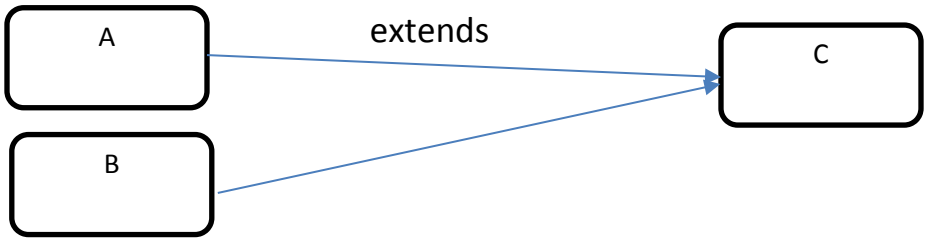
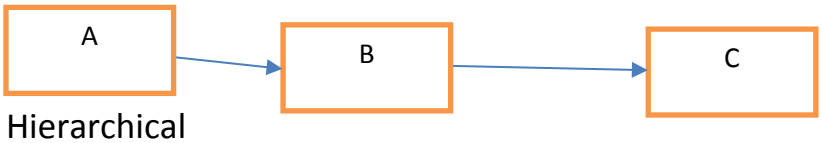
1. Inheritance
2. Types of Inheritance
    a. Single (class A=> class B)

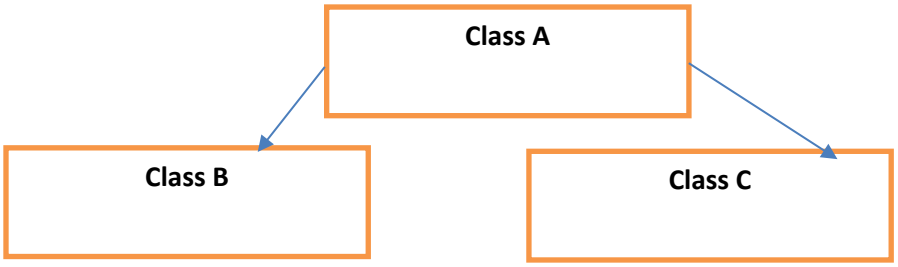
    b. Multiple (not allowed in java because of diamond problem, applicable with interface)



    c. Multilevel (class A => class B => class C)



    d. Hierarchical



3. Types of Relationship
    a. **IS-A** Relationship
        i. Using extends or implements
            1. Ex:-
                a. class Animal
                b. class Mammal extends Animal

c. class Dog extends Animal

2. Mammal IS-A Animal
3. Use of instanceOf keyword.

ii. Always with inheritance

b. HAS-A Relationship

i. Defining reference type in other class.

ii. Class vehicle

iii. Class Speed

iv. Class Van extends Vehicle{
Speed speed;
}

v. Van HAS-A Speed.

c. Parrent reference can point to child object but not vice-versa.

4. Reference variable typecasting
   a. Implicit
      i. Automatically it's being done
      ii. `RefB refBObj = new RefB();`
      iii. `RefA refAObj = refBObj;`
   b. Explicit
      i. Need to specify else it will be error
      ii. Check instanceOf before casting to avoid Exception/error
      iii. RefA refAObj1 = new RefB();
      iv. RefB refB1= (RefB) refAObj1;
5. private – Not inherited from parent to child as its invisible outside class
6. Over-riding
   a. Rules for overriding
      i. Method signature should be same.
      ii. Method body can be changed
      iii. In-case of co-variant return type :- return type should be same type or its sub-class
      iv. Rule for Exception (will cover in exception handling)

1. Can not throws new or broader exception in child class method.

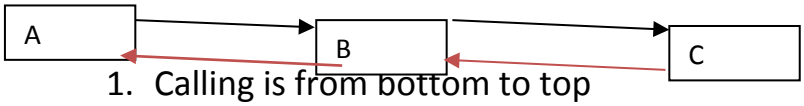
v. Can not narrow the access modifier. (public => private)

b. Super keyword

i. Class methods/variables from immediate super class

1. Practice on accessing super class variable using super.

ii. To call the immediate class cons. C c = new C();



1. Calling is from bottom to top
2. Execution is from top to bottom.
3. Assignment:-
    a. Write a program to show calling and execution sequence when constructor, static & non-static init blocks are used with parent and child class (A=>B=>C).

c. Can not override static method (no compile error when try to override)

d. Use of static and non-static block with super and this .

i. Write a prog to make use of static and non-static block , use of super and this keyword with reference to Inheritance.

7. Over-loading
    a. Method name should be same
    b. There should be different input type or number of input type parameter.
    c. Can have any access modifier
    d. Can modify the return type.

```
public void calcArea(int length ,int width) {
System.out.println("Area of Rectangle");
    }
```

```
int calcArea(int side) {
    System.out.println("Area of Square");
    return side*side;
    }
```

8. final keyword
    a. final variables once declared must be initialized
    b. value cannot be changed
    c. For final method
        i. It cannot be overridden
    d. Final class can not be inherited

1) Static method overloading. Write program.
2) Student , College , University (three class)
    Create main class and there should be a functionality to add and one method to **getstudentDetails** () should print all these values (st_name , st_rollNumber,st_address , college_name , department_id/name , University_name, uni_addres)

1. Abstract class (incomplete class)
    a. Use abstract keyword.
    b. Can not create object of Abstract class
        i. Reason: Class is not complete and if we create an object then, incomplete method can be accessed.
    c. Abstract means incomplete
    d. Can have both complete & incomplete(abstract) methods.
    e. Private & abstract keyword cannot be used together
    f. Abstract and final keyword can not be used together.
    g. Used to have different (abstract method) & default (complete method) behavior

3. Interface
    a. Using interface java have multiple inheritance

b. Class implements an interface
c. By default methods are abstract and public
d. Interface keyword is used to define
e. Interface cannot be instantiated (cannot create object)
f. Interface do not have constructors
g. Can have both abstract and complete (static & default method)
    i. Static & default(keyword)
    ii. Static methods can only be accessed by Interface name.
h. Creates IS-A relationship
i. Variables
    i. By default its public static final
j. Interface extends another interface