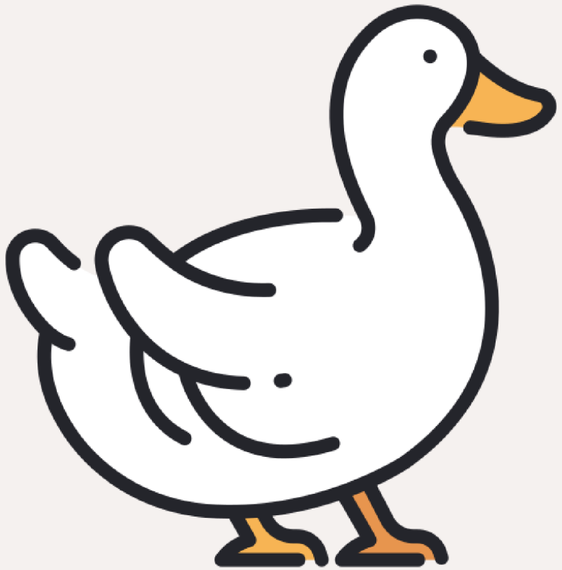
STRATEGY PATTERN

Head First Design Patterns - Eric and Elizabeth Freeman

Members:

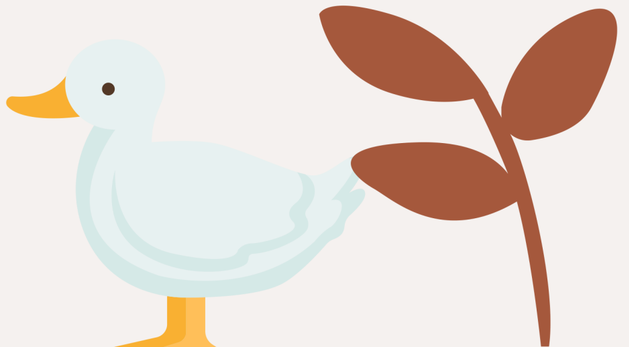
- Nguyen Dung
- Huynh Tan Khai



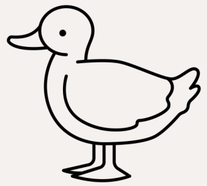
"Ducks are eternal!"

