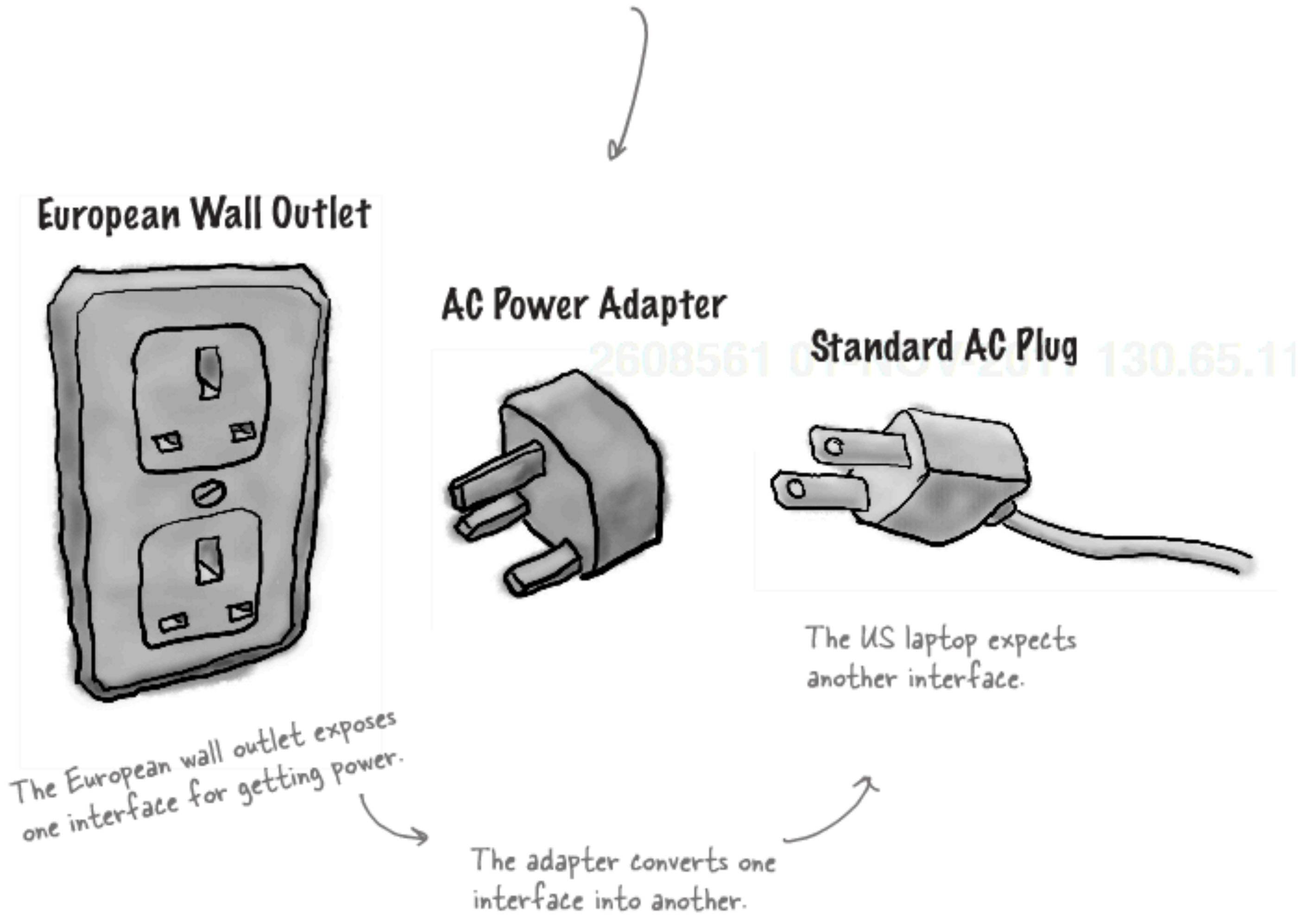
CMPE 202

Gang of Four Design Patterns

Adapter

Adapters all around us

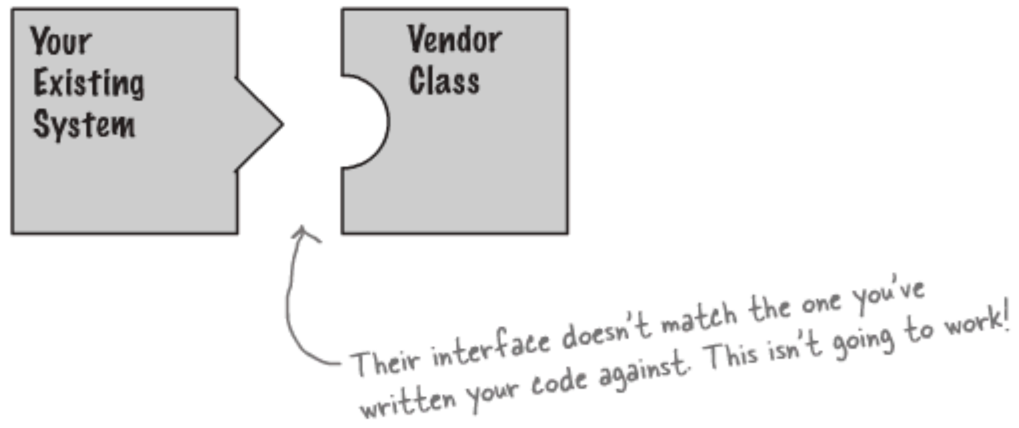
You'll have no trouble understanding what an OO adapter is because the real world is full of them. How's this for an example: Have you ever needed to use a US-made laptop in