POS/409 – C# PROGRAMMING

Better Programming
Professor: Troy Tuckett

PROBLEMS WITH UI-FIRST PROGRAMMING

- Poor programming decisions
  - Discourages object-first coding
  - Breaks MVC

BETTER PROGRAMMING

- A little about UML
- Using the simplest approach
- Refactoring
- Design Patterns

BETTER PROGRAMMING

- Architectural
  - Frameworks
- Design
  - Patterns
- Cross-cutting concerns
  - Exceptions
  - Logging

LITTLE ABOUT UML

Class (instance variables and methods)

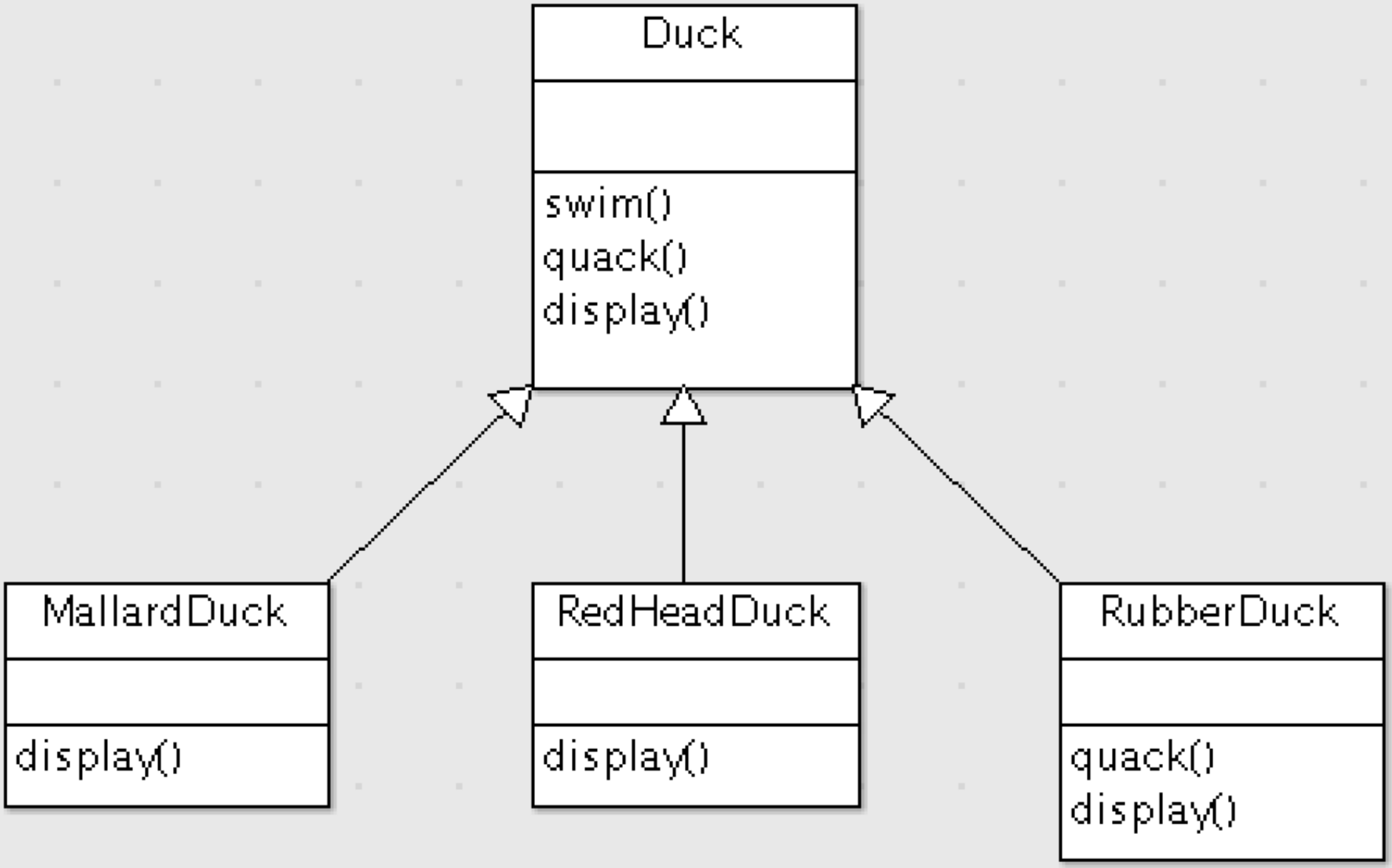
- Solid Lines for Inheritance

Interfaces

- Dotted lines for Implementation

DESIGN PATTERN EXAMPLE

- Duck Simulator


Duck
swim()
quack()
display()
MallardDuck
display()
RedHeadDuck
display()
RubberDuck
quack()
display()

