Inheritance

- To derive New classes from old classes
- The idea of inheritance implements the **is a** relationship.
- This also provides an opportunity to reuse the code functionality and fast implementation time.

Base and Derived Classes

- A class can be derived from more than one classes, which means it can inherit data and functions from multiple base classes.

- The existing class is called the **base** class, and the new class is referred to as the **derived** class.

A derived class inherits all base class following methods

- Constructors, destructors and copy constructors of the base class.
- Overloaded operators of the base class.
- The friend functions of the base class.

Access Control and Inheritance

A derived class can access all the non-private members of its base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class.

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

Syntax

class derived-classname : access-specifier base-class

Where access-specifier is one of **public, protected,** or **private**, and base-class is the name of a previously defined class. If the access-specifier is not used, then it is private by default.

Base Class Member	Deriving from public BC	Deriving from protected BC	Deriving from private BC
Public	Public	Protected	Private
Private	Private	Private	Private
Protected	Protected	Protected	Private

Example

```
 class san
{   protected :
     int a;
};
class ban: public san
{
    public:
       in()
        { a=20;
           cout<<a;  }
}
main()
{
      ban b1;
      b1.in();
}
```

Types of Inheritance

- Simple
- Multilevel
- Multiple
- Hyrbid
- Hierarchical

Ambiguity in Inheritance

- In simple as well as Multilevel

- In Hybrid Inheritance

Polymorphism

- The word **polymorphism** means having many forms. Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance.

poly + morphism

Can be achieved by

- Early Binding
  - Function Overloading
  - Operator Overloading

- Late Binding
  - Virtual Function

Virtual Function

- A virtual function a member function which is declared within base class and is re-defined (Overriden) by derived class.When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.

Rule of Virtual Function

- Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.
- They are mainly used to achieve Runtime polymorphism
- The resolving of function call is done at Run-time.
- They Must be declared in public section of class.