CSC207 - Design Patterns

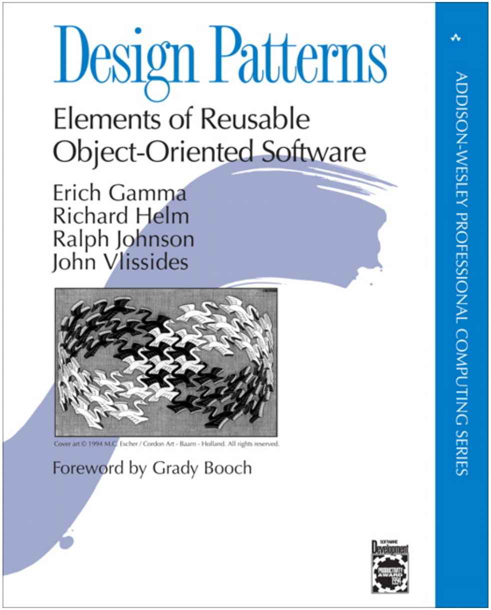
Ilir Dema

Summer 2016

Design Patterns

- A design pattern is a general description of the solution to a well-established problem using an arrangement of classes and objects.
- Patterns describe the shape of code rather than the details.
- Theyre a means of communicating design ideas.
- They are not specific to any one programming language.
- First codified by the Gang of Four in 1995
  - Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
- They are classified in four main groups:
  - Creational, Structural, Behavioral, Concurrency.

Design Patterns



Design Patterns

Elements of Reusable Object-Oriented Software

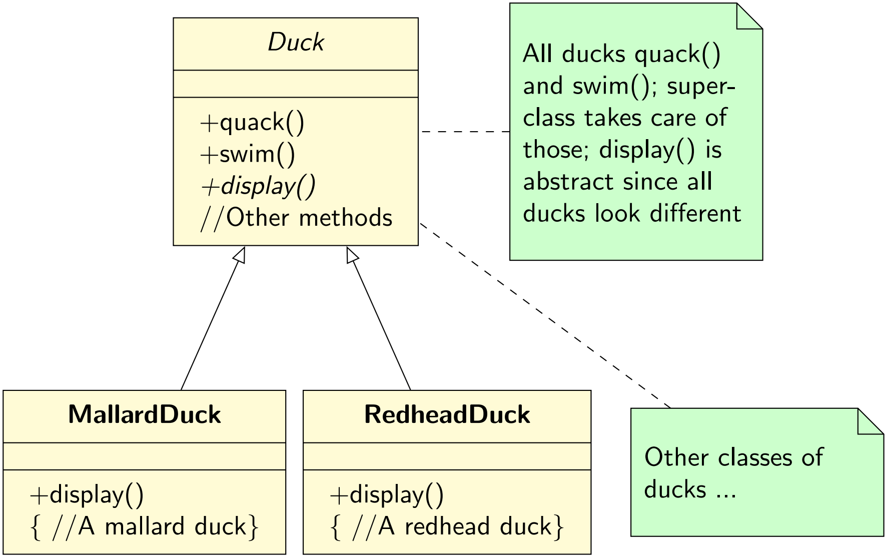
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

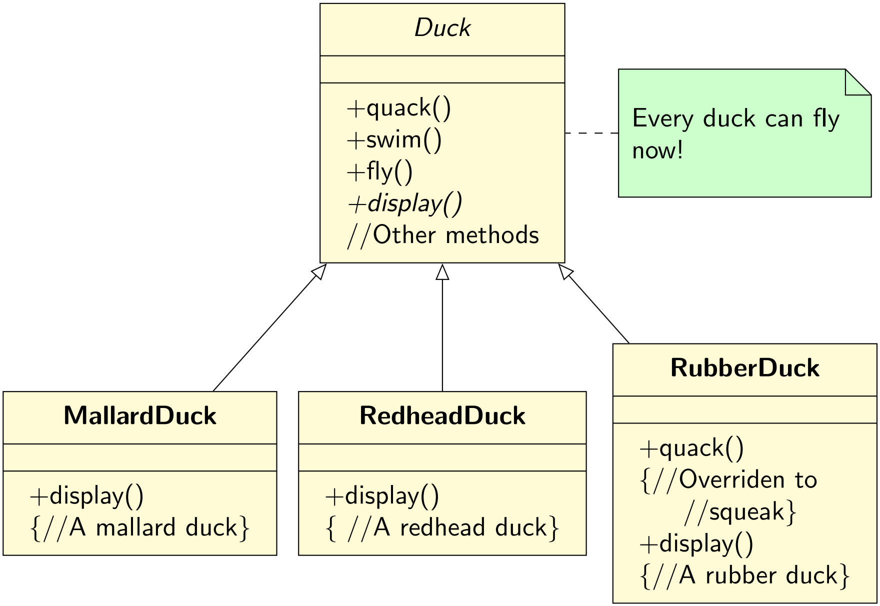
Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Who do we need patterns?


