
3. ranked inheritance
4. structure Inheritance
5. Hybrid Inheritance.

1. Single Level Inheritance

If a derived category is made from just one base category, then such a inheritance is named single level inheritance.

Consider a simple example of single inheritance.



(Figure: 1 - Single Level Inheritance)

The above diagram shows single inheritance. Class A is parent class and class B is consider as a Derived class. Class B has all the properties of its own as well as Base Class (i.e. A)

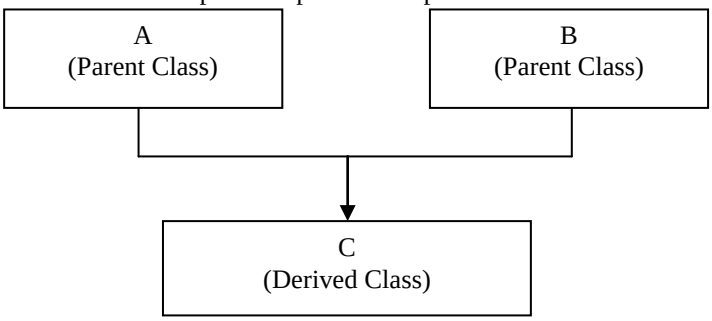
Example of single level inheritance

```
#include <iostream.h>
class A
{
        public:
                int y;
                void read()
                {
                        cout<<"Enter Y : ";
                        cin>>y;
                }
                void print()
                {
                        cout<<"\n Y = "<<y;
                }
};
class B: public A
{
        int p;
        public:
                void read1()
                {
                        read();
                        cout<<"Enter P : ";
                        cin>>p;
                }
                void print1()
                {
                        print();
                        cout<<"\n P = "<<p;
                }
};
void main()
{
        B b;
        clrscr();
        b.read1();
        b.print1();
        getch();
}
Output :
Enter Y : 12
Enter  P : 15
Y = 12
P = 15
```

2. Multiple Inheritance

If a derived class is created from more than one base class, then such an inheritance is called multiple inheritance.

Consider a simple example of Multiple Inheritance.



(Figure: 2 - Multiple Inheritances)

The above diagram shows multiple inheritance. Class A is Parent class and class B is also a parent class and class C is created from two parent class A & B. So class C is called Derived class. Class C has all the properties of its own as well as class A and class B.

Multiple inheritances allow us to merge the features of several parent classes as a starting point for defining new classes. It is just like a child inherits some properties of father and mother both.

Example of Multiple Inheritances

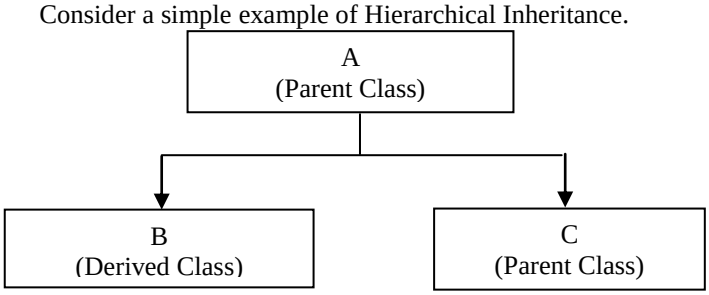
```
#include <iostream.h>
class A
{
public:
int x;
void read()
{
cout<<"Enter X : ";
cin>>x;
}
void print()
{
cout<<"\n X = "<<x;
}
};
class B
{
public:
int y;
void read1()
{
cout<<"Enter Y : ";
cin>>y;
}
void print1()
{
cout<<"\n Y = "<<y;
}
};

class C: public A, public B
{
int z;
public:
void read2()
{
read();
read1();
cout<<"Enter Z : ";
cin>>z;
}
void print2()
{
print();
print1();
cout<<"\n Z = "<<z;
}
};
void main()
{
C c;
clrscr();
c.read2();
c.print2();
getch();
}

Output :
Enter  X : 12
Enter  Y : 15
```

```
Enter  Z :  18
X = 12
Y = 15
Z  =  18
```

3. Hierarchical Inheritance

If more than one derived classes are created from same parent class, then such an inheritance is called Hierarchical inheritance.

Consider a simple example of Hierarchical Inheritance.



