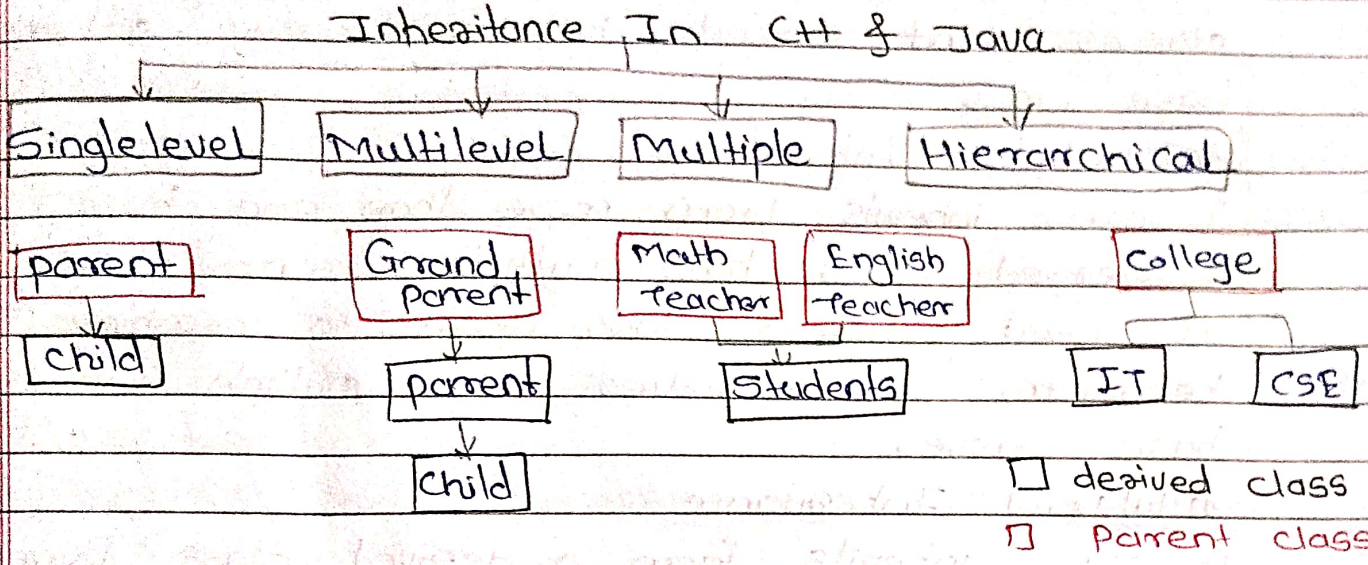
C++

# Assignment No:- 16

Q1] Explain the concept of inheritance and the type of Inheritance according to the architecture.

→




In object-oriented programming inheritance is a mechanism where a class inherits properties and behaviours from another class.
This promotes code reuses, polymorphisms and modularity.

- Base class (parent class)

The class that provides its attributes and methods to other classes.

- Derived class (child class)

The class that inherits the attributes and method of base class
It can also have its own additional attributes and methods.

- Types Of Inheritance.

1] Single Inheritance-
A class inherits from only one base class.
Simplest form of Inheritance

linear relationship between the base class and the derived class.
It used when you need to create specialized classes that extend the functionality of one base class.

2] Multiple Inheritance
A class inherits from more than one base class
A derived class has multiple parent class.
It used when a class needs to combine behaviours or properties from multiple unrelated base classes.

3] Multilevel Inheritance
A class inherits from a derived class, forming a chain.
Linear relationship spanning multiple levels.

4] Hierarchical Inheritance
Multiple classes inherit from a single base class
Tree like structure with a single parent and multiple children.

Q2] Explain the concept of access specifier in details.

→ In c++ access specifiers are keywords used to control the acessibility of members of a class. They define how and where the member of a class can be accessed.
There are three main access specifiers in C++.

1] Public
Members declared under the public access specifier are accessible from anywhere in the program.

This is used when you want the member to be accessible to all parts of the code.
- Accessible from both inside and outside class, provided you have an object of class.

