Java(실습 11주차)

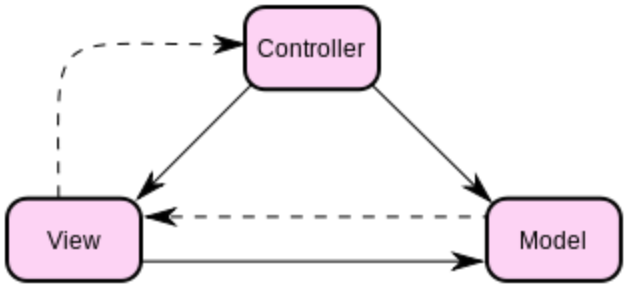
15th May 2014

목표

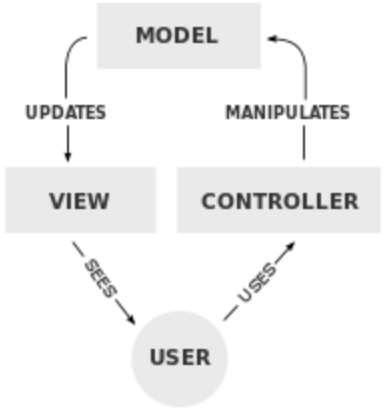
- MVC 를 알아보고 객체지향 프로그래밍에 대한 이해를 높인다
  - MVC (Model-View-Controller)
  - Observer pattern

MVC

- **모델-뷰-컨트롤러**(Model–View–Controller, MVC)
  - 소프트웨어 공학에서 사용되는 아키텍처 패턴이다. 이 패턴을 성공적으로 사용하면, 사용자 인터페이스로부터 비즈니스 로직을 분리하여 애플리케이션의 시각적 요소나 그 이면에서 실행되는 비즈니스 로직을 서로 영향 없이 쉽게 고칠 수 있는 애플리케이션을 만들 수 있다. MVC에서 모델은 애플리케이션의 정보(데이터)를 나타내며, 뷰는 텍스트, 체크박스 항목 등과 같은 사용자 인터페이스 요소를 나타내고, 컨트롤러는 데이터와 비즈니스 로직 사이의 상호동작을 관리한다.

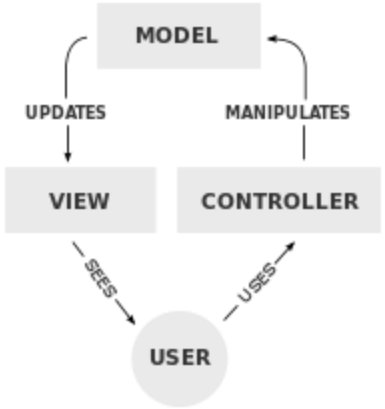


MVC



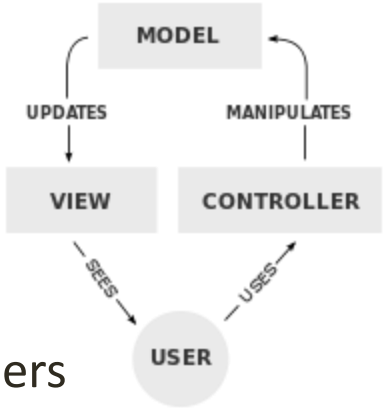
- View
  - 객체의 입력과 출력을 담당
  - 필요에 따라 Model로부터 객체의 상태를 요청할 수 있고 응답 받은 상태에 따라 다른 출력 형식을 가질 수 있으며 Controller에게 전달 시 상태정보를 같이 보낼 수 있다.
  - Model로부터 응답 받은 객체의 상태에 의해 View가 직접 제어, 가공을 하는 일이 없도록 해야한다.
    - View는 상태에 따라 출력형식만 다르게 가야하고 모든 제어나 가공은 Controller

MVC



- Controller
  - 객체 또는 데이터의 흐름을 책임
  - 활성화 된 View로부터 넘겨 받은 메시지(User Action or event) 또는 객체를 파악하여 해당 객체를 어떤 Model로 전달할지를 결정하고 필요한 객체를 데이터를 가공할 Model로 전달

