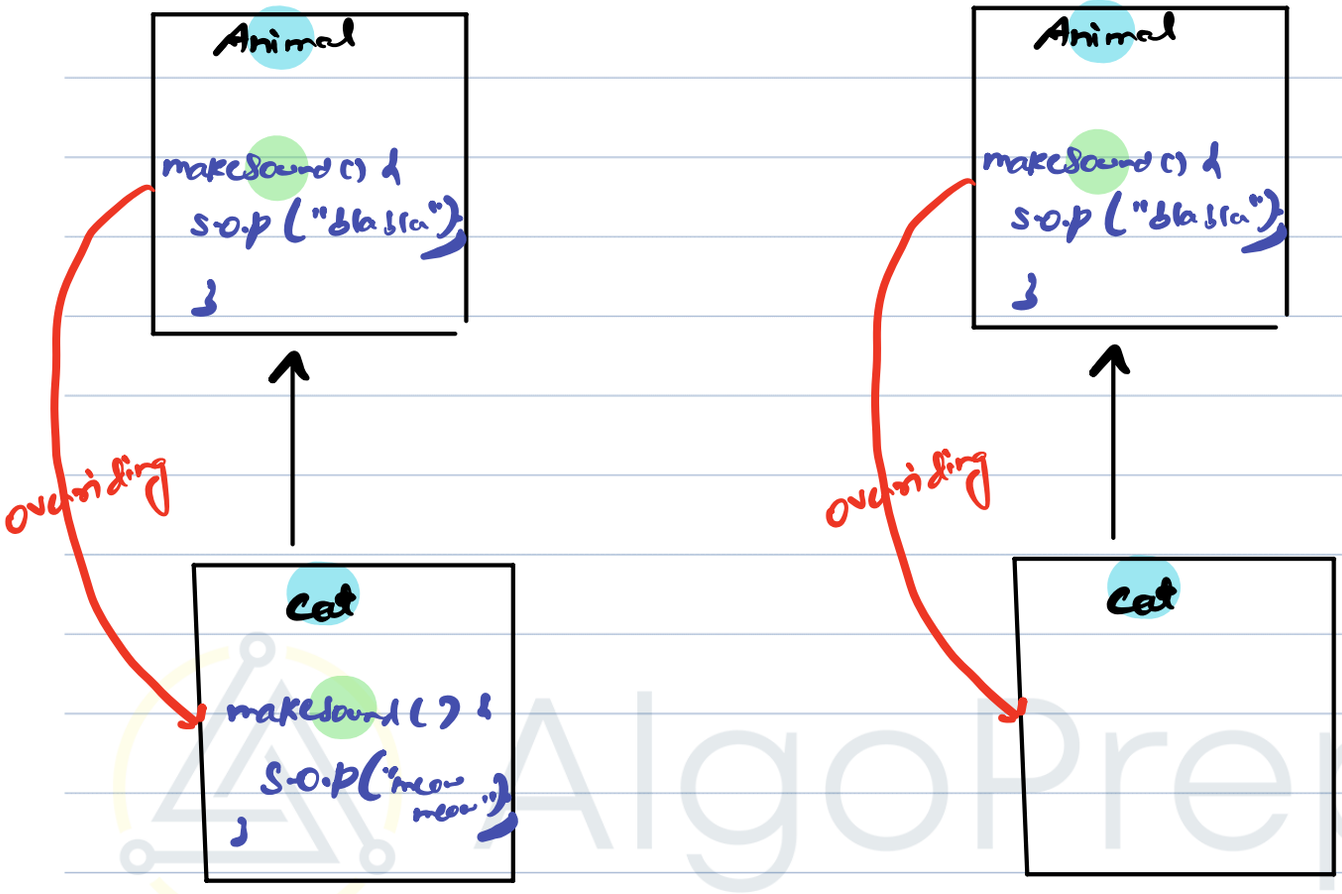
Today's agenda

- Overriding
- Interfaces
- Abstract classes
- Static

* Method overriding → overwrite / replace / change



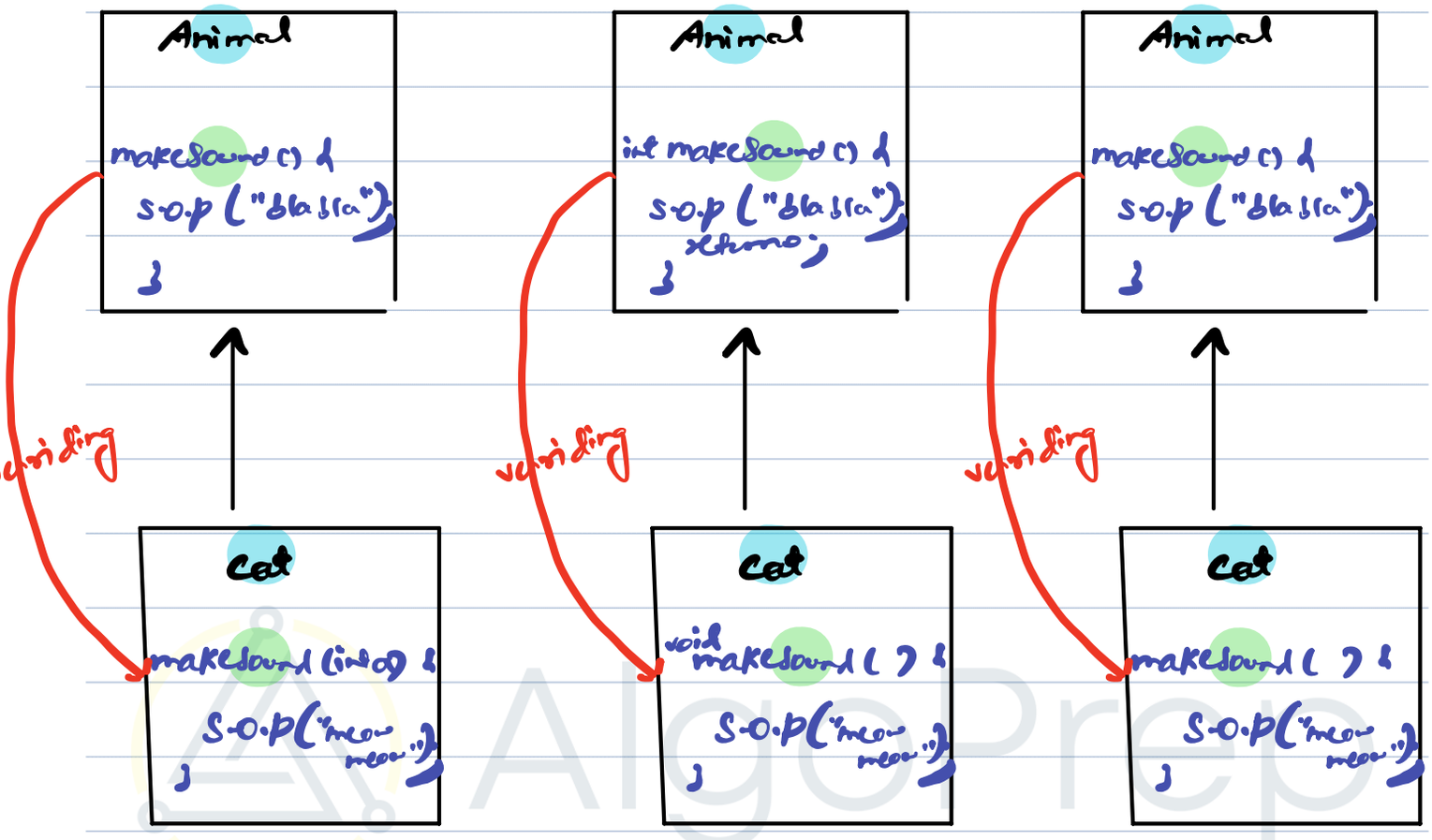

```
Cat obj1 = new Cat();
obj1.makeSound();
```
↓
meow meow

```
Cat obj2 = new Cat();
obj2.makeSound();
```
↓
bla bla

Overriding:

↳ A method with same signature and same return type is present in Parent class as well as child class then method of child class overrides the method of Parent class.

