**Adapter Design Pattern**

Adapter design pattern is used to provide a way for reusing an existing class.

**Intent**

* Convert the interface of a class into another interface clients expect. Adapter lets classes work together that could not otherwise because of incompatible interfaces.

The intent is to convert the interface of existing class to the one which client is expecting.

**Problem**

* Want to use existing system/component but the current system used by client does not have interfaces that are compatible with existing one.

The client has the interface that he wants to use, we have an existing class which can be reused but it has different interface. So either a new class is required or we have to find a way to use the existing one.

**Solution**

* Come up with a class which adapts the interface of existing system/component to the new one which client expects.

The system will adapt the existing interface and will provide it in the way client expects.

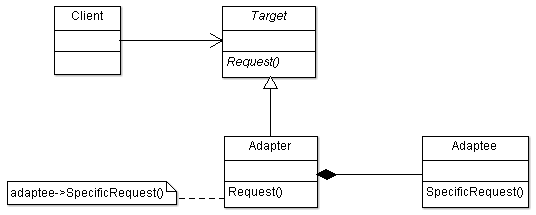
**Where it is applicable?**

* There is an existing class which we want to use in a system and may require to provide interface in the way applicable for the system. Or we can say we want to reuse the class but we don't have compatible interfaces.

**Structure**

**Composition**

Composition structure also called as object adapter.



**Participant Classes**

* Target class provides interfaces to clients and has abstract method Request().
* Adapter class is derived from Target class and implements the method Request().
* Adaptee has the functionality we want to reuse.

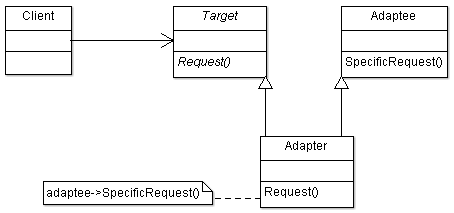
**How they work together?**

* The Adapter class has instance of Adaptee and it adapts the methods of Adaptee class. The method Request() of Adapter class in turn uses the method SpecificRequest() of Adaptee. The client is required to use the abstract method Request() only.
* Suppose we want to use another method which is not available in Adaptee then we can just provide that method in Adapter class itself.
* We can also have multiple adaptees and adapter can have their instance and adapts the functionality it requires.

**Structure**

**Inheritance**

Inheritance structure is also called as class adapter.



**Participant Classes**

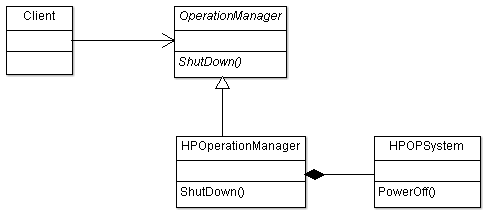
* Target class provides interfaces to clients and has abstract method Request().
* Adapter class is derived from Target and Adaptee. It implements the method Request() and inherits the functionality of Adaptee.
* Adaptee has the functionality we want to reuse.

**How they work together?**

* Adapter class is derived from Adaptee, so that it can adapt the Adaptee methods. The method SpecificRequest() is called in method Request() of class Adapter which is provided as interface to client.

**Example**

We want to reuse already available class for HP System operations.



OperationManager class provides the interface ShutDown() to client. Concrete class HPOperationManager implements the method ShutDown(). The class HPOPSystem already exists and provides the functionality through its method PowerOff(). HPOperationManager has instance of HPOPSystem and uses it in ShutDown() method to call the method PowerOff().