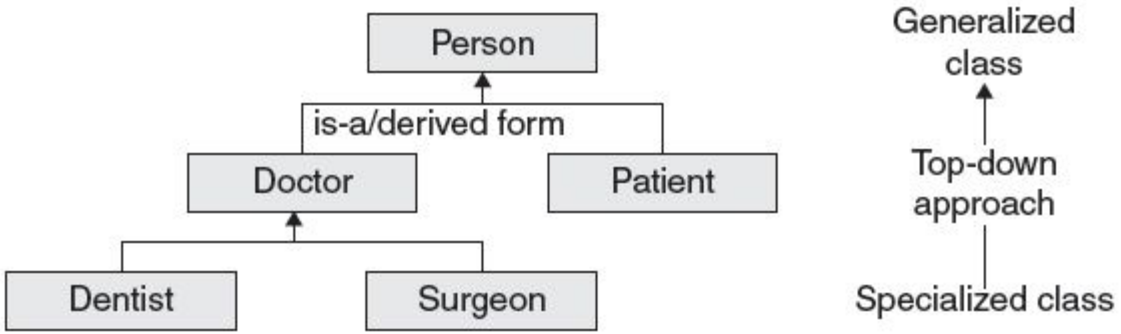
C++
Inheritance

1. Introduction - I

- **The technique of creating a new class from an existing class is called inheritance.**
- **The old or existing class is called the base class and the new class is known as the derived class or sub-class.**
- **The derived classes are created by first inheriting the data and methods of the base class and then adding new specialized data and functions in it.**
- **During the process of inheritance, the base class remains unchanged.**



1. Introduction - II

- Like function overloading, operator overloading is also a form of compile time polymorphism.

- Operator overloading is, therefore, less commonly known as operator ad hoc polymorphism .

- Since different operators have different implementations depending on their arguments. Operator overloading is generally defined by the language, the programmer, or both.

1. Introduction - III

- Private is the highest level of data hiding. When a base class is privately inherited by a derived class, then public members of the base class become private members of the derived class.

- The private members cannot be inherited but the derived class can access them using public member functions of the base class.

- This means that the object of a privately inherited class cannot access the inherited members.

1. Introduction - V

- **Public is the lowest and the most open level of data hiding.**

- **When a base class is publicly inherited by a derived class, then public members of the base class become public members of the derived class.**

- **The private members cannot be inherited but the derived class can access them using public member functions of the base class.**

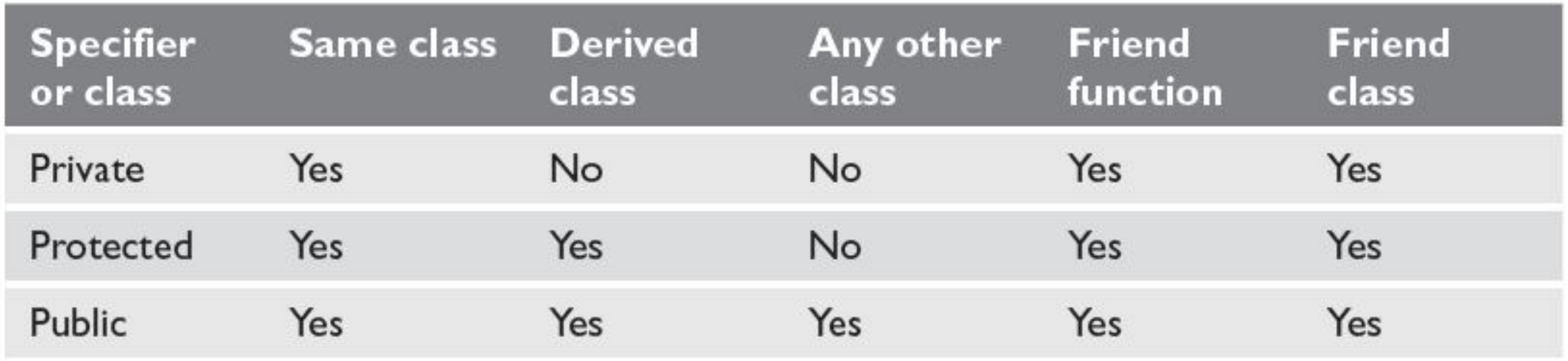
1. Introduction - VI

- **A protected specifier lies between private and public .**

- **A data member or a member function declared as protected can be accessed by the class in which it is defined—as in private —and the class which is immediately derived from it.**

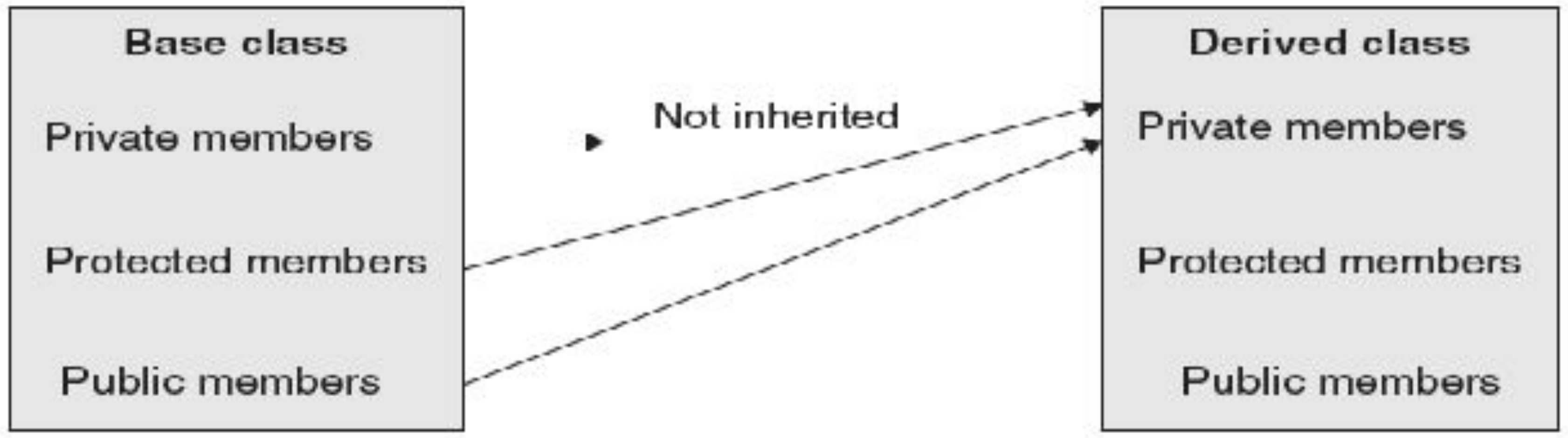
- **No other class except these two can access the protected members of a class.**

