Java로 배우는 디자인패턴 입문

Chapter 2. Adapter
필요한 형태로 수정해서 재활용한다

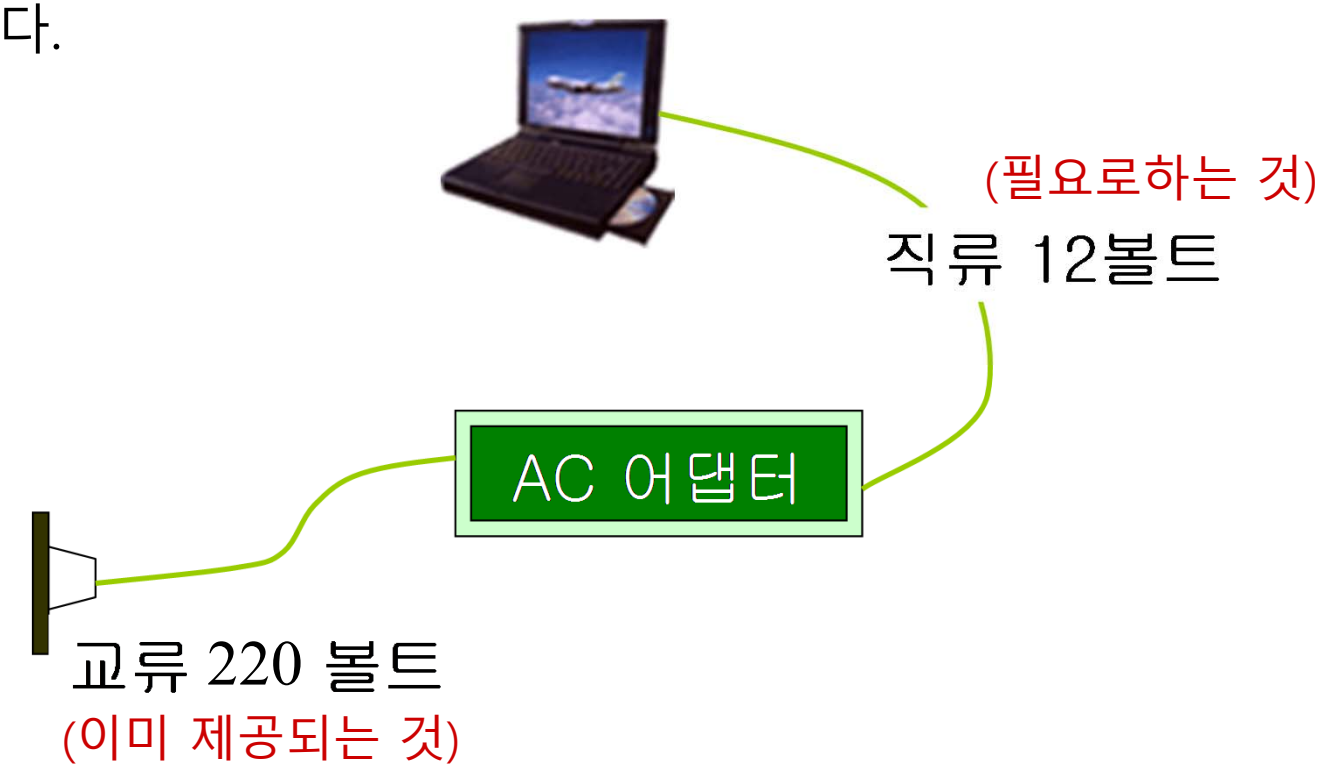
교재: 자바언어로배우는디자인패턴입문(개정판)/YukiHiroshi저/김윤정역/영진닷컴

덕성여자대학교 컴퓨터학과

최 승 훈

01. Adapter 패턴

- AC 어댑터
  - 기존의 교류 220볼트 전기를, 직류 12 볼트로 바꾸어 준다
  - 필요한 건 12볼트인데, 220볼트 밖에 없다. => 어댑터를 사용한다.



