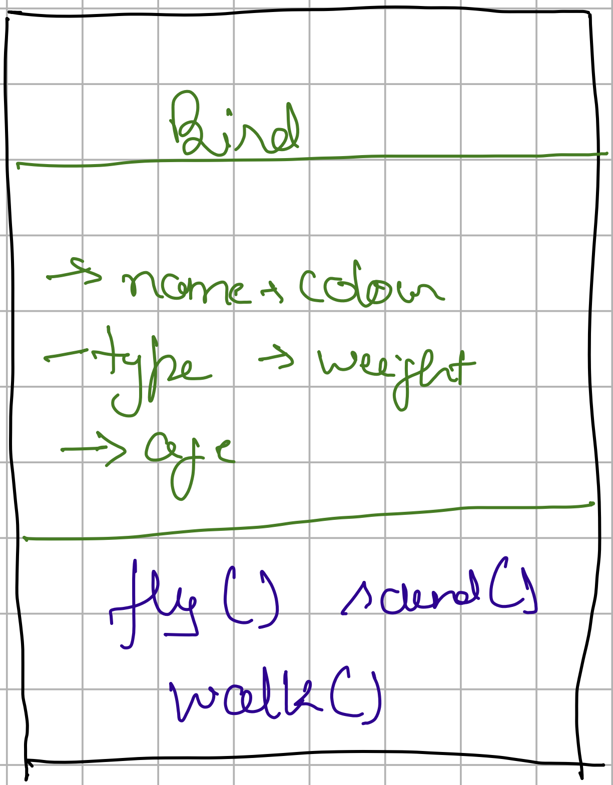
SOLID Design Principals.

S → Single Respsiblity Principal
O → Open for extension Closed for Modification
L → Linkofs Substitution Principal
I → Interface segrigation Principal
D → Dependency Inversion.