1. Introduction - VII

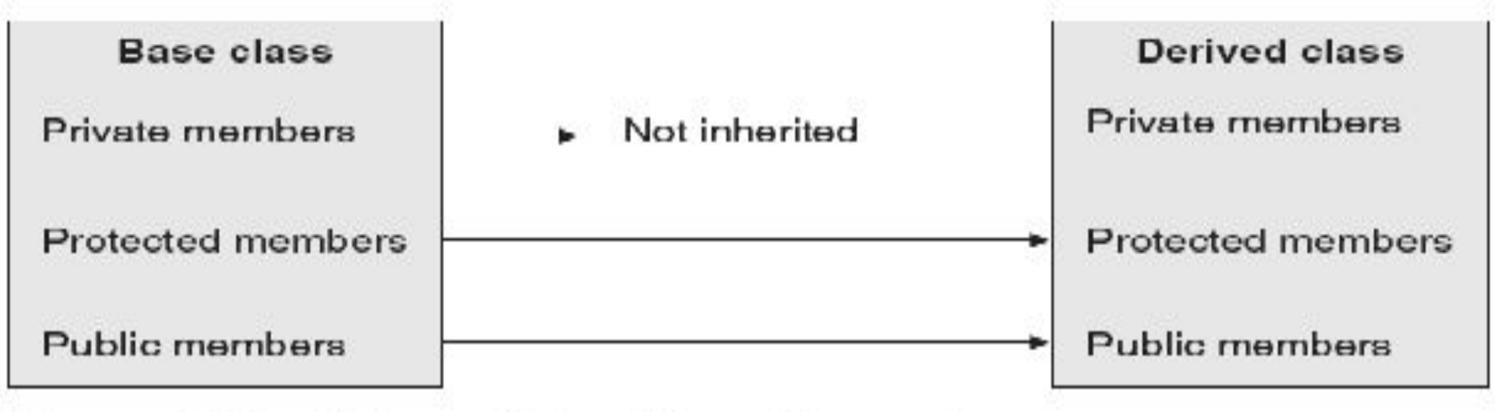


Specifier or class	Same class	Derived class	Any other class	Friend function	Friend class
Private	Yes	No	No	Yes	Yes
Protected	Yes	Yes	No	Yes	Yes
Public	Yes	Yes	Yes	Yes	Yes

