*객체지향 설계와 패턴*  
**LAB ASSIGNMENT #6: 컴포지트, 브리지 패턴**

## Weight 10% 마감 4월 18일(일)

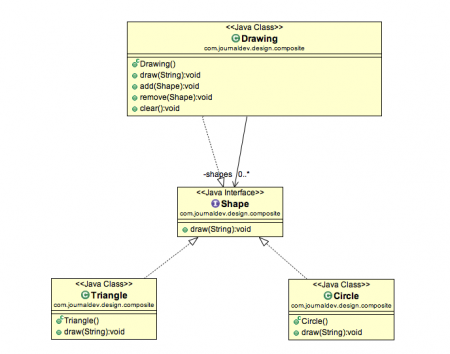
**목적:** 컴포지트, 브리지 패턴

**컴포지트 패턴:**

문제 #1: 컴포지트 패턴에서 Leaf 객체와 Composite 객체가 공유하고 있는 인터페이스가 인터페이스가 아닌 추상 클래스로 표현하는 이유는 무엇인가?

Leaf와 Composite객체 모두가 가지고 있는 attribute가 있을 수 있기 때문이다. file, directory 구조를 예로 들어보자. file, directory는 둘다 이름을 나타내는 name과 차지용량을 나타내는size를 가질 수 있다. 바꿔 말해서 name, size는 각 객체가 가지는 인스턴스 변수로써 이는 추상클래스에선 선언이 가능하지만, 인터페이스에선 불가능하다(인터페이스는 static final로만 변수 선언가능). 또한, file과 directory를 마우스 우클릭으로 속성을 보고자한다면 이는 공통적인 기능으로 추상이 아닌 메소드가 될 수 있다. 즉, 같은 기능을 하는 메소드를 가지고 있다면 추상이 아닌 메소드로 선언할 수 있으며 이는 추상클래스에서만 가능하다. 요약하자면, Leaf객체와 Composite 객체가 동일 속성과 동일 기능을 가진다면 인터페이스보단 추상클래스가 더 효율적이므로 추상클래스로 표현하는 것이 더 낫다.

문제 #2: 다음 그림의 Leaf클래스인 Trangle, Circle 클래스와 Composite 클래스인 Drawing 클래스를 구현하라. 다음에 제시된 클라이언트가 실행될 수 있도록 코딩하라.