Duck
+quack()
+swim()
+display()
//Other methods
All ducks quack() and swim(); super-class takes care of those; display() is abstract since all ducks look different
MallardDuck
+display()
{ //A mallard duck}
RedheadDuck
+display()
{ //A redhead duck}
Other classes of ducks ...

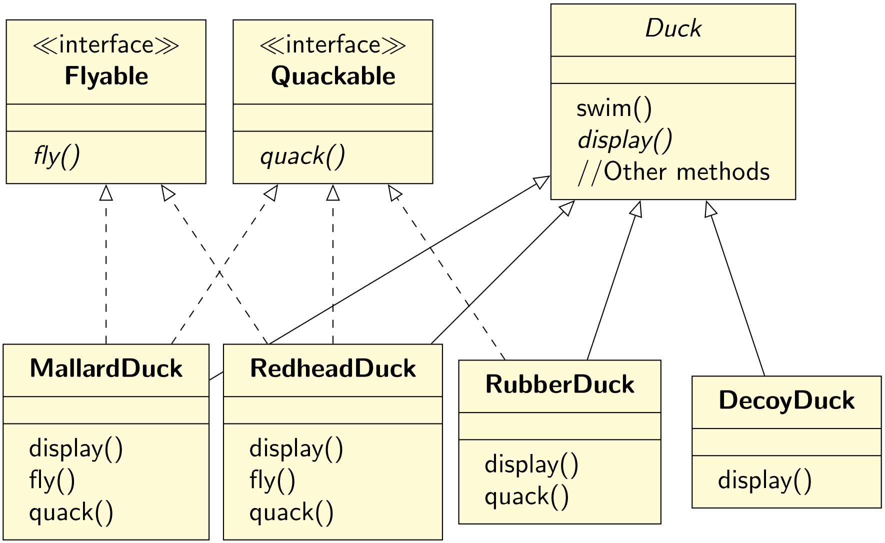
A quick change is needed - ducks need to fly!



An attempt to solve the problem

- Override the `fly()` method in `RubberDuck` same way we have already done with `quack()`.
- But what happens if we add a `DecoyDuck` made of wood? They are not supposed to fly or quack!
- Which of the folowing are disadvantages of using inheritance to provide specific Duck behavior? (Chose all that apply).
  a. Code is duplicated across subclasses.
  b. Runtime behavior changes are difficult.
  c. We can't make a duck dance.
  d. Hard to gain knowledge of all duck behaviors.
  e. Ducks can't fly and quack at the same time.
  f. Changes can unintentionally affect other ducks.

How about interfaces?



