KNU 강원대학교
KANGWON NATIONAL UNIVERSITY

자바프로그래밍2

7주차 실습

인터페이스, 람다식, 패키지

2024-2학기 자바프로그래밍2 3분반
TA 강지호

실습 평가 방법

- 점수 : 100점 만점 기준(문제 별 난이도에 따라 점수 부여)
- 채점 기준 : 완성도, 작동 유무, 일부 오류 등에 따라 감점
  - 프로그램이 동작하지 않거나, 코드 공유, ChatGpt 사용 등의 부정행위 적발 시 0점
  - 소스 코드에 허점, 잘못된 들여쓰기, 일부 입출력 오작동 시 정도에 따라 감점
- 제출 기한 : 실습 당일 23시 59분까지(이후 제출 불가능)
  - 실습 시간(14:00-15:50) 내 제출 시 감점X
  - 18:00 까지 제출 시 채점 점수의 5% 감점
  - 20:00 까지 제출 시 채점 점수의 10% 감점
  - 23:59 까지 제출 시 채점 점수의 20% 감점

실습 제출 방법

- 압축 파일명 : n주차_학번_이름.zip
- 소스 파일 : Eclipse에서 Export한 zip 파일 내 소스 파일(.java)
- 보고서 : 각 문제별 문제 번호 및 소스 코드 실행 결과 화면 캡처한 **pdf 파일** - 부재 시 감점
- 소스 파일과 보고서를 압축하여 주차별로 위 압축 파일명과 같이 e-루리에 제출
- e-루리 접속 오류 등 특별한 사유로 인해 제출하지 못하는 경우
  - rkdwlgh01@naver.com 해당 e-mail을 통해 제출

실습 조교 및 질의응답

- e-루리 Q&A 게시판 활용 (작성 후 e-루리 메시지 시 빠른 응답 가능)
- 실습 TA의 e-mail 활용
  - 강지호 : rkdwlgh01@naver.com

[자바프로그래밍2]
7주차 실습 문제

제출 기한 10.17(목) 23:59 전까지

문제 1 (30점)

- 클래스 **Product**를 아래의 조건에 맞게 구현하고 **Exercise01**에서 테스트하세요.

Product
- name : String - price : int - rating : double
+ Product(name: String, price: int, rating: double) + compareTo(other: Object) : int + toString() : String

Product 클래스 (**Comparable 인터페이스 사용**)

- 필드는 외부에서 접근할 수 없도록 설정

- **생성자** 구현
- **compareTo()** 메소드를 가격과 평점을 기준으로 객체를 비교하도록 **오버라이딩**
- toString() 메소드를 **오버라이딩**하여 객체의 상태를 아래 형식으로 출력
  - " 상품명: name, 가격: price, 평점: rating"

문제 1 (30점)

- **Comparable 인터페이스**는 표준 자바 라이브러리에 아래와 같이 정의되어 있으므로 따로 정의할 필요 없음

```
public interface Comparable {          // 실제로는 제네릭을 사용해서 정의된다.
    int compareTo(Object other);       // -1, 0, 1 반환
}
```

- **compareTo()** 메소드 추가 설명

  Product 객체를 가격을 기준으로 비교하고 가격이 동일하면 평점을 기준으로 비교(각각 아래 형식으로 비교)

  - Product 객체가 비교할 대상보다 가격이 크면 1, 작으면 -1을 반환 (오름차순 정렬)
  - Product 객체가 비교할 대상보다 평점이 크면 -1, 작으면 1, 동일하면 0을 반환 (내림차순 정렬)

문제 1 (30점)

- 클래스 명 : **Product, Exercise01**

Product.java
Exercise01.java

```java
import java.util.Arrays;

public class Exercise01 {

    public static void main(String[] args) {
        Product p1 = new Product("Computer", 300, 4.0);
        Product p2 = new Product("Computer", 300, 4.5);
        Product p3 = new Product("Smartphone", 100, 4.7);

        //compareTo() 테스트
        System.out.println(p1.compareTo(p3));
        System.out.println(p1.compareTo(p2));
        System.out.println(p2.compareTo(p3));

        // 상품 배열 생성 및 정렬
        Product[] products = {p1, p2, p3};
        Arrays.sort(products);

        for (Product product : products) {
            System.out.println(product);
        }

    }

}
```