Q. Design a Bird?



V0

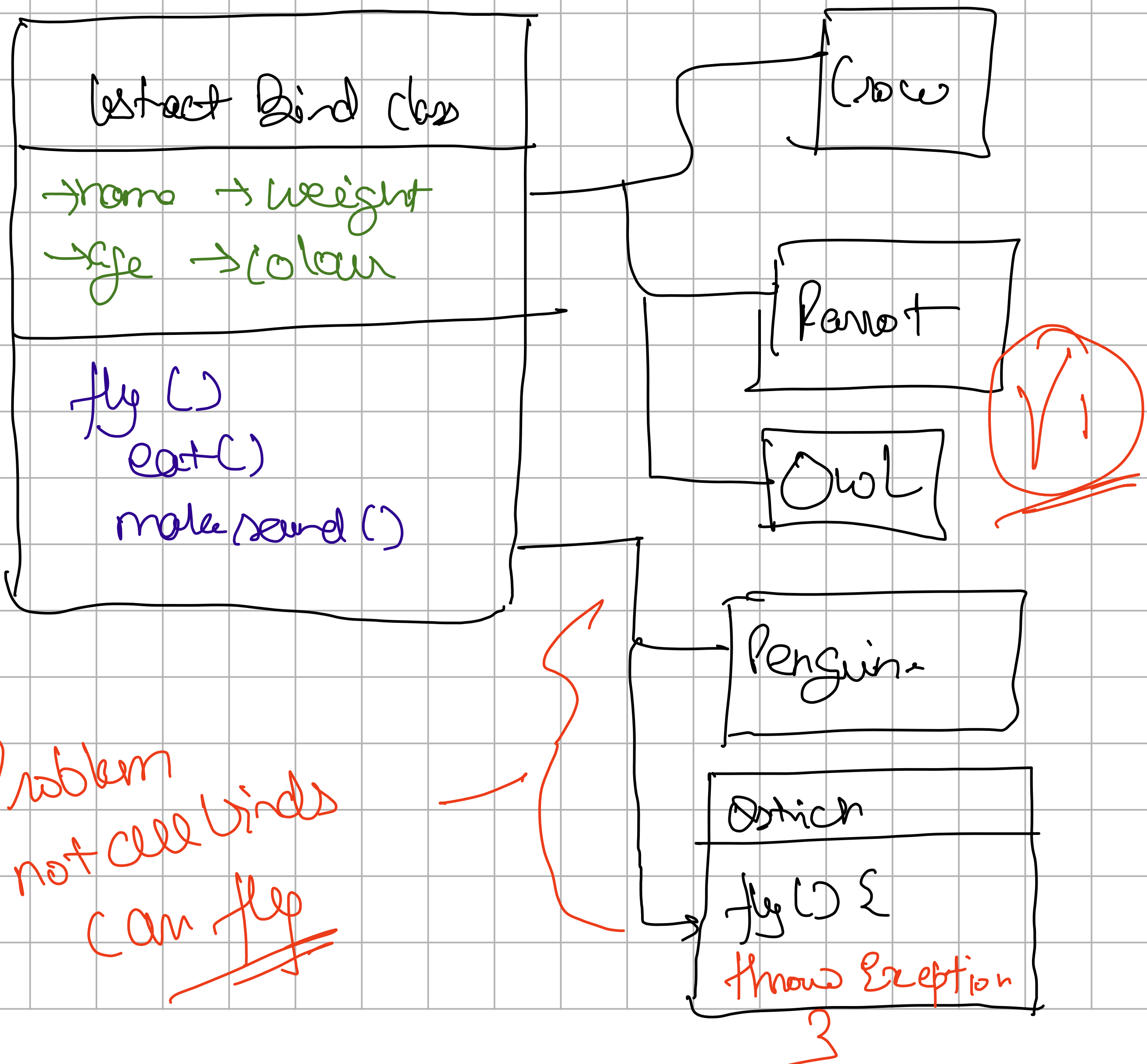
SRP Single Responsibility Principal

Every code unit

Class method Package

Should have exactly 1 responsibility

it should have code to do 1 thing and one reason you might want to change

