

Separating what changes from what stays the same

Where do we start? As far as we can tell, other than the problems with `fly()` and `quack()`, the Duck class is working well and there are no other parts of it that appear to vary or change frequently. So, other than a few slight changes, we're going to pretty much leave the Duck class alone.

Now, to separate the “parts that change from those that stay the same,” we are going to create two *sets* of classes (totally apart from Duck), one for *fly* and one for *quack*. Each set of classes will hold all the implementations of the respective behavior. For instance, we might have *one* class that implements *quacking*, *another* that implements *squeaking*, and *another* that implements *silence*.

We know that `fly()` and `quack()` are the parts of the Duck class that vary across ducks.

To separate these behaviors from the Duck class, we'll pull both methods out of the Duck class and create a new set of classes to represent each behavior.

The Duck class is still the superclass of all ducks, but we are pulling out the fly and quack behaviors and putting them into another class structure.

Now flying and quacking each get their own set of classes.

Various behavior implementations are going to live here.