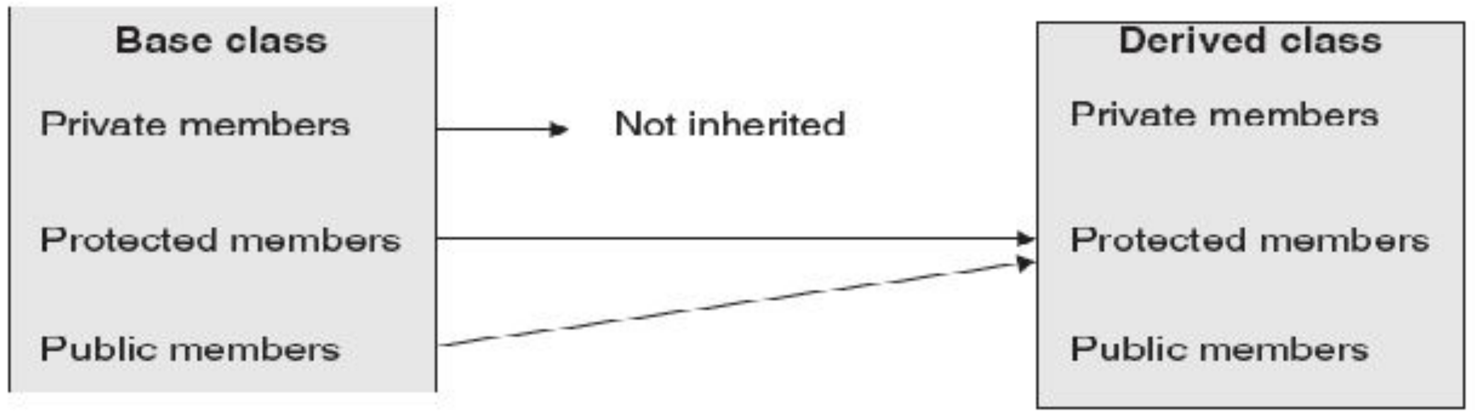
1. Introduction - VIII



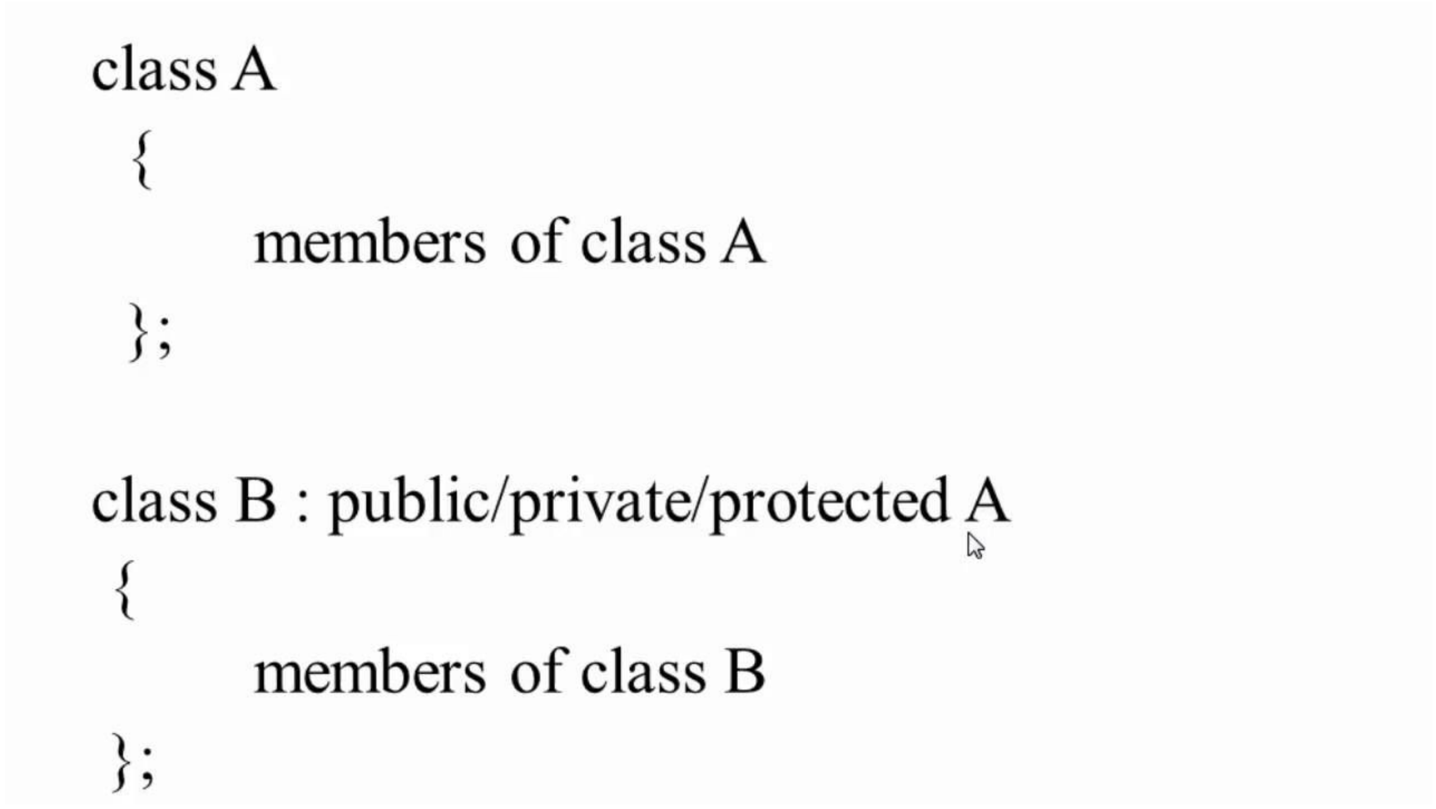
1. Introduction - IX



1. Introduction - X

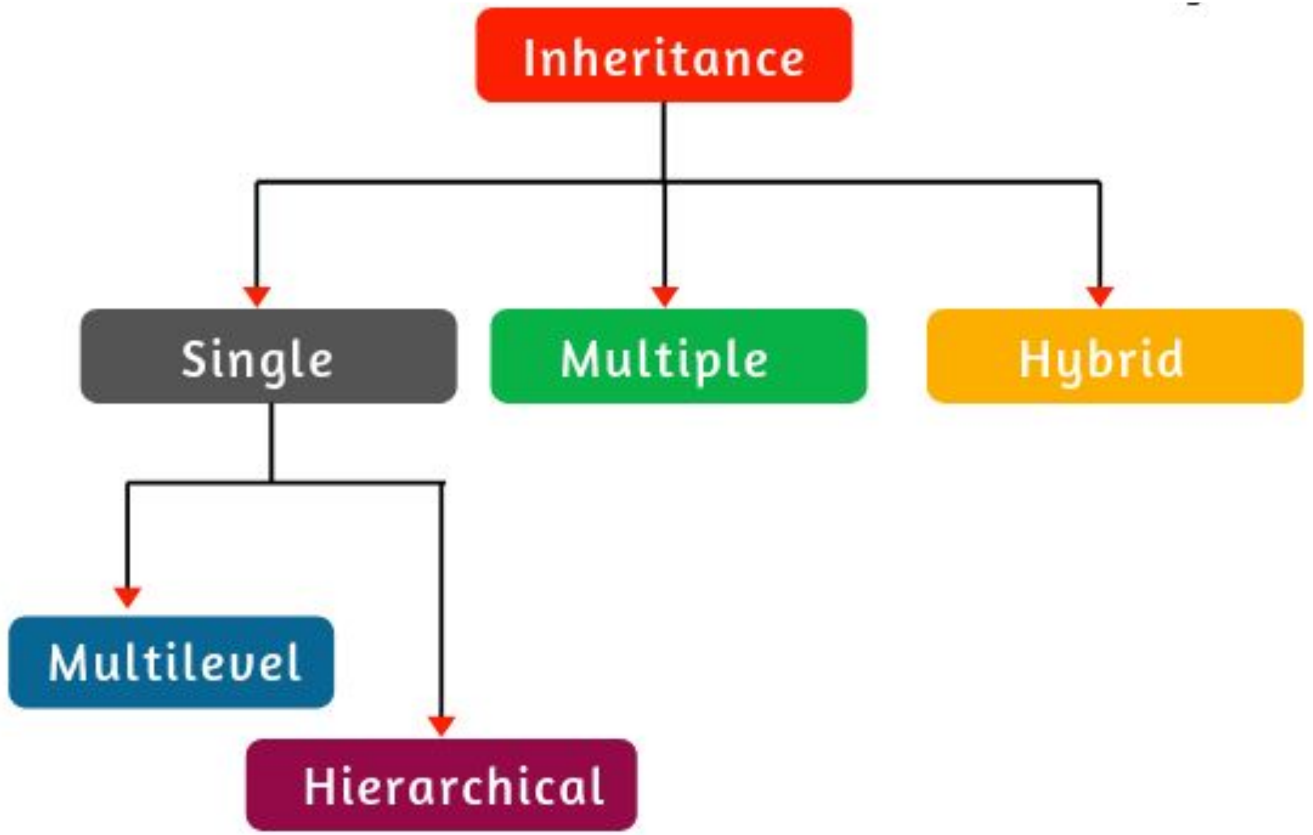


