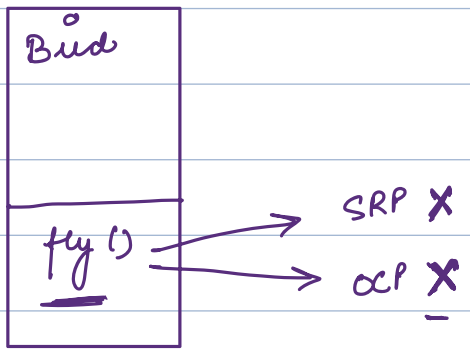
L : Liskov's substitution

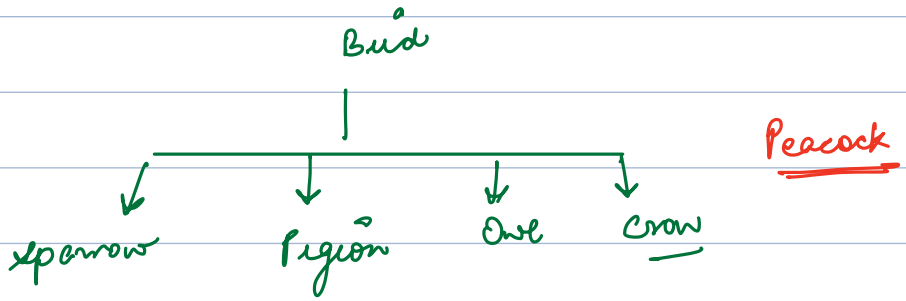
I : Interface segr.

D : Dependency inversion

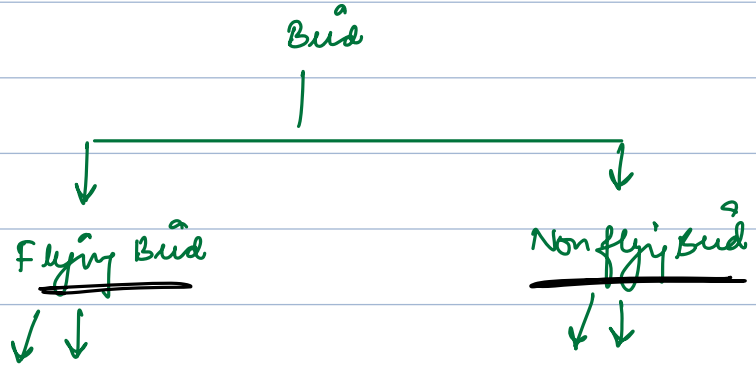
V0



V1



V2



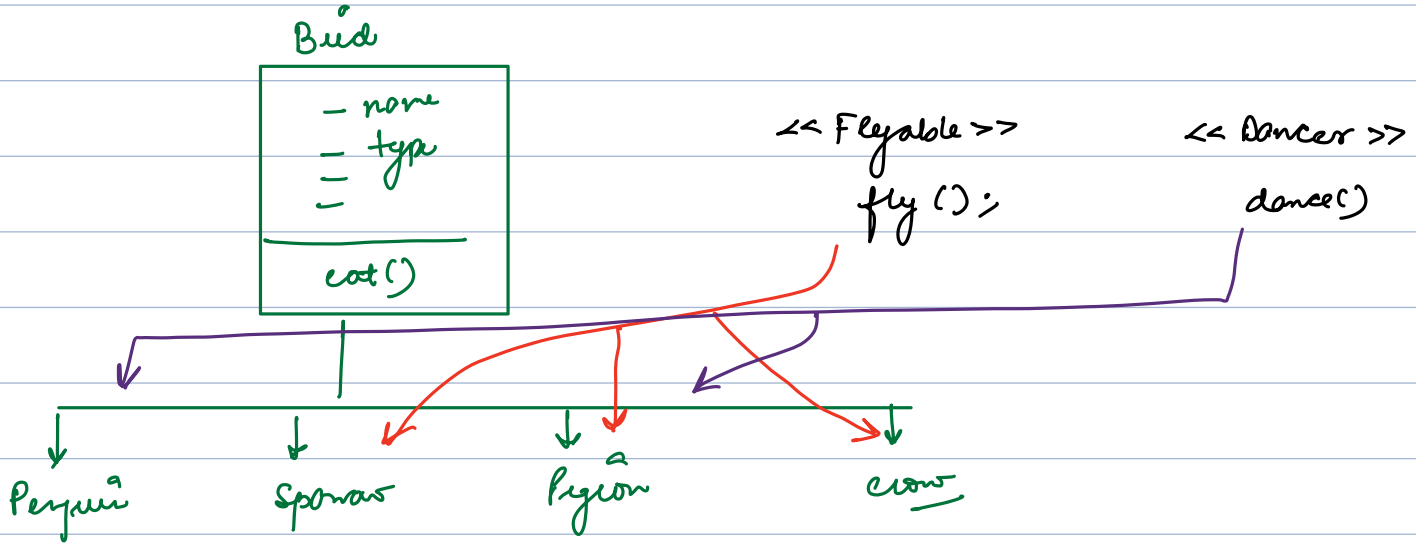
Problem Statement :- Some Birds demonstrate a particular behaviour and some doesn't

List < Birds > ?