- Yves Saint Huynh

Table of Contents

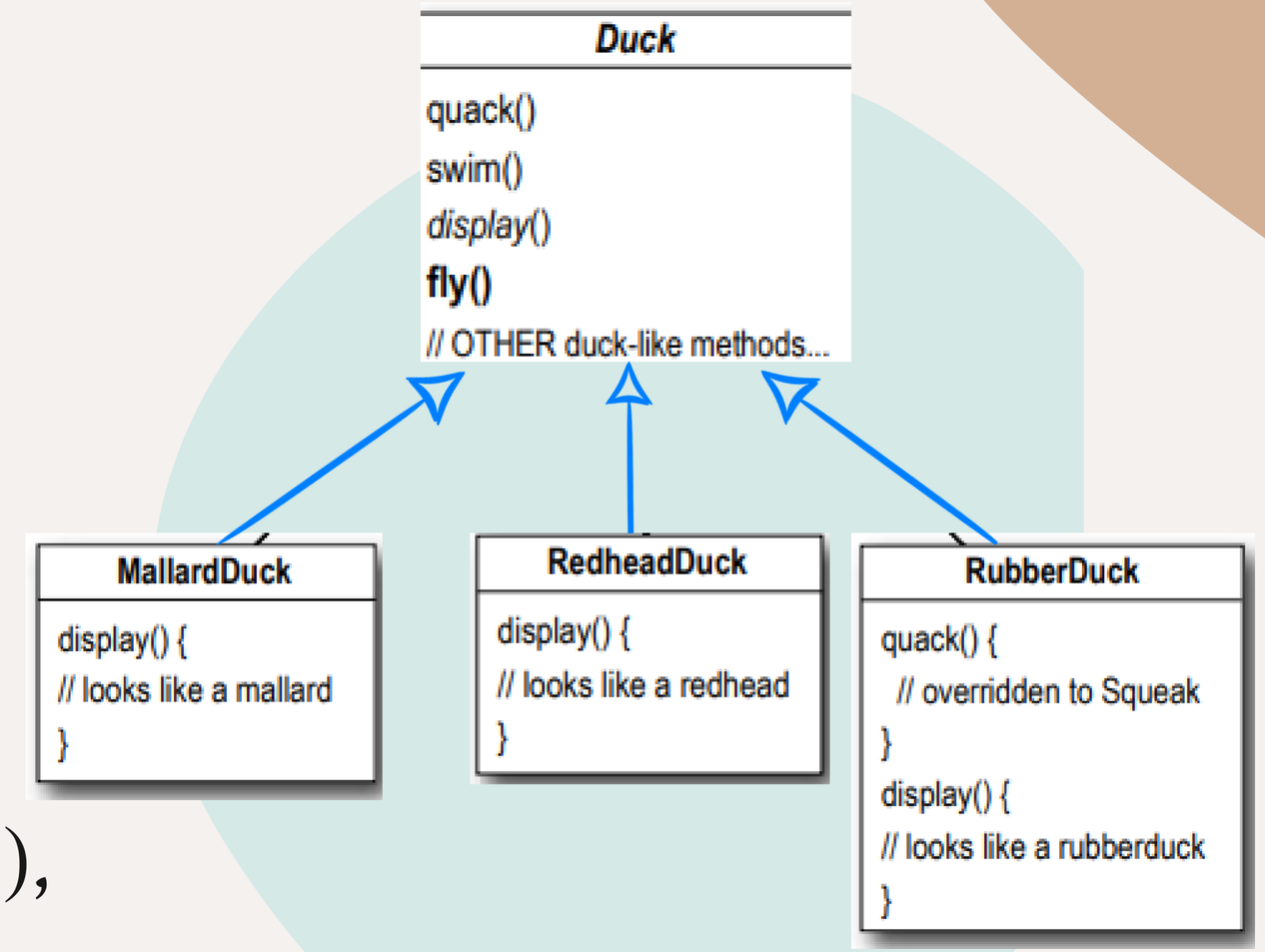


Problem



+ Building a simple SimUDuck app to simulate ducks with a variety of features

+ Class Duck needs quack(), fly(), and swim() functions



RubberDuck

```
quack()  { // squeak}
display() { // rubber duck }
fly() {
  // override to do nothing
}
```

DecoyDuck

```
quack() {
  // override to do nothing
}

display() { // decoy duck}

fly() {
  // override to do nothing
}
```

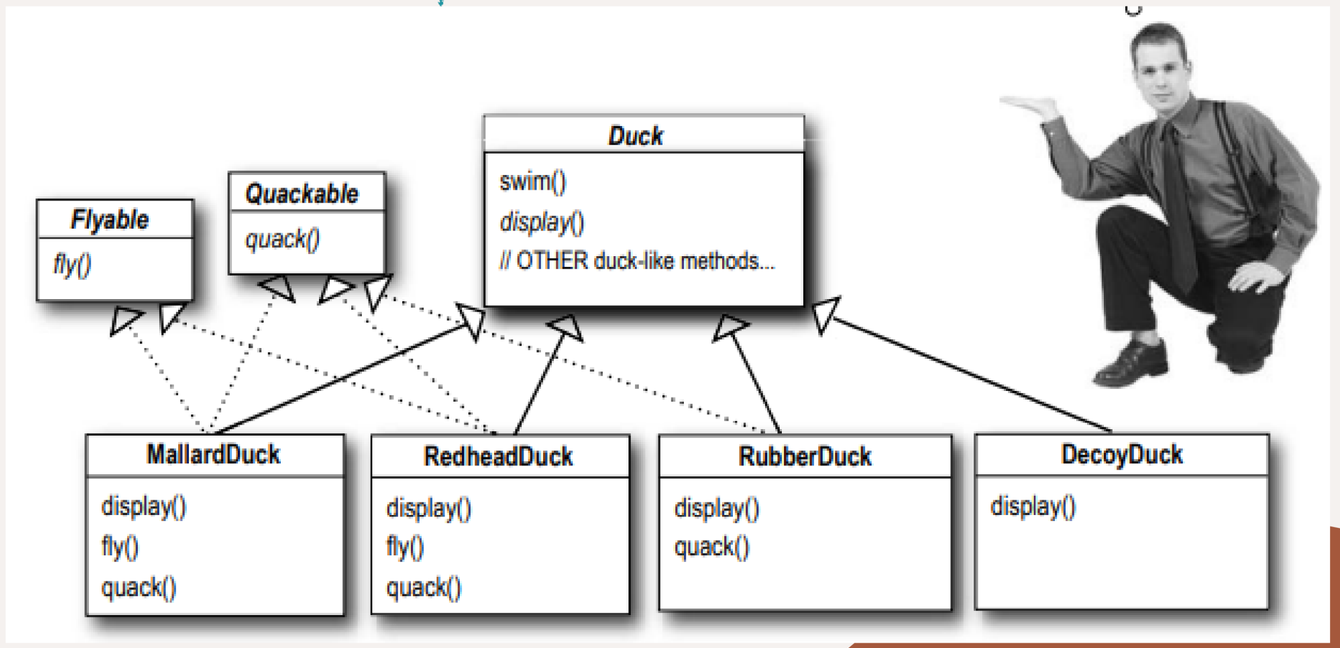
Why inheritance does not work?

