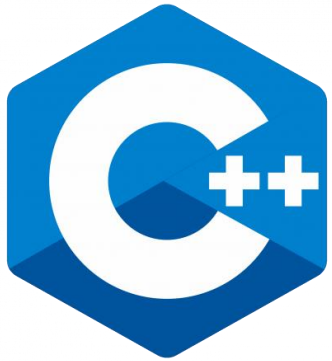


C++ Programming

Trainer : Pradnyaa S Dindorkar

Email: pradnya@sunbeaminfo.com

We did…..

- Design pattern
- Singleton Design pattern
- Operator overloading
- array index operator []
- OOP and its pillars
- Composition

1. Inheritance
2. Object slicing
3. Protected data member
4. Types of inheritance
5. Mode of inheritance
6. Diamond problem
7. Conversion Function

Inheritance

- If "is-a" relationship exist between two types then we should use inheritance.
- Inheritance is also called as " journey from Generalization to Specialization".
- Example: Book is-a product
- During inheritance, members of base class inherit into derived class.
- If we create object of derived class then non static data members declared in base class get space inside it.
- Size of object = sum of size of non static data members declared in base class and derived class.
- If we use private/protected/public keyword to control visibility of members of class by using access Specifier.
- If we use private/protected/public keyword to extend the class then it is called mode of inheritance.
- Default mode of inheritance is private.
  - Example: class Employee : person //is treated as class Employee : private Person
- Example: class Employee:public Person
- In all types of mode, private members inherit into derived class but we can not access it inside member function of derived class.
- If we want to access private members inside derived class then:
  - Either we should use member function(getter/setter).
  - or we should declare derived class as a friend inside base class.

Syntax of inheritance in C++

class Person //Parent class { }; **class Employee : public Person //** Child class { };	In C++ Parent class is called as Base class and child class is called as derived class. To create derived class we should use colon(:) operator. As shown in this code, public is mode of inheritance.

class Person //Parent class
{ **char name[30]; int age;** };
class Employee : public Person //Child class
{ **int empid; float salary;** };
int main(void)
{
Person p;
cout<<**sizeof(p)<<endl;**
Employee emp;
cout<<**sizeof(emp)<<endl;**
return 0;
}
	If we create object of derived class, then all the non- static data member declared in base class & derived class get space inside it i.e. non-static. static data members of base class inherit into the derived class.

Except following functions, including nested class, all the members of base class, inherit into the derived class

- Constructor
- Destructor
- Copy constructor
- Assignment operator
- Friend function.