Peacock Penguin Sparrow

1. Only the birds having the behaviour should have the methods
2. create the list of the birds that support a behaviour

interface



List < Flyable > ___ ;

contain all the birds which can fly

Liskov's substitution principle

Object of any class should be as-is substitutable in a variable of parent-type without requiring any code changes.

```
Bird b = new Bird();
         Sparrow();
         Pigeon();
         Penguin();
```

```
b.fly();
```

↓

- Throw the error
- leave it empty

Interface segregation Principle

Some Birds can fly

Some Birds can dance

All the birds which can fly can also dance

& vice versa

(1)

<<FlyDance>>

- fly()
- dance()

(2)

<<Flyable>>

fly()

<<Dancer>>

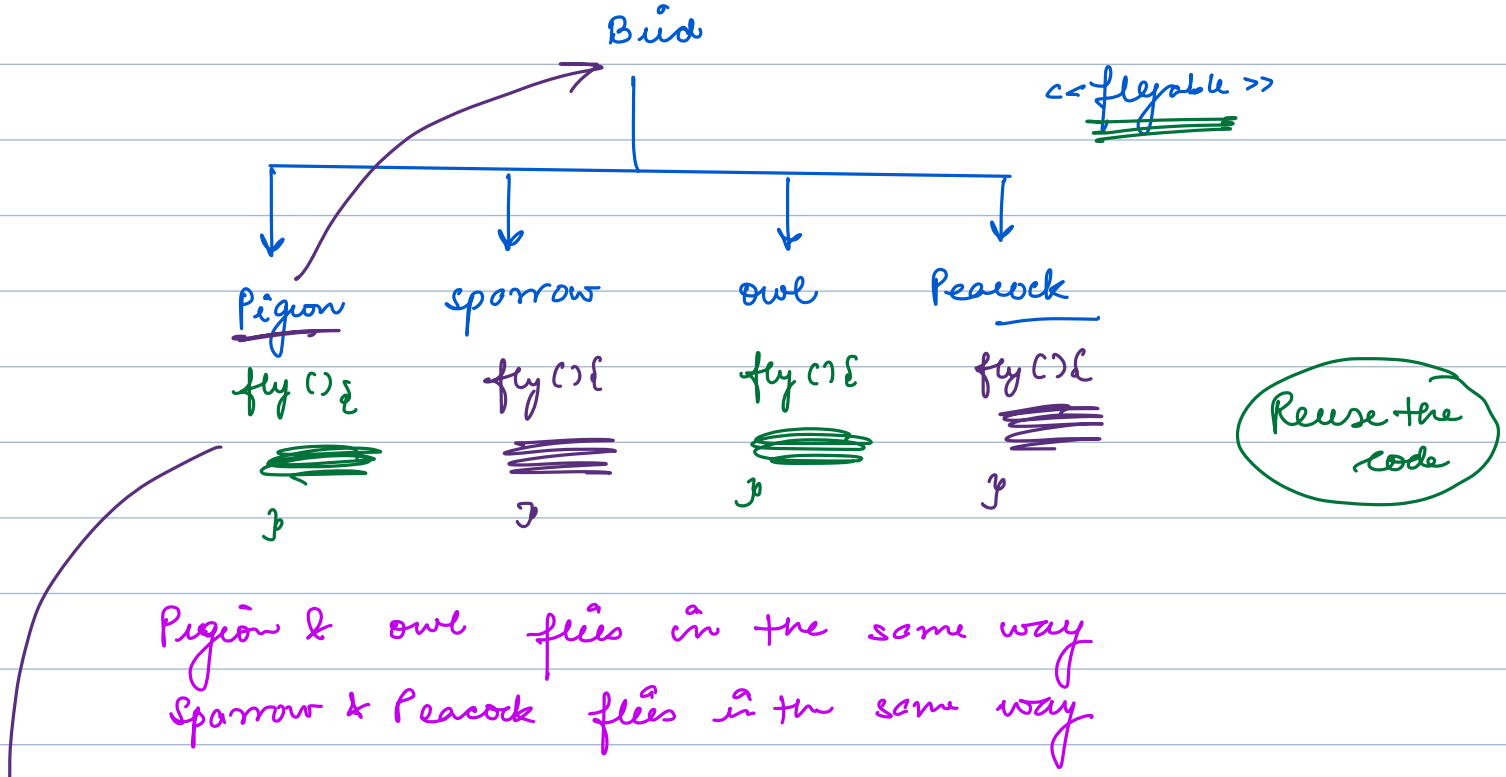
dance()