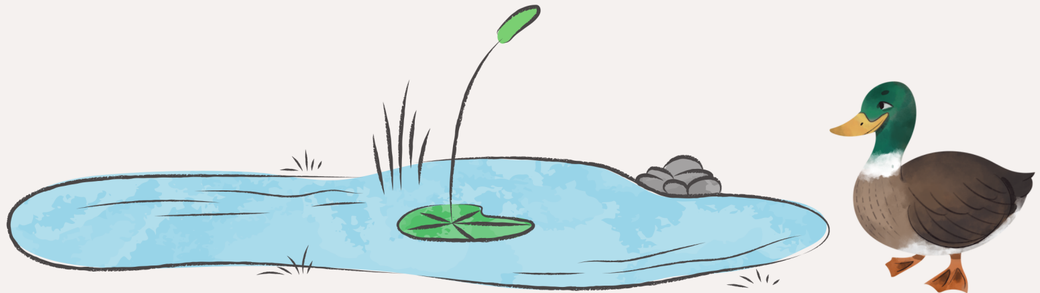
+ Wait! If we think again, not all ducks can quack and fly!

+ Even when we can always override them, the code has to be considerably modified every time a new duck is added.

Other Problem?

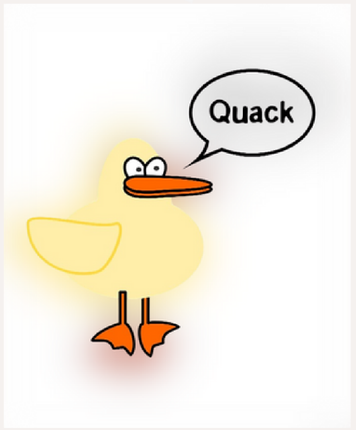
- Sometimes the interface can be a little bit messy .
- But what if the customers want to add new flying and quacking styles to the duck?





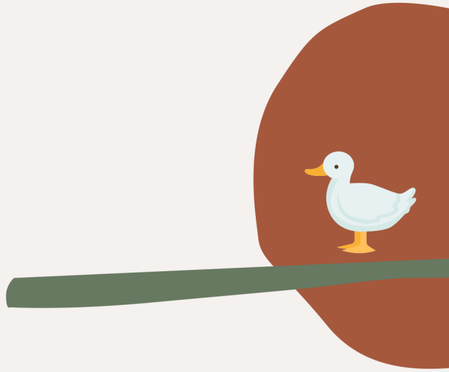
Strategy Pattern Design Principle

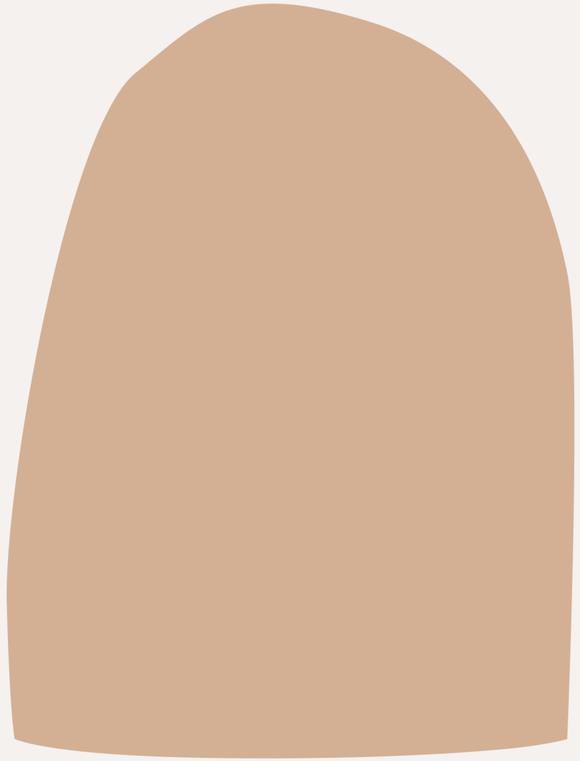
Identify the aspect of your application that vary and separate them from what stays the same.



Separating what changes and what stays.

Pull both fly() and quack() method out of the Duck class, and create a new set of classes to represent each behavior.





Demo

Thank you!