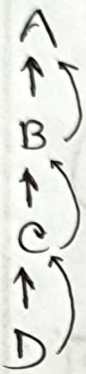
CSE391 Advanced Programming
22-01-26 | class - 06

Multi-Level Inheritance



→ super

*Constructor Chaining

```
class A{
    String name;
    A(String n){ name = n}
}

class B extends A{
    String name1;
    B(String n, String n1){
        super(n);
        name1 = n1;
    }
}

class C extends B{
    String name2;
    C(String n, String n1, String n2){
        super(n,n1);
        name2=n2;
    }

public class Main{
    psvm(..){
        C ob1 = new C("AB","CD","EF");
```

```
class Grandfather{
    private String name;
    private int age;
    Grandfather(String n, int a){
        name=n; age=a;
    }
    void show(){
        System.out.println(name);
        System.out.println(age);
    }

class Father extends Grandfather{
    private String name1;
    private int age1;
    Father(String n1, int a1){ super(n,a); name1=n1; age1=a1;}
    void show(){ super.show();
        System.out.println(name1);
        System.out.println(age1);
```

```
class Son extends Father{
    private String name2;
    private int age2;
    Son(String n2, int a2){ name2=n2; age2=a2;}
    String n, int a, String n1, int a1, super(n,a,n1,a1);

    void show(){
        super.show();
        System.out.println(name2);
        System.out.println(age2);
    }
}

public class Main{
    public static void main(String[] args){
        Son s = new Son(
                "Gnd", 73,
                "Dad", 48,
                "Me", 23
            );
        s.show();
    }
}
```

#1 Dynamic Method Dispatch / Runtime Polymorphism **

super class়র ref variable sub class়র obj কে point করতে পারে।

```
class A{
    Void show(){ System.out.println("class A"); }
    Void show2(){ System.out.println("another method of A"); }
}
class B extends A{
    void show(){ System.out.println("class B"); }
    void show1(){ System.out.println("Another method."); }
}
public class Main{
psvm{
    B ob1 = new B();
    A ob2 = ob1;   // A ob2;
    // ob2 = ob1
    ob2.show();   // "class B"
    ob2.show1();  will cause error
}
}
```

{ abstract method হলে subclass এ ঐ method override করা বাধ্যতামূলক }

final Keyword

usage:

1. as constant
2. declare a class as 'final'
   - ↳ no ~~~~ other class will be able to inherit that class
3. declare a method as 'final'
   - ↳ nothing else will be able to override that method.

```
class Vehicle {
    void start() {
        System.out.println("Vehicle started");
    }
}

class Car extends Vehicle {
    void start() {
        System.out.println("Car started");
    }
}

class Bike extends Vehicle {
    void start() {
        System.out.println("Bike started.");
    }
}

public class Main {
    public static void main(String[] args) {
        Car c1 = new Car();
        Vehicle v1 = c1;
        v1.start();
        Bike b2 = new Bike();
        v1 = b2;
        b2.start();
    }
}
```