MVC



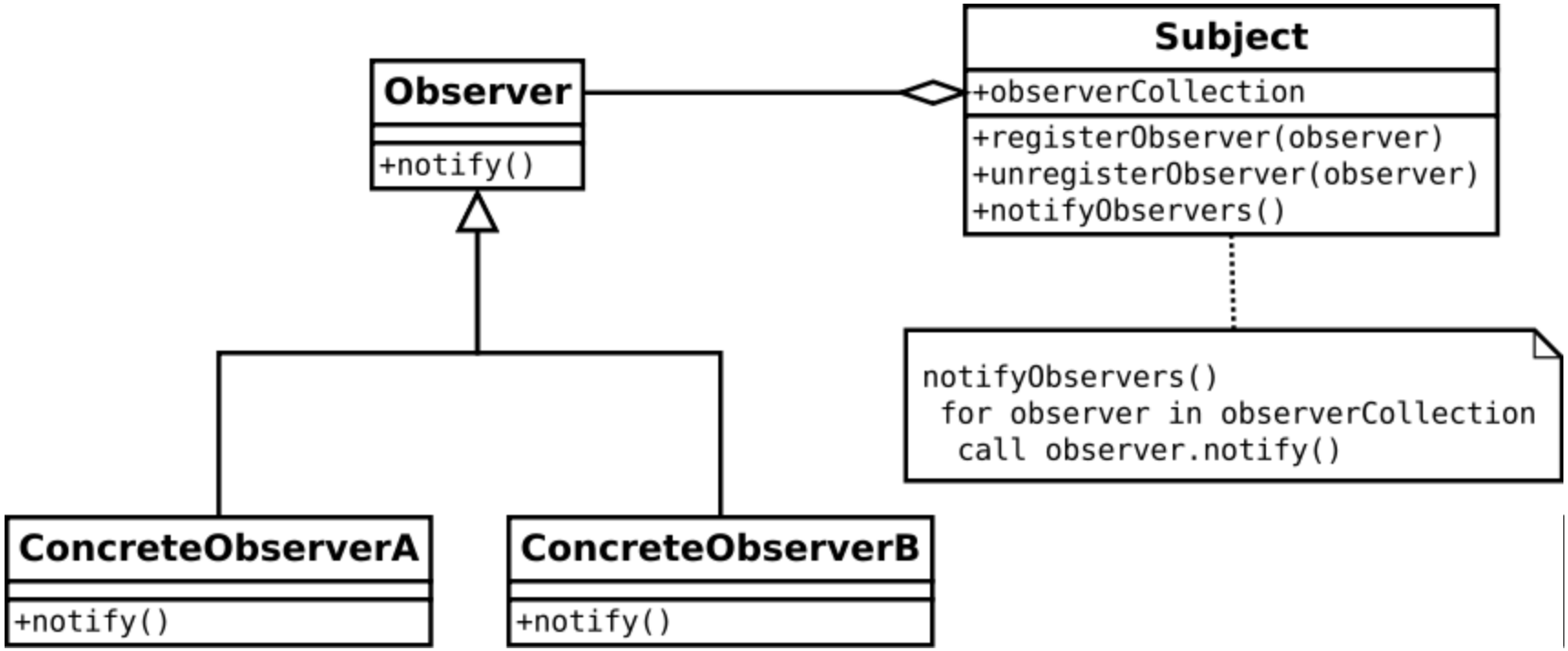
- Model
  - A **model** notifies its associated views and controllers
  - when there has been a change in its state.
  - This notification allows the views to produce updated output, and the controllers to change the available set of commands. A *passive* i mplementation of MVC omits these notifications, because the appli cation does not require them or the software platform does not su pport them.

