ADAPTER DESIGN PATTERN (WRAPPER)

We talked about the Facade design pattern yesterday which provides a wrapper over a sub-system (or a set of functionality.

Adapter design pattern also provides a wrapper over the functionality of a class, the difference is that Adapter convert the interface of a class into another interface which clients expect.

Adapter, the structural design pattern, lets classes work together that could not otherwise because of incompatible interfaces. Hence it is an absolute wrapper over a class (and not a wrapper over a functionality as Facade pattern).

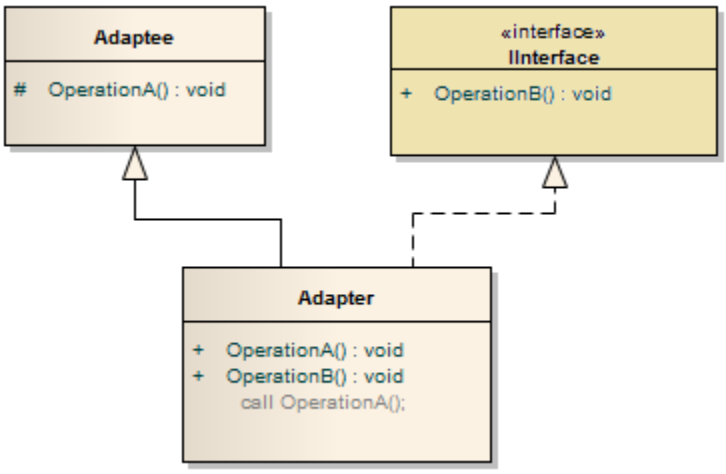
This of a real life example from where the name of this pattern has come. Suppose you are from America. You power adapters will be like

(with rectangular pins, unlike the round pin adapters in India). You come to India, and all the plugs have round pins like shown below.

This means that the interface of library is not compatible with the way our code can call it. Hence, we need an adapter.

Pictures say a lot more than words, I hope the concept of Adapter is clear. Now lets get back to computer science:

Adapter is about creating an intermediary abstraction that translates, or maps, the old component to the new system. Clients call methods on the Adapter object which redirects them into calls to the legacy component.



Please note the difference between Adapter design pattern and Facade design pattern which is a wrapper over the functionality, i.e a subsystem and not just a wrapper to change the interface.

Source: http://www.ritambhara.in/adapter-design-pattern-wrapper/