Strategy Pattern:

Strategy Pattern defines a family of algorithms, encapsulates each state, and makes

them interchangeable. Strategy lets the algorithms vary independently from

clients that use it.

Duck1:

When implemented without patterns

Duck2:

Interface classes for QuackBehavior and FlyBehavior. Seperate the behaviors that

change.

Duck3:

Flybehavior and QuackBehavior packages to implement the behaviors

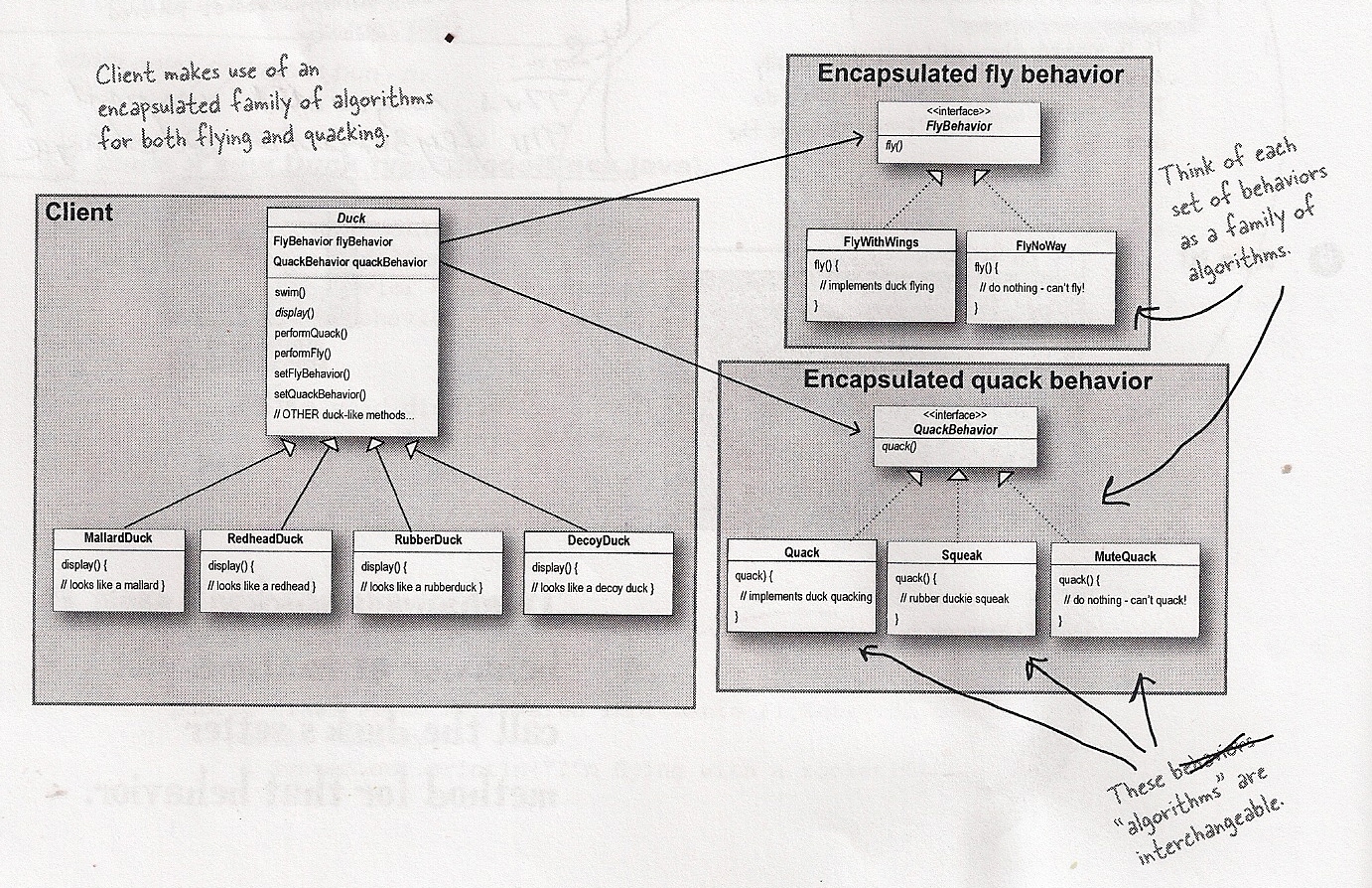
Duck 4:

Implement fly and quack behaviors dynamically.Two new methods, setFlyBehavior and

setQuackBehavior added to the Duck class.

- Create a new Duck type called ModelDuck

- Create a new FlyBehaviro type called FlyRocketPowered



How is it done:

* flybehavior package implements the FlyBehavior interface. All types of fly behavior classes implement the FlyBehavior interface and need to implement the fly() method.
* quackbehavior package implements QuackBehavior interface. All types of quack behavior classes implement the QuackBehavior interface and need to implement the quack() method.
* The abstract Duck class has Flybehavior and QuackBehavior variables. Any class inheriting from the Duck class has to define its own FlyBehavior and QuackBehavior.