(Figure: 3 - Hierarchical Inheritance)

The above diagram shows hierarchical inheritance. Class A is parent class and class B and class C are derived class. Class B has properties of its own and class A. Class C has all the properties of its own as well as class A.

Example of Hierarchical Inheritance

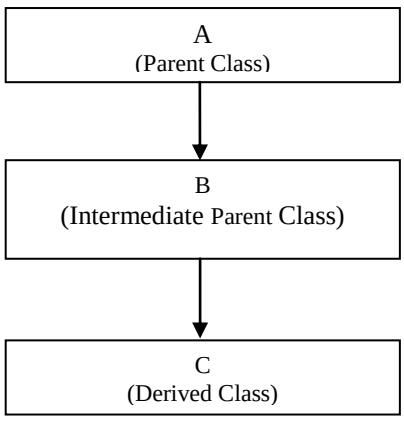
```
#include <iostream.h>
class A
{
        public:
                int x;
                void read()
                {
                        cout<<"Enter X : ";
                        cin>>x;
                }
                void print()
                {
                        cout<<"\n X = "<<x;
                }
};
class B:public A
{
        public:
                int y;
                void read1()
                {
                        read();
                        cout<<"Enter Y : ";
                        cin>>y;
                }
                void print1()
                {
                        print();
                        cout<<"\n Y = "<<y;
                }
};
class C: public A
{
```

```
        int z;
        public:
                void read2()
                {
                        read();
                        cout<<"Enter Z : ";
                        cin>>z;
                }
                void print2()
                {
                        print();
                        cout<<"\n Z = "<<z;
                }
};
void main()
{
        B b;
        C c;
        clrscr();
        b.read1();
        c.read2();
        b.print1();
        c.print2();
        getch();
}

Output :
Enter  X : 5
Enter  Y : 11

Enter  X :  20
Enter  Z :  25
X = 5
Y = 11

X = 20
Z  =  25
```

4. Multi Level Inheritance

If a derived class is created from another derived class (intermediate base class) then such an inheritance is called multilevel inheritance.

Consider a simple example of Multi level Inheritance.



(Figure: 4 - Multilevel Inheritance)

The above diagram shows multilevel inheritance. Class A is Base class and class B is created from class A. Class B has properties of its own and class A. Class C is created from class B(intermediate base class) so class C has the properties of its own as well as class A and class B.

Example of Multilevel Inheritance

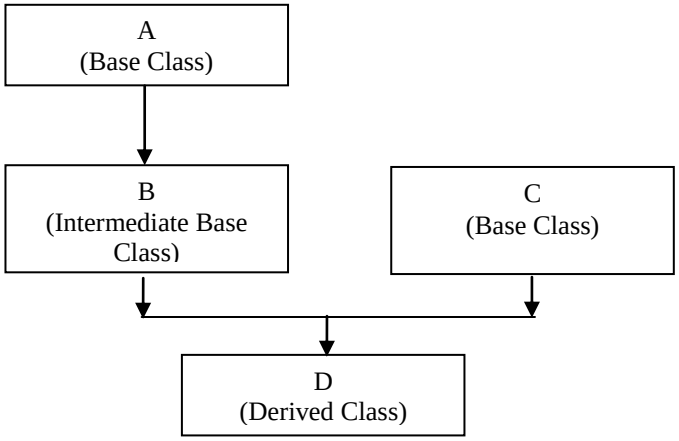
```
#include <iostream.h>
class A
{
        public:
                int x;
                void read()
                {
                        cout<<"Enter X : ";
                        cin>>x;
                }
                void print()
                {
                        cout<<"\n X = "<<x;
                }
};
class B:public A
{
        public:
                int y;
                void read1()
                {
                        read();
                        cout<<"Enter Y : ";
                        cin>>y;
                }
                void print1()
                {
                        print();
                        cout<<"\n Y = "<<y;
                }
};
class C: public B
{
        int z;
        public:

                void read2()
                {
                        read1();
                        cout<<"Enter Z : ";
                        cin>>z;
                }
                void print2()
                {
                        print1();
                        cout<<"\n Z = "<<z;
                }
};
void main()
{
        C c;
        clrscr();
        c.read2();
        c.print2();
        getch();
}

Output :
Enter  X : 10
Enter  Y : 20
```

```
Enter  Z :  30
X = 10
Y = 20
Z  =  30
```

5. Hybrid Inheritance

Combination of one or more inheritance is known as Hybrid Inheritance.

