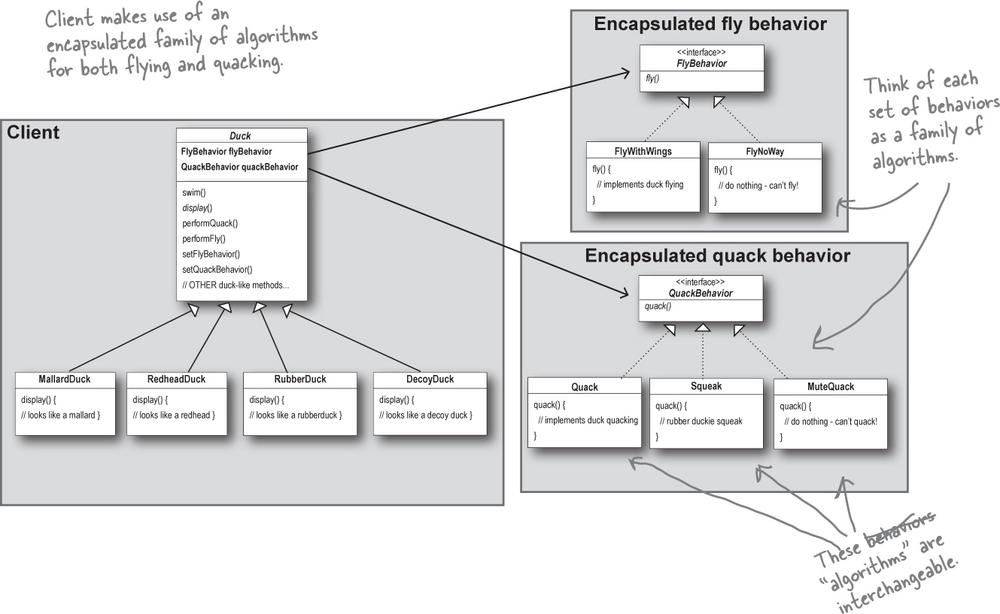
**Strategy Pattern**

In Strategy pattern, a class behaviour or its algorithm can be changed at run time.

This pattern encapsulates the behaviours that can be changed at runtime!



|  |
| --- |
| NOTE |
| We are using a Strategy Pattern to implement the variable behaviours of ourduck  🡪 This tells us the duck behaviour has been encapsulated into its own set of classes that can be easily expanded and changed, even at runtime if needed. |

# Has-A Relationship : Composition

* Each duck has a FlyBehaviour and a QuackBehaviour
* When we put two classes together in this way, we are using Composition.
* Instead of inheriting their behaviour, the duck gets their behaviour by being composed with the right behaviour object.
* Favour composition over inheritance!
* Composition gives more flexibility. Not only does it let us encapsulate a family of algorithms into our own set of classes, but it also lets us change behaviour at runtime