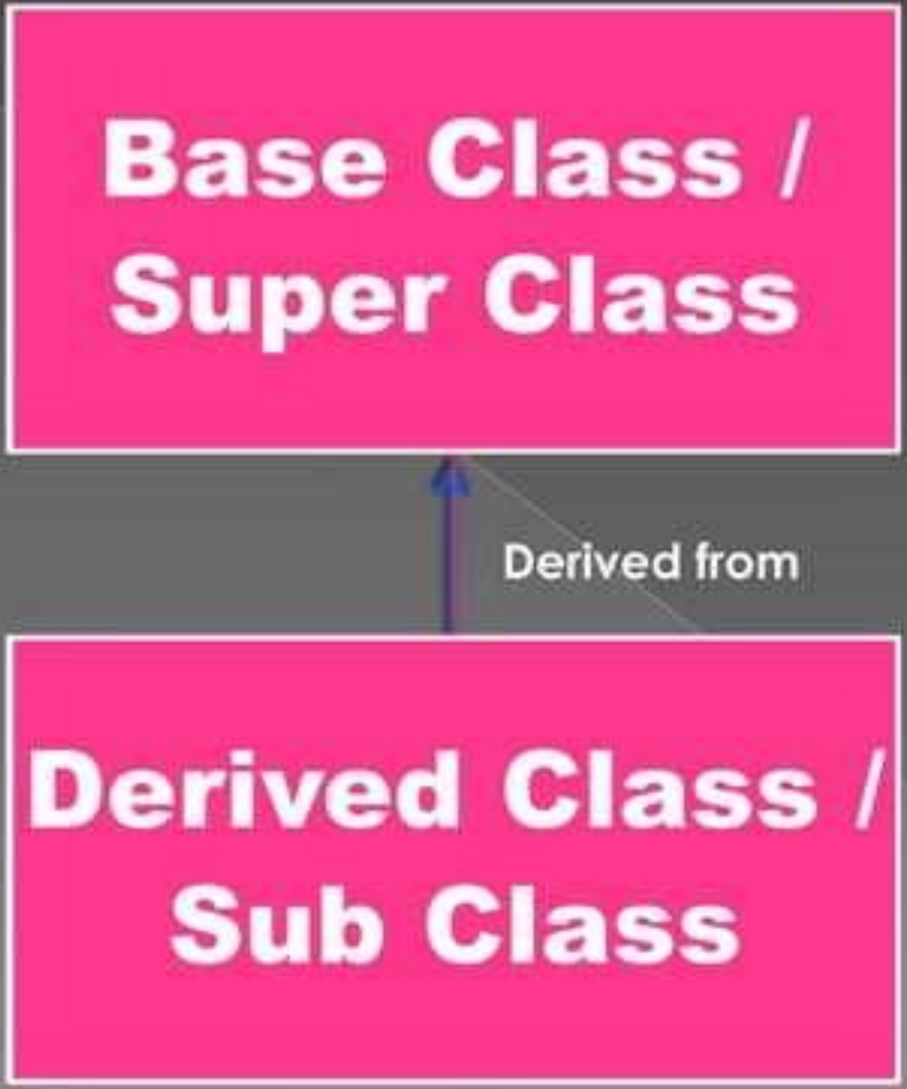
INHERITANCE

Prepared by : SELVIN JOSY BAI. S

One of the most important concepts in object-oriented programming is that of inheritance.

INHERITANCE - DEFINITON

- It is the capability of one class to inherit properties from another class.
- The technique of building new classes from the existing classes is called inheritance.


Base Class / Super Class
Derived from
Derived Class / Sub Class

Base Class - DEFINITION

- It is the class whose properties are inherited by another class.
- It is also called **Super Class**.

Derived Class - DEFINITION

- It is the class that inherit properties from base class(es).
- It is also called **Sub Class**.
- It inherits all the properties of the base class and can add additional features to the derived class.