a European country? Then you've probably needed an AC power adapter...



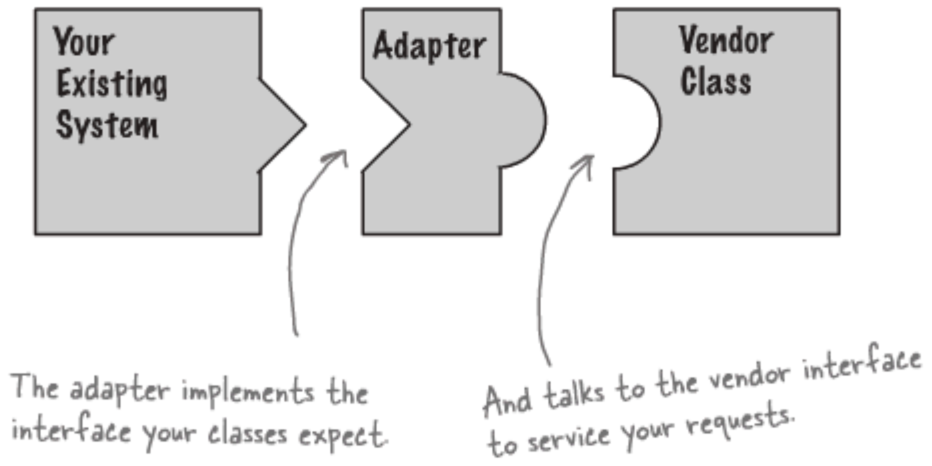
European Wall Outlet
AC Power Adapter
Standard AC Plug
The US laptop expects another interface.
The European wall outlet exposes one interface for getting power.
The adapter converts one interface into another.

