object oriented programming concepts

1. Encapsulation
2. Inheritence
3. Polymorphism
4. Abstraction

Encapsulation

- Encapsulation is a one of the important Principal of java.
- Encapsulation means binding of data and methods into a single entity, i.e inside the class is called Encapsulation.
- To do this, you must make all data members of that class private.
- Through Encapsulation information Hidding is achieved.

```
class person {
    int age;
    string name;      } State or data or variables

    walk();
    eat();            } Behaviour or operations or methods.
}
```

- Java supports encapsulation by default.

The main advantages of Encapsulation are

(1) we can achieve security.
(2) Enhancement will become very easy.
(3) improves modularity of the program.
(4) High cohesion and low coupling
(5) immutable class creation.
(6) Unit test

Need of Encapsulation ?

```
Public class student {
        int age;
}

Student s1 = new Student();
s1.age = 30;  ✓

Student s2 = new Student();
s1.age = -70;  ✗ No negative age
```

So, Through Encapsulation the data is made private and use the setter method to set the data from outside the class.

```
Public class Student {
    Private int age;
    Public void setAge (int age)
    {
        if (age > 0)
        { age = age;
        }
        else { age = 0; }
    }
}
```

(-ve) value for age is not allowed, and data is private only member & the class only can access the private data.

* Tightly Encapsulated class?
  - A class is said to be tightly encapsulated if every data member declared as private.
  - Provided the setter and getter methods for access ~~add~~ data outside the class.

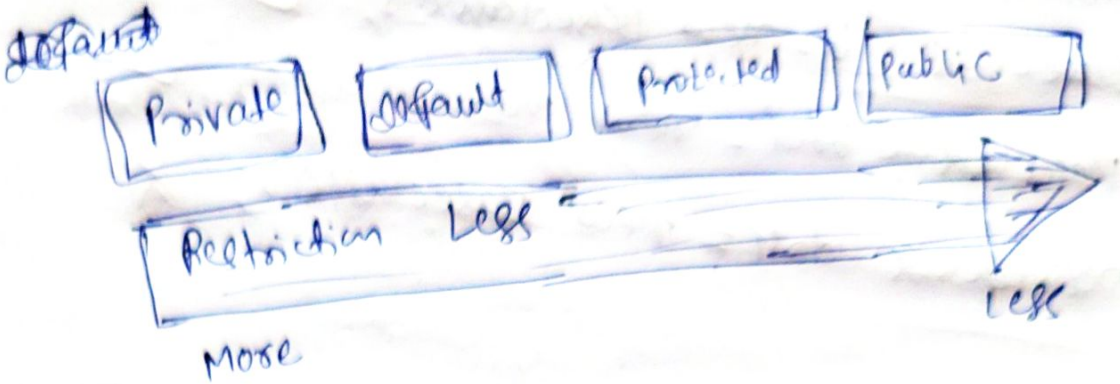
Inheritance

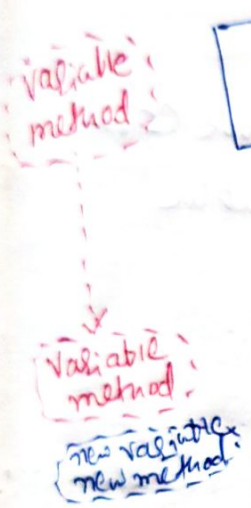
- The process by which one class acquires the properties (instance variables) and functionalities of another class is called Inheritance"
- The existing class is called superclass / Parent class / Base class.
- New class is called subclass / derived class / child class
- Inheritance is a important property or features of OOPs
- main goal of Inheritance is code reusability. i.e already existing functionalities can be used to Enhance the features in new Product / software / application.
- Inheritance represents IS-A Relationship is also known as a parent-child relationship.

Advantages of Inheritance

1. Application development time is less
2. Application take less memory
3. Execution time is less so performance increases
4. Redundancy (repetition) of the code is reduced
5. Code reusability.

In the Inheritance scope of access specifier increasing is allowed but decreasing not allowed.