01. Adapter 패턴

- Adapter 패턴
  - 이미 제공되어 있는 것을 그대로 사용할 수 없는 경우
  - '이미 제공되어 있는 것'과 '필요한 것' 사이의 간격을 메우는 디자인 패턴
  - Wrapper 패턴이라고도 한다.

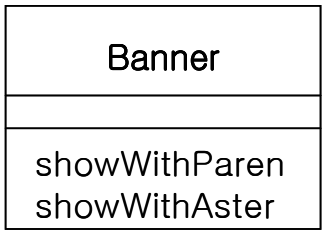
- 두 가지 종류의 Adapter 패턴
  - 상속(inheritance)을 이용한 Adapter 패턴
  - 위임(delegation)을 이용한 Adapter 패턴

02. 예제 프로그램 1 – 상속을 이용한 것

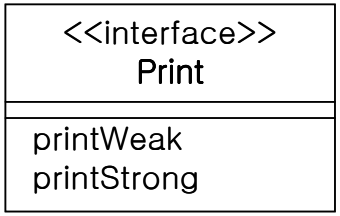
- Banner 클래스
  - showWithParen( ): 문자열 앞뒤에 괄호를 쳐서 표시하는 메소드
  - shwoWithAster( ): 문자열 앞뒤에 '*'를 붙여서 표시하는 메소드
  - 이 두 메소드를 '이미 제공되어 있는 것'으로 가정한다.
- Print 인터페이스
  - printWeak( ): 문자열을 약하게 표시(괄호를 붙임)
  - printStrong( ): 문자열을 강하게 표시('*' 붙임)
  - 이 두 메소드를 직류 12볼트와 같은 '필요한 것'이라고 가정한다.
- 목표
  - Banner 클래스라는 기존의 클래스를 이용해서, Print 인터페이스를 충족시키는(즉, 구현하는) 클래스를 만들고자 한다.

02. 예제 프로그램 1 – 상속을 이용한 것

- 이미 제공되는 것:

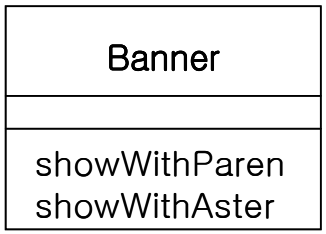


- 필요로 하는 것:

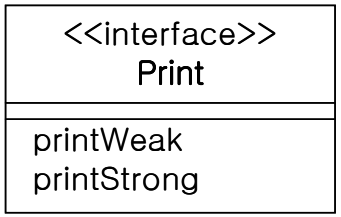


필요한 메소드는, Print 인터페이스에 정의되어 있는
printWeak와 printStrong이다.
그러나, 같은 일을 하는 메소드를 Banner 클래스가 제공한다.
⇒ Banner 클래스의 메소드들을 재활용하는 Adapter 클래스를 만든다.
⇒ PrintBanner 클래스

02. 예제 프로그램 1 – 상속을 이용한 것



이미 존재하며 수정하지 못하는 클래스



필요한 메소드를 선언한 인터페이스

Client

Print 인터페이스에 선언된 메소드를 이용해서 Banner 클래스가 가지고 있는 메소드를 이용하고자 한다.

02. 예제 프로그램 1 – 상속을 이용한 것

- PrintBanner 클래스
  - Banner 클래스를 상속하면서, Print 인터페이스를 구현한다.

