Here’s how the client uses the adapter,

1. The client makes a request to the adapter by calling a method on it using the target interface.
2. The adapter translates the request into one or more calls on the adaptee using the adaptee interface.
3. The client receives the results of the call and never knows there is an adapter doing the translation.

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

***Types of Adapter:***

***Object Adapter:***

<<interface>>Target

request()

Client

Adaptee

Specific request()

Adapter

request()

***Class Adapter: (***It is not possible in java, because of multiple inheritance)

Adapter

Request()

Adaptee

specificRewuest()

Target

Request()

Client

* When you need to use an existing class and its interface is not the one you need, use an adapter.
* An adapter changes an interface into one a client expects.
* Implementing an adapter may require little work or a great deal of work depending on the size and complexity of the target interface.
* There are two forms of the Adapter Pattern: object and class adapters. Object adapters require composition. Class adapters require multiple inheritance.
* An adapter wraps an object to change its interface, a decorator wraps an object to add new behaviors and responsibilities, and a façade wraps a set of objects to simplify.