Syntax:

```
class Baseclass {
    ..... // Base class methods and variables
}
class Derivedclass extends Baseclass {
    ..... // old features
    ..... // derived class methods & variables.
                New features.
}
```

Object creation?

```
Derivedclass d = new Derivedclass();   ✓
Baseclass    b = new Baseclass();      ✓
Baseclass    bref = new Derivedclass(); ✓
Derivedclass dref = new Baseclass();   ✗
```

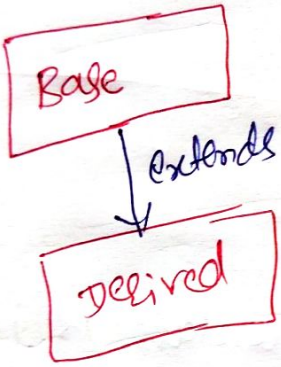
Keep In mind

* we can't assign superclass to subclass
* we can't extend final class
* we have private constructor only in the class we can't extend that class, if both private and public constructor we can extend that class.
* class can't extend itself
* java doesn't support multiple and Hybrid inheritance that is achived through interface.
* we can override Base class method in derived class.
* constructor are not inherited.
* Private members of class are not getting inherited in subclass.
* super keyword is used to accel Base class variables and methods.
* this keyword is used to access current class variables and method.
* final class can't be inherited.

Types of Inheritance

1. simple/single Inheritance.
2. Multi-level Inheritance
3. Hierarchical Inheritance
4. multiple Inheritance
5. Hybrid Inheritance.

1) Single Inheritance



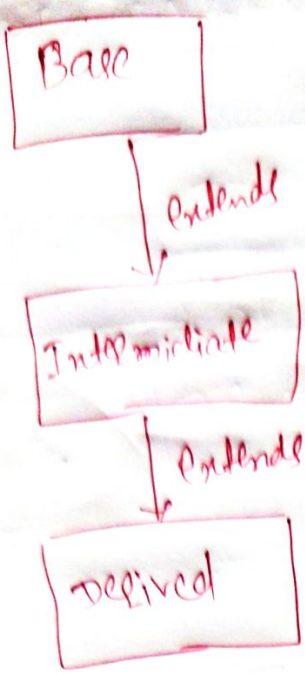
Syntax:

```
class Base {
    ---
    ---
}
class Derived extends Base {
    ---
    ---
    ---
}
```

There will be only one superclass and one subclass.

2) multi-level Inheritance

- One superclass can have only one direct subclass and many indirect subclasses.
- when a class extends a class, which in turn extends another class, is called multilevel inheritance.

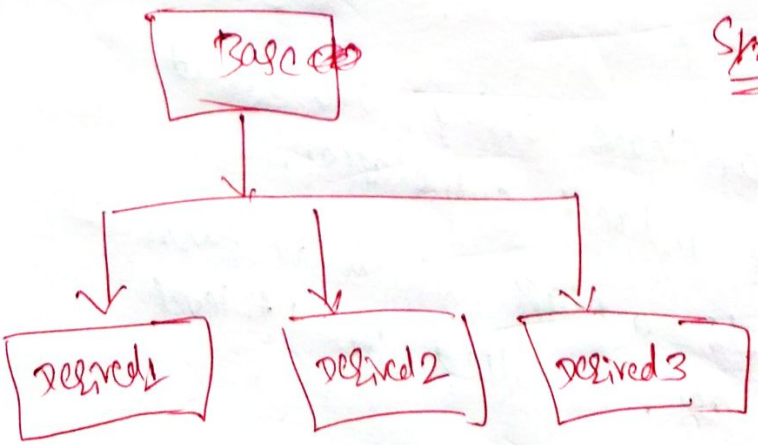


Syntax:

```
class Base {
    ...
    ...
    ...
}
class Intermidiate extends Base {
    ...
    ...
    ...
}
class Derived extends Intermidiate {
    ...
    ...
}
```

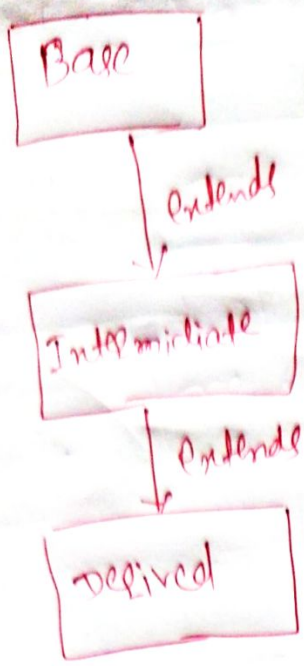
(3) Hierarchical Inheritance

- In such kind of Inheritance one class is inherited by many sub classes.
- One super class can have many direct sub classes.
- One subclass can have only one direct super class.