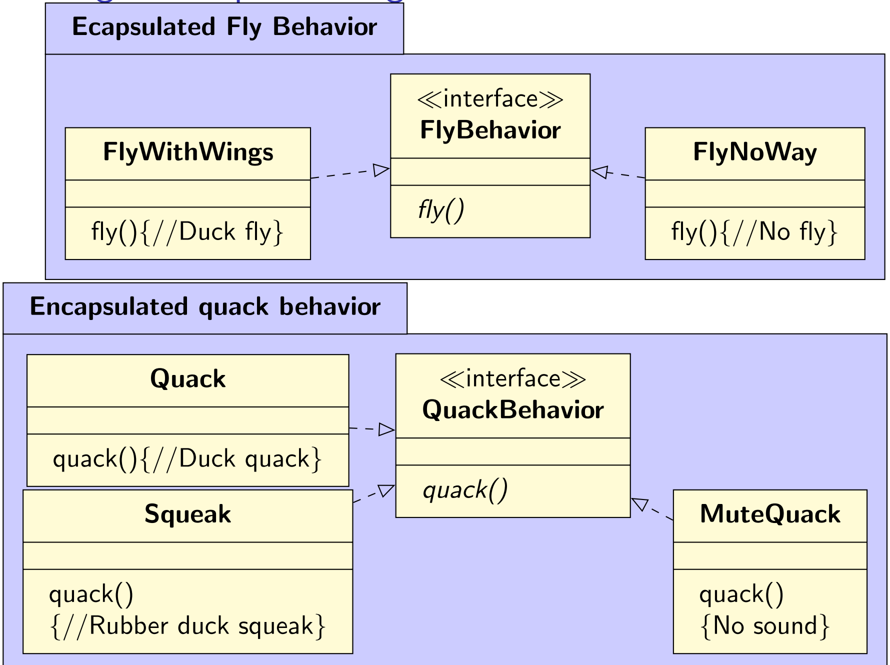
The solution

- The use of interfaces solves part of the problem (no flying rubber ducks), it completely destroys reuse of code, creating a huge maintenace problem.
- This can be addressed by using a **design principle**:
  - Identify the aspects of your application that vary and spearate them from what stays the same.
  - Or, take the parts that vary and encapsulate them so later you can modifyor extend them without affecting those that don't.
- Let's do it:
  - The parts of `Duck` that vary are `fly()` and `quack()`.
  - Pull them out of `Duck` class and create a new set of classes to represent each behavior.

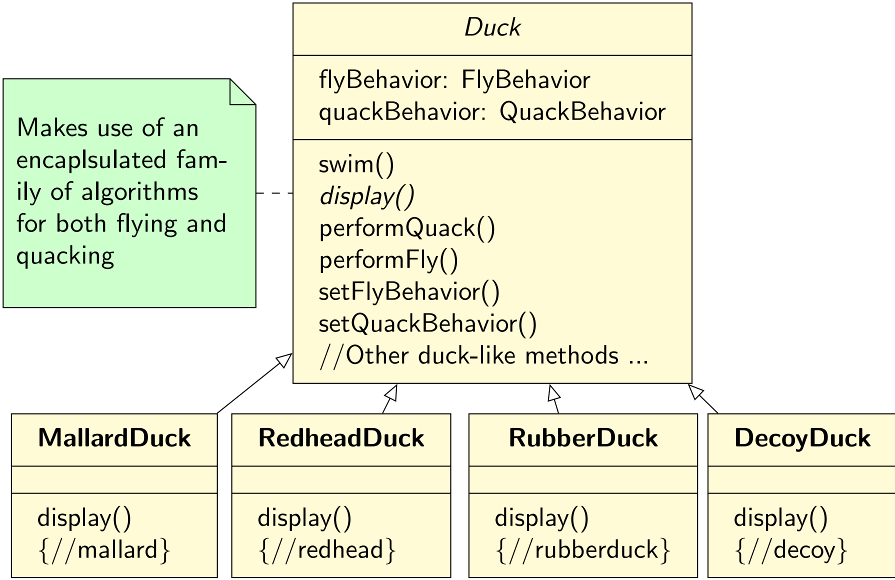
More design principles

- Meanwhile we need to keep things flexible, so we might want to assign a fly behavior to a `MallardDuck` at instantiation time.
- To successfully do so, we need to **program to an interface, not to an implementation**.
- We also need to **favour the composition over inheritance**.

Separating and implementing duck behaviors



Duck **has a** `FlyBehavior` and a `QuackBehavior`



The strategy pattern

- **The strategy Pattern** defines a family of algorithms, encaplsulates each one, and makes them interchangeable.
- Strategy lets the algorithm vary independently from clients that use it.
- For an example, see the solution of the duck problem.
- Exercise:
  - A duck call is a device that hunters use to mimic the calls (quacks) of ducks. How would you implement your own duck call that does not inherit from the Duck class?

Where are we so far?

- OO Basics
  - Abstraction
  - Encapsulation
  - Polymorphism
  - Inheritance
- OO Design Principles
  - Encapsulate what varies
  - Favor composition over inheritance
  - Program to interfaces, no implementations.
- OO Design patterns
  - We did see strategy pattern
  - We will see a few more in the near future.

Observer Design Pattern

- Problem:
  - Need to maintain consistency between related objects.
  - Two aspects, one dependent on the other.
  - An object should be able to notify other objects without making assumptions about who these objects are.
- Example - Magazine subscription:
  - A magazine publisher goes into business and stars publishing magazines
  - You subscribe to the publisher - every time there is a new issue, it gets delievred to you, fors as long as you remain a subscriber.
  - You unsubsribe when you do not want the magazine anymore.
  - Meanwhile the publisher remains in business, other people keep subscribing and ubsubscribing to its magazine.