2] Private

Members declared under the private access specifier are accessible only within the class. They are not accessible directly from outside the class.
- This is used to implement data hiding
- Access only by a member function of same class.

3] Protected
- Members declared under the protected access specifier behave like private members, but they can also be accessed by derived classes
- This is used when you want to restrict access to members but still allow derived classes to inherit and use them.
- Accessible within the class itself and in derived classes, but not accessible outside the class.

Q3] What is difference between private and protected access specifier.

→ • Private

Members declared as private are only accessible within the same class. They cannot be accessed or modified outside the class, even by derived class.

• Public

~~Used to enforce encapsulation by restricting access~~

• Protected

Member declared as protected are accessible

within the same class and in derived class However, they are not accessible from outside the class hierarchy

- Private

Used to enforce encapsulation by restricting access to sensitive data and internal implementation details.

Protected -

Provides a balance between encapsulation and inheritance, allowing derived classes to work with base class members while still restricting access from external code.

Q4] What is default access specifier, if it is not written?

→ The default access specifier is private. This means that if we don't explicitly specify an access specifier, members of class will be private by default.

e.g.

```
class MyClass
{  int x;  // default is private
   public:
      int Y;
};
```

Q5] What are the types of inheritance according to access specifier.

→ In C++, inheritance types based on access specifiers determine how the base class members are treated in the derived class

1) Public Inheritance
- when a class inherits another class using the public access specifier
- Public members of base class become public in the derived class.
- Protected members of the base class remain protected in the derived class.
- Private members of base class are not accessible directly in the derived class.

2) Protected Inheritance
when a class inherits another class using the protected access specifier.
- Public members of base class become protected in the derived class.
- Protected members of the base class remain protected in the derived class.
- Private members of the base class are still not accessible directly in the derived class.

3) Private Inheritance
when a class inherits another class using the private access specifier.
- Public members of the base class become private in the derived class.
- Protected members of the base class become private in the derived class.
- Private members of the base class are still not accessible directly in the derived class.

Q6 Explain the constructor and destructor calling sequence in case single level, multi level and multiple inheritance.

→ Single level

- Constructor call sequence-

The base class constructor is called first, followed by the derived class constructor.

- Destructor call sequence-

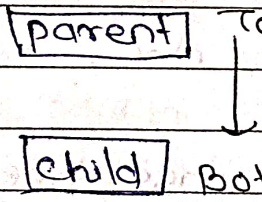
The derived class destructor is called first, followed by base class destructor.

- Multi level Inheritance

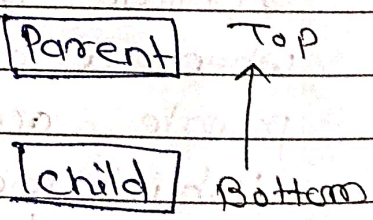
Constructor are called from top of the hierarchy to the bottom.

Destructor are called from bottom of the hierarchy to the top.

Constructor calling sequence




Destructor calling sequence




- Multiple Inheritance

Constructor calling sequence

Constructors of all base classes are called in the order they are declared in derived class.

Then the constructor of the derived class is called.

Destructor call sequence
The destructor of derived class is called first then destructor of all base classes are called in the reverse order of their declaration.

Q7) Draw object layout and class diagram of below code snippets and explain its internal working in detail. Explain type of inheritance in the below code snippet.