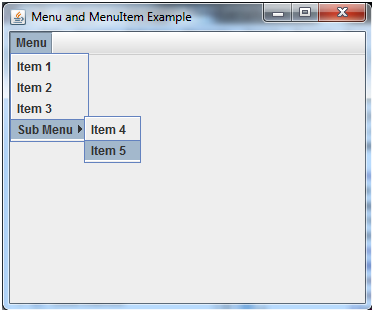
<TestCompositePattern.java>

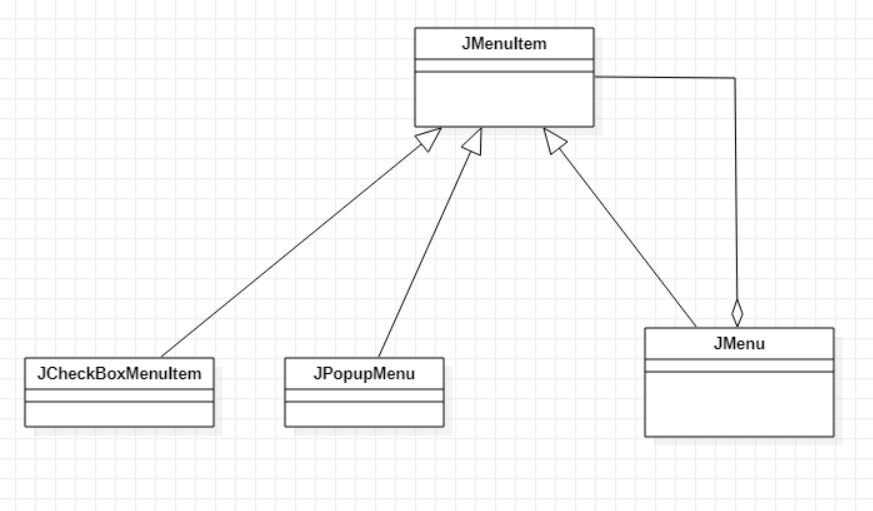
|  |
| --- |
| package com.journaldev.design.test;  import com.journaldev.design.composite.Circle;  import com.journaldev.design.composite.Drawing;  import com.journaldev.design.composite.Shape;  import com.journaldev.design.composite.Triangle;    public class TestCompositePattern {      public static void main(String[] args) {          Shape tri = new Triangle();          Shape tri1 = new Triangle();          Shape cir = new Circle();            Drawing drawing = new Drawing();          drawing.add(tri1);          drawing.add(tri1);          drawing.add(cir);            drawing.draw("Red");            drawing.clear();            drawing.add(tri);          drawing.add(cir);          drawing.draw("Green");      }  }  <출력 결과>  Drawing Triangle with color Red  Drawing Triangle with color Red  Drawing Circle with color Red  Clearing all the shapes from drawing  Drawing Triangle with color Green  Drawing Circle with color Green |

|  |  |
| --- | --- |
|  |  |

문제 3. javax.swing 패키지의 MenuElement 인터페이스와 JMenu 및 JMenuItem 클래스를 공부해보라. 이러한 클래스가 컴포지트 디자인 패턴의 예가 되는가? 예가 된다면 왜 그런지 설명하라. 클래스 간의 관계를 보여주는 클래스 다이어그램을 그려라.

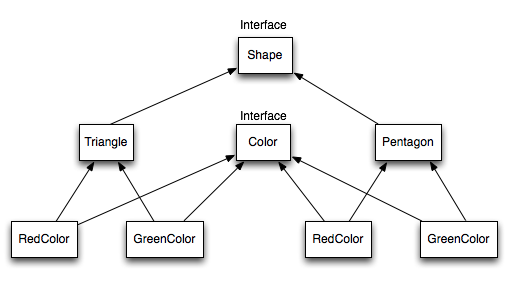
JMenu는 JMenuItem을 base로 하여 만든 sub class이다. JMenu클래스 정의에선 JMenuItem을 인스턴스변수로 가지고 있다. 따라서 JMenu와 JMenuItem는 컴포지트 디자인 패턴의 예가 된다. 아래와 같이 JMenu 안의 구성요소로 또 다시 JMenu가 들어간 경우를 쉽게 상상할 수 있다.





**브리지 패턴**

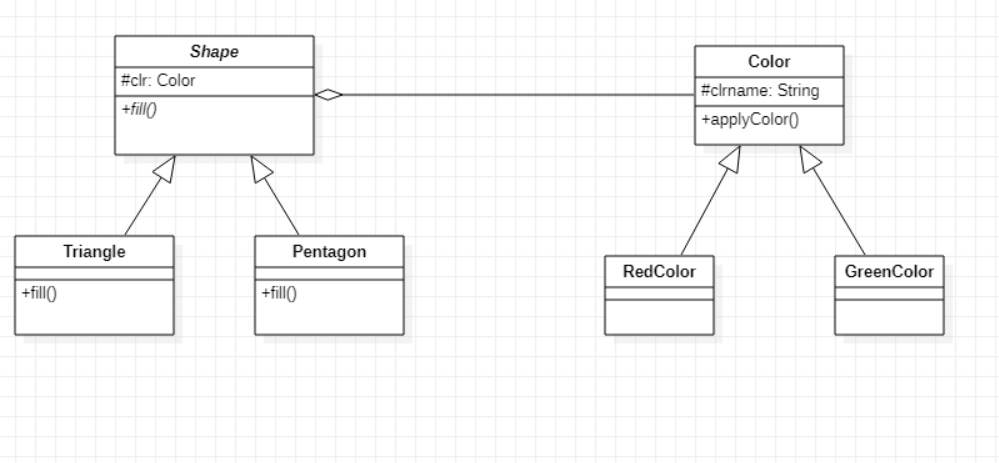
문제 #4: Triangle, Pentagon 클래스의 컬러를 다양하게 하기 위하여 다음과 같은 상속을 사용하였다.