Object oriented adapters

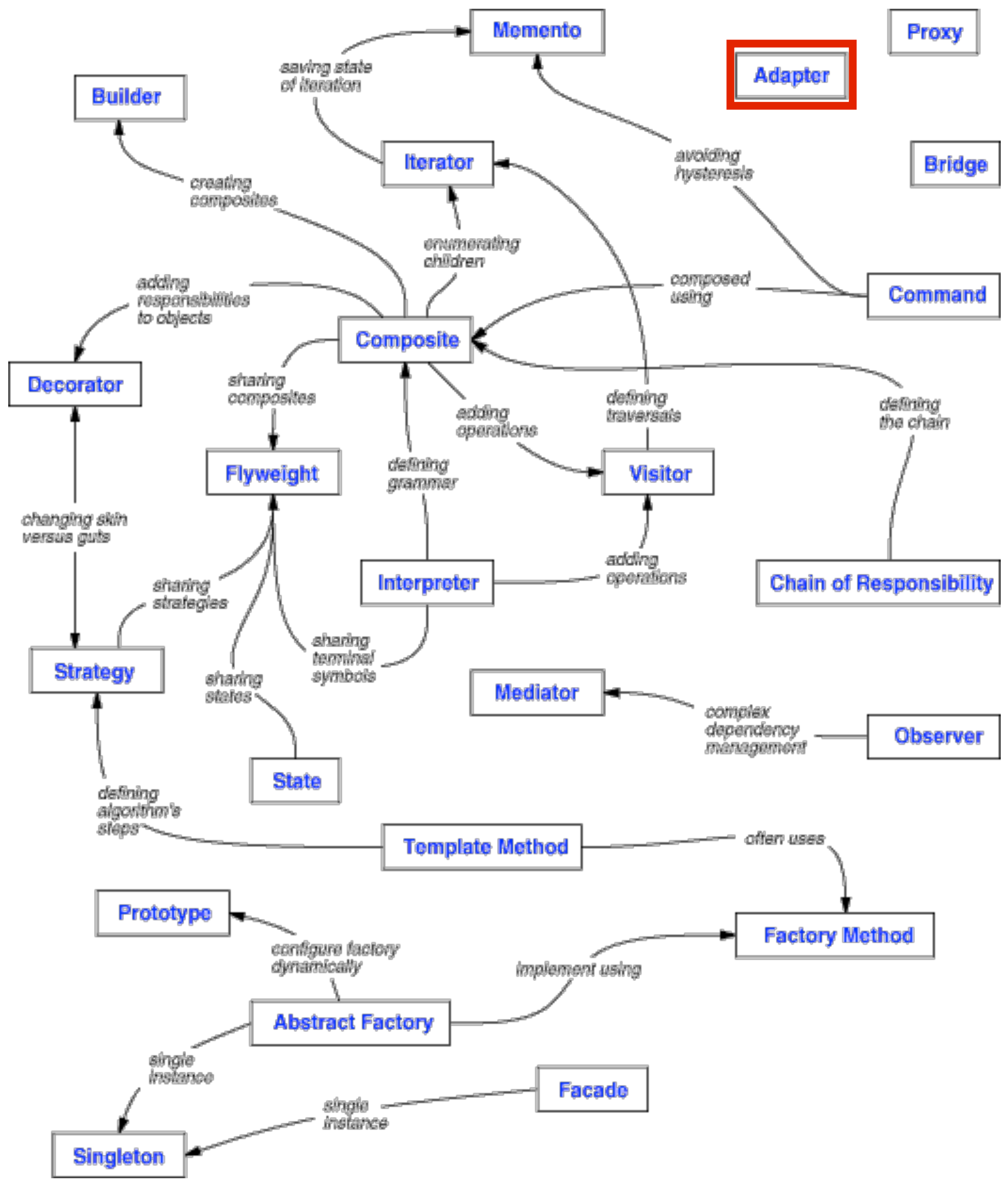
Say you've got an existing software system that you need to work a new vendor class library into, but the new vendor designed their interfaces differently than the last vendor:

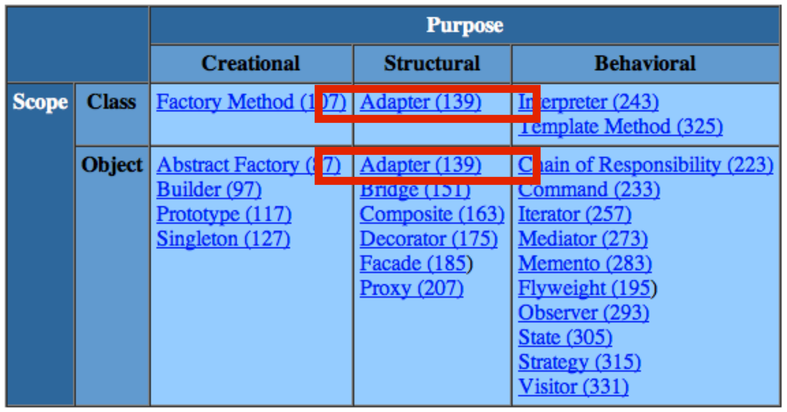

Your Existing System
Vendor Class
Their interface doesn't match the one you've written your code against. This isn't going to work!

