

Client makes use of an encapsulated family of algorithms for both flying and quacking.

Client

```
class Duck {
    FlyBehavior flyBehavior
    QuackBehavior quackBehavior

    swim()
    display()
    performQuack()
    performFly()
    setFlyBehavior()
    setQuackBehavior()
    // OTHER duck-like methods...
}
```

MallardDuck

```
display() {
    // looks like a mallard
}
```

RedheadDuck

```
display() {
    // looks like a redhead
}
```

RubberDuck

```
display() {
    // looks like a rubberduck
}
```

DecoyDuck

```
display() {
    // looks like a decoy duck
}
```

Encapsulated fly behavior

```
<<interface>>
FlyBehavior
fly()
```

FlyWithWings

```
fly() {
    // implements duck flying
}
```

FlyNoWay

```
fly() {
    // do nothing - can't fly!
}
```

Think of each set of behaviors as a family of algorithms.

Encapsulated quack behavior

```
<<interface>>
QuackBehavior
quack()
```

Quack

```
quack() {
    // implements duck quacking
}
```

Squeak

```
quack() {
    // rubber duckie squeak
}
```

MuteQuack

```
quack() {
    // do nothing - can't quack!
}
```

These behaviors "algorithms" are interchangeable.