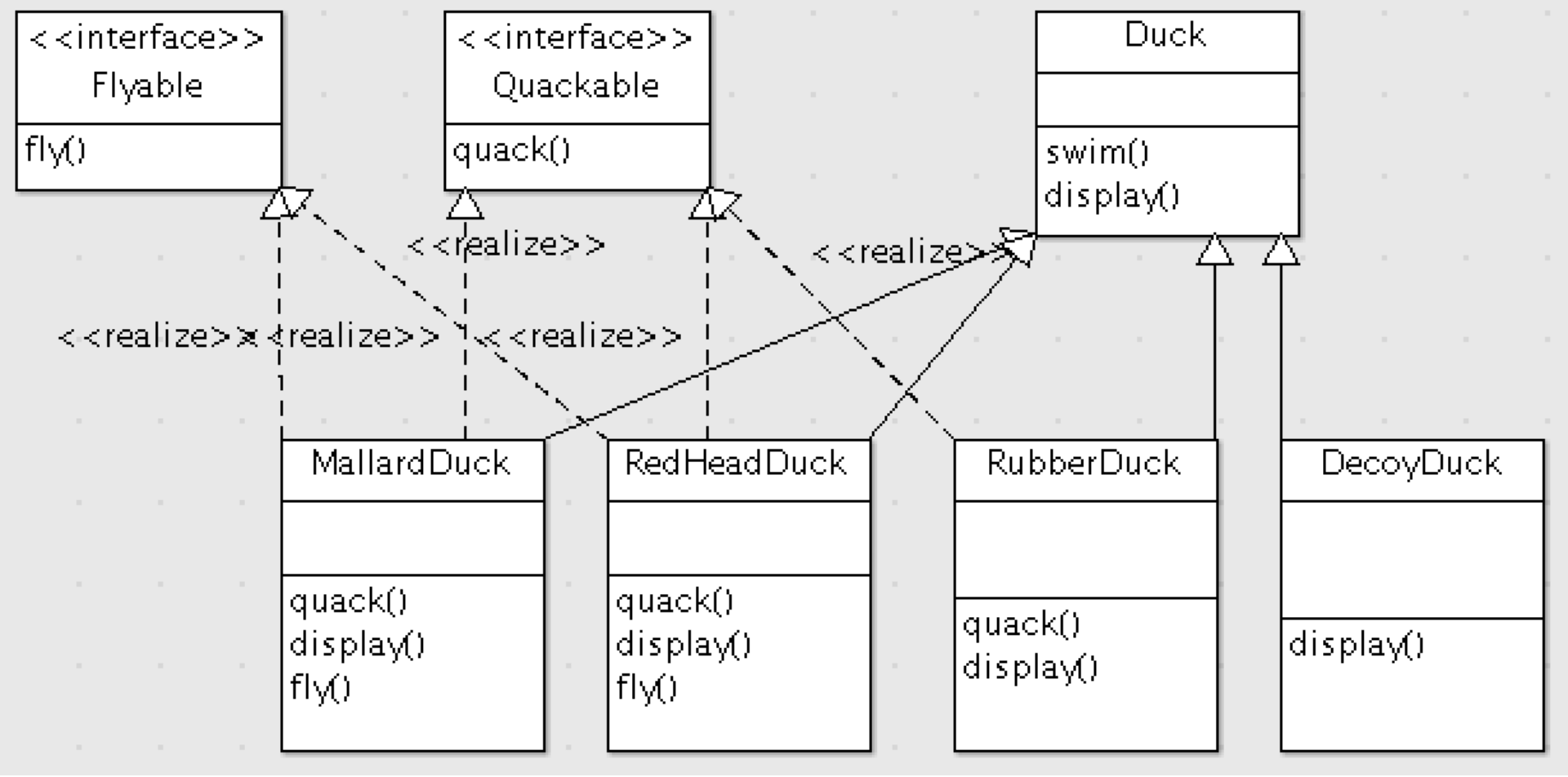
DUCKS

What about fly()?