Advantages INHERITANCE

- CODE REUSABILITY
- EASY TO IMPLEMENT REAL WORLD MODELS
- TRANSITIVE NATURE

VISIBILITY MODES

- It is any of the access labels: private, public or protected.
- It defines the accessibility of the members of the base class within the derived class.
- If the visibility mode is not specified, it will be taken as private by default.

Difference between Private & protected members

- Private members can never be inherited. Only the public and protected members can be inherited to the derived class. This is the difference between the private and protected members.

VISIBILITY MODES

Base Class Visibility	Derived class visibility		
	Public derivation	Private derivation	Protected derivation
Private	Not inherited	Not inherited	Not inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected

Accessibility of members by various objects :

Access Specifier of members	Accessible from own class	Accessible from derived class	Accessible from objects outside class
PRIVATE	YES	NO	NO
PUBLIC	YES	YES	YES
PROTECTED	YES	YES	NO

Making private member inheritable

- Private members of the base class cannot be inherited with any of the visibility mode.
- One is by making the visibility mode of private members as public, but they will be exposed to the outside world.
- Another is to convert the private members into protected so that they will be hidden from the outside world but can be inherited.

Friendship Vs Derivation

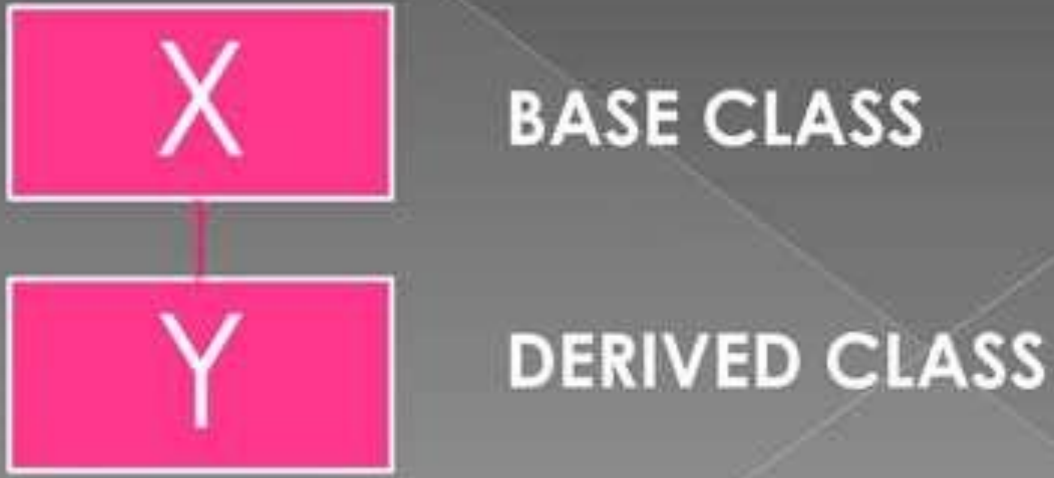
Friendship	Derivation
Provide access to private and protected members.	Private members cannot be derived into another class.
A non-member function can make friendship to a class	It shares the features of base class and adds some more attributes
Two independent classes can have friendship, where friend function acts as bridge between them.	The derived classes are created with the help of base class. Derived class is a special instance of base class.
Friendship is not Derivation	Derivation is a kind of Friendship.

Types of INHERITANCE

1. SINGLE INHERITANCE
2. MULTIPLE INHERITANCE
3. HIERARCHICAL INHERITANCE
4. MULTILEVEL INHERITANCE
5. HYBRID INHERITANCE

1. SINGLE INHERITANCE

- Derivation of a class from only one base class is called SINGLE Inheritance.
- In the figure class Y is derived from class X.