Syntax:

```
class Base {
    ...
}
class Derived1 extends Base
{
    ...
}
```

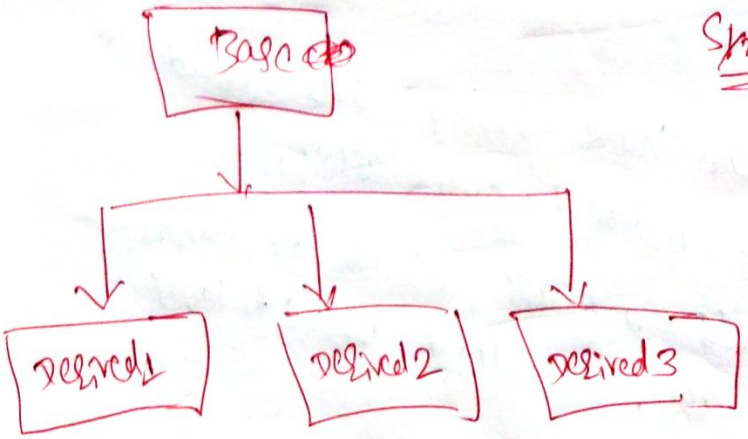


Syntax:

```
class Base {
    ...
    ...
    ...
}
class Intermidiate extends Base {
    ...
    ...
    ...
}
class Derived extends Intermidiate {
    ...
    ...
}
```

(3) Hierarchical Inheritance

- In such kind of Inheritance one class is inherited by many sub classes.
- One super class can have many direct sub classes.
- one subclass can have only one direct super class.



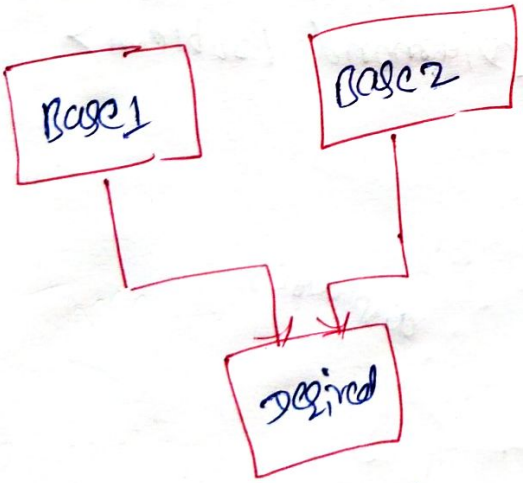
Syntax:

```
class Base {
    ...
}
class Derived1 extends Base
{
    ...
}
```

```
class Derived2 extends Base
{
---
---
}
class Derived3 extends Base
{
---
---
---
}
```

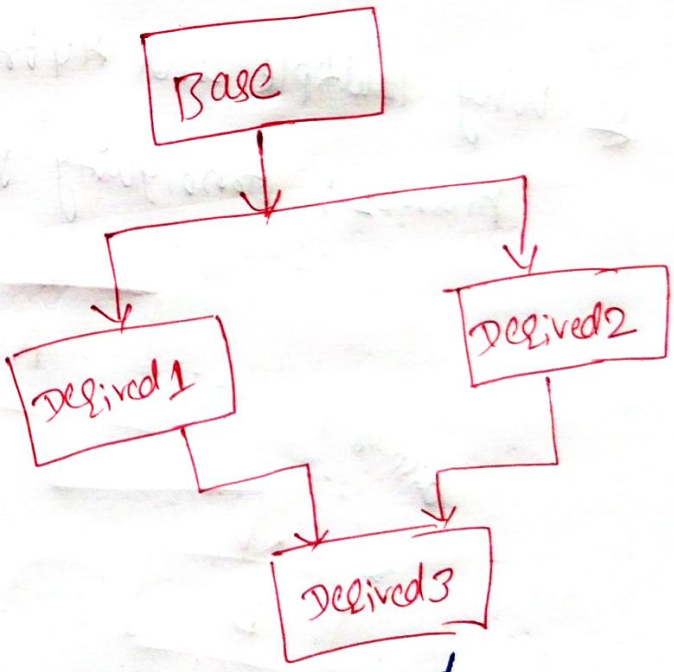
④ Multiple Inheritance
⑤ Hybrid Inheritance

} Multiple and Hybrid Inheritance not allowed in Java, But can achieve through Interface concept.