2. Inheritance - Syntax

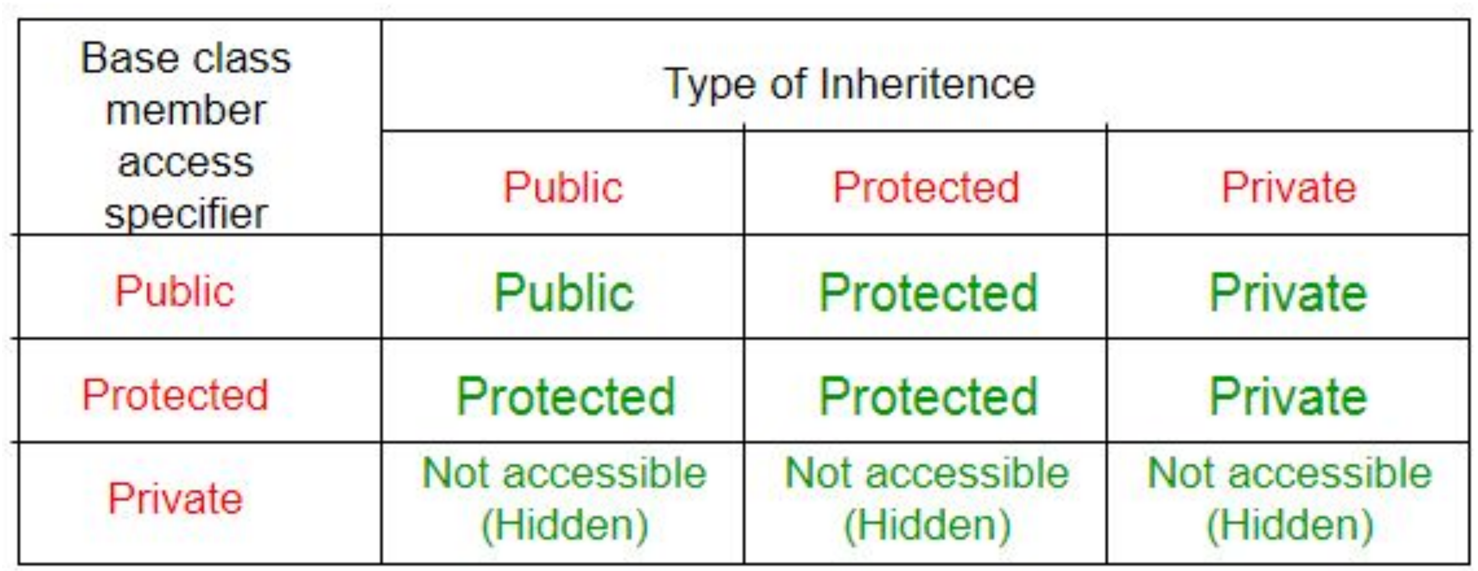


```
class A
 {
        members of class A
 };

class B : public/private/protected A
 {
        members of class B
 };
```