Syntax for SINGLE Inheritance

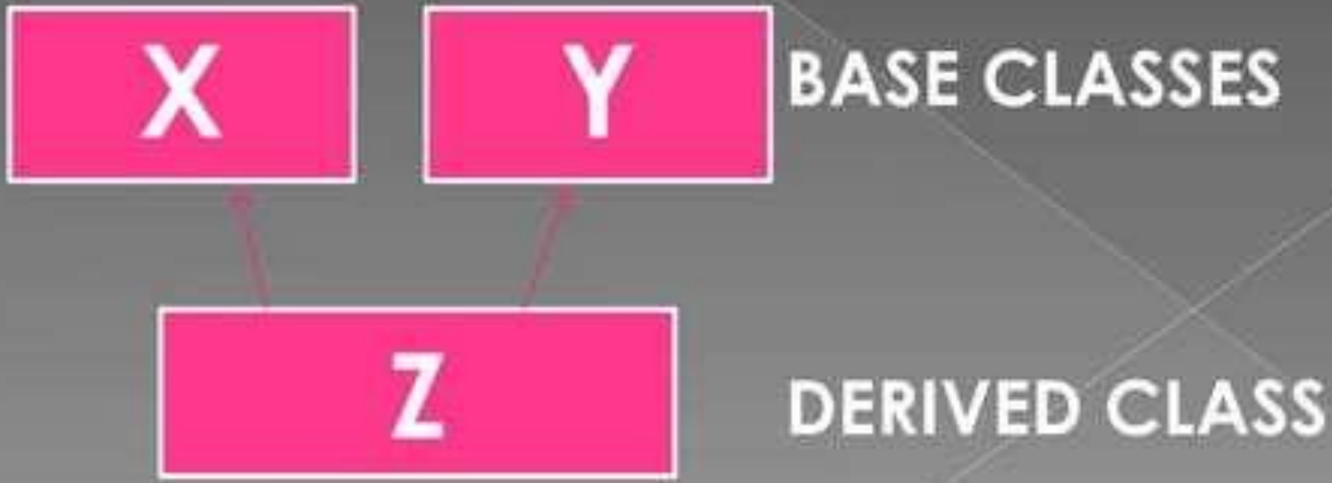
```
class DerivedClassName : [visibility mode]
  BaseClassName
{
  //DataMembers and MemberFunctions;
}
```

Example:

```
class Automobile : public Vehicle
{
  //DataMembers and MemberFunctions;
}
```

2. MULTIPLE INHERITANCE

- Derivation of a class from SEVERAL (TWO OR MORE) base classes is called MULTIPLE Inheritance.
- In the figure class Z is derived from both the classes X & Y.



Syntax for MULTIPLE Inheritance

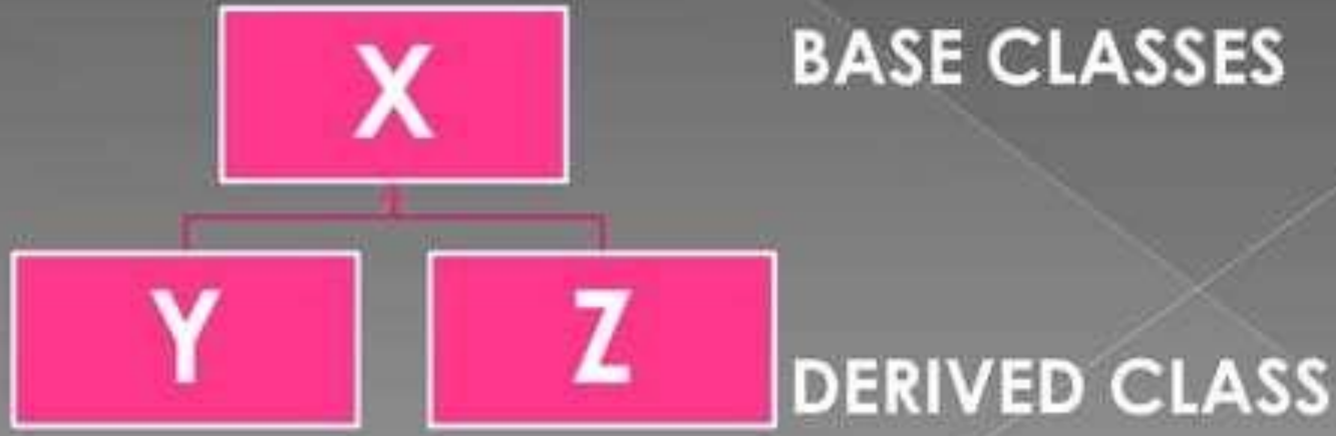
```
class DerivedClassName : [visibility mode]
   BaseClassName1, [visibility mode] BaseClassName1
{
   //DataMembers and MemberFunctions;
}
```