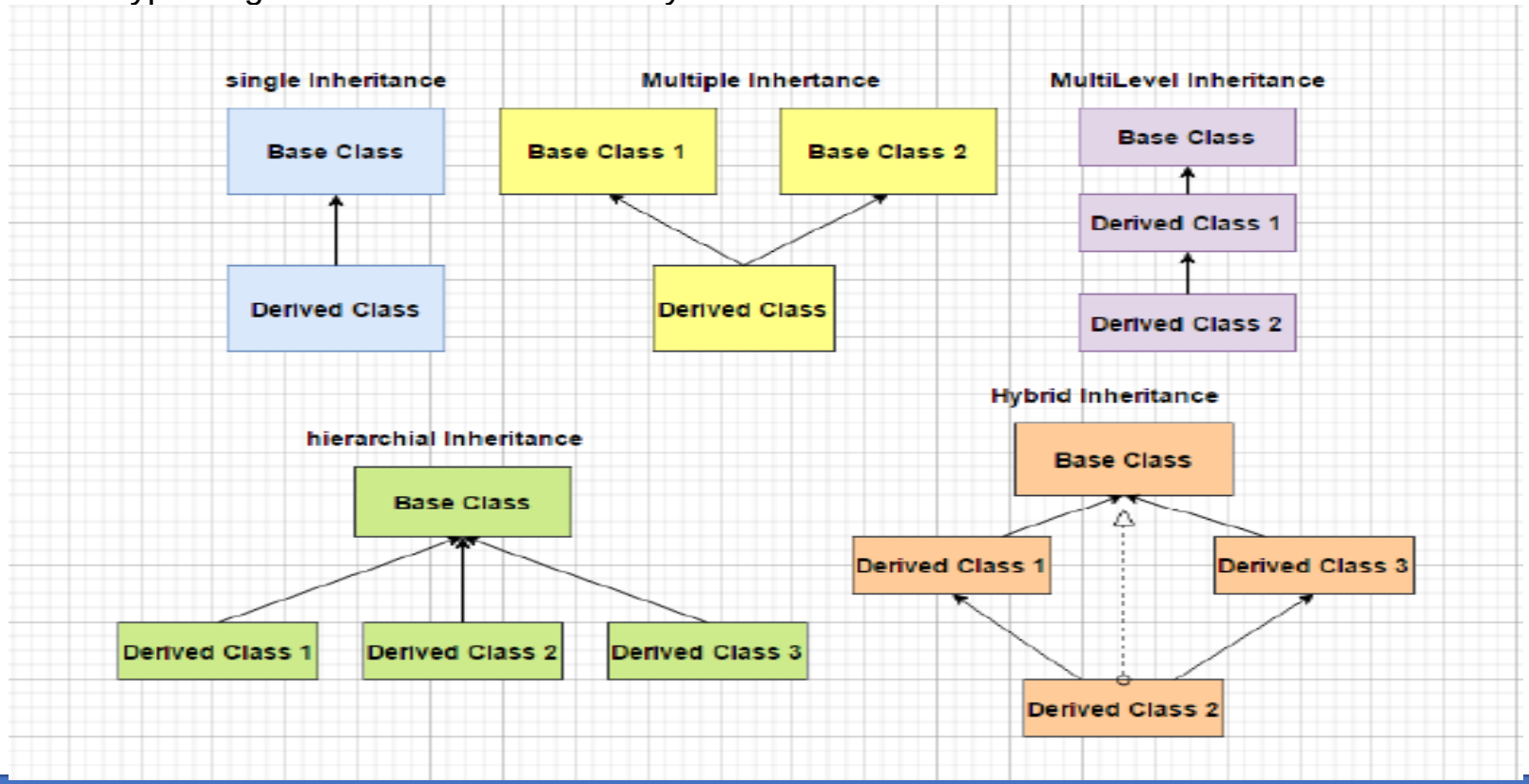
Protected Data member

- The protected access specifier allows the base class members to access onto derived class.
- However, protected members are not accessible from outside the class and global functions like main().
- Protected members in a class are similar to private members as they cannot be accessed from outside the class.
- But they can be accessed by derived classes or child classes while private members cannot.