Ex:



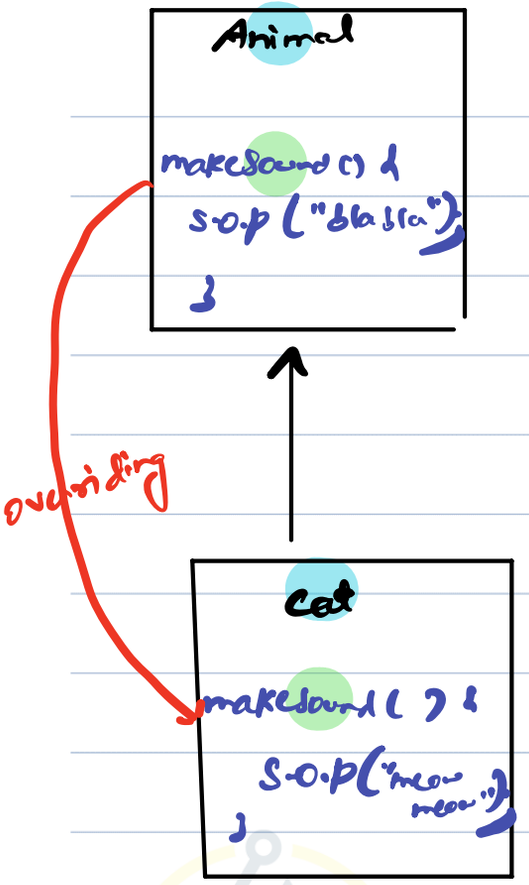
✳ method overriding ✗✗

✳ method overloading ✓

✳ error

✳ method overriding

```
Cat obj1 : new Cat();
    Obj1.makSound();
    Obj1.makeSound(10);
```



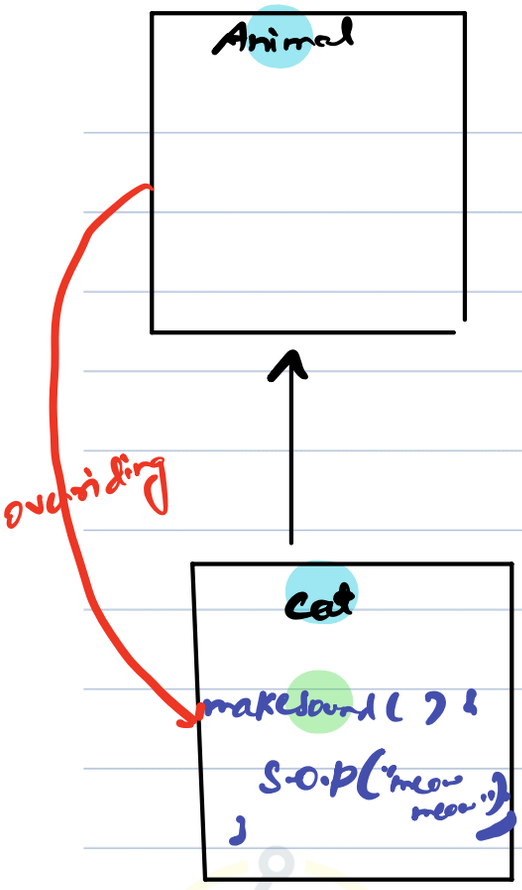

```
Animal a = new Animal();
a.makeSound(); → bla bla
```

```
Cat c = new Cat();
c.makeSound(); → meow meow
```

```
Cat c = new Animal();
                ↳ → error
c.makeSound();
```

```
Animal A = new Cat();
A.makeSound(); → meow meow
```

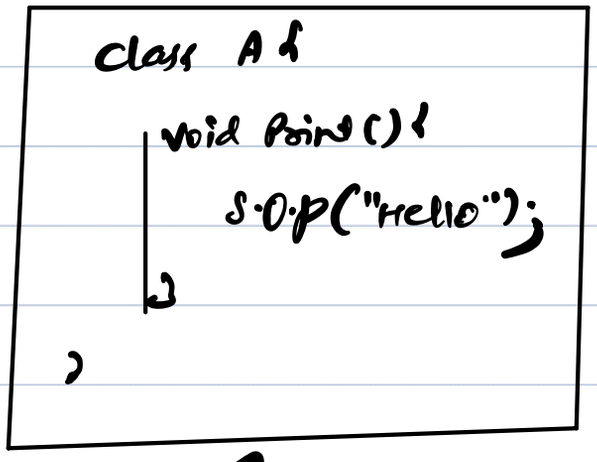
↳ the method that gets pointed is of the object that is stored in the variable and not the data type of variable.