- 전원 예와 예제 프로그램 비교

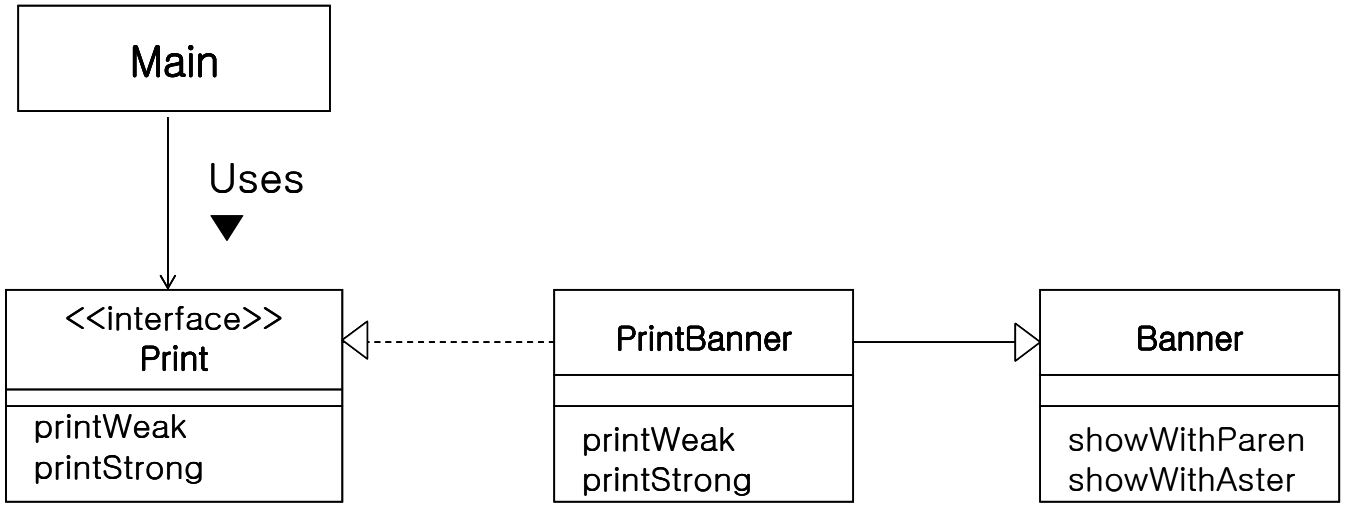
	전원의 비유	예제 프로그램
제공되어 있는 것	교류 100볼트	Banner클래스 (showWithParen,showWithAster)
교환장치	어댑터	PrintBanner클래스
필요한 것	직류 12볼트	Print 인터페이스(printWeak, printStrong)

02. 예제 프로그램 1 – 상속을 이용한 것

- Banner 클래스 소스 (sample1/Banner.java)
- Print 인터페이스 소스 (sample1/Print.java)
- PrintBanner 클래스 소스 (sample1/PrintBanner.java)
  - Banner 클래스를 상속, Print 인터페이스를 구현
  - printWeak( ): 상속받은 showWithParen( )을 호출한다.
  - printStrong( ): 상속받은 showWithAster( )를 호출한다.

02. 예제 프로그램 1 – 상속을 이용한 것

- 클래스 다이어그램



02. 예제 프로그램 1 – 상속을 이용한 것

- Main 클래스 소스 (sample1/Main.java)
  - PrintBanner의 인스턴스를 Print 인터페이스 형의 변수에 할당
    - Main 클래스는, Print 인터페이스를 사용해서 프로그래밍했음
    - 실제 일을 수행하는 Banner 클래스의 메소드는, Main 클래스에서는 완전히 은폐되어 있다.
  - Main 클래스를 수정하지 않고도, Banner 클래스의 구현을 변경할 수 있다.

03. 예제 프로그램 2 – 위임을 이용한 것

- 위임
  - 내가 할 일을 누군가에게 맡긴다.
  - 예제에서:
    - PrintBanner는 자신이 할 일을 Banner 클래스의 인스턴스에게 맡긴다.

- 예제1과 달리, Print를 클래스로 가정하면,
  - PrintBanner가 Print와 Banner의 하위 클래스로 정의할 수 없다. (다중 상속을 자바는 지원하지 않기 때문에)
  - 위임을 사용해야 한다.