Example:

```
class CHILD : public FATHER, public MOTHER
{
   //DataMembers and MemberFunctions;
}
```

3. HIERARCHICAL INHERITANCE

- Derivation of SEVERAL classes from SINGLE base class is called HIERARCHICAL Inheritance.
- In the figure the classes Y & Z is derived from the same class X.



Syntax for HIERARCHICAL Inheritance

```
class DerivedClassName1 : [visibility mode] BaseClassName
{
    ----------;
}

class DerivedClassName2 : [visibility mode] BaseClassName
{
    ----------;
}
```

Example:

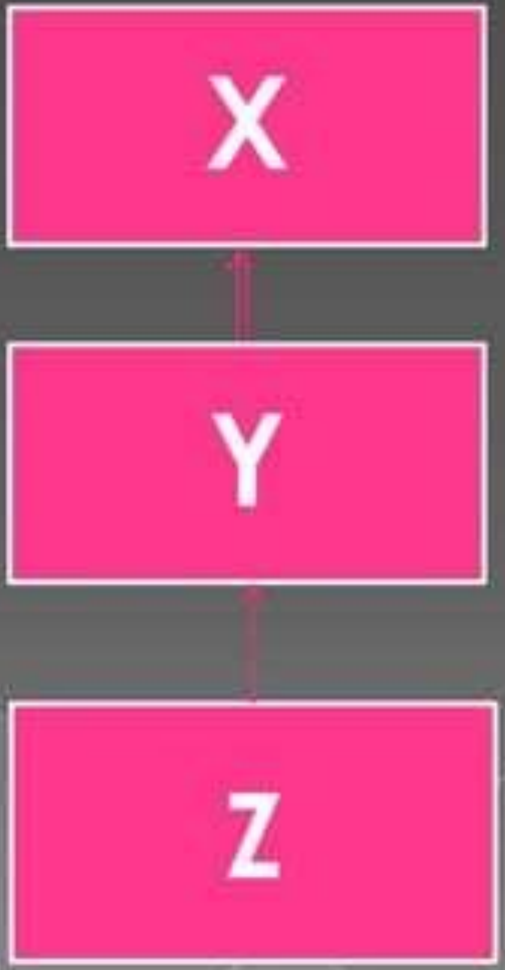
```
class Y : public X
{
---------;
}
```

Example:

```
class Z : public X
{
---------;
}
```

4. MULTILEVEL INHERITANCE

- When a sub class is derived from a base class which itself is derived from another class, it is known as MULTILEVEL Inheritance.
- In the figure the class Z is derived from class Y, which is a derived class that is inherited from the class X.



Syntax for MULTILEVEL Inheritance

```
class DerivedClassName1 : [visibility mode] BaseClassName
{
    ----------;
}

class DerivedClassName2 : [visibility mode] DerivedClassName1
{
    ----------;
}
```

Example:

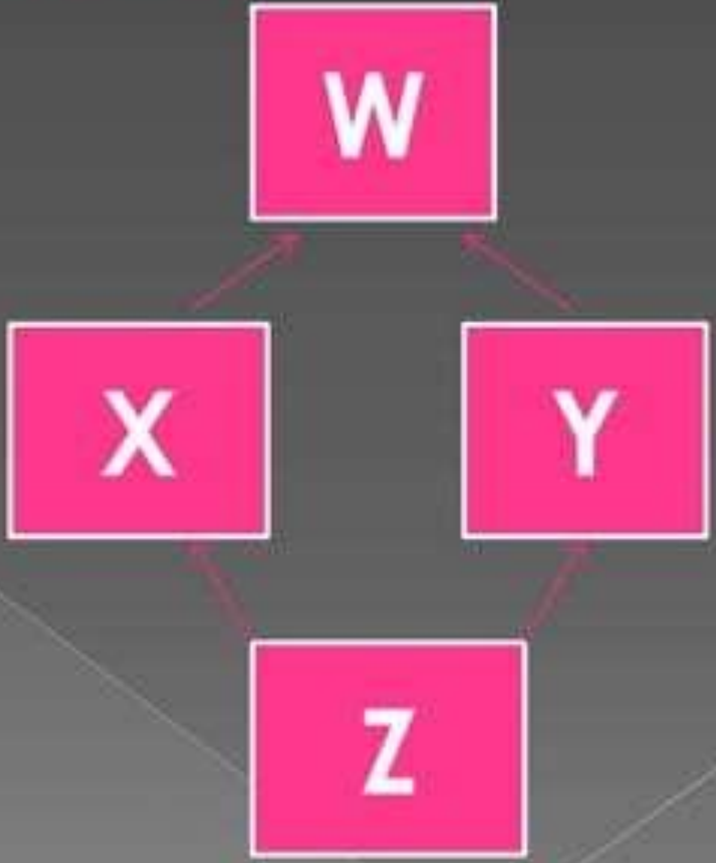
```
class Y : public X
{
---------;
}
```

Example:

```
class Z : public Y
{
---------;
}
```

5. HYBRID or MULTIPATH INHERITANCE

- Derivation of a class involving more than one form of Inheritance is known as HYBRID inheritance.
- As it is the derivation of a class from other derived classes, which are derived from the same base class.



Constructors and Destructors in INHERITANCE

- The derived class need have a constructor as long as the base class has a no-argument constructor.
- If the base class has constructors with arguments, then it is mandatory for the derived class to have a constructor and pass the arguments to the base class constructor.

.....contd.

- When an object of a derived class is created, the constructor of the base class is executed first and later the constructor of the derived class.
- Unlike constructors, destructors in the class hierarchy are invoked in the reverse order of the constructor invocation.

ABSTRACT CLASS

- It is one that has no instances and is not designed to create objects.
- It is only designed to be inherited from.
- It specifies an interface at a certain level of inheritance and provides a framework or skeleton, upon which other classes can be built.

VIRTUAL BASE CLASS

- When classes are derived in the form of hybrid inheritance, there can be a problem by which multiple copies of the base class members come in the lowest level derived class through the various intermediate subclasses. Here comes the virtual base class for rescue.

```
class A
{ public:
     int a;
};

class B : public A
{ public:
     int b;
};

class C : public A
{ public:
     int c;
};

class D : public B, public C
{ public:
     int d;
};
```

Class B contains a and b

Class C contains a and c

Class D contains a,b, a,c, & d

```
class A
{ public:
      int a;
};

class B : virtual public A
{ public:
      int b;
};

class C : virtual public A
{ public:
      int c;
};

class D : public B, public C
{ public:
      int d;
};
```

only one copy of A will be inherited

Class B contains a and b

Class C contains a and c

Class D contains a,b, c, & d

CONTAINERSHIP or CONTAINMENT

- In inheritance, if the class D is derived from the class B, it is said that D is a kind of B; the class D has all the properties of B in addition to the features of its own.
- In OOP, the containership occurs when an object of one class is contained in another class as a data member.
- In other words, a class can contain objects of other classes as its members.

Example

```
class ABC
{
    int a;
    float b;
    public:
        void fab();
};
class PQR
{
    int p;
    float q;
    public:
       ABC ob1;
        void fab();
};
```