How does this design work with RubberDuck and DecoyDuck?

PROBLEMS WITH INHERITANCE

- Code is Brittle
- Side effects
- Breaks Encapsulation
- Poor maintainability


<<interface>>
Flyable
fly()
<<interface>>
Quackable
quack()
Duck
swim()
display()
<<realize>>
<<realize>>
<<realize>>
<<realize>>
<<realize>>
MallardDuck
quack()
display()
fly()
RedHeadDuck
quack()
display()
fly()
RubberDuck
quack()
display()
DecoyDuck
display()

PROBLEMS WITH INTERFACES

- No implementation
- No code, so no reuse

DESIGN PRINCIPLE

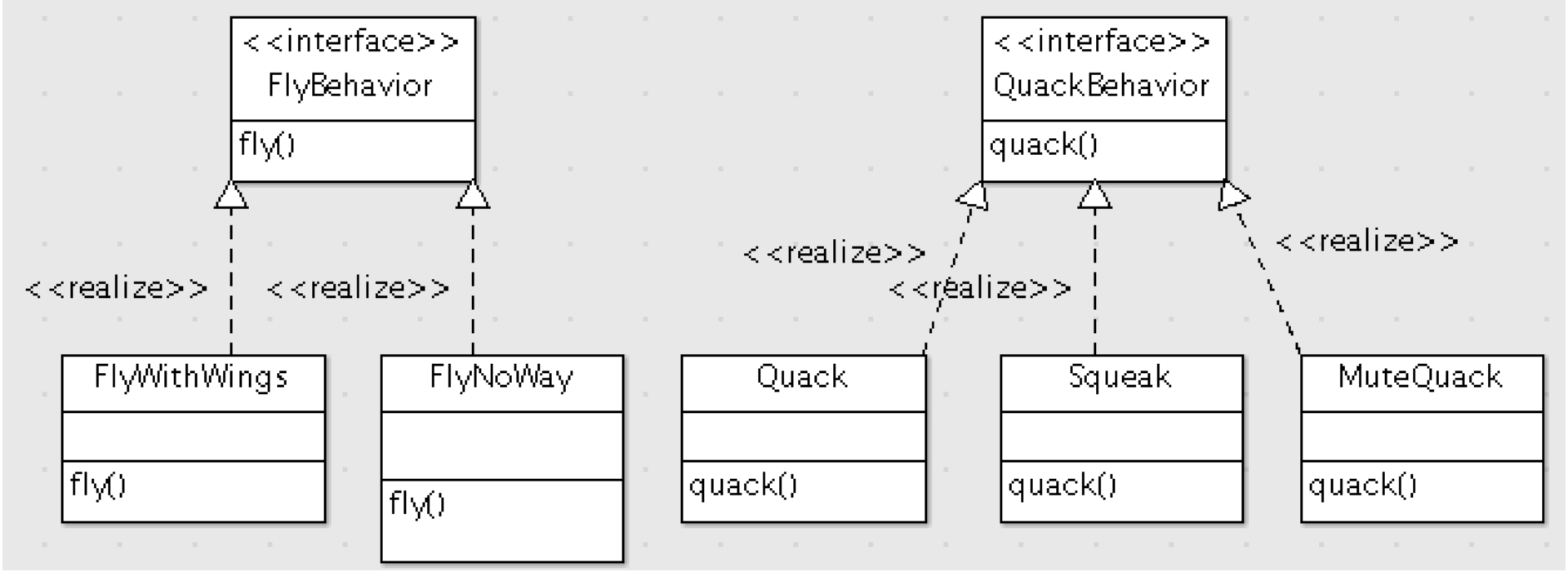
- Separate those things that change often from those that don't