3. Inheritance - Classification

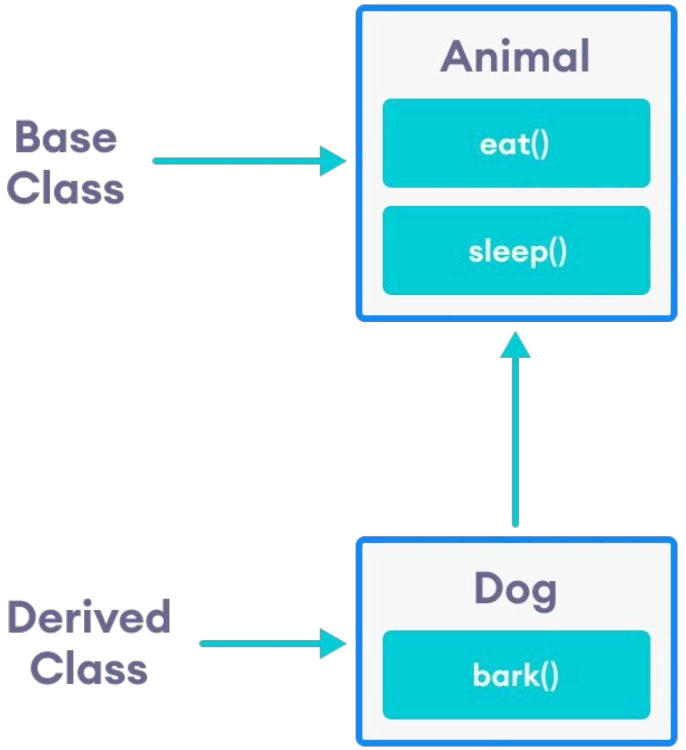


4. Inheritance - Classification

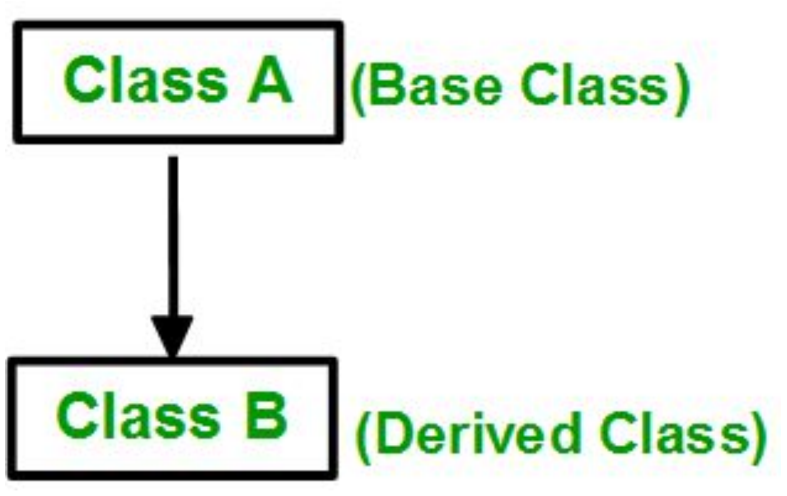


Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

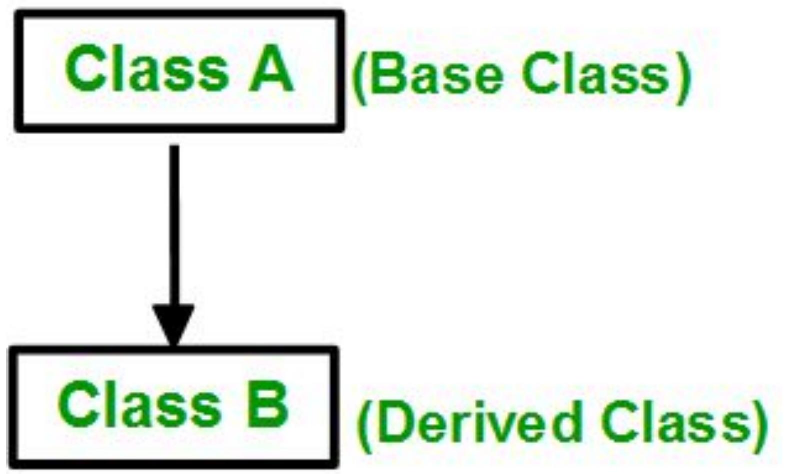
5. Inheritance - Concept

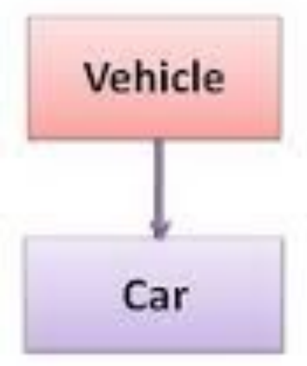


