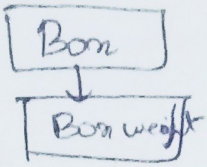
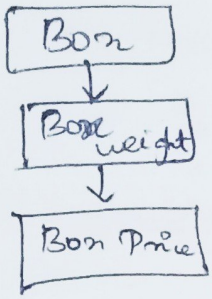
# Day-10

## Types of inheritance:

① Single inheritance: One class extends another class




② Multi-level inheritance: One class can inherit derived class, derived class become parent for another class




Bon Price can inherit Bon weight & Bon

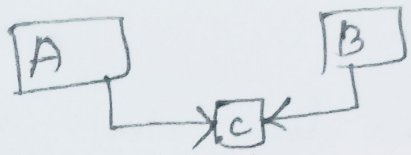
Eg:

```
class Animal {
    void eat() {
        System.out.println("Animal can eat");
    }
}
class Dog extends Animal {
    void Bark() {
        System.out.println("Dog Barks");
    }
}
class Puppy extends Dog {
    void weep() {
        System.out.println("Puppy weeps");
    }
}
public class Main {
    public static void main(String[] args) {
        Puppy p1 = new Puppy();
        p1.eat(); -> We call eat in Animal class.
    }
}
```

Op:

Animal can eat.

③ Multiple Inheritance: One class extend more than 1 class



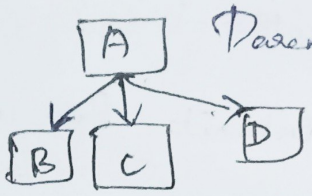

Not allowed in Java

We will do this interface

C can access A & B

```
C obj = new C();
C.m   // O/p: will be confused
```

④ Hierarchical Inheritance: One class inherited by many classes




Parent class

child class

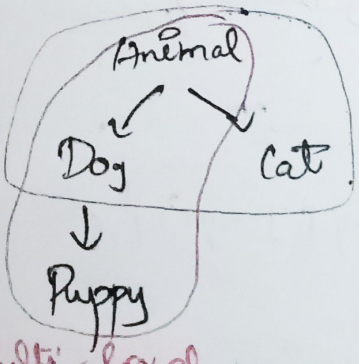
Eg:

```
Class Animal {                    Parent
    void eat() {
        System.out.println("Animal can eat");
    }
}
Class Dog extends Animal {        child ①
    void Bark() {
        System.out.println("Dog Barks");
    }
}
Class Cat extends Animal {        child ②
    void meow() {
        System.out.println("Cat Meow");
    }
}
```

Both Dog & Cat can Eat

⑤ Hybrid Inheritance: Both Multiple & Hierarchical combine

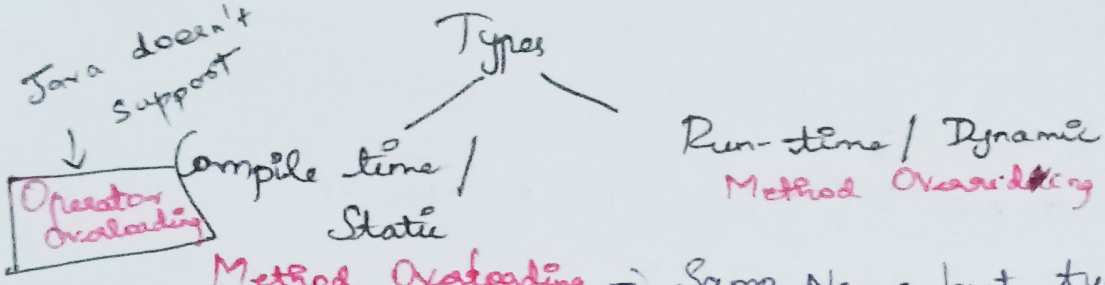
NOT in Java




Hierarchical

Multi-level

**Polymorphism:** Many ways to represent — Greek word




Method Overloading ⇒ Same Name but types, arguments, Return type can be different, these functions can be overloaded.

Eg: Multiple Constructors

Eg: Method Overloading

```
class Helper {                                   Same Class name
    static int Multiply (int a, int b) {         Static are overloaded
        return a*b;                              We can also add extra int c
    }                                            a*b*c;
    static double Multiply (double a, double b) {
        return a * b;
    }
}
class Polymorphism {
    public static void main (String [] args) {
        Syetem.out.println(Helper.Multiply(2,4));      2,4,1   Based on the I/p as an argument
        Syetem.out.println(Helper.Multiply(5.5, 6.3));
    }
}
```

O/p: 8
34.65

**Method Overriding:**

Run-time polymorphism.

In which the compiler can allow a child class to implement a specific method already provided in the parent class.

Eg:

```
class Parent {
    public void show(){
        System.out.println("Inside Parent Class");
    }
}
Class subclass1 extends Parent {
    public void show(){
        System.out.println("Inside Subclass1");
    }
}
Class subclass2 extends Parent {
    public void show(){
        System.out.println("Inside subclass2");
    }
}
public class Main {
    public static void main(String [] args){
        Parent p;
        p = new subclass1();
        p.show();
        p = new subclass2();
        p.show();
    }
}
```

Here we called subclass 1 & 2
So, Parent Class does not display.

Op:

Inside subclass1
Inside subclass2