(1) 위와 같은 구조로 설계하였을 때 어떤 문제점이 있는가?

Shape에 Rectangle, Color에 BlueColor가 추가된다면 총 9개의 Color 객체가 필요하게 됨(Triangle – Red, Green, Blue, Pentagon-Red,Green,Blue, Rectangle-Red,Green,Blue). 만약 Shape종류와 Color 종류가 늘어나면 그만큼의 조합수가 늘어나게된다. 즉, Shape 종류가 M개, Color종류가 N개라 하면 필요한 Color객체는 M \* N개가 필요하게 된다. 따라서 변경에 열려있게 된다.

(2) 위의 설계를 브리지 패턴을 적용하여 리팩토링하라(단 Shape에는 Color를 지정하는 생성자와 컬러를 적용하여 인쇄하는 applyColor() 메소드가 있어야 함).



(3) 다음 테스트 프로그램으로 아래와 같은 실행 결과가 나오도록 (2)의 설계를 코딩하라.

public class Pentagon extends Shape{

public Pentagon(Color c) {

super(c);

}

@Override

public void applyColor() {

System.out.print("Pentagon filled with color ");

color.applyColor();

}

}

<출력 결과>

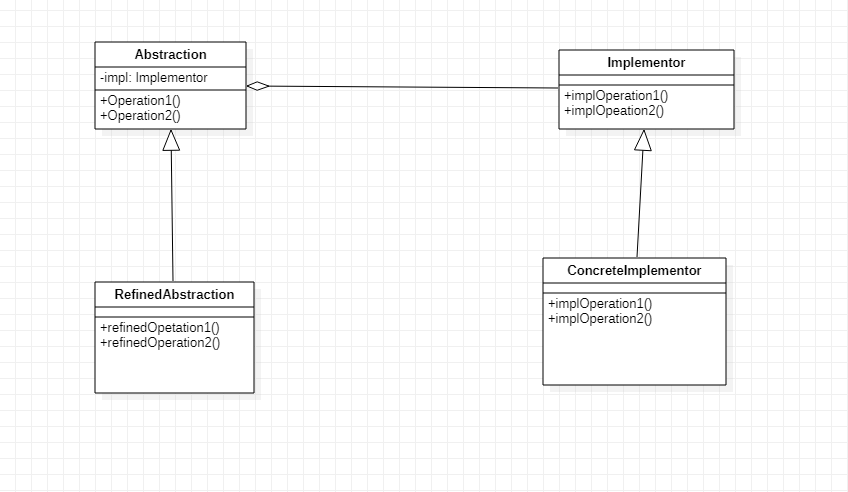
Triangle filled with color red.

Pentagon filled with color green.

문제 5.

(1) 브리지 패턴은 어떤 때 사용하는지 그 의도를 설명하고 UML로 그린 후 패턴의 구성요소를 설명하라.

패턴 자체 다이어그램을 그릴 것.



Abstraction : 기능 계층의 최상위 클래스. 구현 부분에 해당하는 클래스의 인스턴스를 가지고 해당 인스턴스를 통해 구현부분의 메소드를 호출