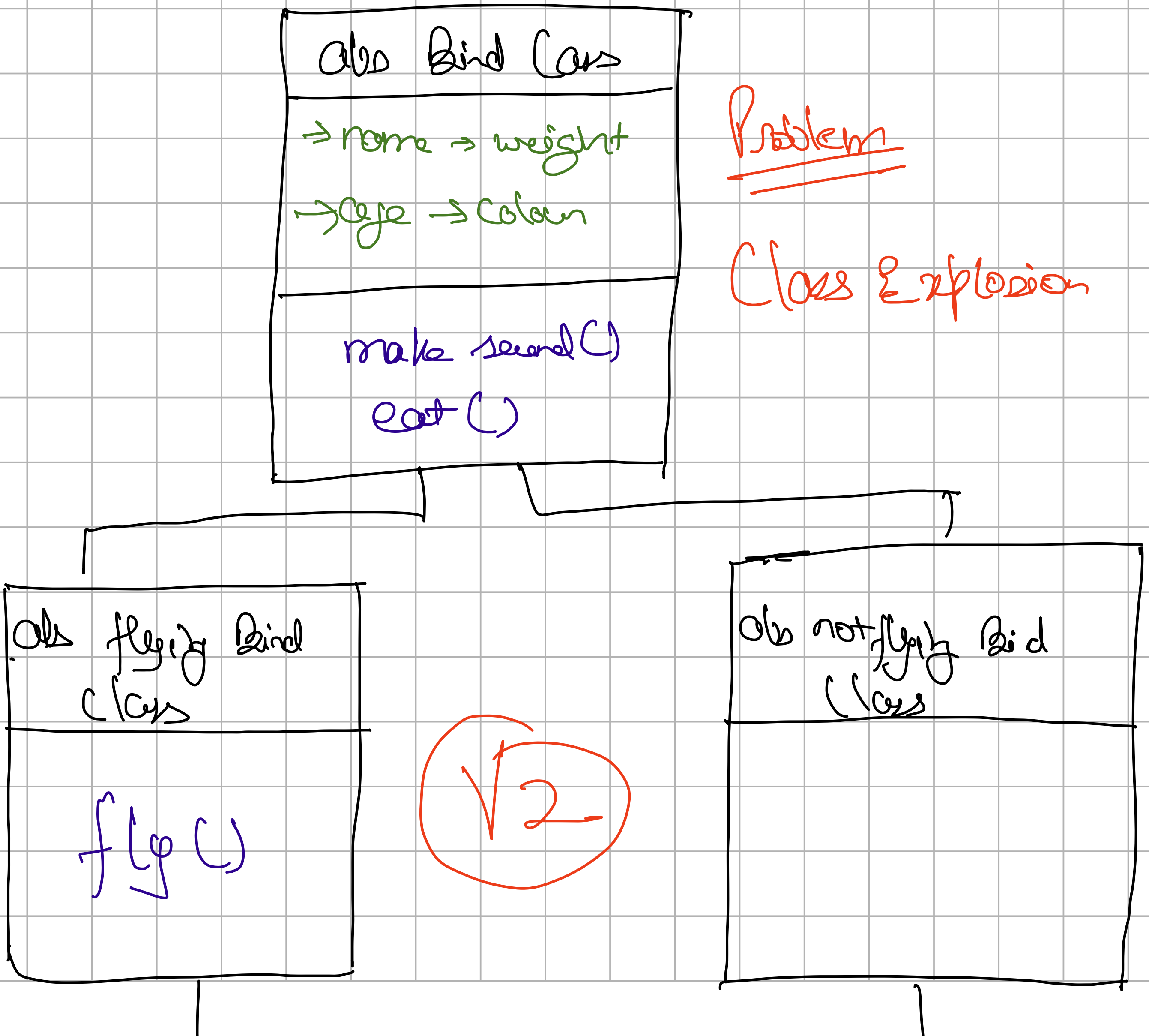


② Open Closed Principal

⇒ A code base should be open for extension but closed for modification

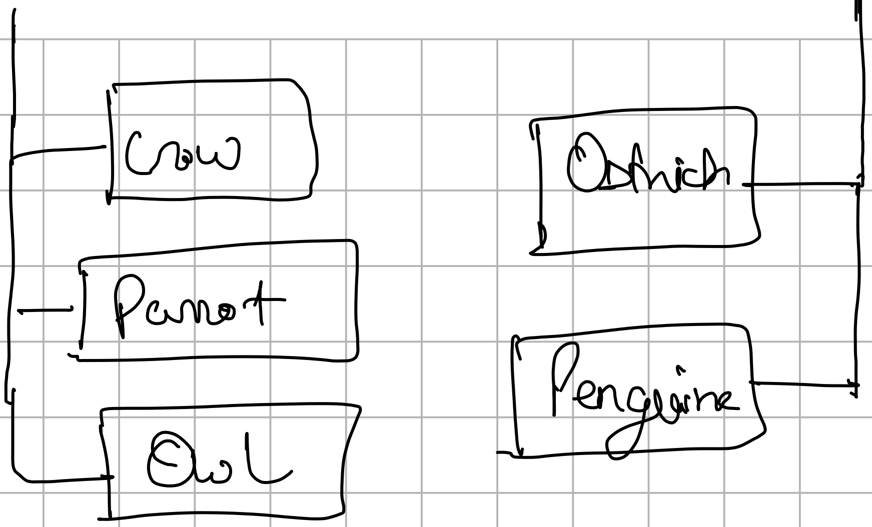
⇒ Easy to add features
⇒ But new features should not require much change to already existing code base