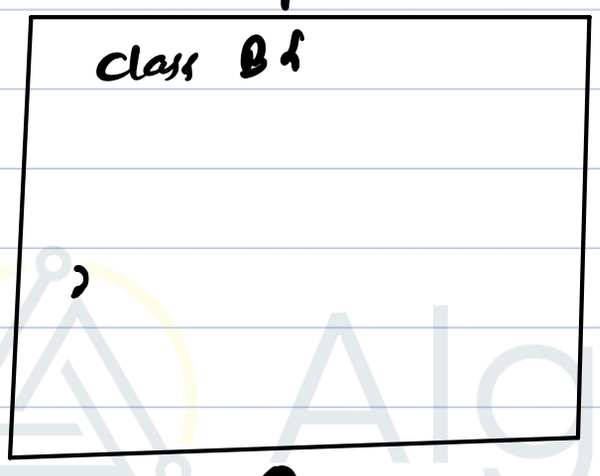


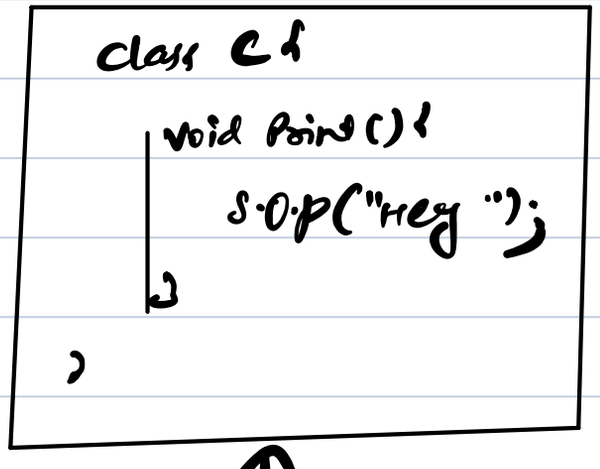
Animal A : new Cat();

A.makeSound(); → error

Q.







A obj1 = new D();
obj1.Print();
Is Hey

A obj2 = new B();
obj2.Print();
Is Hello

B obj3 = new D();
obj3.Print();
Is Hey

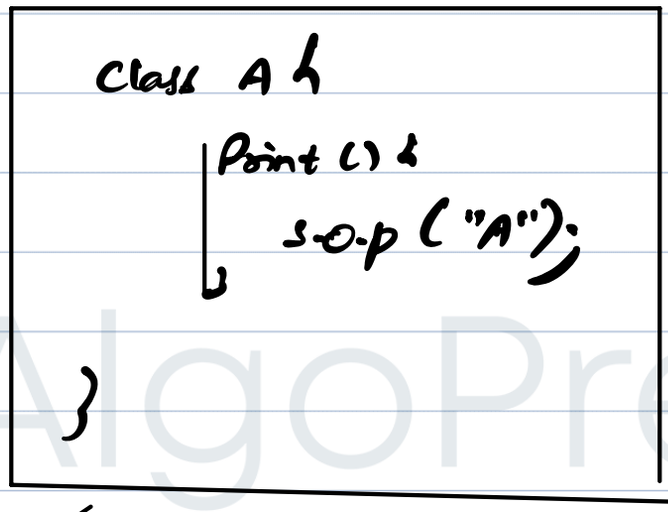
* Multilevel inheritance.

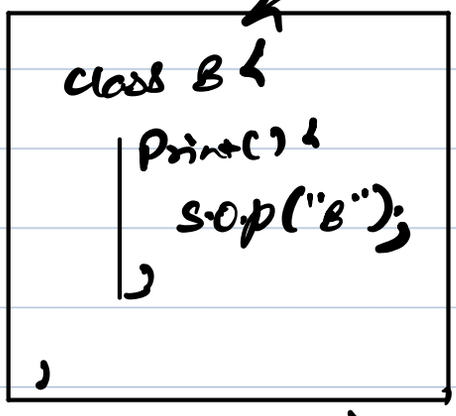
↳ yes, supported in java

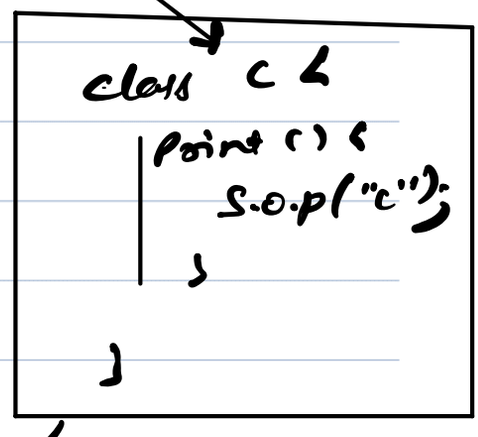
* multiple inheritance

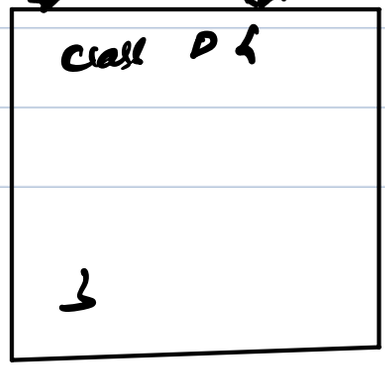
↳ no, not supported in java.

