CS F213 - Object Oriented Programming

J. Jennifer Ranjani

email: jennifer.ranjani@pilani.bits-pilani.ac.in

Chamber: 6121 P, NAB

Consultation: Appointment by e-mail

https://github.com/JenniferRanjani/Object-Oriented-Programming-with-Java

BITS Pilani

Pilani Campus

innovate achieve lead

BITS Pilani
Pilani Campus

Structural Design Patterns

- It is about organizing different classes and objects to form larger structures and provide new functionality.

innovate achieve lead

BITS Pilani

Pilani Campus

Adapter Design Pattern

Adapter Pattern - Intent

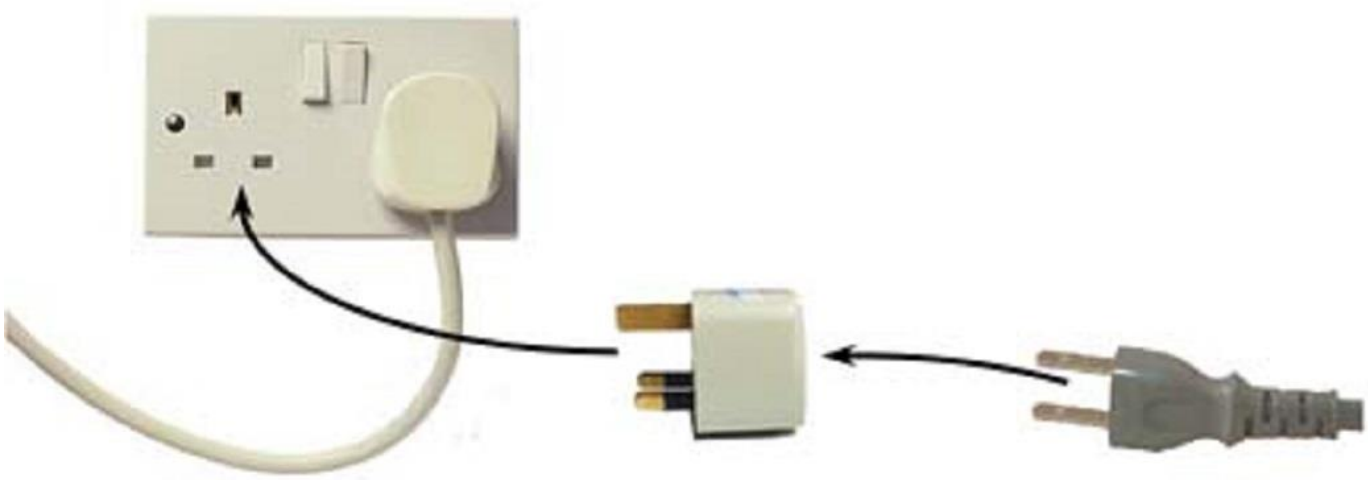
- Convert the interface of a class into another interface client expects.
- Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.
- Wrap an existing class with a new interface.

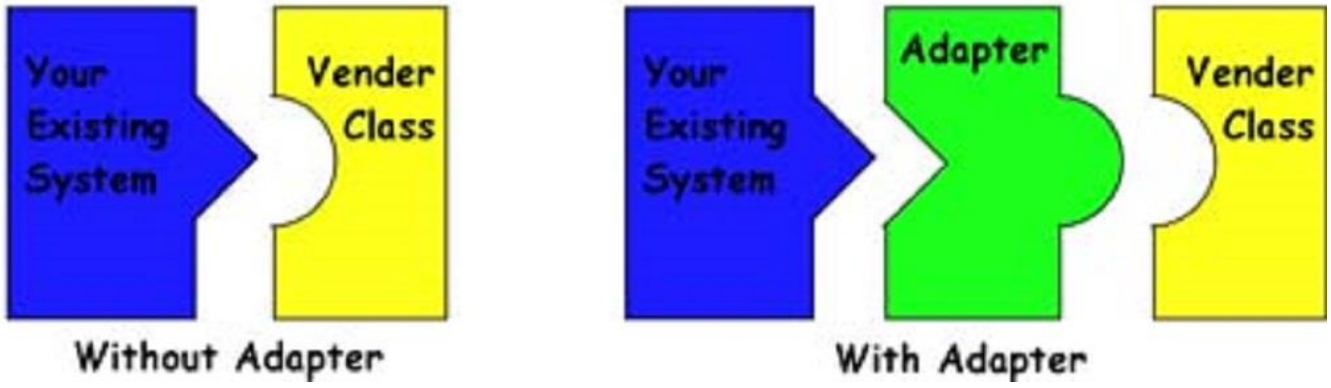
Adapter Pattern

- Adapter pattern work as a bridge between two incompatible interfaces.
- It combines the capability of two independent interfaces.
- Objects joining these unrelated interfaces is called an Adapter.