6. Inheritance - Single Inheritance

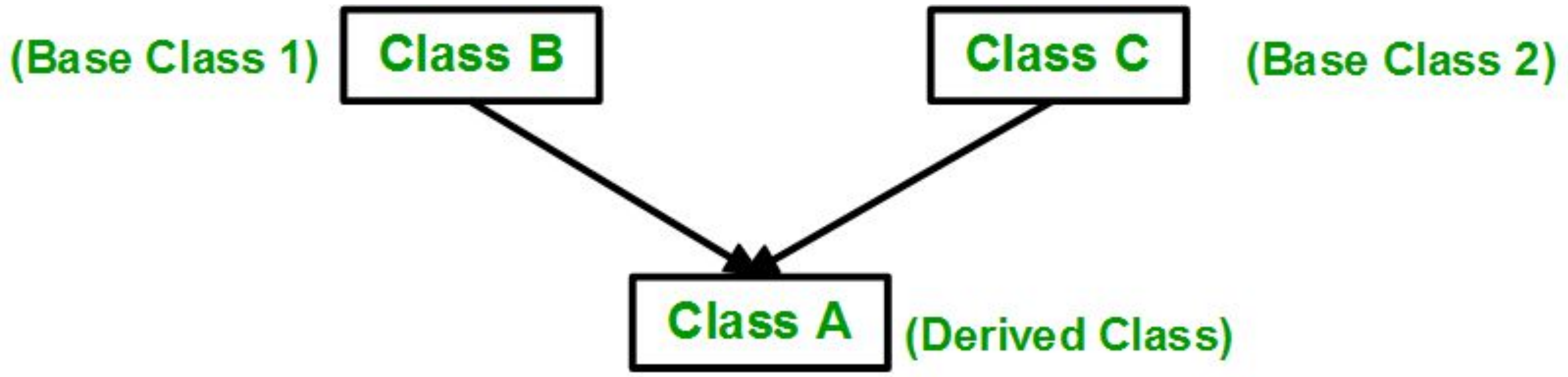


7. Inheritance - Single Inheritance

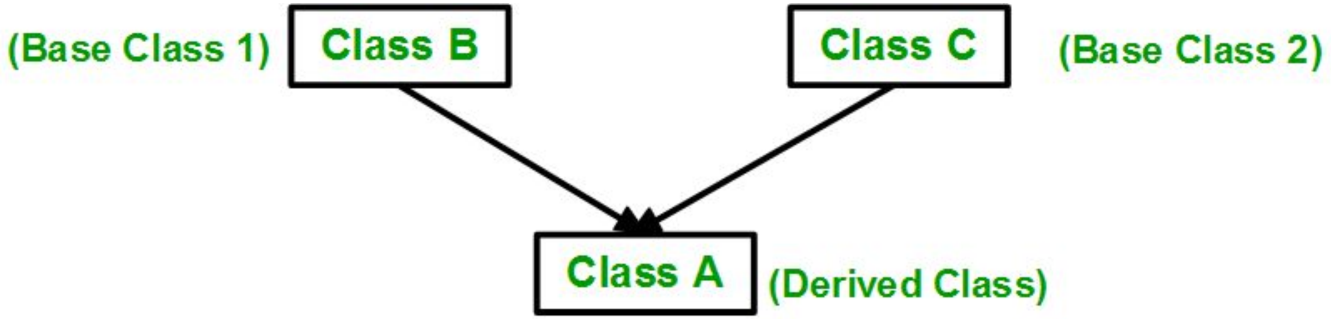


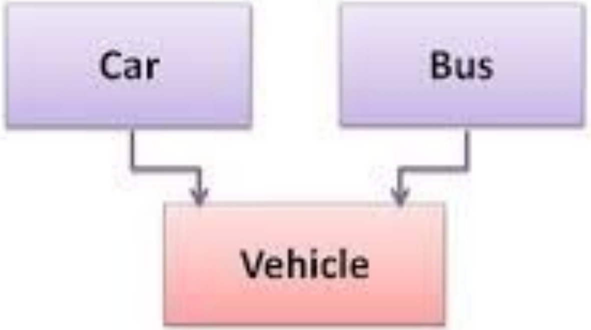


8. Inheritance - Multiple Inheritance

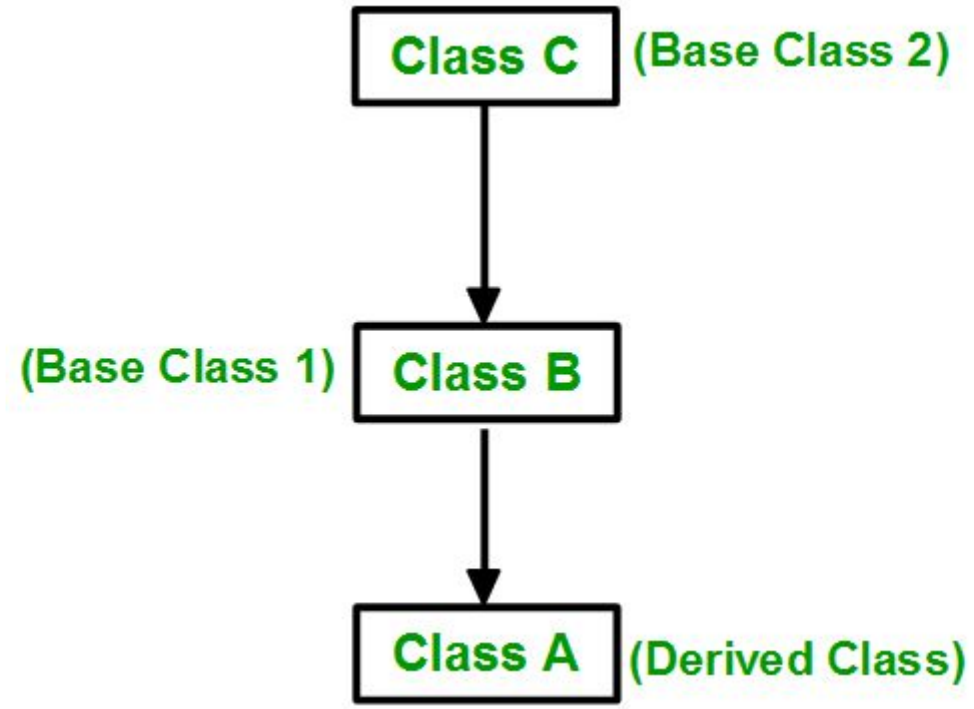


9. Inheritance - Multiple Inheritance