Okay, you don't want to solve the problem by changing your existing code (and you can't change the vendor's code). So what do you do? Well, you can write a class that adapts the new vendor interface into the one you're expecting.


Your Existing System
Adapter
Vendor Class
The adapter implements the interface your classes expect.
And talks to the vendor interface to service your requests.

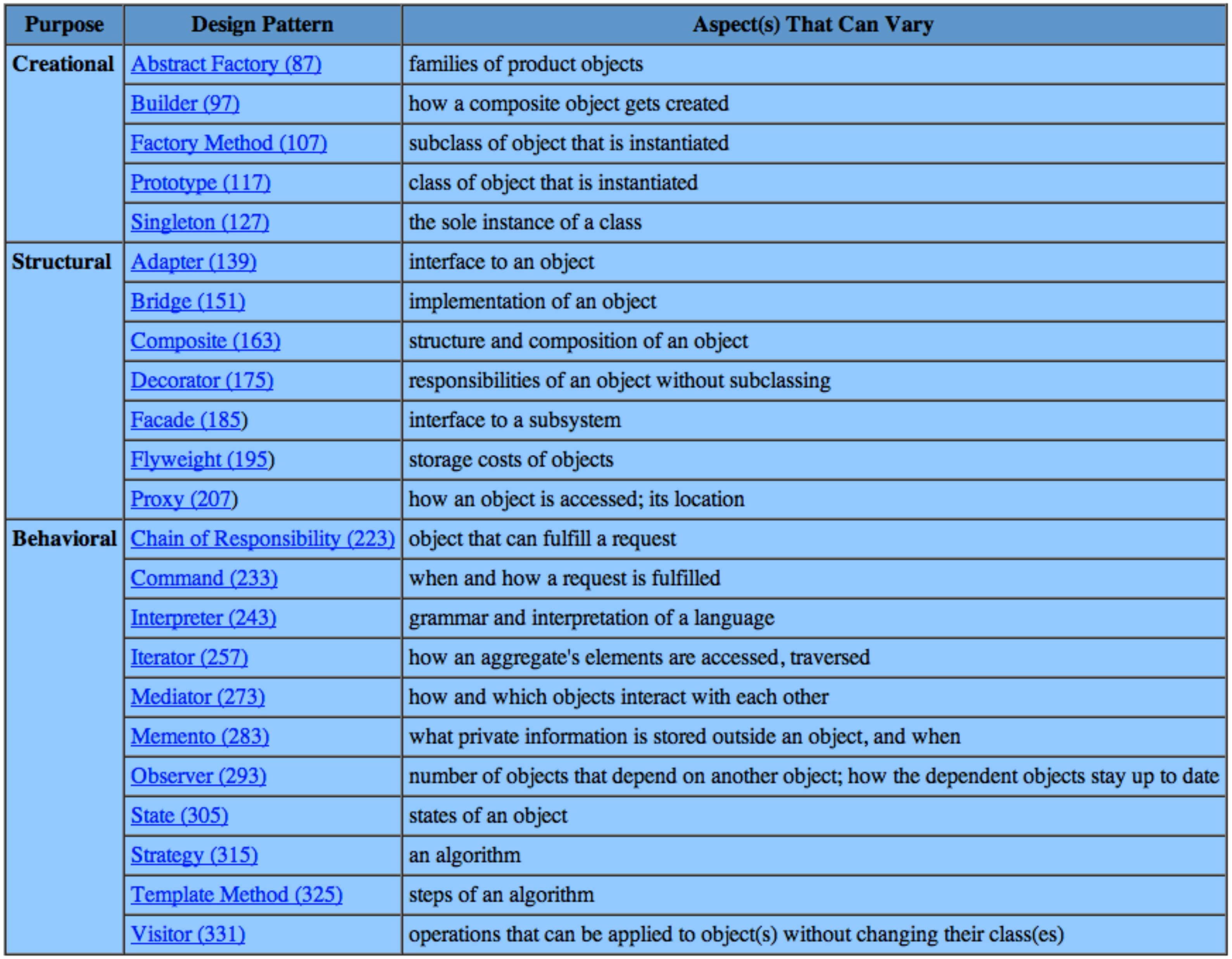
Adapter





		Purpose		
		Creational	Structural	Behavioral
Scope	Class	Factory Method (107)	Adapter (139)	Interpreter (243) Template Method (325)
	Object	Abstract Factory (87) Builder (97) Prototype (117) Singleton (127)	Adapter (139) Bridge (151) Composite (163) Decorator (175) Facade (185) Proxy (207)	Chain of Responsibility (223) Command (233) Iterator (257) Mediator (273) Memento (283) Flyweight (195) Observer (293) State (305) Strategy (315) Visitor (331)

Design Pattern Catalog



Purpose	Design Pattern	Aspect(s) That Can Vary
Creational	Abstract Factory (87)	families of product objects
	Builder (97)	how a composite object gets created
	Factory Method (107)	subclass of object that is instantiated
	Prototype (117)	class of object that is instantiated
	Singleton (127)	the sole instance of a class
Structural	Adapter (139)	interface to an object
	Bridge (151)	implementation of an object
	Composite (163)	structure and composition of an object
	Decorator (175)	responsibilities of an object without subclassing
	Facade (185)	interface to a subsystem
	Flyweight (195)	storage costs of objects
	Proxy (207)	how an object is accessed; its location