Note: Not allowed

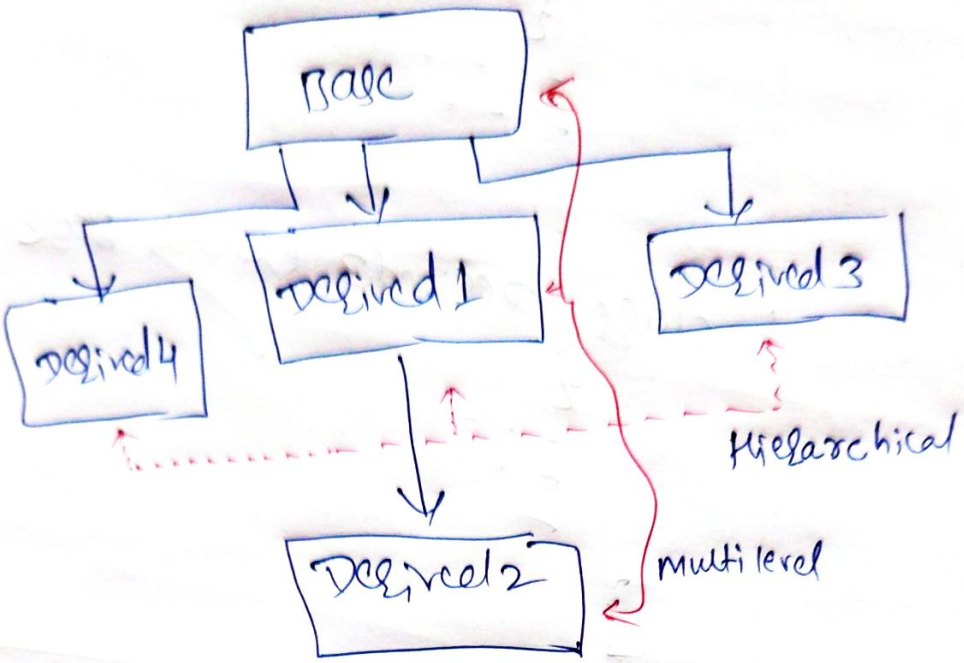
Multiple Inheritance



Note: Not allowed

Hybrid Inheritance.

Hybrid Inheritance (like this allowed)

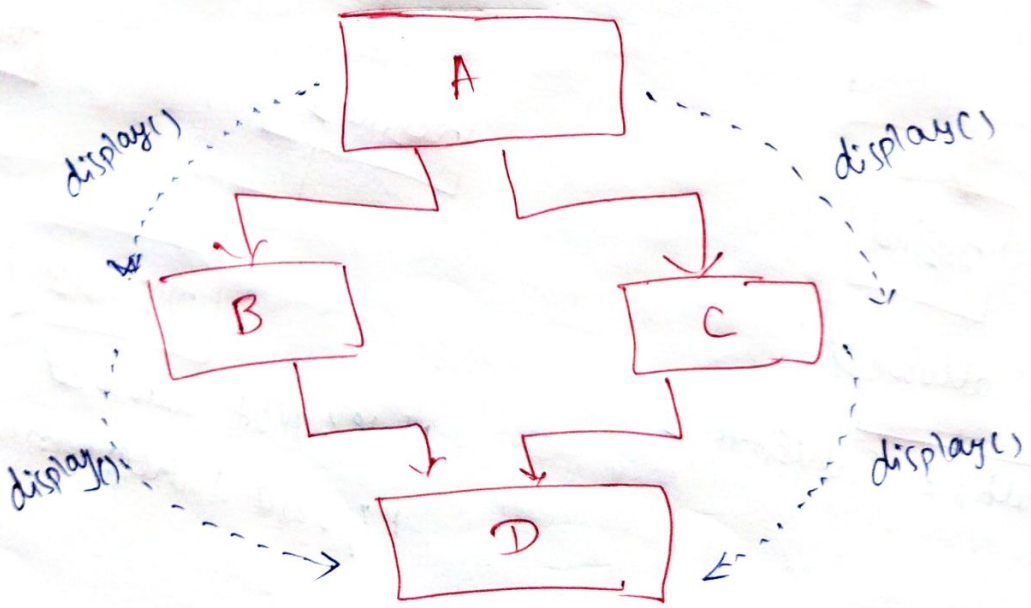


Allowed: Multilevel + Hierarchical = Hybrid
Single + Multilevel = Hybrid
} allowed

Not allowed: Multiple + Hierarchical = Hybrid
Multiple + Multilevel = Hybrid
} not allowed

* Why multiple and Hybrid are not allowed in java?

Answer: Ambiguity problem. <Diamond Problem>



class D which is derived from class B and C gets two copy of display() method.
during runtime which display method has to be called is it from the class B or from C. So it's an ambiguity problem because of that multiple and Hybrid are not allowed in java.

```
class B {
        void display()
        {
        SOP("I am from B class");
        }
}
class C {
        void display()
          {
          SOP("I am from C class");
          }
}
class D extends B, C {
        {
        // Both the methods with same name
        // are inherited to D, this causes ambiguity
        // Therefore multiple Inheritance not supported
        }
```

Not allowed