⇒ Instead add new classes/methods

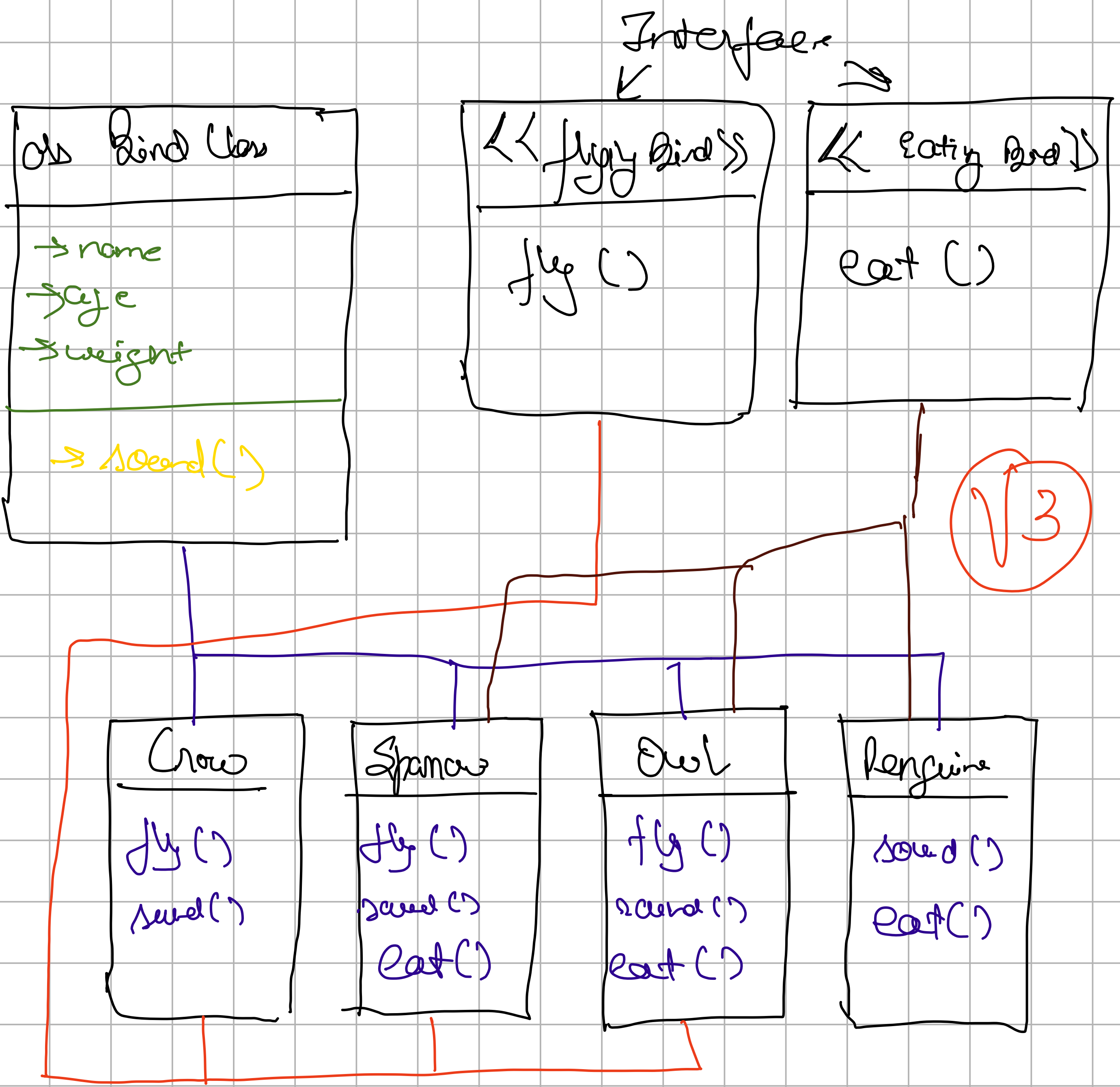


Problem

Class Explosion



(2) LSP → Liskov's Substitution Principal

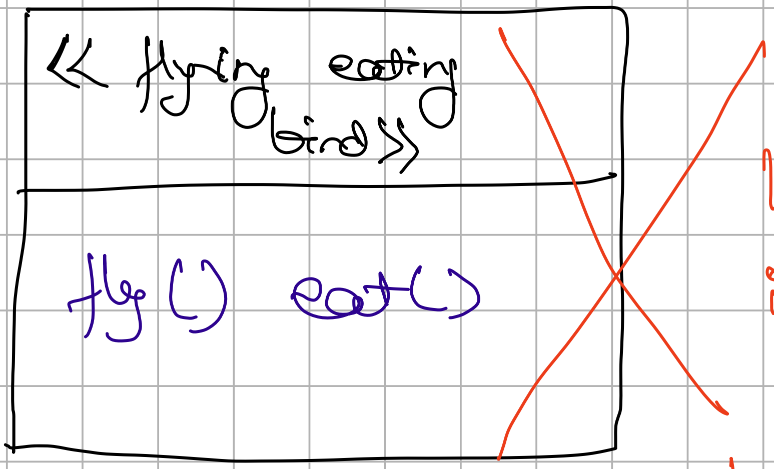


4 Interface Segregation Principle

