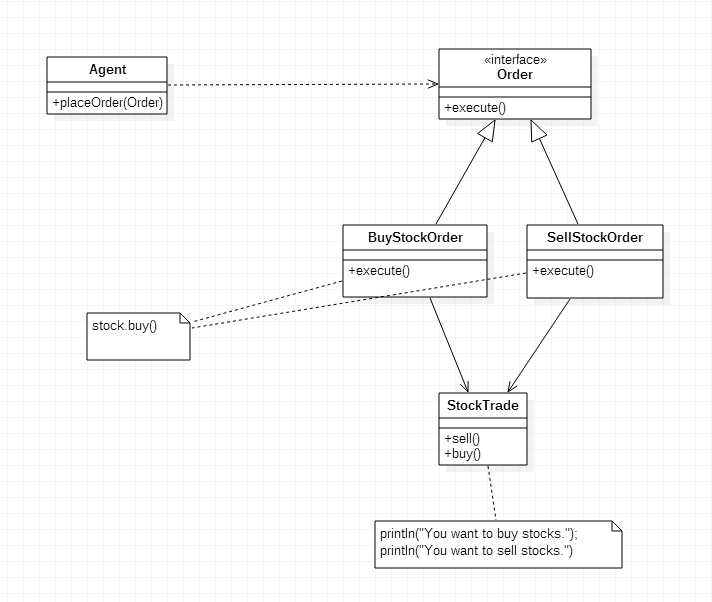
*객체지향 설계와 패턴*  
**LAB ASSIGNMENT #11**

## Weight 10% 마감 6월 4일(금)

**목적: 커맨드 패턴, 데코레이터, 반복자**

문제 #1: 커맨드 패턴

다음은 주식을 거래하기 위한 프로그램의 클래스 설계이다.



(1) 위 설계를 만족하는 각 클래스를 Java 언어로 코딩하라.

(2) 다음 클라이언트 프로그램으로 실행시켜 보고 결과를 캡처하라.

// Client

public class Client {

public static void main(String[] args) {

StockTrade stock = new StockTrade();

BuyStockOrder buyStockOrder = new BuyStockOrder(stock);

SellStockOrder sellStockOrder = new SellStockOrder(stock);

Agent agent = new Agent();

agent.placeOrder(buyStockOrder ); // Buy Shares

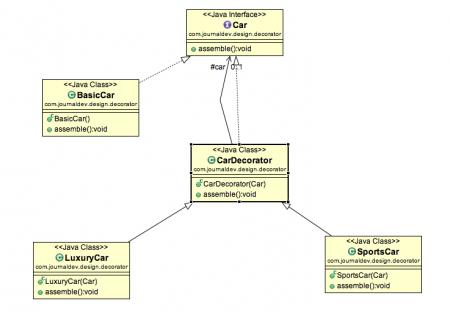
agent.placeOrder(sellStockOrder ); // Sell Shares

}

}

문제 2: 데코레이터 패턴

여러 종류의 자동차를 구현하고 싶다. Basic car에 장식으로 Sport car 또는 Luxury car를 확장할 수 있게 다음과 같이 데코레이터 패턴으로 설계하였다.



위 설계를 코딩하고 다음과 같은 프로그램으로 테스팅하라. 제시된 출력 결과가 나와야 한다.

package com.journaldev.design.test;

import com.journaldev.design.decorator.BasicCar;

import com.journaldev.design.decorator.Car;

import com.journaldev.design.decorator.LuxuryCar;

import com.journaldev.design.decorator.SportsCar;

public class DecoratorPatternTest {

public static void main(String[] args) {

Car sportsCar = new SportsCar(new BasicCar());

sportsCar.assemble();

System.out.println("\n\*\*\*\*\*");

Car sportsLuxuryCar = new SportsCar(new LuxuryCar(new BasicCar())); sportsLuxuryCar.assemble();

}

}

출력 결과:

Basic Car. Adding features of Sports Car.

\*\*\*\*\*

Basic Car. Adding features of Luxury Car. Adding features of Sports Car.

문제 3: 반복자 패턴

다음은 Topic 자료에 대하여 반복자 패턴을 적용하기 위한 코드의 일부이다.

public class Topic

{

    private String name;

    public Topic(String name) {

        super();

        this.name = name;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

}

public interface Iterator<E>

{

void reset(); // reset to the first element

E next(); // To get the next element

E currentItem(); // To retrieve the current element

boolean hasNext(); // To check whether there is any next element or not.

}

public class Main

{

    public static void main(String[] args)

    {

        Topic[] topics = new Topic[5];

        topics[0] = new Topic("topic1");

        topics[1] = new Topic("topic2");

        topics[2] = new Topic("topic3");

        topics[3] = new Topic("topic4");

        topics[4] = new Topic("topic5");

        List<Topic> list = new TopicList(topics);

        Iterator<Topic> iterator = list.iterator();

        while(iterator.hasNext()) {

            Topic currentTopic = iterator.next();

            System.out.println(currentTopic.getName());

        }

    }

}

(1) 위의 메인 프로그램이 수행되기 위하여 필요한 클래스를 추가하여 설계하고 UML 클래스 다이어그램으로 나타내라.

(2) 코드가 제시되지 않은 클래스에 대하여 구현하고 실행하라.

(3) 실행 결과를 캡처하라.

**문제 4: 데코레이터 패턴**

어느 채식 전문 식당에서 메뉴를 주문하는 프로그램을 개발하려고 한다. 기본적으로 채식류(Vegetarian Food)을 주문할 수 있고 Non-Vegetarian 메뉴를 주문하면 채식에 구운 치킨과 치킨 커리(Roasted Chicken and Chicken Curry)가 장식된다. 또한 중국 음식을 주문하면 채식 음식에 볶음밥(Fried Rice)과 야채튀김(Manchurian)이 나온다.

(1) DecoratorPatternCustomer 라는 메인 프로그램이 메뉴를 준비한다고 할 때 Food, VegeFood와 데코레이션을 위한 Food의 클래스를 설계하여 클래스 다이어그램으로 그려라.

(2) 아래와 같은 실행 결과가 나올 수 있게 위의 설계를 코딩하라.

========= Food Menu ============

1. Vegetarian Food.

2. Non-Vegetarian Food.

3. Chinese Food.

4. Exit

Enter your choice: 1

Veg Food

50.0

========= Food Menu ============

1. Vegetarian Food.

2. Non-Vegetarian Food.

3. Chinese Food.

4. Exit

Enter your choice: 2

Veg Food With Roasted Chicken and Chicken Curry

200.0

========= Food Menu ============

1. Vegetarian Food.

2. Non-Vegetarian Food.

3. Chinese Food.

4. Exit

Enter your choice: 3

Veg Food With Fried Rice and Manchurian

**제출물**

제출하여야 할 것:

1. UML로 표현한 설계와 파악한 패턴과 어디에 적용되었는지 표시하고 어떤 효과가 있었는지 기술하라.

2. 구현한 결과:

\* 원시코드

\* 실행 한 후의 결과 화면

**평가**

과제 #11다음과 같은 기준으로 평가할 것임:

1. 패턴 마이닝의 정확성

2. 구현의 정확성

3. 설계, 구현 결과물에서 볼 수 있는 설계에 대한 이해 수준