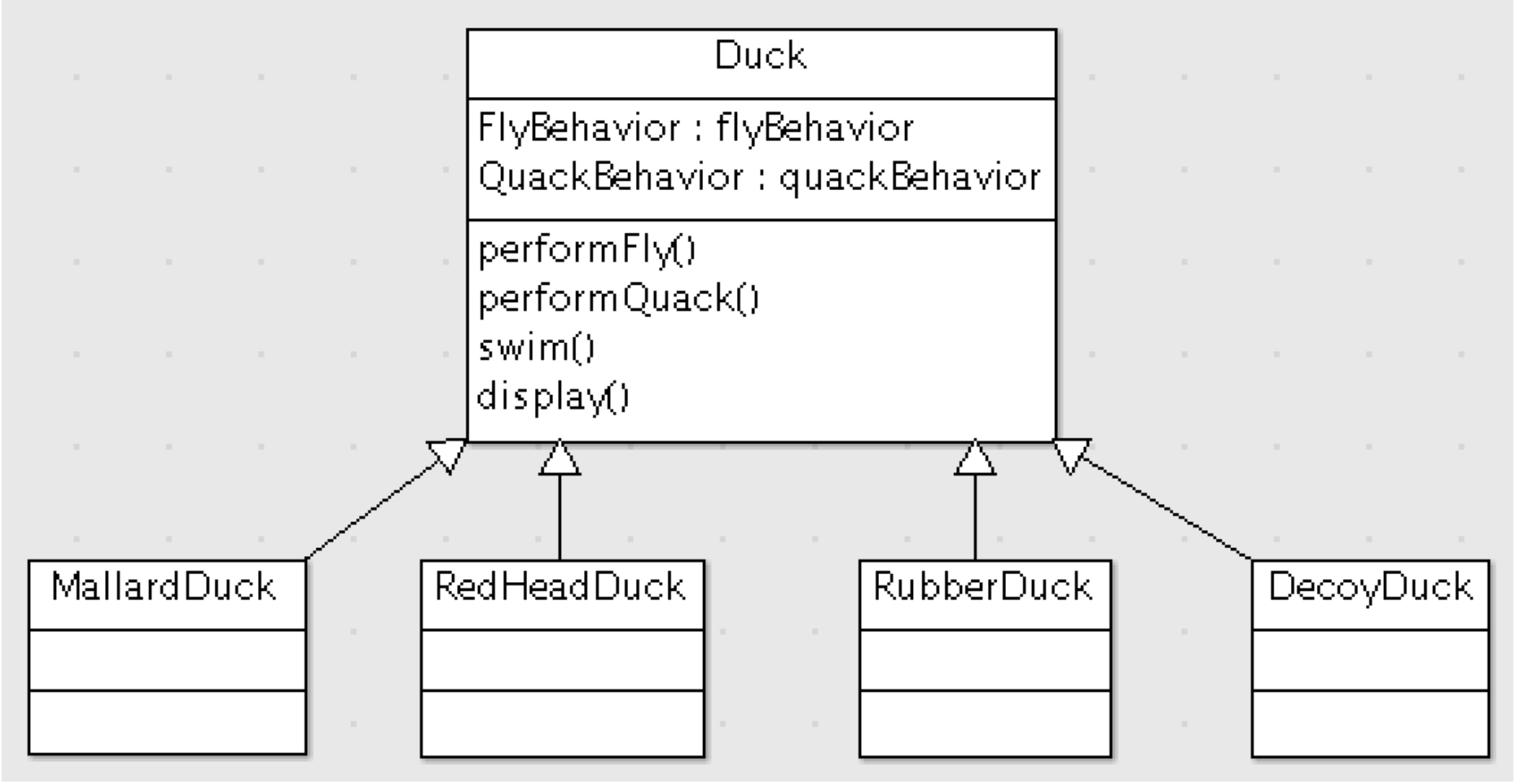
DUCKS

- Put duck behaviors in separate classes
- Use an interface for each behavior instead of hard implementations (in superclass or specialized impl)
- We can use polymorphism to determine behavior dynamically.

DESIGN PRINCIPLE

- Code to an interface instead of an implementation


<<interface>>
FlyBehavior
fly()
<<interface>>
QuackBehavior
quack()
<<realize>>
<<realize>>
<<realize>>
<<realize>>
<<realize>>
FlyWithWings
fly()
FlyNoWay
fly()
Quack
quack()
Squeak
quack()
MuteQuack
quack()


Duck
FlyBehavior : flyBehavior
QuackBehavior : quackBehavior
performFly()
performQuack()
swim()
display()
MallardDuck
RedHeadDuck
RubberDuck
DecoyDuck

DESIGN PRINCIPLE

- Prefer Composition over Inheritance

Constructor of each subclass

```
public MallardDuck()
    quackBehavior - new Quack();
    flyBehavior = new FlyWithWings()
```

JUST COMPLETED STRATEGY PATTERN