객체지향프로그래밍및실습 – 9주차 실습활동지(10월30일)

성명: _____ 학과: _____ 학번: _____ 실습반: _____

제출방식: 보고서를 pdf 형식으로 제출한다(report9.pdf).

제출기한: 10월 31일(금) 23시59분까지 (지연 제출 0점 처리함, 미완성이라도 반드시 제출하길 바람)

I. Objectives

1. Abstract class와 inheritance, polymorphism에 대한 관계를 이해한다.
2. Interface와 polymorphism의 관계를 이해하고 활용할 수 있다.

II. Exercises (15점)

1. 자동차 통행료를 계산하는 코드를 작성하고자 한다. 아래 요구사항을 읽고 물음에 답하시오.

- Car 클래스와 Bus 클래스의 공통 superclass로 Vehicle 클래스를 추가할 것
- 모든 차량의 거리 요율(distance rate)는 동일한 값을 가짐.

1) 승용차(Car) 통행료 계산 공식
   - 통행료(toll) = 통행 거리 * 거리 요율(distance rate) * 배기량 요율(volume rate)
   - 배기량 요율 = 차량 배기량(volume) ÷ 기준 배기량 (base volume constant)

2) 버스(Bus) 통행료 계산 공식
   - 통행료(toll) = 통행 거리(distance) * 거리 요율(distance rate)) * 승객 요율(passenger rate)
   - 승객 요율 = 승차한 승객수 ÷ 기준 승객수 (base passenger constant)

```
public class Car
{
    public static final double BASE_VOLUME = 2000.0;  // 기준 배기량
    private static int distanceRate = 50;              // 거리 요율

    private String no;
    private int departure;
    private int destination;
    private int volume;                                // 차량 배기량

    public Car(String no, int departure, int destination, int volume)
    {
        this.no = no;
        this.departure = departure;
```

```
        this.destination = destination;
        this.volume = volume;
    }
    public int calcTollFee()                              // 통행료 계산
    {
         double volumeRate = volume / BASE_VOLUME;
         int distance = getDestination() - getDeparture();
         int toll = (int) ( distance * distanceRate * volumeRate );
         return toll;
    }
    public String getNo() { return no; }
    public int getDeparture() { return departure; }
    public int getDestination() { return destination; }
    public int getVolume() { return volume; }

    public static void setDistanceRate(int d) { distanceRate = d; }
    public static int getDistanceRate() { return distanceRate; }
}

public class Bus
{
    public static final double BASE_PASSENGER = 20.0;  // 기준 승객수
    private static int distanceRate=50;                // 거리 요율

    private String no;
    private int departure;
    private int destination;

    private int passengers;                        // 승객수

    public Bus(String no, int departure, int destination, int passengers)
    {
        this.no = no;
        this.departure = departure;
        this.destination = destination;
        this.passengers = passengers;
    }

      public int calcTollFee()                                    // 통행료 계산
      {
         double passengerRate = passengers / BASE_PASSENGER;
         int distance = destination - departure;
         int toll = (int) (distance * distanceRate * passengerRate);
         return toll;
```

```
    }
    public String getNo() { return no; }
    public int getDeparture() { return departure; }
    public int getDestination() { return destination; }
    public int getPassengers() { return passengers; }
    public static void setDistanceRate(int d) { distanceRate = d; }
    public static int getDistanceRate() { return distanceRate; }
 }

// VehicleTest.java
public class VehicleTest
{
   public static void main(String[] args)
   {
     Vehicle[] vs = new Vehicle[3];
     vs[0] = new Car("c1111", 0, 200, 3000);   // 3000은 volume
     vs[1] = new Car("c2222", 100, 300, 2000); // 2000은 volume
     vs[2] = new Bus("b3333", 200, 400, 40);   // 40은 승객수

     for(Vehicle v: vs)
     {
        System.out.printf("%s : %d%n", v.getNo(), v.calcTollFee());
     }
   }
}
```

가) Car 및 Bus 클래스의 공통 superclass인 Vehicle 클래스를 정의하고자 한다. 아래 Vehicle 클래스를 채우고 Car 및 Bus 클래스에 코드를 추가하거나, 불필요한 부분을 삭제, 또는 수정하여 완성하시오. 단, 추가한 부분은 밑줄을 치고, 삭제한 부분은 취소선을 그을 것. 수정할 문장은 취소선을 그은 후 추가할 것. [전체 코드] [5점]

- Car 및 Bus의 공통적인 부분들을 Vehicle 클래스로 옮길 것.
- Car 및 Bus 클래스의 메소드를 수정할 것.

```
public abstract class Vehicle
{
   // ...  variables ...


   // ... constructor ...


     // ... 메소드 ...
```

```
}
```

나) VehicleTest의 실행 결과는? [실행결과][1점]

다) VehicleTest에서 polymorphism이 어떻게 활용되고 있는지 설명하고 polymorphism의 장점을 Vehicle 예제를 들어 설명하시오. [실행결과][2점]

2. 1번 문제에서 작성한 Car, Bus, Vehicle 클래스와 아래 VehicleSortTest코드를 이용하여 아래 물음에 답하시오.

```
// VehicleSortTest.java
import java.util.*;
public class VehicleSortTest
{    public static void main(String[] args)
    {
      Vehicle[] vs = new Vehicle[3];
      vs[0] = new Car("c1111", 0, 200, 3000);   // 3000은 volume
      vs[1] = new Car("c2222", 100, 300, 2000); // 2000은 volume
      vs[2] = new Bus("b3333", 200, 400, 40);   // 40은 승객수

      Arrays.sort(vs);  // (A)

      for(Vehicle v: vs)
      {
         System.out.printf("%s : %d%n", v.getNo(), v.calcTollFee());
      }
    }
}
```

가) Vehicle 객체를 번호(no)순으로 정렬하도록 Vehicle class를 수정해보시오. VehicleSortTest class는 그대로 이용한다. [수정된 메소드, 실행결과] [1점]

나) (가)번에서 수정한 Vehicle 클래스를 그댈로 이용하면서 toll fee순으로 정렬할 수 있는 Comparator class (TollComparator)를 구현하시오. VehicleSortTest.java 파일의 (A) 문장을 아래 문장으로 변경하여 테스트하시오. [수정된 클래스, 실행결과] [2점]

```
      Arrays.sort(vs, new TollComparator());  //(A)
```

3. Payable interface 예제(첨부파일: Payable.java, Employee.java, SalariedEmployee.java, Invoice.java, PayableTest.java)를 실행해 본 후 다음 물음에 답하시오.

가) 대출금(loan)도 지불해야 할(payable) 대상으로 추가하고자 한다. 아래 조건과 코드를 고려하여 Loan class를 적절하게 정의하시오. [Loan 클래스, 실행결과][3점]

- 대출금에 대한 지불금액(payment amount)은 대출원금(principal)과 이자(interest)를 합한 금액임
- 이자를 설정할 수 있음(setInterest 메소드)
- Loan 객체 정보의 출력은 아래와 같이 하면 됨

  loan: 대출명
  principal: 1000.00
  interest: 100.00 // 초기값은 0

```
// in the main method of PayableTest class
...
Payable[] payableObjects = new Payable[5];
...
payableObjects[4] = new Loan("청년도약", 1000.0);  //청년도약은 대출명이며,
                                                   // 1000.0은 대출원금을 나타낸다.
Loan loan = (Loan)payableObjects[4];
loan.setInterest(100.0);                           // 이자 100.0을 설정
...
```