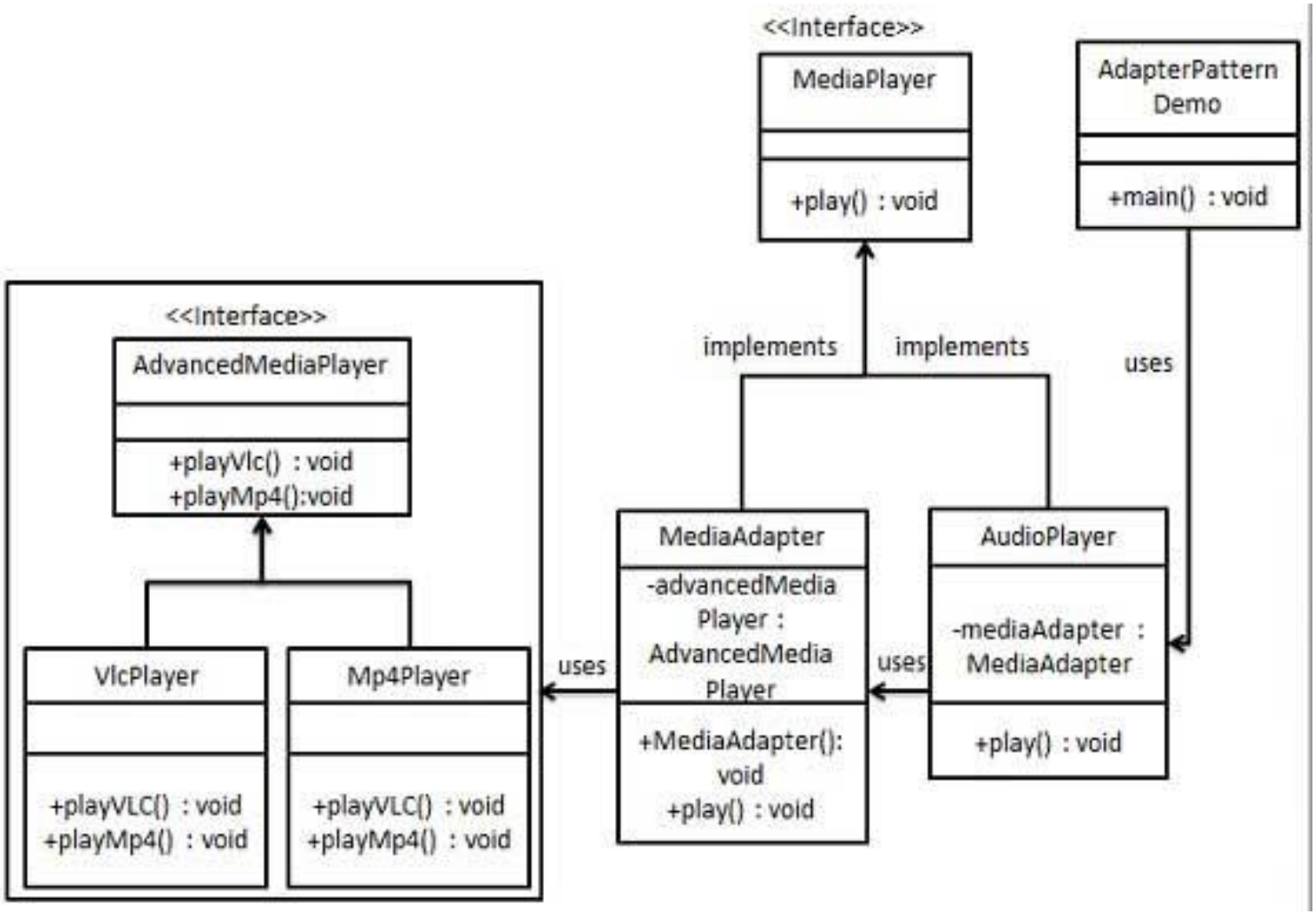
- Analogy:
  - Mobile charger: Battery needs 3V to charge and the normal socket produces 240V. Charger works as an adapter between mobile socket and the wall socket.

Adapter Pattern…





Adapter Pattern - Example



How is it done?

innovate achieve lead

- Identify the players: object (adaptee) provides the functionality desired by the client, but doesn't implement the interface (target) required by the client.
- Identify the interface that the client requires
- Design a wrapper class that can match the adaptee to the client.
- The adapter/wrapper class has a instance of the adaptee class
- The adapter/wrapper class maps the client interface to the adaptee interface
- The client uses the new interface.