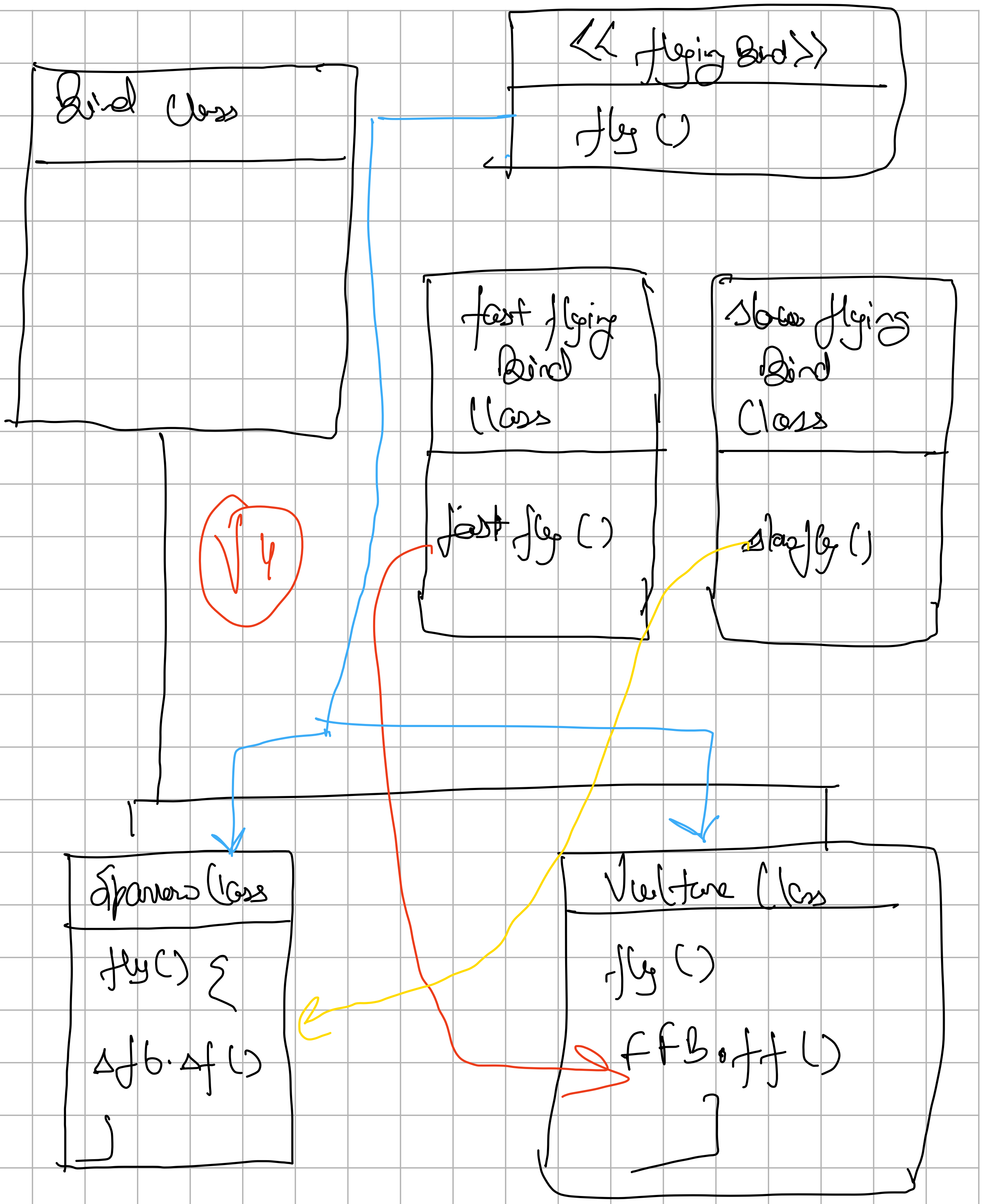
Interface should be as light as possible

It should have less no of methods, ideally only 1

Should have more than 1 method, if they are logically related to each other.