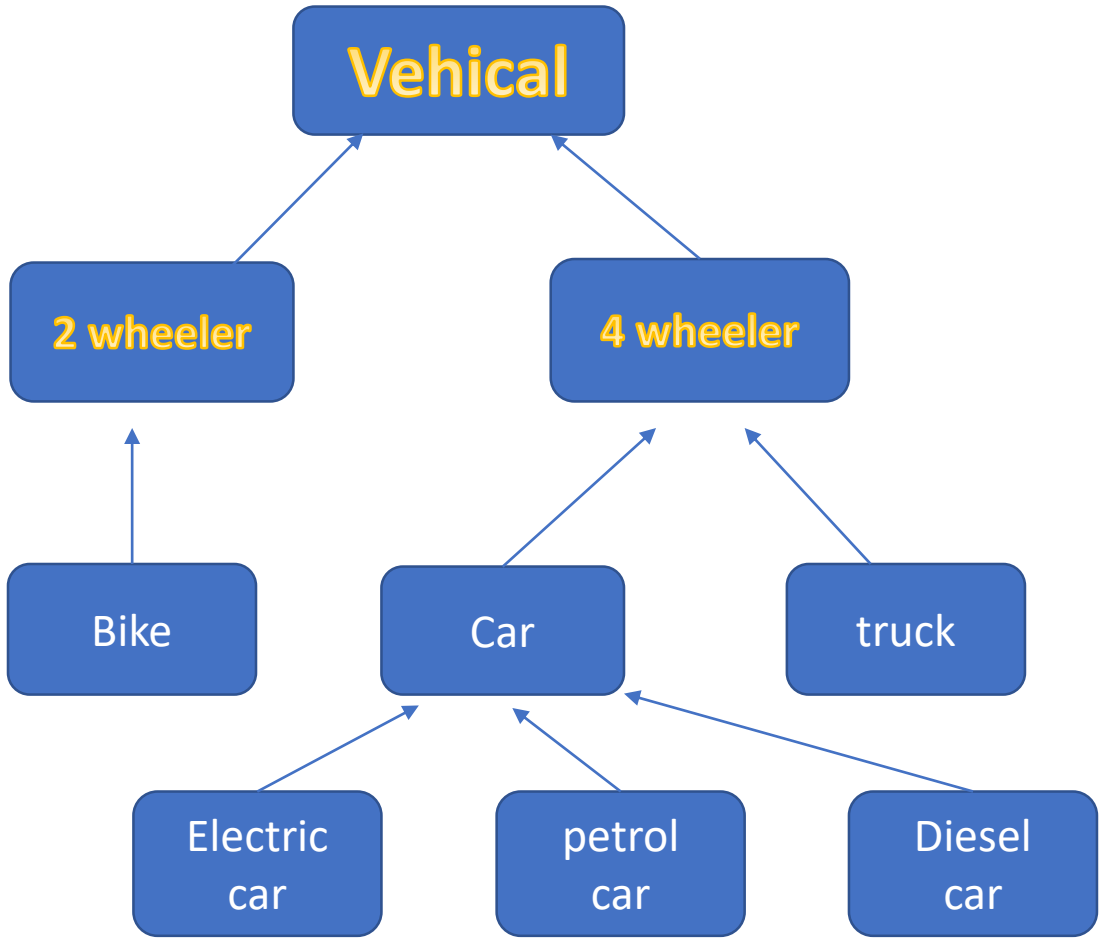
Types of Inheritance

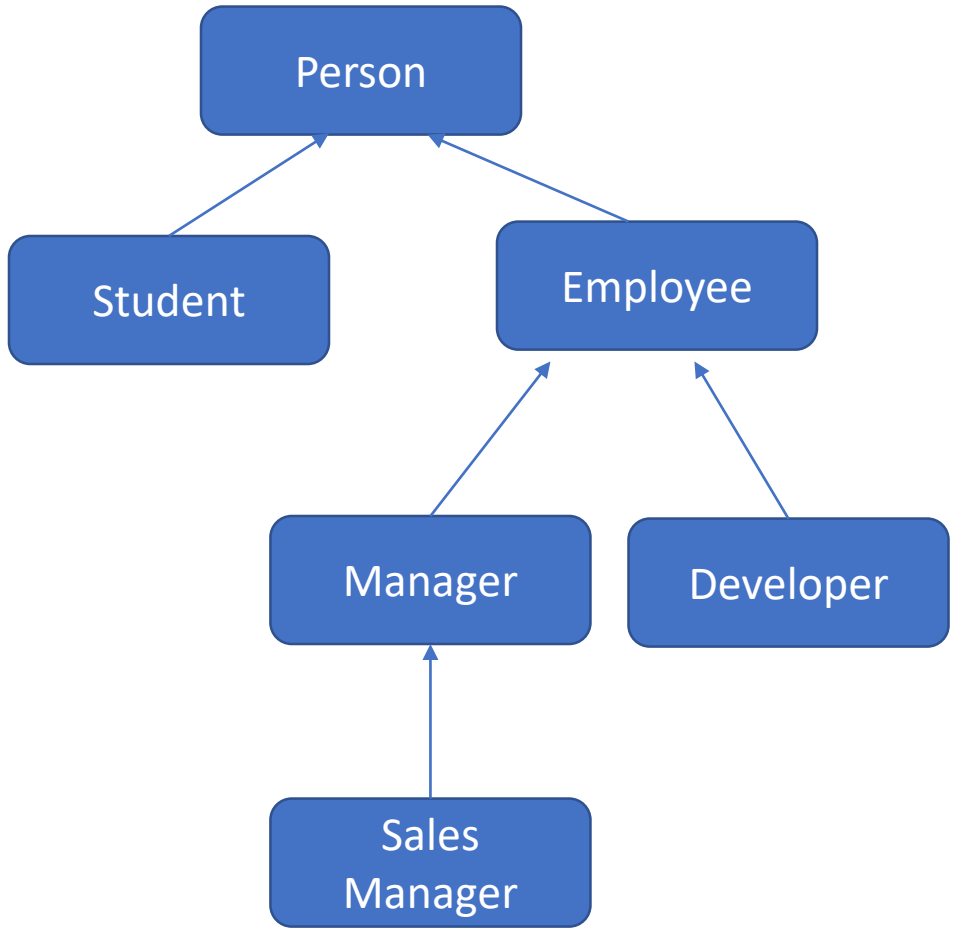
- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance

