INHERITANCE

Uday Kumar, Ayush Jain
Student, B.Tech, Cs-It
Dronacharya College Of Engineering, Gurgaon

Abstract- Inheritance has proved its importance in software development as modification of program becomes very easy through the help of inheritance. So in this research paper we're discussing about inheritance, types of inheritance like single inheritance, multiple inheritance, Multi-Level inheritance, multi-path inheritance, hierarchical inheritance, hybrid inheritance, examples of inheritance and applications of inheritance.

Index Terms- Inheritance introduction, types of inheritance, examples of inheritance, applications of inheritance.

I. INTRODUCTION

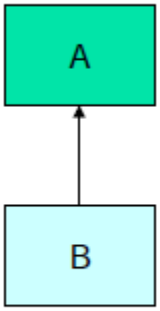
One of the fundamental ideas behind object-oriented programming is that code should be reusable. However, existing code often does not do exactly what you need it to. Perhaps the most obvious way to proceed is to change the existing code to do what you want. However, if we do this, we will no longer be able to use it for it's original purpose, so this is rarely a good idea.

A slightly better idea is to make a copy of some or all of the existing code and change it to do what we want. This mechanism of deriving a new class from existing class is called "inheritance". The old class is known as "base" class, "super" class or "parent" class"; and the new class is known as "sub" class, "derived" class, or "child" class.

The inheritance allows subclasses to inherit all properties (variables and methods) of their parent classes.

II. SINGLE INHERITANCE

In single inheritance onle one class per derived class exists and single inheritance requares a small amount of run time overhead when used with dynamic binding.



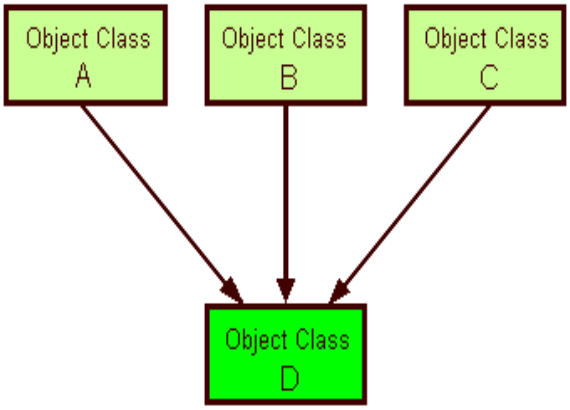
(a) Single Inheritance

Syntax:
class A
{/* ... stuff here ... */};
class B: [access-specifier] A
{/* ... stuff here ... */};

III. MULTIPLE INHERITANCE

A class can inherit properties from more than one class which is known as multiple inheritances. This form of inheritance can have several super classes i.e. multiple Inheritance is the ability of a class to have more than one base class (super class).

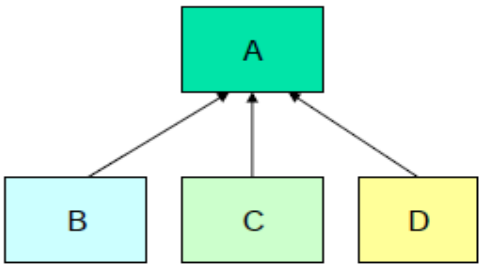
In multiple inheritance we can combine classes into new class to inherit the 'useful' parts and redefine whatever needs to be redefined from which we can reuse existing classes.



Syntax of multiple inheritance is:
Class D: visibility A, visibility B …….
{
…….. //(body of D)
};

IV. HIERARCHICAL INHERITANCE

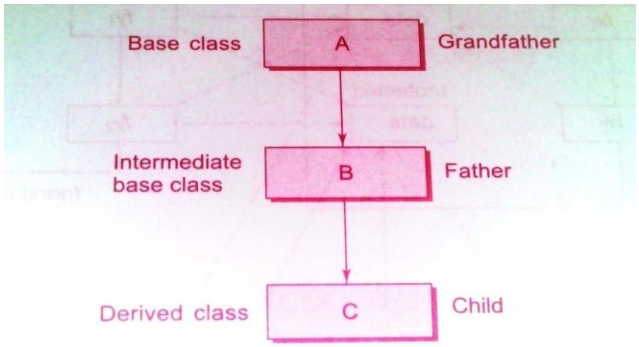
When we derive many classes from one class it is known as hierarchical inheritance. In Hierarchical inheritance the base class will include all the features that are common to the subclasses. A subclass can be constructed by inheriting the properties of the base class, also a subclass can serve as a base class for the lower level classes and so on.



(c) Hierarchical Inheritance

V. MULTILEVEL INHERITANCE

A class can be derived from another derived class which is known as multilevel inheritance. In this figure class A serves as a base class for derived class B, which in turn serves as a base class for derived class C. The class B is known as intermediate base class since it provides link for inheritance between A and C.



Multilevel inheritance is declared as:
Class A
{

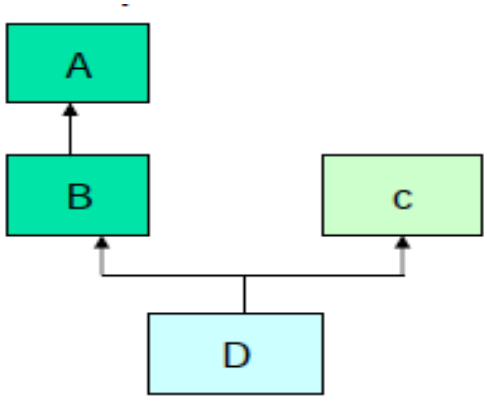
}; //base class
Class B: public A
{

}; //B derived from A
Class C: public B
{

}; //C derived from B

VI. HYBRID INHERITANCE

When we apply more then one type of inheritance to derive a inherited class it is known as Hybrid Inheritance.



(b) Hybrid Inheritance

Declaration is:
Class D : public B, public C
{

};

Where class B itself is derived from A,

```
Class B : public A
{
----------
----------
};
```

VII. APPLICATIONS OF INHERITANCE

- Helps us to break software into manageable pieces.
- Existing classes can be morphed to design new classes i.e. code reuse.
- Enables us to group different types of objects together and do some action on all of them.
- Usually this allows a form of specialization of one type of object from the one being inherited.
- Setting up a protocol. A class can declare a number of methods that its subclasses are expected to implement. The class might have empty versions of the methods, or it might implement partial versions that are to be incorporated into the subclass methods. In either case, its declarations establish a protocol that all its subclasses must follow.

VIII. CONCLUSION

Since the friend functions and friend classes of the base class are not inherited during using any type of inheritance so we can program a inheritance class in which we can inherit properties of friend function and friend classes.

REFRENCES

[1]: http://www.udtallas.edu
[2]: http://www.ijarcsms.com
[3]: http://www.cloudbus.org
[3]: https://developer.apple.com
[4]: http://www.dre.vanderbilt.edu
[5]: E-balaguruswami
[6]: http://www.compsci.hunter.cuny.edu