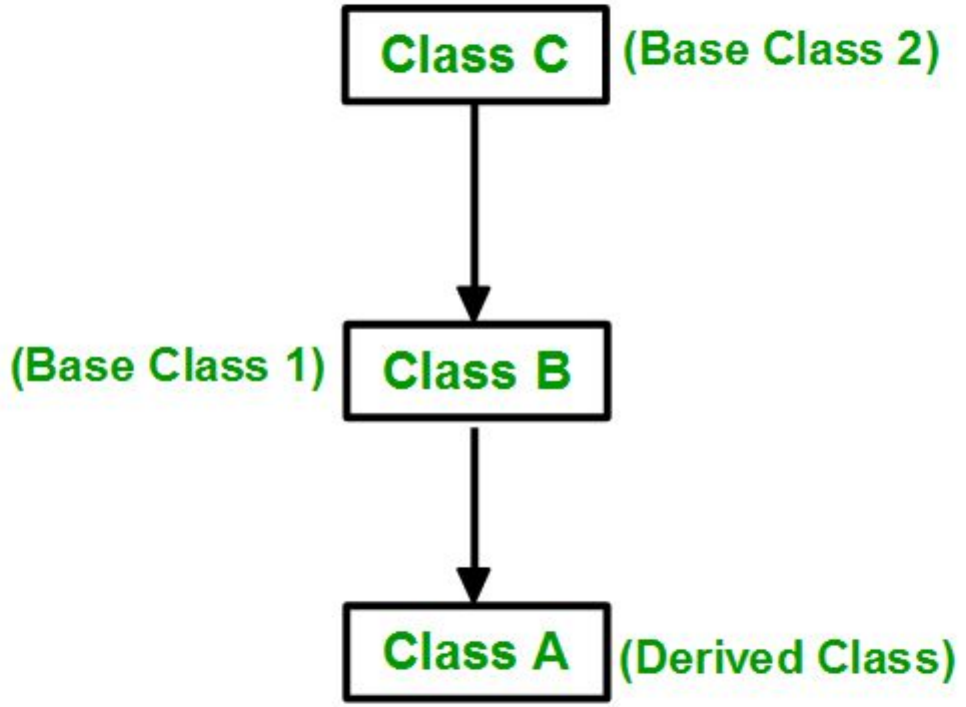


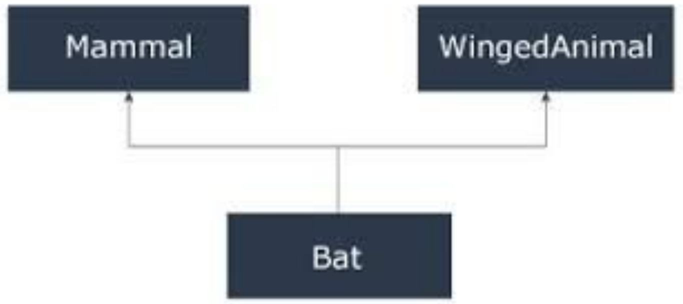


10. Inheritance - Multilevel Inheritance

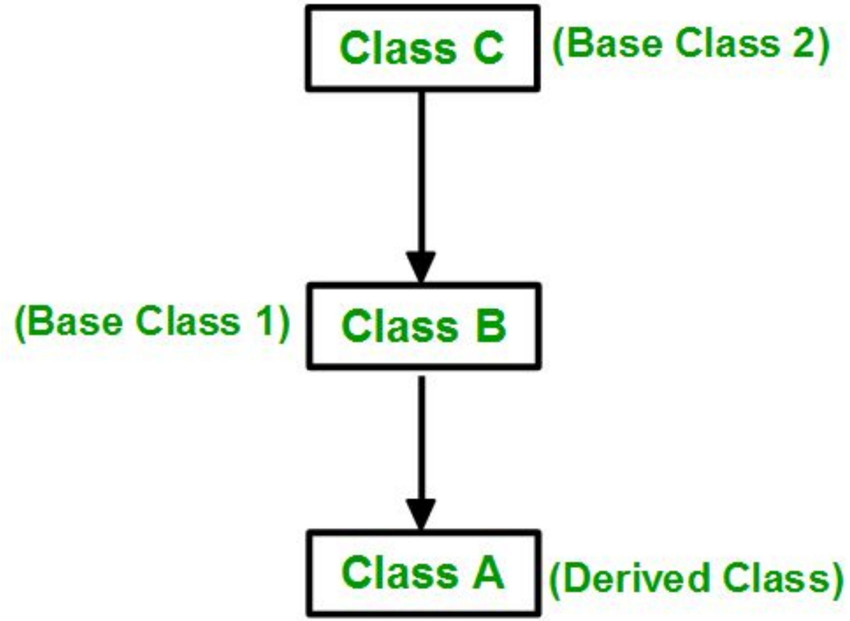


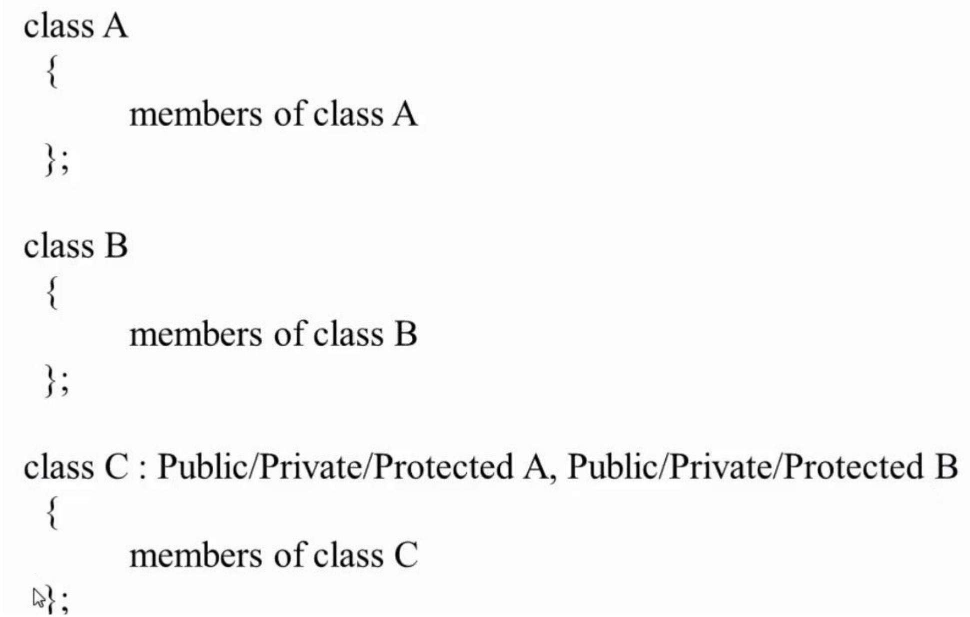
11. Inheritance - Multilevel Inheritance





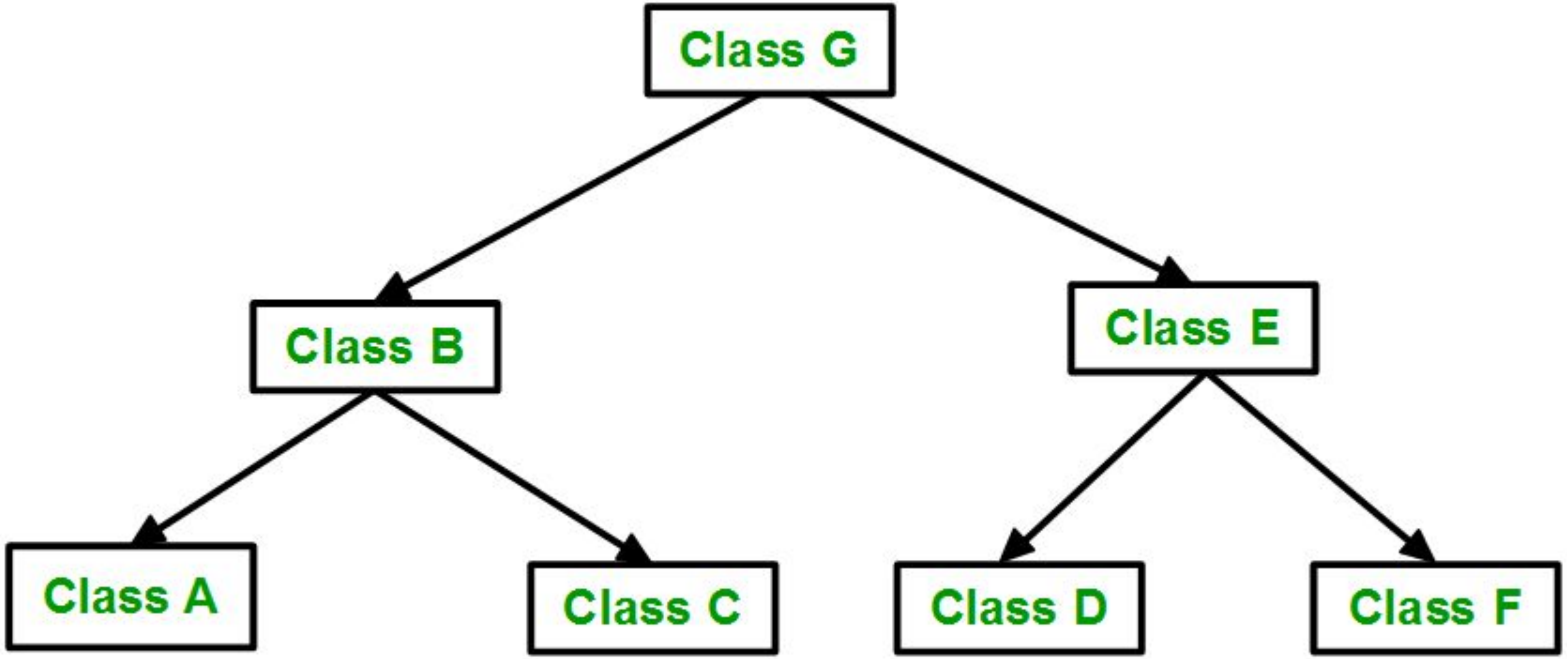
12. Inheritance - Multilevel Inheritance