Behavioral	Chain of Responsibility (223)	object that can fulfill a request
	Command (233)	when and how a request is fulfilled
	Interpreter (243)	grammar and interpretation of a language
	Iterator (257)	how an aggregate's elements are accessed, traversed
	Mediator (273)	how and which objects interact with each other
	Memento (283)	what private information is stored outside an object, and when
	Observer (293)	number of objects that depend on another object; how the dependent objects stay up to date
	State (305)	states of an object
	Strategy (315)	an algorithm
	Template Method (325)	steps of an algorithm
	Visitor (331)	operations that can be applied to object(s) without changing their class(es)

Intent

Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.

Also Known As

Wrapper

Applicability

Use the Adapter pattern when

- you want to use an existing class, and its interface does not match the one you need.
- you want to create a reusable class that cooperates with unrelated or unforeseen classes, that is, classes that don't necessarily have compatible interfaces.
- *(object adapter only)* you need to use several existing subclasses, but it's impractical to adapt their interface by subclassing every one.
  An object adapter can adapt the interface of its parent class.

Participants

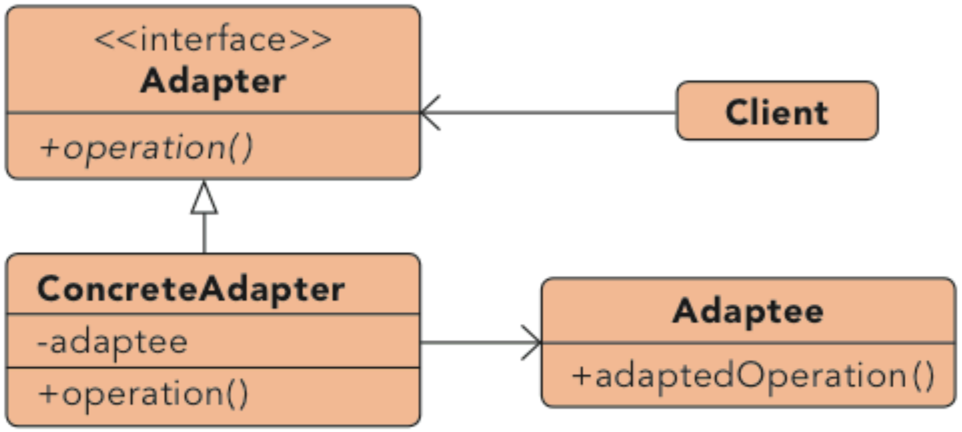
- **Target** (Interface)
  - defines the domain-specific interface that Client uses.
- **Client**
  - collaborates with objects conforming to the Target interface.
- **Adaptee** (Interface)
  - defines an existing interface that needs adapting.
- **Adapter**
  - adapts the interface of Adaptee to the Target interface.

