*객체지향 설계와 패턴*  
**LAB ASSIGNMENT #5**

## Weight 10% 마감 4월 12일(월)

**목적:** 인터페이스, 어댑터 패턴, 퍼싸드 패턴

**인터페이스:**

문제 #1: Java 언어에서 추상 클래스와 인터페이스의 차이점 세 가지를 적어라.

1. 추상클래스는 클래스로의 다중상속이 불가능하지만 인터페이스는 클래스로의 다중상속이 가능하다.
2. 추상클래스는 추상이 아닌 메소드를 가질 수 있지만 인터페이스는 오직 추상메소드만 가질 수 있다.
3. 추상클래스는 추상클래스 내에 생성자를 정의할 수 있지만, 인터페이스는 생성자를 정의할 수 없다.

문제 #2: 다음 RocketSim 인터페이스에 대한 설명 중 옳은 것은 ‘True’ 옳지 않는 것은 ‘False’로 표시하라.

package com.oozonoiz.simulation;

import com.ooozinoz.units;

interface RocketSim

{

abstract Length apogee();

public Force thrust();

}

(1) RocketSim 인터페이스에서 apogee()만 명시적으로 abstract라고 선언되어 있지만 두 메소드 모두 abstract이다. ( True )

(2) RocketSim 인터페이스에서 thrust()만 명시적으로 public라고 선언되어 있지만 두 메소드 모두 public이다. ( True )

(3) 모든 인터페이스는 public이다. 따라서 RocketSim도 명시적으로 public이라고 선언되어 있지는 않지만 public이다. ( False )

(4) RocketSim 인터페이스를 상속하여 RocketSimSolid라는 인터페이스를 생성할 수 있다. ( True )

(5) 모든 인터페이스는 적어도 하나의 메소드는 반드시 가지고 있어야 한다. ( False )

(6) 인터페이스는 구현 클래스에서 반드시 선언해야 하는 인스턴스 필드를 선언할 수 있다. ( False )

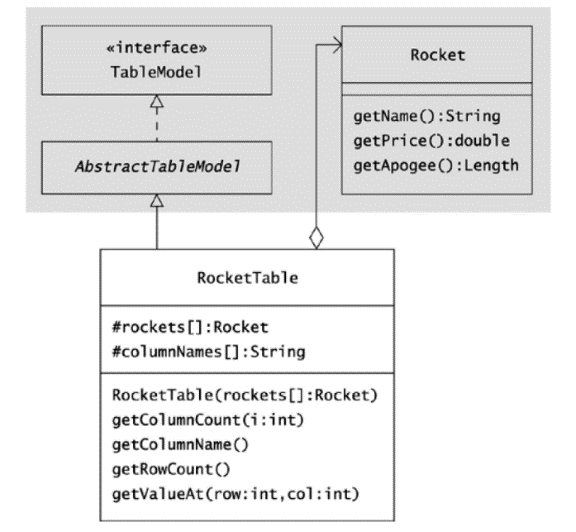
(7) 인터페이스를 인스턴스화 할 수는 없지만, 인터페이스 정의에 생성자 메소드를 선언하여, 구현 클래스에서 이와 동일한 시그네처를 지닌 생성자를 제공하도록 할 수 있다. ( False )

**어댑터 패턴**

문제 #3: Rocket이 가진 정보를 Swing이 가진 TableModel을 이용하여 그림 1과 같은 테이블로 디스플레이하고 싶다. 그러면 다음 설계와 같이RocketTable 이라는 어댑터를 만들어 Rocket 클래스를 TableModel 인터페이스에 맞게 조절하면 된다.



그림 1. JTable을 이용하여 로케트 관련 정보를 디스플레이 한 사례



Rocket 객체의 배열이 TableModel 역할을 할 수 있도록 조정해 줄 수 있게 RocketTable 메소드의 코드를 채워라.

package com.oozinoz.applications;  
import javax.swing.table.\*;  
  
public class RocketTable extends AbstractTableModel {  
 protected Rocket[] rockets;  
 protected String[] columnNames =  
 new String[] { "Name", "Price", "Apogee" };  
 public RocketTable(Rocket[] rockets) {  
 this.rockets = rockets;  
 }  
 public int getColumnCount() {  
 return columnNames.length;  
 }  
 public String getColumnName(int i) {  
 return columnNames[i];  
 }  
 public int getRowCount() {  
 return rockets.length;  
 }  
 public Object getValueAt(int row, int col) {  
 if(columnNames[col] == "Name")  
 return rockets[row].getName();  
 else if (columnNames[col] == "Price")  
 return rockets[row].getPrice();  
 else if(columnNames[col] == "Apogee")  
 return rockets[row].getApogee();  
 else  
 return null;  
 }  
}