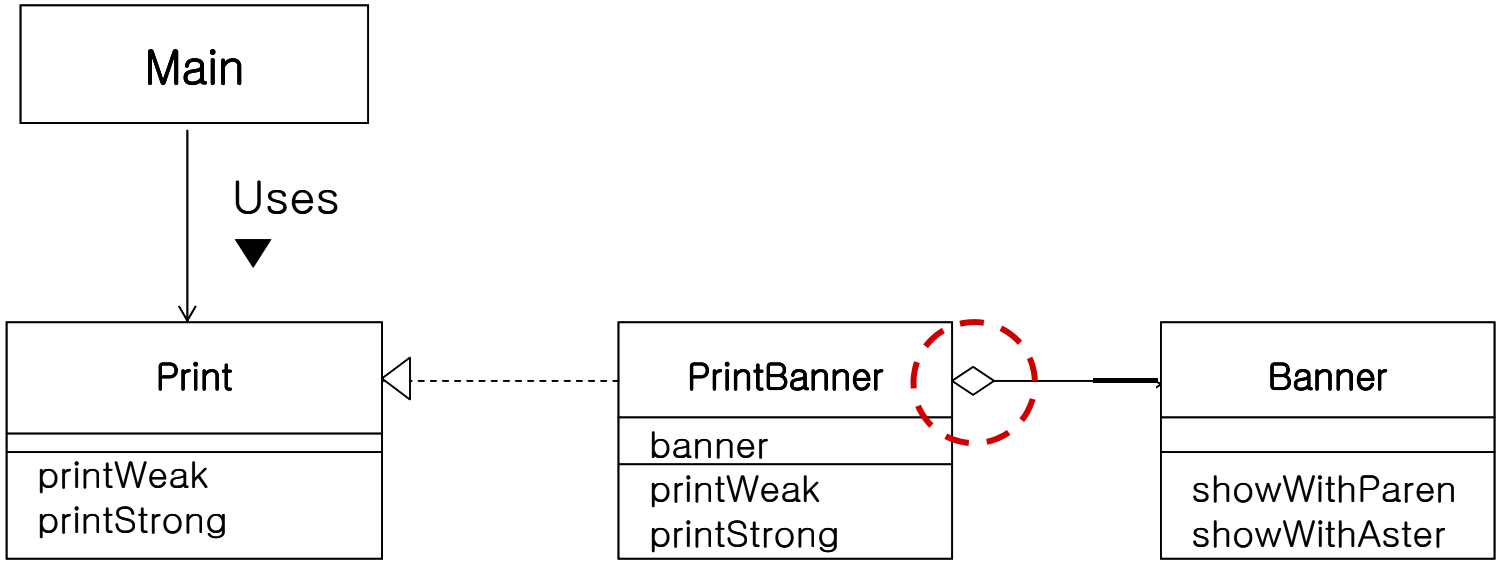
03. 예제 프로그램 2 – 위임을 이용한 것

- Print 클래스 (sample2/Print.java)
  - 추상 클래스로 정의됨

- PrintBanner 클래스 (sample2/PrintBanner.java)
  - banner 필드가 Banner 클래스의 인스턴스를 가진다.
  - printWeak( ):
    - banner의 showWithParen()을 호출한다.
    - PrintBanner 자신이 일을 처리하지 않고, banner에게 위임한다.
  - printStrong( ):
    - banner의 showWithAster()을 호출한다.
    - PrintBanner 자신이 일을 처리하지 않고, banner에게 위임한다.

