

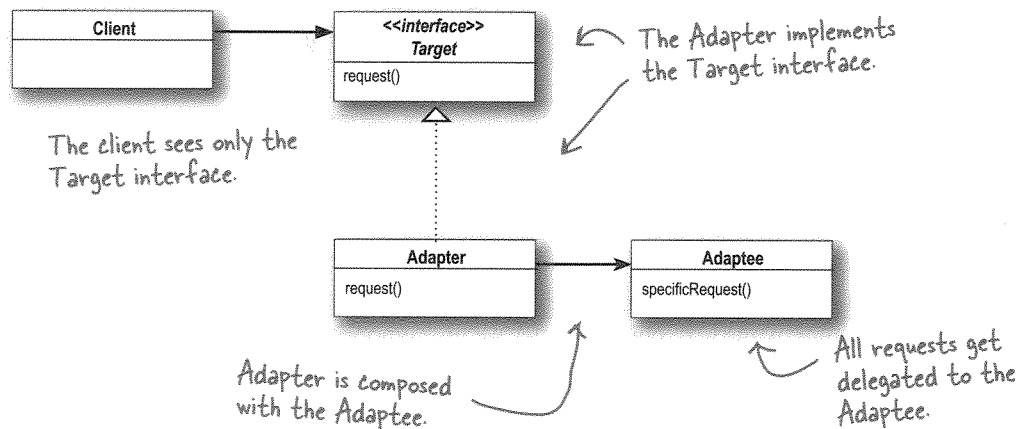
## Adapter Pattern defined

Enough ducks, turkeys and AC power adapters; let's get real and look at the official definition of the Adapter Pattern:

**The Adapter Pattern** converts the interface of a class into another interface the clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.

Now, we know this pattern allows us to use a client with an incompatible interface by creating an Adapter that does the conversion. This acts to decouple the client from the implemented interface, and if we expect the interface to change over time, the adapter encapsulates that change so that the client doesn't have to be modified each time it needs to operate against a different interface.

We've taken a look at the runtime behavior of the pattern; let's take a look at its class diagram as well:



The Adapter Pattern is full of good OO design principles: check out the use of object composition to wrap the adaptee with an altered interface. This approach has the added advantage that we can use an adapter with any subclass of the adaptee.

Also check out how the pattern binds the client to an interface, not an implementation; we could use several adapters, each converting a different backend set of classes. Or, we could add new implementations after the fact, as long as they adhere to the Target interface.