innovate achieve lead

BITS Pilani

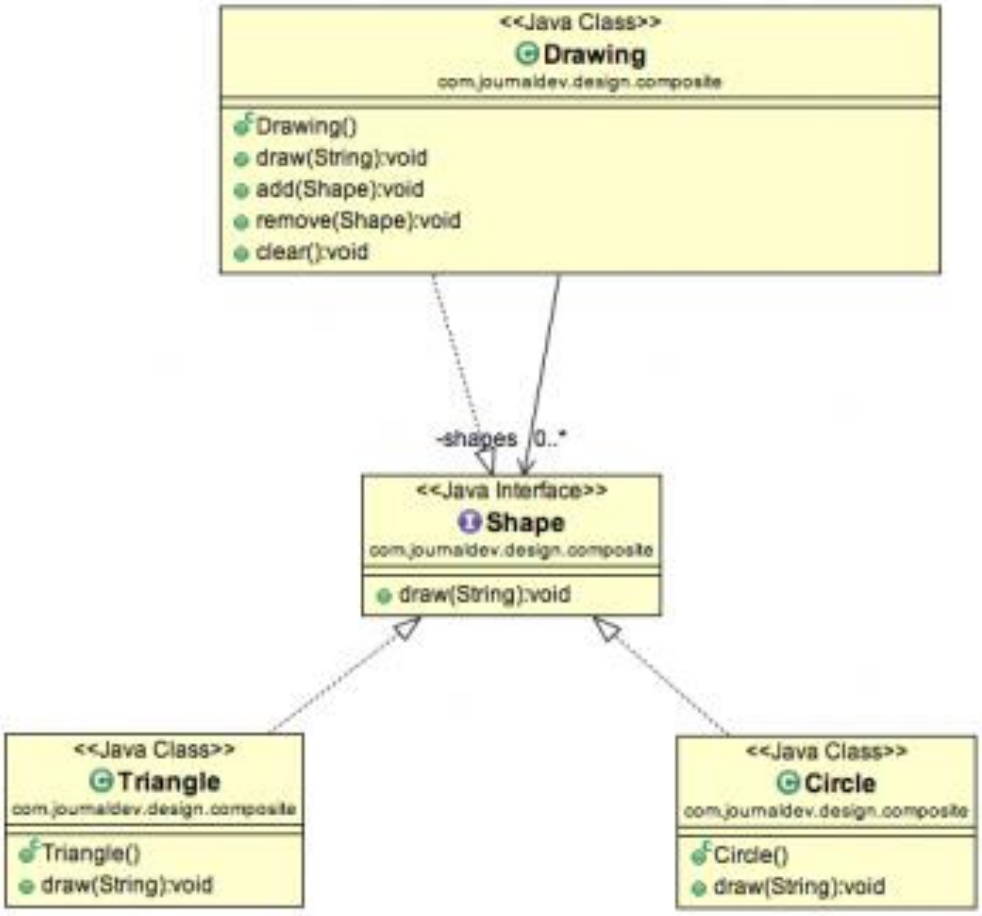
Pilani Campus

Composite Design Pattern

Consists of

- **Base Component:**
  - It is the interface for all objects in the composition, client uses the base component to work with the objects in the composition.
  - It can be an interface or an abstract class with some methods common to all the objects.
- **Lead:**
  - Defines the behavior for the elements in the composition
  - Building block for the composition and implements the base component
  - It doesn't have references to other components
- **Composite:**
  - Consists of leaf elements and implements operations in the base component

Example design



innovate achieve lead

BITS Pilani

Pilani Campus

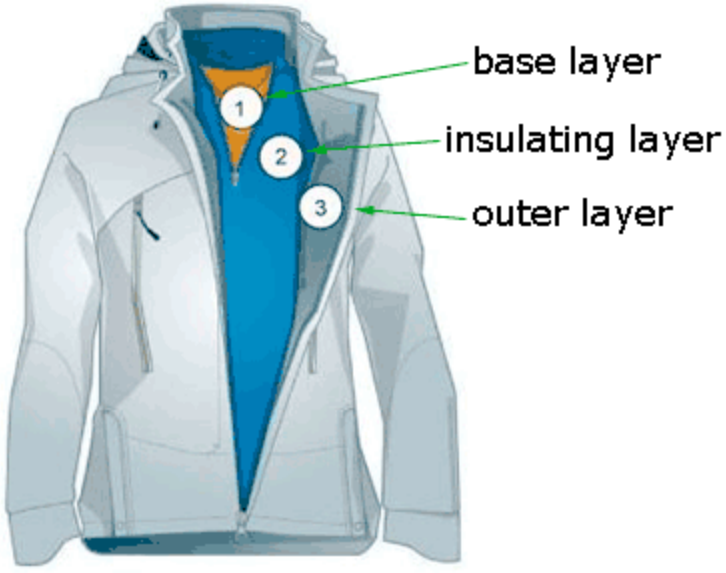
Decorator Design Pattern

Use

- It modifies the functionality of an object at run-time. At the same time other instances of the same class will not be affected.
- Inheritance or composition is used to extend the behavior of an object but this is done at compile time.
- We can add any new functionality or remove any existing behavior at run time.

Analogy

BufferedInputStream bis=new BufferedInputStream(new FileInputStream(new File("abc.txt")));



Pros and Cons

Pros:

- Provides a flexible alternative to subclassing for extending functionality
- Allows behavior modification at runtime rather than making changes in the existing code.
- Provides solution to permutation issues because we can wrap a component with any number of decorators.
- Supports a principle that classes should be open for extension but closed for modification.

Cons:

- Uses many small objects and overuse can be complex.
- Complicates the process of instantiating components.
- Multiple layers of decorator chain pushes its true intent beyond

Design Example