Consider a simple example of hybrid inheritance.



(Figure: 5 - Hybrid Inheritance)

Accessibility in Public Inheritance

Accessibility	private variables	protected variables	public variables
Accessible from own class?	yes	yes	yes
Accessible from derived class?	no	yes	yes
Accessible from 2nd derived class?	no	yes	yes

Accessibility in Protected Inheritance

Accessibility	private variables	protected variables	public variables
Accessible from own class?	yes	yes	yes
Accessible from derived class?	no	yes	yes (inherited as protected variables)
Accessible from 2nd derived class?	no	yes	yes

Accessibility in Private Inheritance

Accessibility	private variables	protected variables	public variables
Accessible from own class?	yes	yes	yes
Accessible from	no	yes	yes
derived class?		(inherited as private variables)	(inherited as private variables)
Accessible from 2nd derived class?	no	no	no

III. CONCLUSION

The mechanism (process) of account a replacement category from existing (old) category is termed inheritance, by mistreatment inheritance we will utilize the options of existing category which is that the most vital idea in C++. All the inheritance has its own options and its use to produce users to reusability ideas powerfully, to save lots of time and cut back the complexness. In this paper we've to check the higher than 5 styles of inheritance. We have to search out that inheritance is central ideas in C++ that permits account a category from multiple categories at a time.

REFERENCES

1. b3e6ba44.pdf
2. Bjarne Stroustrup, The C++ Programming Language, 4th edition, Pearson Education Inc.
3. E Balagurusamy, Object oriented Programming with C++, 6th Edition, New Delhi: Tata McGraw-Hill Publishing Company Limited.
4. http://www.ijarcsms.com/docs/paper/volume1/issue2/V1I2-0005.pdf
5. https://pdfs.semanticscholar.org/a61d/a617de6cc75ae1bbbc0b02bea7ba
6. https://www.geeksforgeeks.org/inheritance-in-c/
7. https://www.programiz.com/cpp-programming/public-protected-private-inheritance
8. https://www.researchgate.net/profile/Bjarne_Stroustrup/publication/2396782
9. Multiple_Inheritance_for_C/links/00b7d514319dc8875a000000/Multiple-Inheritance-for-C.pdf
10. www.ijarcsms.com
11. www.tutorial-cpp-programming.blogspot.com
12. www.ijaetmas.com
13. www.ijarcsse.com
14. www.designurge.com
15. www.coursehero.com
16. www.w3schools.in
17. www.geeksforgeeks.org
18. www.sonalikothari.com
19. E. S. F. Najumudheen, Rajib Mall, Debasis Samanta. "Test coverage analysis based on an object-oriented program model", Journal of Software Maintenance and Evolution: Research and Practice, 2011
20. citeseerx.ist.psu.edu
21. docplayer.net
22. www.computernotes.in
23. pankajtiwarii.blogspot.com
24. www.rgcetpdy.ac.in I
25. marifaty.blog
26. cphstl.dk

AUTHORS PROFILE

Dr. Mehul Patel is the person who has knowledge of various subjects. He has completed his Ph. D, MBA, MCA and M. Sc in Value Education & Spirituality and Various Certificate Courses. Presently he is serving in C.P.Patel & F.H.Shah Commerce College, ANAND since 3 Years. He has 20 years experience in various categories like research, system analysis, and customization at user end. He taught various faculties like computer science, management, statistics, and mathematics and operation research. He produced **24 Research papers, 3 books** published and attended 29 Seminar/Workshops. He is **review member** in various international journals like JETIR, IJCRT, IARA and IJSRSET. As well as **Editor in International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET)**, Print ISSN: 2395-1990, Online ISSN: 2394-4099, **Associate Editor** of International Journal of Advance & Innovative Research (ISSN: 2394-7780) and International Journal of Research in Management and Social Sciences (ISSN: 2322 – 0899). **Associate Editor** of International Journal of Commerce & Management Research ISSN: 2455-1627, **Associate Editor** of National Journal of Multidisciplinary Research & Development ISSN**:** 2455-9040, and **Editorial Member in** International Journal for Innovative Research in Multidisciplinary Field (ISSN: 2455-0620) & International Journal of Research Culture Society (ISSN: 2456-6683). All are UGC Approved.