Collaborations

- Clients call operations on an Adapter instance. In turn, the adapter calls Adaptee operations that carry out the request.

ADAPTER

Class and Object Structural



Purpose

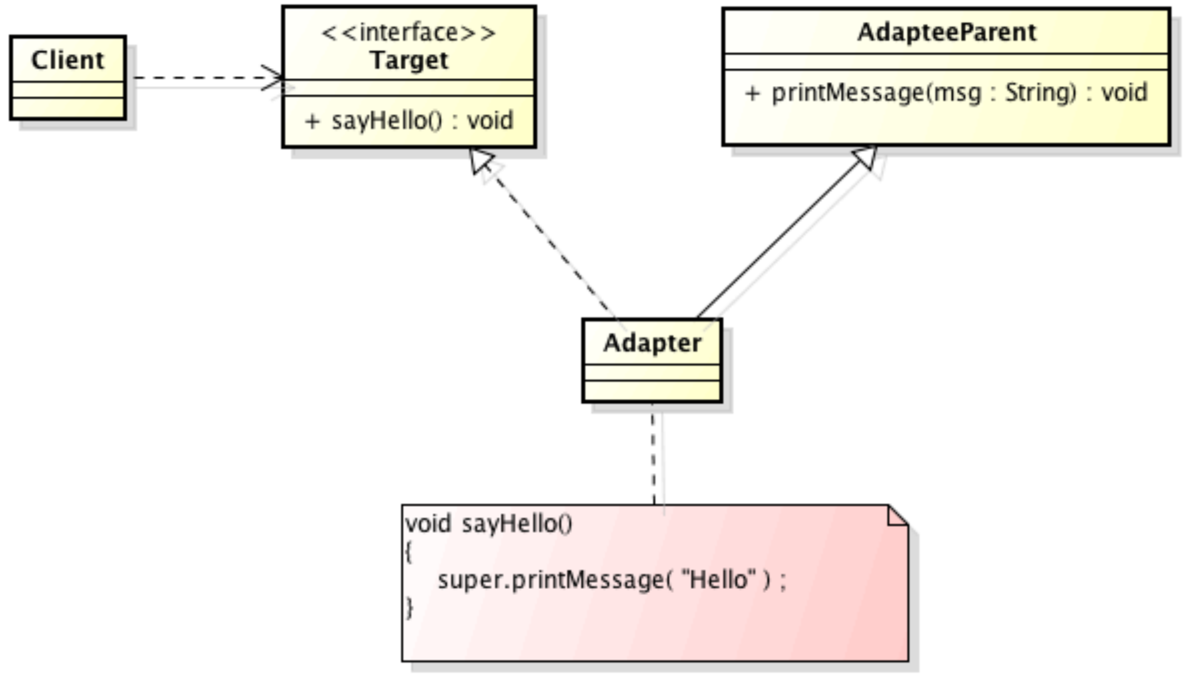
Permits classes with disparate interfaces to work together by creating a common object by which they may communicate and interact.

Use When

- A class to be used doesn't meet interface requirements.
- Complex conditions tie object behavior to its state.
- Transitions between states need to be explicit.