03. 예제 프로그램 2 – 위임을 이용한 것

- 클래스 다이어그램

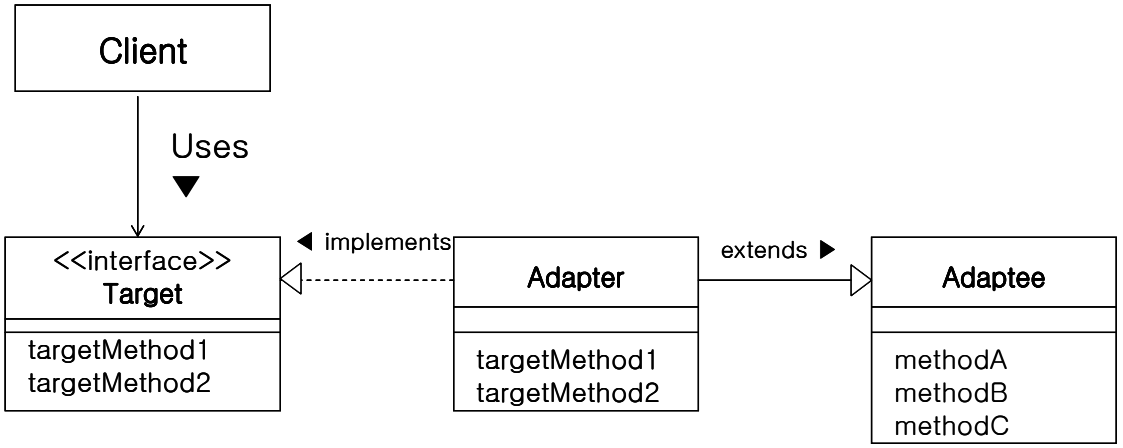


04. Adapter 패턴에 등장하는 역할

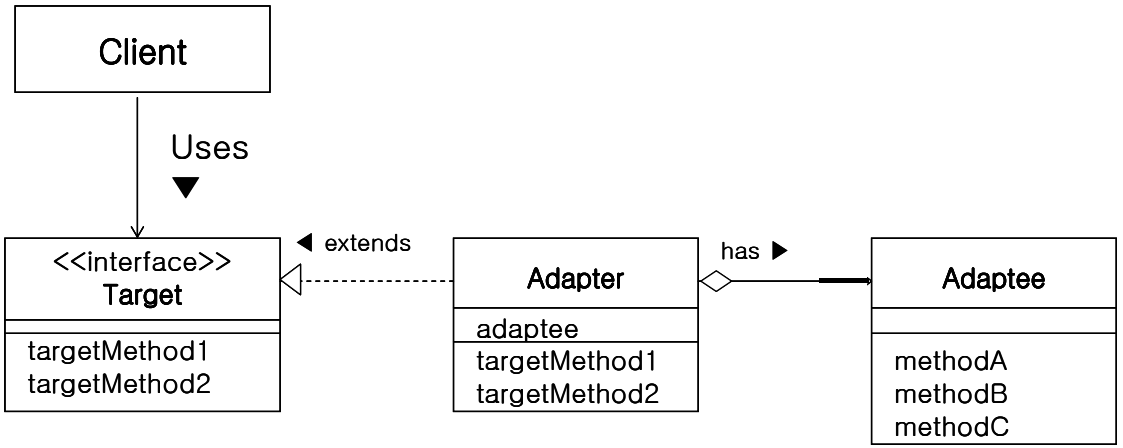
- Target(대상)의 역할
  - 필요한 메소드를 제공(선언)하는 역할
  - 예제: 직류 12 볼트 또는 Print 인터페이스나 클래스
- Client(의뢰자)의 역할
  - Target 역할의 메소드를 이용하는 역할
  - 예제: 12 볼트로 동작하는 노트북이나 Main 클래스
- Adaptee(변경되는 측)의 역할
  - 이미 준비되어 있는 메소드를 제공하는 역할
  - 예제: 교류 100볼트 전원이나 Banner 클래스
- Adapter의 역할
  - Target 역할을 실제로 충족시키는 역할
  - 예제: 어댑터나 PrintBanner 클래스

04. Adapter 패턴에 등장하는 역할

- 상속을 이용한 패턴



- 위임을 이용한 패턴



05. 독자의 사고를 넓혀주는 힌트

- 어떤 경우에 사용할까
  - 기존의 클래스를 수정하지 않고도 필요한 클래스로 만들 때 사용함
  - 기존의 클래스가 충분히 테스트 되어 있을 때 더욱 좋다
    - 재사용할 때 어댑터 패턴 적용함

- 만약 소스가 없더라도
  - 기존의 클래스는 손을 대지 않고, 새로운 인터페이스에 기존의 클래스를 맞출 수 있다.
  - 특히, 기존 클래스의 소스 코드 없이, 메소드의 프로토타입만 알면 어댑터 패턴을 적용할 수 있다.
    - 메소드 프로토타입: 메소드의 이름, 인자 개수, 인자형, 반환형 등