If we combine any two or more types together then it is called as hybrid inheritance.



Inheritance is also called as " journey from Generalization to Specialization".





Mode of inheritance

- If we use private, protected and public keyword to manage visibility of the members of class then it is called as access specifier.
- But if we use these keywords to extends the class then it is called as mode of inheritance.
- C++ supports private, protected and public mode of inheritance. If we do not specify any mode, then default mode of inheritance is private.

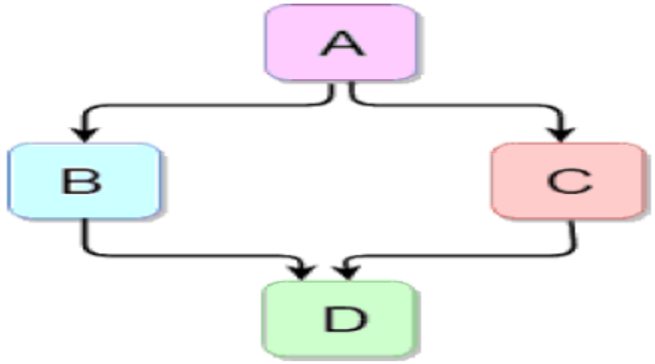
Mode of inheritance

Mode of inheritance (read "--->" as becomes)

	Base	Derived
public mode:		
	Public --->	Public
	protected --->	Protected
	private --->	NA
protected mode:		
	Public --->	Protected
	protected --->	Protected
	private --->	NA
private mode:		
	Public --->	private
	protected --->	private
	private --->	NA

Diamond Problem

- As shown in diagram it is hybrid inheritance. Its shape is like diamond hence it is also called as diamond inheritance.
- Data members of indirect base class inherit into the indirect derived class multiple times. Hence it effects on size of object of indirect derived class.
- Member functions of indirect base class inherit into indirect derived class multiple times. If we try to call member function of indirect base class on object of indirect derived class, then compiler generates ambiguity error.
- If we create object of indirect derived class, then constructor and destructor of indirect base class gets called multiple times.
- All above problems generated by hybrid inheritance is called diamond problem.



Solution to Diamond Problem– Virtual Base Class

- If we want to overcome diamond problem, then we should declare base class virtual i.e. we should derive class B & C from class A virtually. It is called virtual inheritance. In this case, members of class A will be inherited into B & C but it will not be inherited from B & C into class D.

```
class A { };
class B : virtual public A
{ };
class C : virtual public A
{ };
class D : public B, public C
{ };
```

Object slicing

- When a derived class object is assigned to a base class object in C++, the derived class object's extra attributes are sliced off (not considered) to generate the base class object; and this whole process is termed **object slicing**.

- when extra components of a derived class are sliced or not used and the priority is given to the base class's object this is termed object slicing.

- Class base{};
- Class derived :public base {};

```
Main() {
 base b ; derived d ;
 b=d;     //object slicing.
}
```

Thank You