퍼싸드 패턴

문제 #4: 다음의 코드<Simpleformat.java>를 읽고 이해한 후 프로그램을 컴파일하여 실행해 보라.

import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.GregorianCalendar;  
import java.text.DecimalFormat;  
import java.text.MessageFormat;  
class SampleFormats {  
 public static void main(String[] args) {  
 /\* Date format. Examples:  
 4 Jul 1997  
 10 Mar 1979 \*/  
 SimpleDateFormat dateFormat = new SimpleDateFormat("d MMM yyyy");  
 /\* A pretty integer format. Examples:  
 4,235,682  
 5,487  
 42  
 \*/  
 DecimalFormat intFormat = new DecimalFormat("#,###");  
 /\* A monetary format. Examples:  
 $ 4,235,670  
 $ 345.23  
 $ 0.48  
 \*/  
 DecimalFormat dollarFormat = new DecimalFormat("$ #,##0.##");  
/\* A person format using first, last, and user names. Examples:  
 Weinman, Jerod [weinman]  
 Davis, Janet [davisjan]  
 Coahran, Marge [coahranm]  
 The {x} syntax means replace with the x'th index of an array.  
 \*/  
 MessageFormat personFormat = new MessageFormat("{0}, {1} [{2}]");  
  
 /\* Tests \*/  
 System.*out*.println("Date: " + dateFormat.format ( new GregorianCalendar(2005, 7,4).getTime()));  
   
  
 // dateFormat.format() expects a Date object. Use  
 // GregorianCalendar to construct a Calendar object and the  
 // getTime() method to convert it to a Date.  
 System.*out*.println("Integer: " + intFormat.format(1234567) );  
 System.*out*.println("Integer: " + intFormat.format(67.7) );  
 System.*out*.println("Money: " + dollarFormat.format(9812345.678) );  
 System.*out*.println("Money: " + dollarFormat.format(9812345) );  
 System.*out*.println("Money: " + dollarFormat.format(.78) );  
 System.*out*.println("Person: " + personFormat.format  
 ( new Object[]{"Weinman","Jerod","weinman"} ) );  
 // We can create and initialize an Object array on the fly, as  
 // shown above.  
 }  
}

여러 가지 클래스의 복잡한 옵션 대신에 간단한 인터페이스로 사용할 수 있도록 퍼싸드 패턴을 적용하여 다음과 같은 <basicformat.java> 프로그램의 Body를 완성하고 주어진 main을 이용하여 실행하라.

import java.text.SimpleDateFormat;  
 import java.util.Date;  
 import java.util.GregorianCalendar;  
 import java.text.DecimalFormat;  
 import java.text.MessageFormat;  
*/\*\* Specifies basic formatting for dates, integers, money, and people \*/*public class BasicFormat {  
 public static String formatDate( Date date ) {  
 // BODY  
 }  
 */\*\* Formats a date  
 \** ***@param*** *year Full integral value of the year, e.g. 1945, 2008  
 \** ***@param*** *month Zero-based month index, e.g. 0 is January, 6 is July  
 \** ***@param*** *day One-based day of the month  
 \*/* public static String formatDate( int year, int month, int day ) {  
 // BODY  
 }  
 public static String formatLong ( long number ) {  
 // BODY  
 }  
 public static String formatMoney ( double amount ) {  
 // BODY  
 }  
 public static String formatPerson ( String firstName, String lastName,  
 String userName) {  
 // BODY  
 }  
}

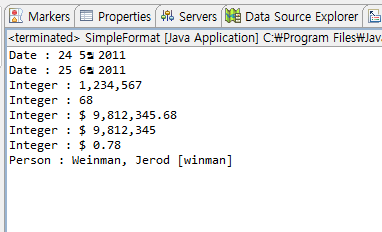
****

그림 2 실행 화면 사례

**제출 방법**

1. 문제 1, 2는 word 파일에 답을 적어 파일을 제출한다.
2. 문제 3는 주어진 파일에 메소드 body 완성 후 실행하고 문제 4는 주어진 클라이언트를 사용하라.
3. 원시코드의 헤더 커멘트에 반드시 자신의 이름을 표시하고 간단히 설명하여야 한다. e-Class에 모든 원시코드를 업로드 한다. 파일 이름은 학번과 이름을 넣을 것.