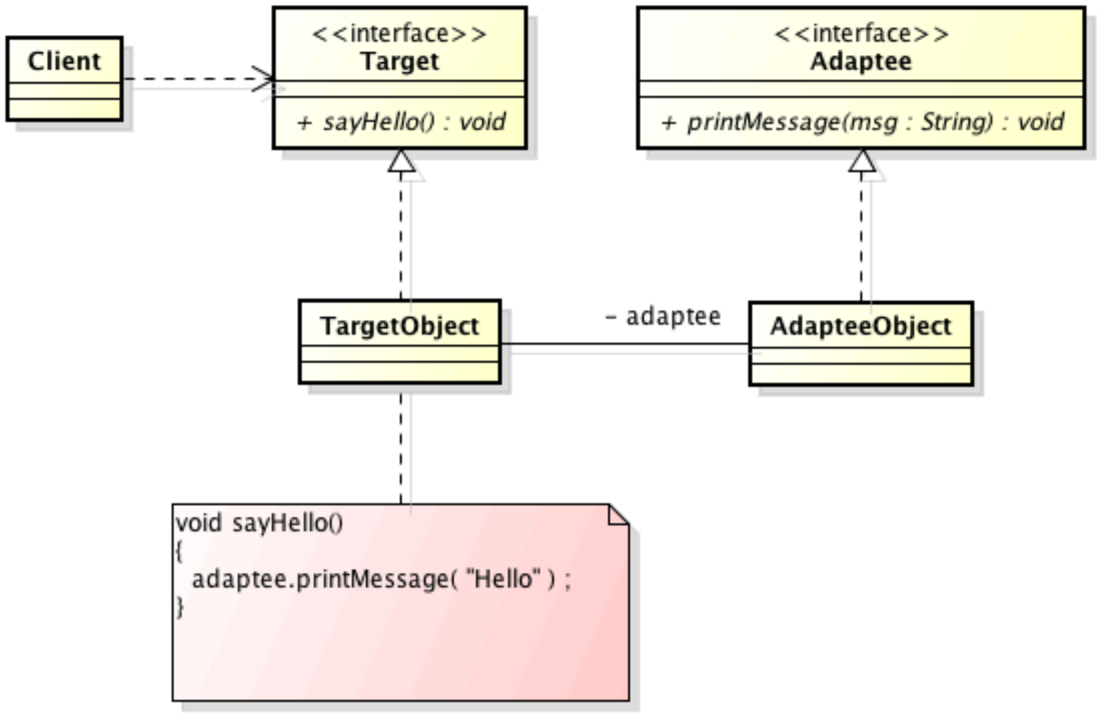
```
public class Client {

    public void runTest()
    {
        Target obj = new Adapter() ;
        obj.sayHello() ;
    }

}
```

```
public class AdapteeParent {

    public void printMessage(String msg) {
        System.out.println( msg );
    }

}
```



```
public class Adapter extends AdapteeParent implements Target {

    /**
     * @see adapter.classAdapter.Target#sayHello()
     */
    public void sayHello() {
        super.printMessage( "Hello" );
    }

}
```

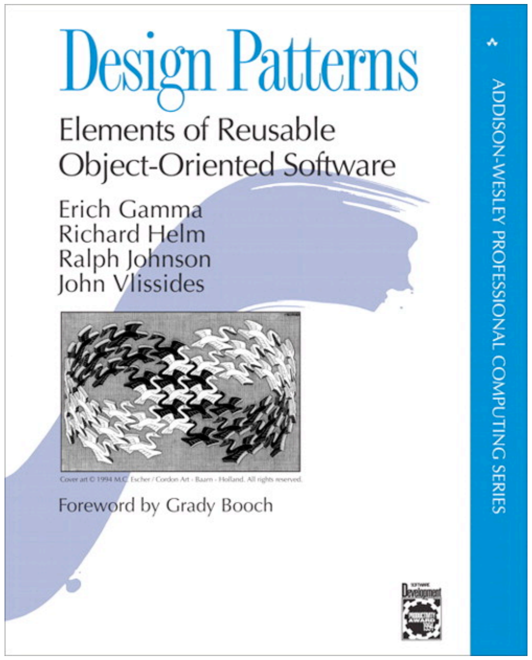
```
public class Client {

    public void runTest()
    {
        Target obj = new TargetObject() ;
        obj.sayHello() ;
    }

}
```