Wrong as it violates and have multiple not logically related methods.

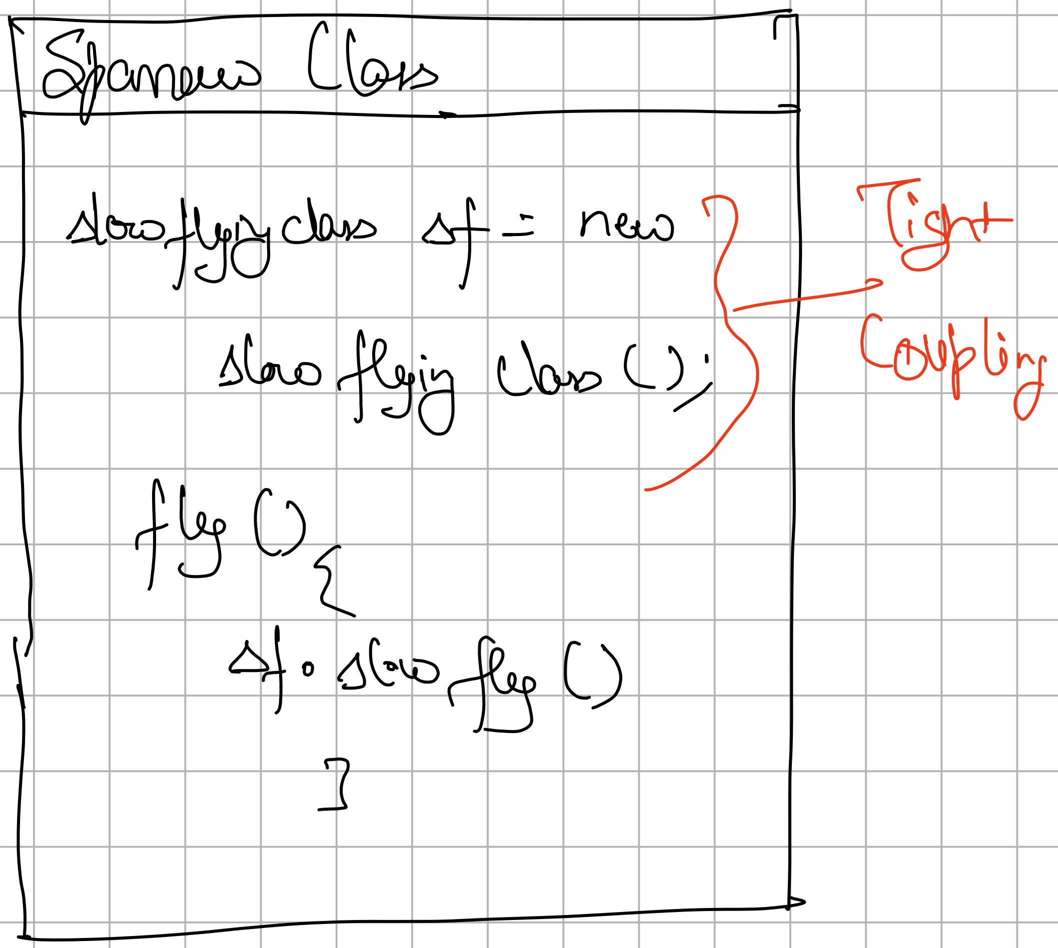

Bird Class
<< Flying Bird >>
fly ()
fast flying Bird Class
fast fly ()
slow flying Bird Class
slowfly ()
√4
Sparrow Class
fly() {
sfb.sf()
}
Vulture Class
fly ()
ffB.ff ()
}

5 Dependency Injection

No two non abstract class should idelly be dependent on each other directly

They should depend on each other via an interface.