↳ diamond problem









D extends C, B {

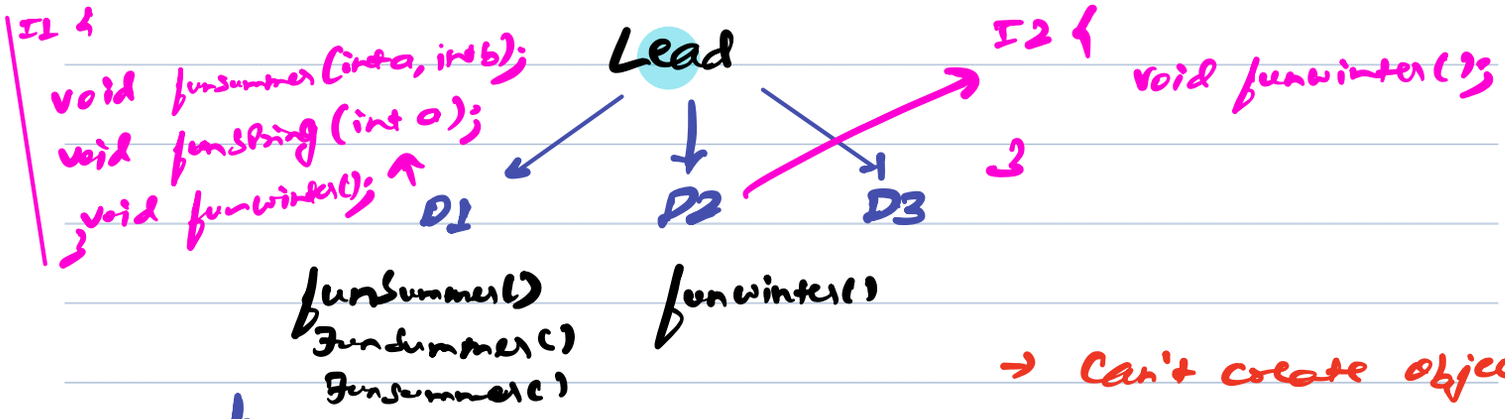
}

D obj : new D();

obj . Print();

Interfaces

↳ contract / blueprint of behaviours.



→ Can't create object of interface directly.

Class S implements I1, I2 {

→ One class can implement any no. of Interfaces.

```
void funsummer(int a, int b) {
    ⋮
}
void funspring(int a) {
}
void funwinter() {
}
}
```

→ Interface is just a contract.
↳ implementation of methods will be when you "implements" the interface in a class.

Eg:

```
class Pair implements Comparable<Pair>{

    int compareTo(){

    }
}
```

```
Pair[] arr;
Arrays.sort(arr);
```

Note: In Interfaces, multi-level inheritance as well as multiple inheritance is possible.

abstract

↳ Partial interface

```
abstract class animal {
    String name;
    int age;

    void makesound() {
        s.o.p ("blabla");
    }

    abstract void eat();
}
```

↳ objects can't be created.

```
abstract class dog extends animal {
    void test() {

    }
}
```

↳ inheritance of abstract class is possible but the child class is also going to be abstracts

Abs Class	Interfaces
abs + methods inheritance is possible within class.	methods implements in class.

Static

↳ Creating method or attributes directly for a Class rather than it's objects.

Eg: Math.abs(), Math.max()

```
Class Math {
    static int abs (int a) {
        :
    }
    static int max (int a, int b) {
        :
    }
}
```


↳ Can you access non-static from static method.

↓

No

↳ Can you access static from non-static method

↓

yes

int[] arr = {10, 20, 20};

~~bubble sort~~ Quick sort

Arrays.sort(arr);

if(arr[j] > arr[j+1]){

swap();

}

{ 10 30 40 20 }

if(condition())

10
a
obj1

int[] arr;

Pair[] arr;

{ (10,a) (30,b) (40,c) (50,d) }

20
b
obj2

Arrays.sort(arr);