RefinedAbstraction : Abstraction에 의해 정의된 인터페이스를 확장. 기능 계층에서 새로운 부분을 확장한 클래스

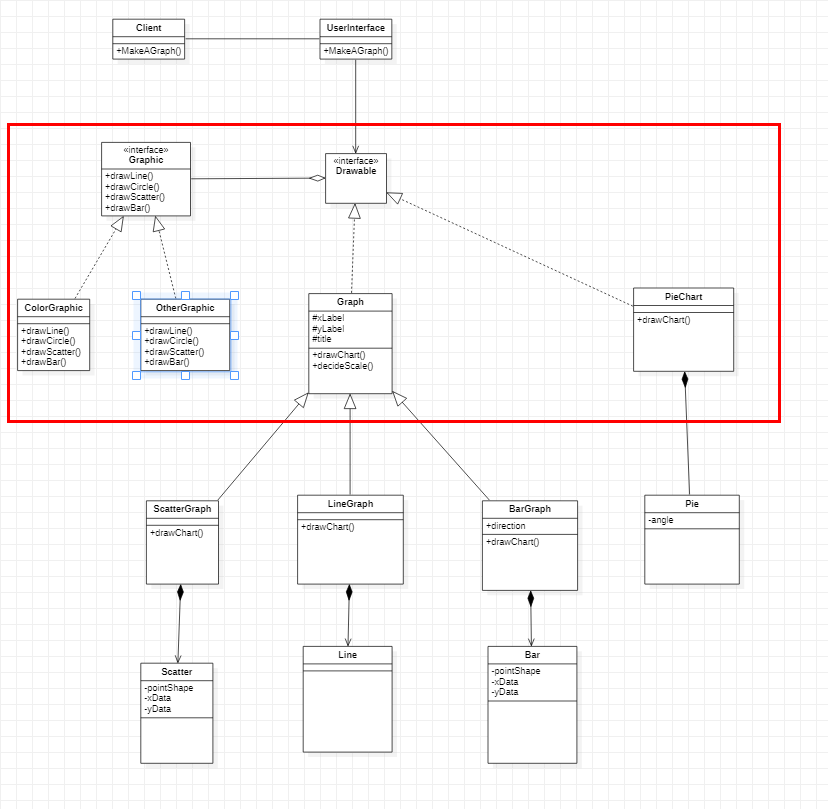
Implementor : 구현 클래스를 위한 인터페이스를 정의. Abstraction의 기능을 구현하기 위한 인터페이스를 정의

ConcreteImplementor : Implementor 인터페이스를 구현. 실제 기능을 구현한다

(2) (10 점) 실습에서 다룬 MakeAGraph 응용 문제에 대하여 브리지 패턴을 적용하여 설계하라. 단 UML로 대략적인 클래스와 속성, 오퍼레이션 및 관계를 표현하라.

“주어진 자료값을 이용하여 여러 가지 다양한 그래프를 그려주는 프로그램 MakeAGraph 를 작성해 줄 것을 요구 받았다고 하자. 예를 들어 볼링 점수가 시간이 흐르면서 어떻게 향상되었는지 막대그래프나 라인 그래프로 그려 보고 싶다. 또는 최근에 눈 온 날, 비 온 날, 갠 날을 막대 그래프로 그리거나 내가 쓴 돈이 어디에 썼는지 파이 차트로 그리고 싶을 것이다.

MakeAGraph 의 초기 버전은 클라이언트가 두 개의 단순 모드 즉 scatter plot mode (점을 찍는 산포도)와 bar graph mode(막대 그래프)를 생성한다. Java 애플리케이션으로 만들어 콘솔에서 실행될 수 있어야 한다(즉 텍스트 모드에서 보여지고 GUI 모드는 필요하지 않다).”



빨간색 부분에서 브릿지 패턴을 적용하여 설계하였다. 개념적인 부분은 Drawable로 추상화했고 구체적인 구현은 Graphic 인터페이스를 구현해서 이루어지도록 하였다.

**제출 방법**

1. 문제 1, 2, 3은 word 파일에 답을 적어 파일을 제출한다.
2. 문제 4는 주어진 클라이언트를 사용하여 실행될 수 있도록 클래스를 생성, 코딩한다.
3. 원시코드의 헤더 커멘트에 반드시 자신의 이름을 표시하고 간단히 설명하여야 한다. e-Class에 모든 원시코드를 업로드 한다. 파일 이름은 학번과 이름을 넣을 것.