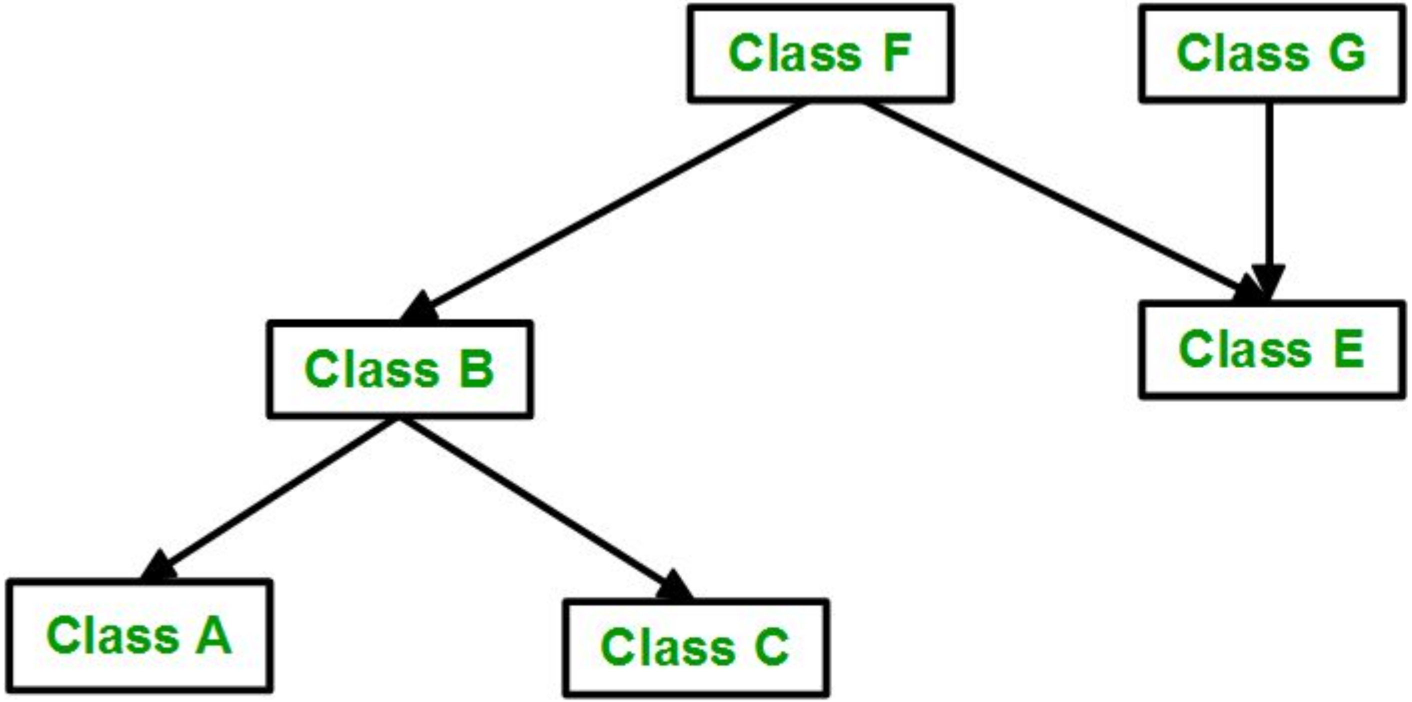


```
class A
 {
        members of class A
 };

class B
 {
        members of class B
 };

class C : Public/Private/Protected A, Public/Private/Protected B
 {
        members of class C
 };
```

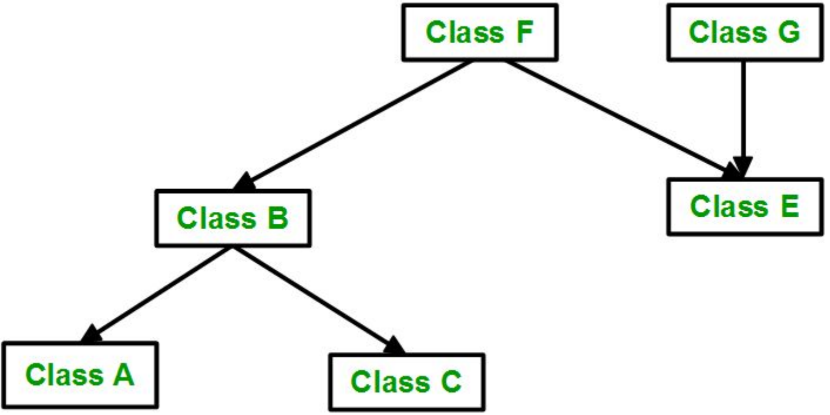
13. Inheritance - Hierarchical Inheritance

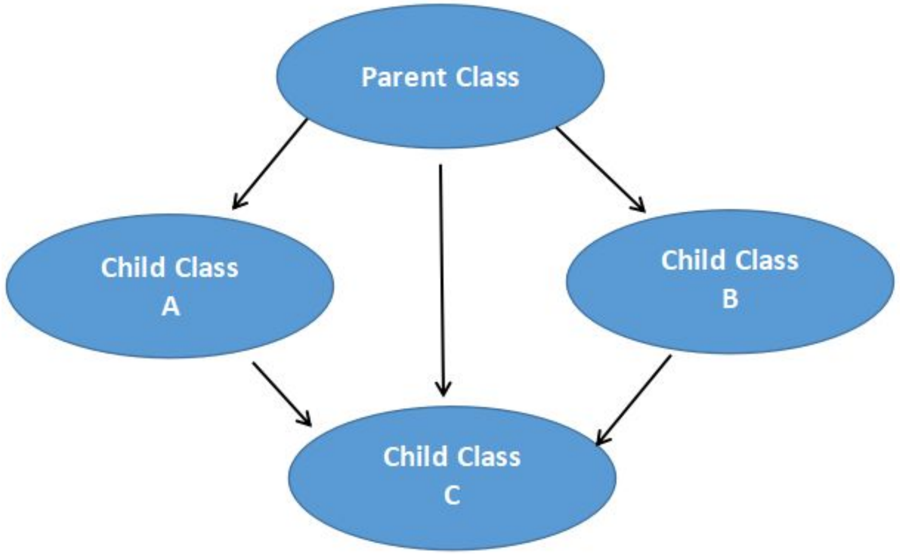


14. Inheritance - Hybrid Inheritance



15. Inheritance - Hybrid Inheritance


Class F
Class G
Class B
Class E
Class A
Class C


Parent Class
Child Class A
Child Class B
Child Class C