CERN open days event

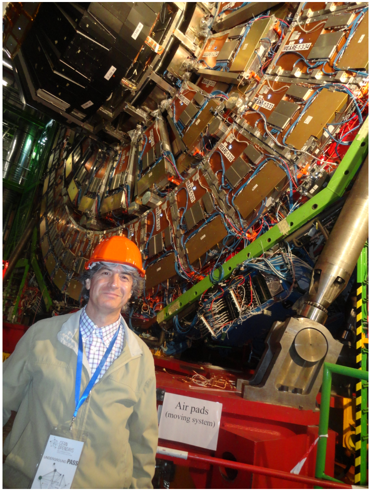
CERN opens its doors September 28th - 29th 2013

15/08/2013

Reservations to visit the underground facilities are now open. You have the opportunity to visit one of the LHC experiments (ALICE, ATLAS, CMS or LHCb), the tunnel of the LHC machine (Radio Frequency or Accelerator Systems) or the SPS accelerator. Tickets will be progressively made available over a period of four weeks.

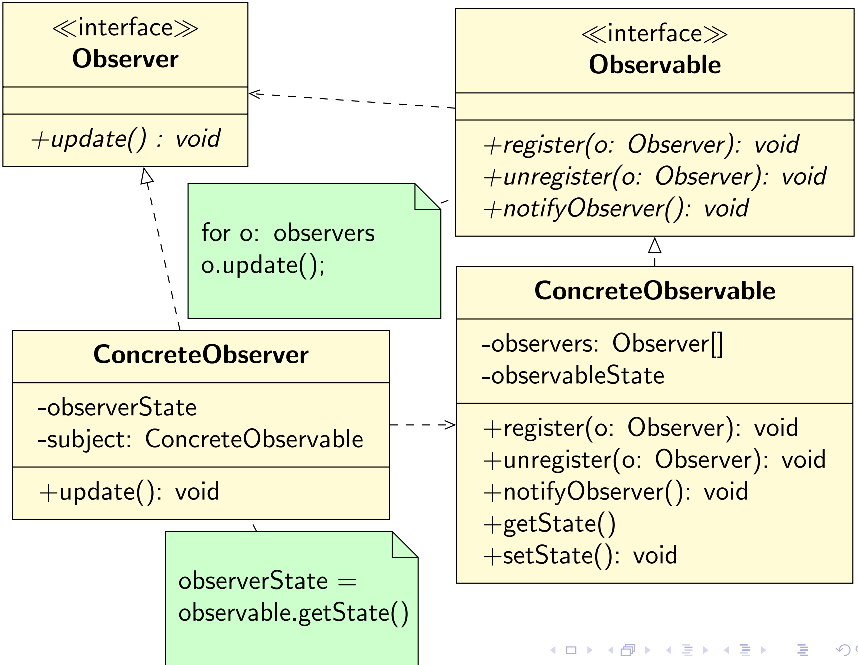
Source: `http://opendays2013.web.cern.ch/?page=5`

How do I get a ticket?

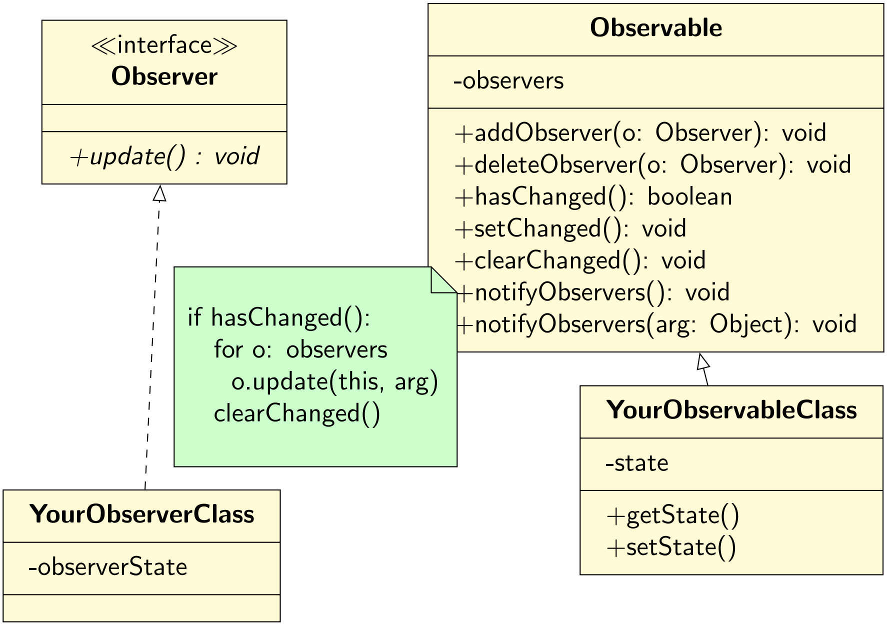


Well, I did get it the old fashion way - clicking at their web site at 2am (am European Time).

Observer Pattern UML



Observer: Java implementation



```
if hasChanged():
    for o: observers
      o.update(this, arg)
    clearChanged()
```