06. Adapter 패턴과 관련된 패턴

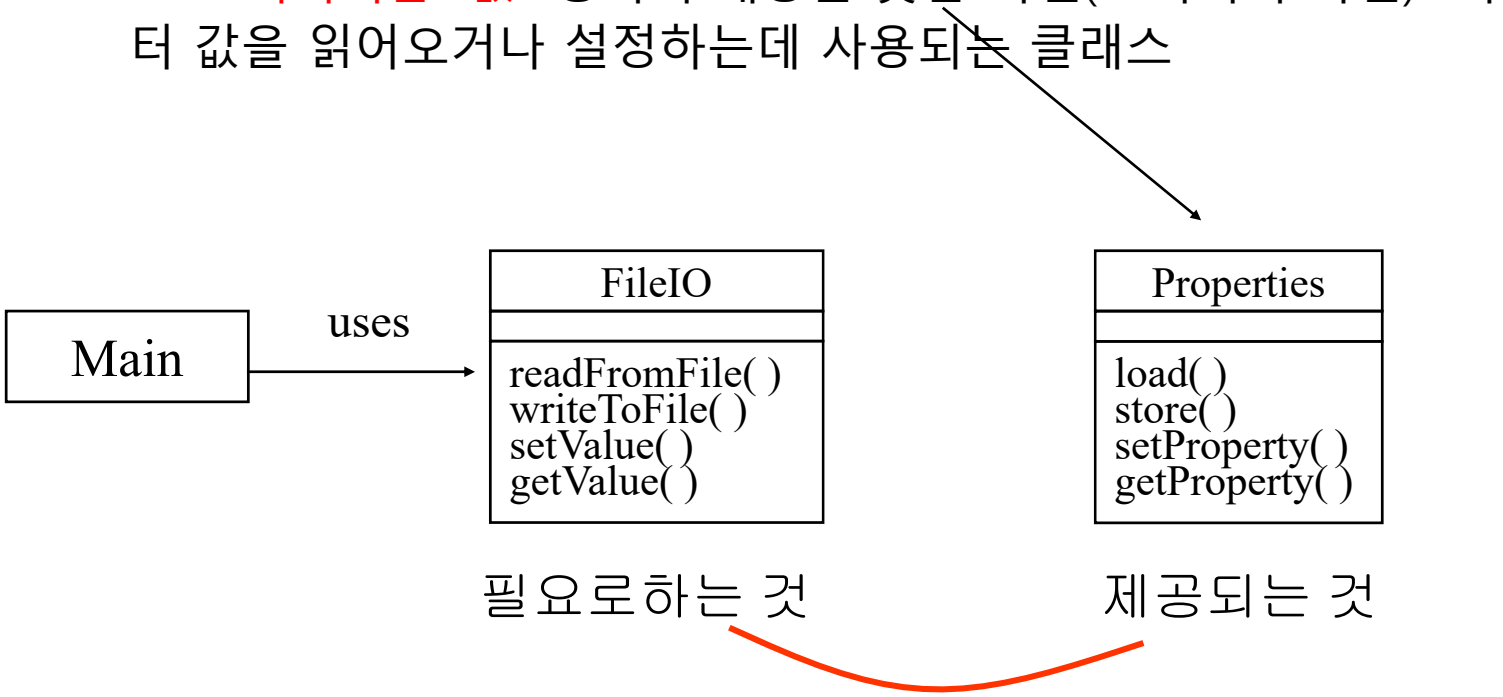
- Bridge 패턴(9장)
- Decorator 패턴(12장)

07. 이 장에서 학습한 내용

- 인터페이스(API)가 다른 두 개의 사이에 들어가 그 사이를 메우는 Adapter 패턴
  - API: Application Programming Interfaces

연습문제

- 2-2
  - 기존에 제공되는 클래스: java.util.Properties
    - "프로퍼티이름=값" 형식의 내용을 갖는 파일(프러퍼티 파일)로부터 값을 읽어오거나 설정하는데 사용되는 클래스



연습문제

- 2-2
  - 어댑터 FileProperties: Properties 상속, FileIO 구현
    - Main 클래스는, FileIO의 인터페이스를 호출한다.
    - FileIO를 구현한 FileProperties의 각 메소드들은 Properties의 해당 메소드를 호출한다.