→

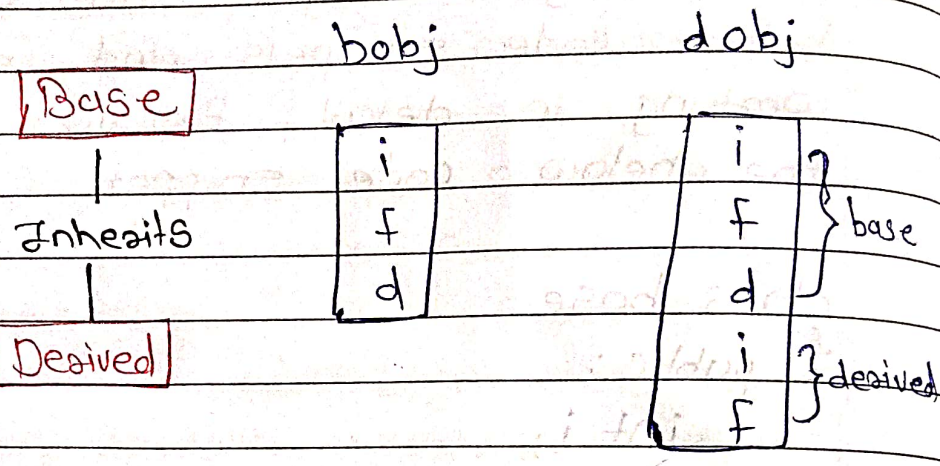
```
class base
{ public:
    int i;
    foat f;
    double d;
    void fun()
    { }
    void gun()
    {}
};
class Derived : public base
{ public:
    int i;
    double d;
    void sun()
    { }
};
int main()
{ base bobj;
  Derived dobj;
return 0;
}
```

Base class

| i, f, d | ← characteristics |
|---|---|
| fun(), gun() | ← Behaviours |

Derived class

| i, f, d, i, f | |
|---|---|
| fun(), gun(), sun() | |

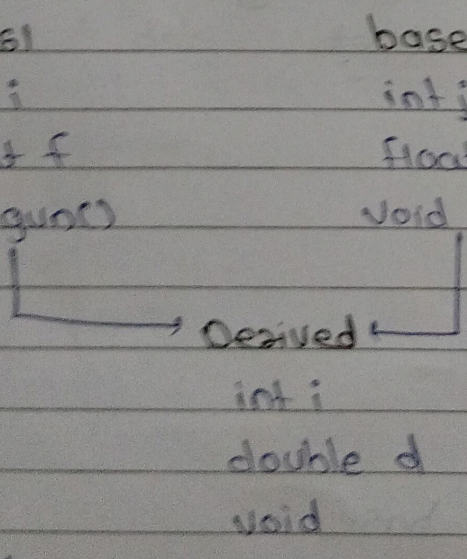



Inheritance Type-
Derived : public base
That means single level Inheritance with public access.
Thus the public and protected member of base are inherited by derived class.

Q.8] Draw object layout and class diagram of below snippets and explain its internal working in details.

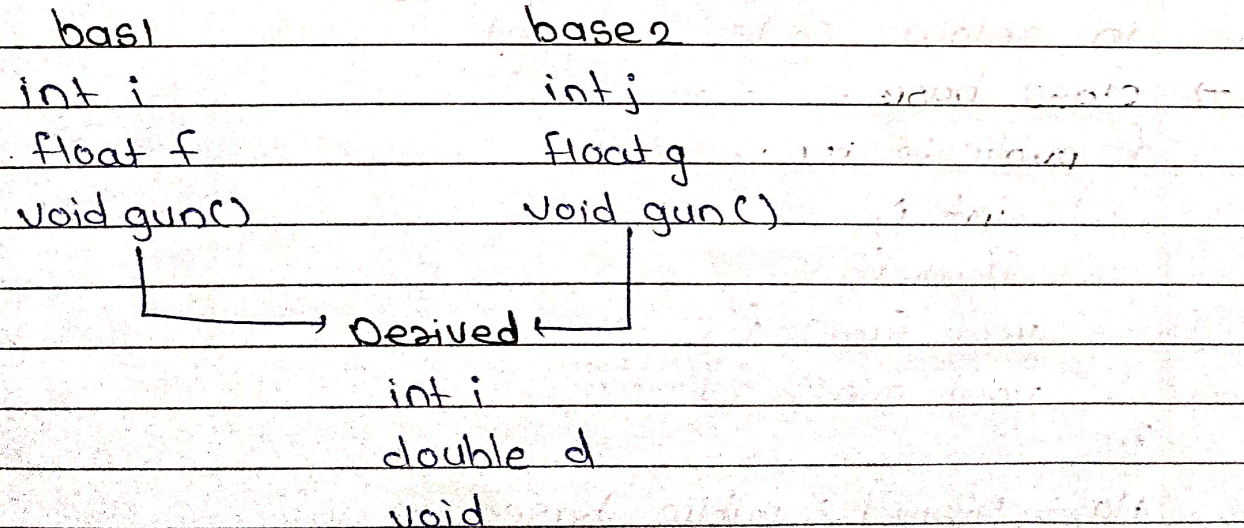
```
class base1
{ public :
    int i;
    float f;
    void gun()
    {}
};
```

```
class base2
{ public :
    int j;
    float g;
    void gun()
    { }
};
class Derived : public base1, base2
{ public :
    int i;
    double d;
    void sun()
    {}
};
int main()
{ Derived dobj;
  return 0;
}
```




