**Structural Design Pattern**

Structural patterns explain how to assemble objects and classes into larger structures while keeping these structures flexible and efficient.

1. **Adapter Design Pattern**

Adapter pattern works as a bridge between two incompatible interfaces. This type of design pattern comes under structural pattern as this pattern combines the capability of two independent interfaces.

This pattern involves a single class which is responsible to join functionalities of independent or incompatible interfaces.

A real-life example could be a case of card reader which acts as an adapter between memory card and a laptop. You plug-in the memory card into card reader and card reader into the laptop so that memory card can be read via laptop.

**Scenario Need to Solve by Adapter Design Pattern:**

We are demonstrating use of Adapter pattern via following example in which an audio player device can play mp3 files only and wants to use an advanced audio player capable of playing vlc and mp4 files.

**Implementation**:

We have a MediaPlayer interface and a concrete class AudioPlayer implementing the MediaPlayer interface. AudioPlayer can play mp3 format audio files by default.

We are having another interface AdvancedMediaPlayer and concrete classes implementing the AdvancedMediaPlayer interface. These classes can play vlc and mp4 format files.

We want to make AudioPlayer to play other formats as well. To attain this, we have created an adapter class MediaAdapter which implements the MediaPlayer interface and uses AdvancedMediaPlayer objects to play the required format.

AudioPlayer uses the adapter class MediaAdapter passing it the desired audio type without knowing the actual class which can play the desired format.

An Adapter Pattern says that just "converts the interface of a class into another interface that a client wants".

In other words, to provide the interface according to client requirement while using the services of a class with a different interface.

The Adapter Pattern is also known as Wrapper.

**Advantage of Adapter Pattern**

* It allows two or more previously incompatible objects to interact.
* It allows reusability of existing functionality.

**Usage of Adapter pattern:**

It is used:

* When an object needs to utilize an existing class with an incompatible interface.
* When you want to create a reusable class that cooperates with classes which don't have compatible interfaces.
* When you want to create a reusable class that cooperates with classes which don't have compatible interfaces.

**Architectural Flow Diagram:**

* **Target Interface**: This is the desired interface class which will be used by the clients.
* **Adapter class**: This class is a wrapper class which implements the desired target interface and modifies the specific request available from the Adaptee class.
* **Adaptee class**: This is the class which is used by the Adapter class to reuse the existing functionality and modify them for desired use.
* **Client**: This class will interact with the Adapter class.

Reference:

1. JavaTpoint
2. Refactoring Guru
3. Tutorial Point