ex Tight Coupling Wrong

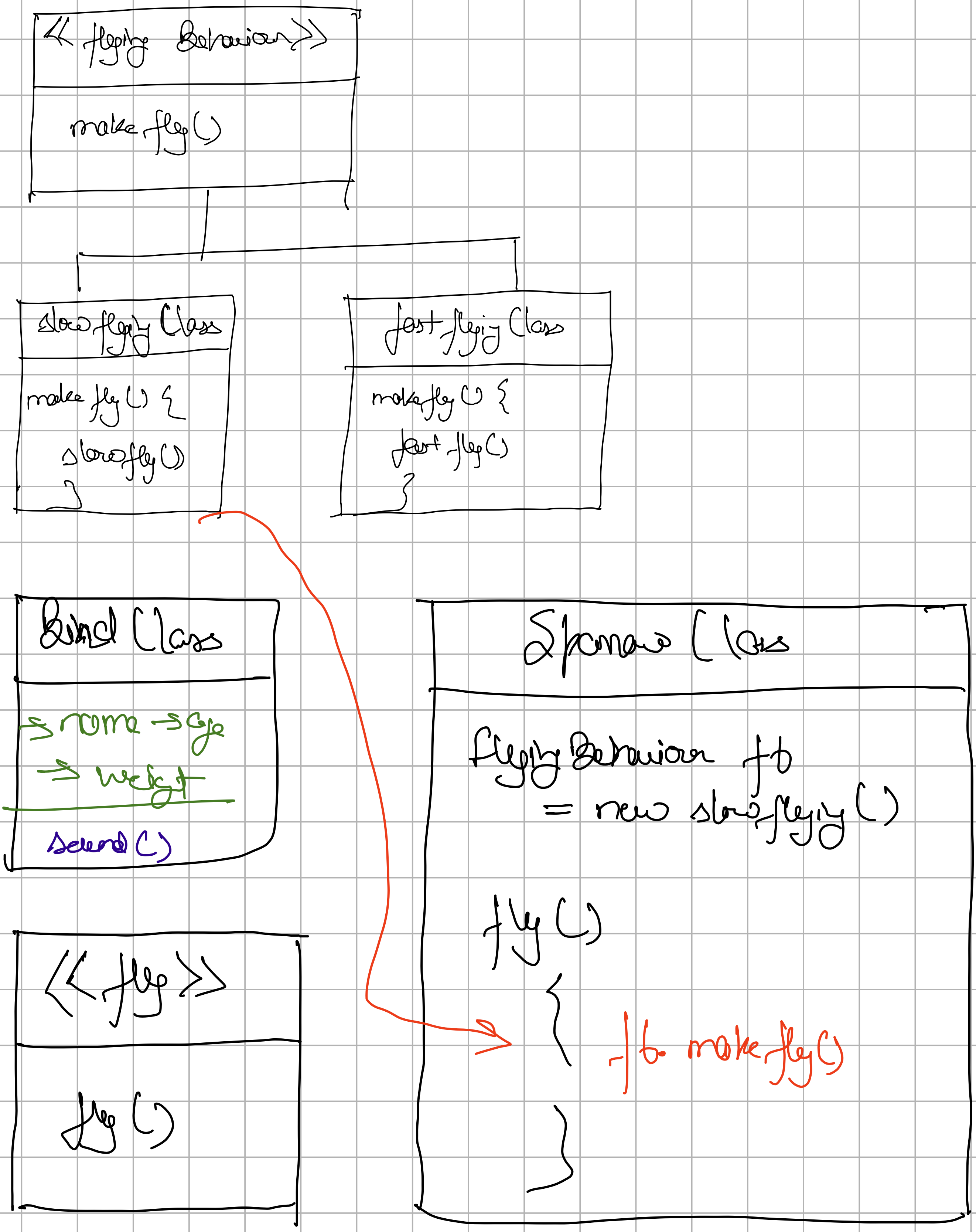


Sparrow Class

```
slowflyingclass sf = new
        slowflying class();

fly() {
    sf.slowfly()
}
```

Correct Way



<< flying Behavior >>
make fly()
slow flying Class
make fly() {
slow fly()
}
fast flying Class
make fly() {
fast fly()
}
Bird Class
-> name -> age
-> weight
sound()
<< fly >>
fly()
Sparrow Class
flying Behavior fb
= new slow flying()
fly()
{
fb. make fly()
}