Observer pattern

- 정의
  - 객체의 상태 변화를 관찰하는 관찰자들, 즉 옵저버들의 목록을 객체에 등록하여 상태 변화가 있을 때마다 메서드 등을 통해 객체가 직접 목록의 각 옵저버에게 통지하도록 하는 디자인 패턴이다. 주로 분산 이벤트 핸들링 시스템을 구현하는 데 사용된다. 발행/구독 모델로 알려져 있기도 하다.

Observer pattern

- Class diagram



Observer pattern

- 구현
  - 이 패턴의 핵심은 옵서버 또는 리스너(listener)라 불리는 하나 이상의 객체를 등록하거나 자신을 등록시킨다. 그리고 관찰되는 객체(또는 주제)에서 발생하는 이벤트를 전달한다.
  - **등록**(register), **제거**(unregister) Method
    - 새로운 리스너를 등록
    - 이벤트가 발생할 때 알려줄 객체 목록에서 옵서버를 제거
  - 옵서버 패턴이 많이 쓰인 시스템에서는 순환 실행을 막는 메카니즘이 필요하다. 이벤트 X가 있을 때 A가 B를 갱신한다고 가정해보자. B는 이 처리를 위해 A를 갱신한다. 그래서 다시 A로 하여금 이벤트 X를 발생하게 한다. 이같은 상황을 막기 위해 이벤트 X가 처리된 후에는 A가 이것을 다시 발생시키지 않는 방법이 요구된다.

EventSource.java

```
import java.util.Observable; // 이 부분이 옵서버입니다.
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class EventSource extends Observable implements Runnable
{
	public void run()
	{
		try
		{
			final InputStreamReader isr = new InputStreamReader( System.in );
			final BufferedReader br = new BufferedReader( isr );
			while( true )
			{
				final String response = br.readLine();
				setChanged();
				notifyObservers( response );
			}
		}
		catch (IOException e) { e.printStackTrace(); }
	}
}
```

ResponseHandler.java

```
import java.util.Observable;
import java.util.Observer; /* 여기가 처리기 */
public class ResponseHandler implements Observer
{
	private String resp;
	public void update (Observable obj, Object arg)
	{
		if (arg instanceof String)
		{
			resp = (String) arg;
			System.out.println("\nReceived Response: "+ resp );
		}
	}
}
```

myapp.java

```
public class MyApp
{
	public static void main(String args[])
	{
		System.out.println("Enter Text >");
		// create an event source - reads from stdin
		final EventSource evSrc = new EventSource();
		// create an observer
		final ResponseHandler respHandler = new ResponseHandler();
		// subscribe the observer to the event source
		evSrc.addObserver( respHandler );
		// starts the event thread
		Thread thread = new Thread(evSrc);
		thread.start();
	}
}
```

Reference

http://docs.oracle.com/javase/7/docs/api/java/util/Observer.html
http://docs.oracle.com/javase/7/docs/api/java/util/Observable.html
http://en.wikipedia.org/wiki/Observer_pattern

Assignment

지난 과제였던 커피 주문 프로그램을 observer pattern을
이용해 주문하는 형식으로 변환하는 프로그램을 작성하여 보자

입력
  순서
   1. 기본 커피 종류 (HouseBlend, DarkRoast, Espresso, Decaf)
   2. 데코레이트 입력을 end가 입력 될때까지 받는다
      - Milk, Mocha, Soy, Whip
   3. 제거 기능을 추가한다.
      - 제거는 Decorator 패턴에 remove(…) 메소드를 정의하여
        구현한다.
   4. 지불 해야할 금액을 듣고 지불 금액을 입력
출력
   입력받은 커피의 종류를 응답하고 입력이 끝나면 금액 계산후
  거스름돈을 출력한다.

Assignment

입출력 예)

Beverage
Mocha -> 입력
Beverage is Mocha -> 출력
Decorator
Soy -> 입력
Mocha + Soy -> 출력
....
End -> 입력
Mocha + Soy + ...
Cost is 5000 -> 출력
10000 -> 입력
Change is 5000 -> 출력
...

Q&A