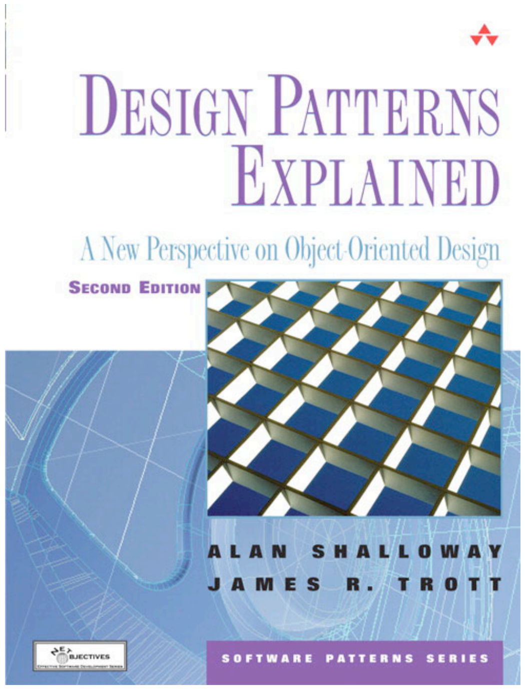
```
public class TargetObject implements Target {

    private AdapteeObject adaptee;

    public TargetObject()
    {
        adaptee = new AdapteeObject() ;
    }

    /**
     * @see adapter.objectAdapter.Target#sayHello()
     */
    public void sayHello() {
        adaptee.printMessage("Hello");
    }

}
```

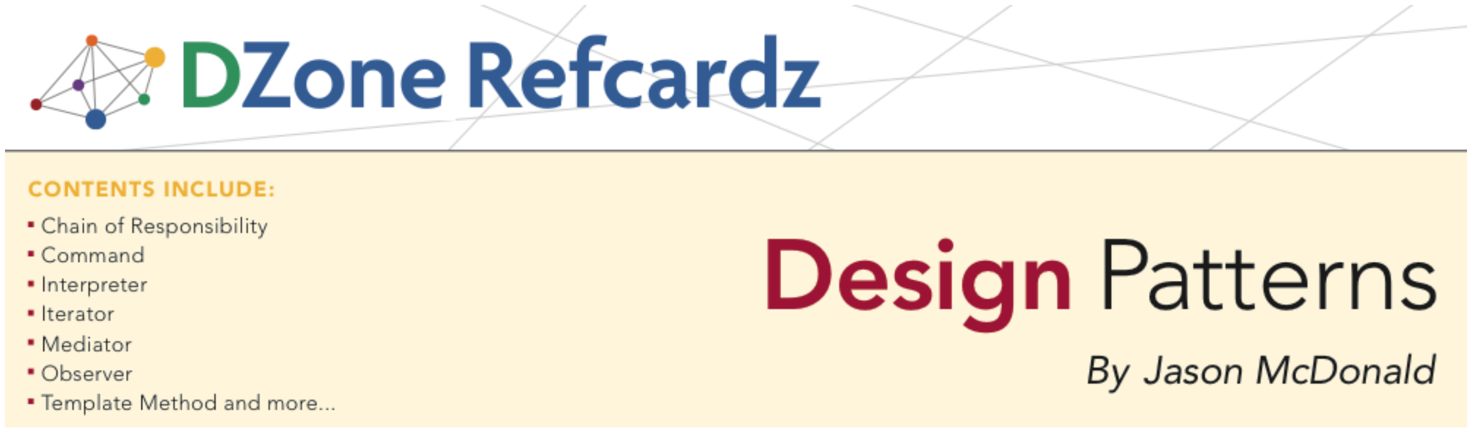
```
public class AdapteeObject implements Adaptee {

    private TargetObject targetObject;

    /**
     * @see adapter.objectAdapter.Adaptee#printMessage(java.lang.String)
     */
    public void printMessage(String msg) {
        System.out.println( msg ) ;
    }

}
```

Resources for this Tutorial


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

A Brain-Friendly Guide
Head First Design Patterns
Avoid those embarrassing coupling mistakes
Learn why everything your friends know about Factory pattern is probably wrong
Discover the secrets of the Patterns Guru
Load the patterns that matter straight into your brain
Find out how Starbuzz Coffee doubled their stock price with the Decorator pattern
See why Jim's love life improved when he cut down his inheritance
O'REILLY
Eric Freeman & Elisabeth Freeman
with Kathy Sierra & Bert Bates


Design Patterns Explained
A New Perspective on Object-Oriented Design
Second Edition
Alan Shalloway
James R. Trott
Software Patterns Series


DZone Refcardz
CONTENTS INCLUDE:
- Chain of Responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Observer
- Template Method and more...

Design Patterns
By Jason McDonald