Derived has memory allocated for:
All members of base1, base2 and it own members.

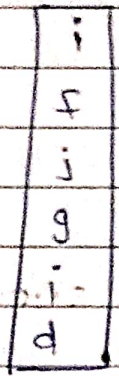
```
class base2
{ public :
    int j;
    float g;
    void gun()
    { }
};
class Derived : public base1, base2
{ public :
    int i;
    double d;
    void sun()
    {}
};
int main()
{ Derived dobj;
  return 0;
}
```




Derived has memory allocated for:
All members of base1, base2 and it own members.

Derived has access to gun() from bothe base1 base2.
sun() from itself.
If there's ambiguity, it must be resolved by explicitly specifying
base1::gun() or base2::gun()

dobj

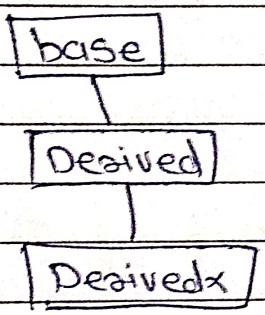



Q9] Draw object layout and class diagram of below code snippets and explain its internal working in details. Explain the type of inheritance in below code snippet.

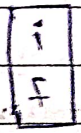
→
```
class base
{ public :
    int i;
    float f;
    void fun() {}
    void gun() {}
};
class Derived : public base
{ public :
    int i;
    double d;
    void sun() {}
};
```

```
class Derivedx : public Derived
{ public:
      int k;
      void run() {}
};
int main()
{ base bobj;
   Derived dobj;
   Derivedx dxobj;
return 0;
}
```

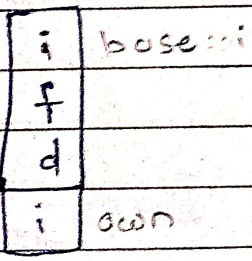
This is an example of Multilevel Inheritance



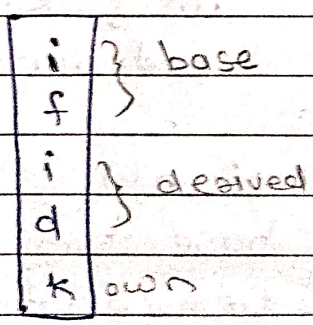

base bobj



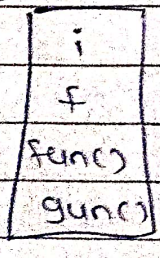

Derived dobj




Derivedx dxobj

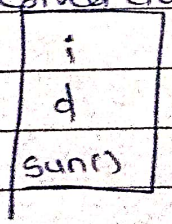



base class



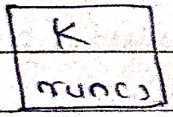

inherits

Derived class




inherits

Derivedx class



Q10 Draw object layout and class diagram of below code snippets and explain its internal working in details. Explain type of Inheritance.

→

```
class base
{ public :
    int i;
    float f;
    double d;
    void fun() {}
    void gun() {}
};
class Derived1 : public base
{ public:
    int x, y;
    void sun () {}
};
class Derived2 : public base
{ public:
    int j, k;
    void run() {}
};
int main()
{ base bobj;
  Derived dobj;
  Derived2 dobj2;
return 0;
}
```

Type of Inheritance is Hierarchical Inheritance.
- Both Derived1 and Derived2 include the members and functions of base.
Additional members from each derived class are stored sequentially in memory.
- Derived1 and Derived2 objects have access to all base members and functions due to public inheritance.
- They also have their own additional functions