- Interface should be as light as possible
- as less methods as possible
- ideally only 1 method

Functional interfaces

Lamba expressions

Dependency Inversion Principle



Pigeon & owl flies in the same way

Sparrow & Peacock flies in the same way

```
PigeonOwlFlyingBehaviour {

    makefly() {

    }

}
```

```
SparPeacFlyingBeh {

    makefly() {

    }

}
```

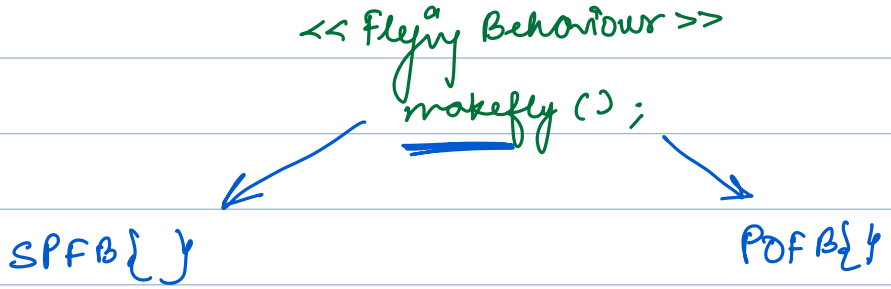
Pigeon

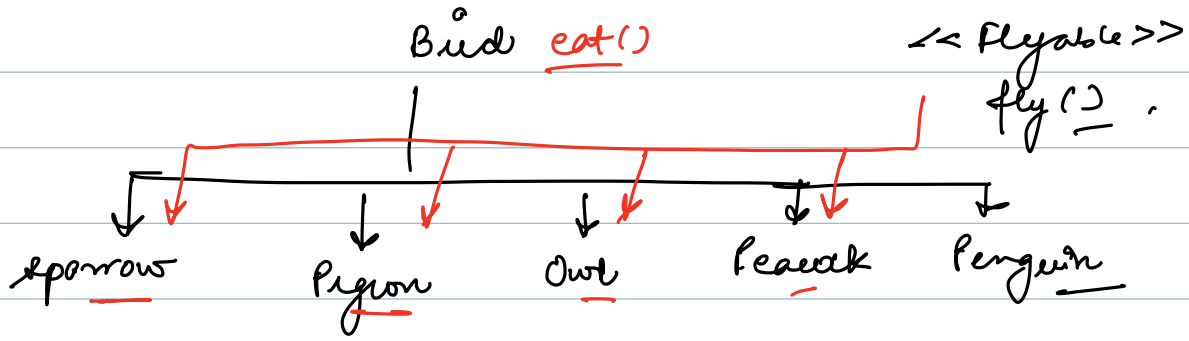
```
FlyingBehaviour fb
PigeonFB pofb = new POFB();

fly() {
    pofb.makefly();
}
```

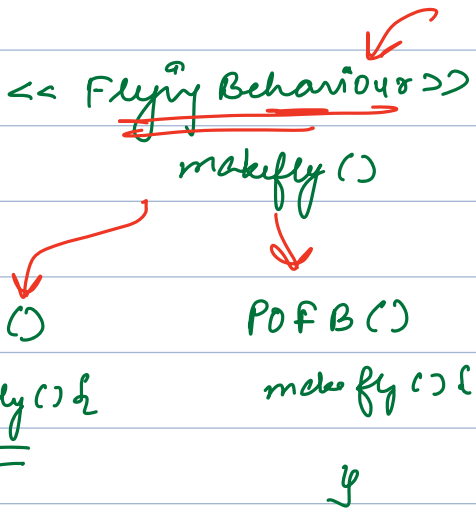
Sparrow

```
flyingBehaviour fb =
SPFB spfb = new SPFB();

fly()
{
    spfb.makefly();
}
```

→ No 2 concrete classes should be dependent upon each other, they should be dependent via interfaces.




Bird
eat()
<< Flyable >>
fly()
Sparrow
Pigeon
Owl
Peacock
Penguin


<< FlyingBehaviour >>
makefly()
SPFB()
makefly(){
}
POFB()
makefly(){
}

Dependency injection

If class A has an attribute of class B

A depends on B

```
Pigeon {
    FlyingBehaviour fb;
    Pigeon(FlyingBehaviour fb) {
        this.fb = fb;
    }
    void fly() {
        fb.makefly();
    }
}
```

```
PofB b = new PofB();
Pigeon p = new Pigeon(pofb);
```

Assignments → 20 oct 2023

contest → Monday