실행결과 예시 :

```
1
1
1
상품명: Smartphone, 가격: 100, 평점: 4.7
상품명: Computer, 가격: 300, 평점: 4.5
상품명: Computer, 가격: 300, 평점: 4.0
```

문제 2 (35점)

- 인터페이스 **Delivery**와 인터페이스를 구현하는 클래스 **Walking, Bicycle, Motorbike**를 조건에 맞게 구현하고 **Exercise02**에서 테스트하세요.

<<interface>> **Delivery**
+ *MAX_DISTANCE* : int = 20 //최대 거리 + *MIN_DISTANCE* : int = 0 //최소 거리
+ calculateETA(distance: double) : double + startDelivery(distance: double) : void + calculateFuel(distance: double) : double + calculateCost(distance: double) : double + isDistanceValid(distance: double) : boolean

Delivery 인터페이스

- 상수 정의
- **추상 메소드** calculateETA(), startDelivery()
- **디폴트 메소드** calculateFuel(), calculateCost()
- **정적 메소드** isDistanceValid()

문제 2 (35점)

- 추상 메소드 (인터페이스에서 구현X)

  calculateETA()는 도착 예상 시간을 계산하여 반환하는 메소드

  - 예상 시간 = 거리 / 속도

  startDelivery()는 배달 거리가 유효한지 확인 후 각 정보를 출력하는 메소드

  - 유효하다면 배달 수단, 예상 시간(분), 배달 비용을 출력(각 시간과 비용은 정수 부분만(소수점 이하 0번째))
  - 유효하지 않다면 `유효 거리가 아닙니다. 해당 거리는 배달할 수 없습니다.` 출력

- 디폴트 메소드 (아래와 같이 각 메소드의 디폴트 구현)

  calculateFuel()는 연료 소모량을 계산하여 반환하는 메소드

  - 디폴트는 0을 반환하도록 구현 (도보와 자전거는 연료가 필요 없음)

  calculateCost()는 배달 비용을 계산하여 반환하는 메소드

  - 배달 비용 = (거리 * 5 + 연료 소모량 * 10) * 100

- 정적 메소드

  isDistanceValid()는 입력된 배달 거리가 유효 거리에 있는지 확인하는 메소드

  - 입력 거리가 최소 거리와 최대 거리 사이에 있는지 확인하여 유효 여부를 반환

문제 2 (35점)

Walking
- speed: double //속도(km/h)
추상 메소드 구현

Walking 클래스

- 필드는 외부에서 접근할 수 없도록 설정하고 5로 초기화
- 실행 결과 예시에 맞게 추상 매소드 구현

Bicycle
- speed: double //속도(km/h)
추상 메소드 구현

Bicycle 클래스

- 필드는 외부에서 접근할 수 없도록 설정하고 15로 초기화
- 실행 결과 예시에 맞게 추상 매소드 구현

Motorbike
- speed: double //속도(km/h) - fuelEfficiency: double //연료 효율성
추상 메소드 구현 디폴트 메소드 calculateFuel() 오버라이딩

Motorbike 클래스

- 필드는 외부에서 접근할 수 없도록 설정
- 속도는 40, 연료 효율성은 20으로 초기화
- 실행 결과 예시에 맞게 추상 매소드 구현
- calculateFuel() 오버라이딩 – **[연료 소모량 = 거리 / 연료 효율성]**

문제 2 (35점)

- 클래스 명 : **Delivery, Walking, Bicycle, Motorbike, Exercise02**

Delivery.java
Walking.java
Bicycle.java
Motorbike.java
Exercise02.java

```java
import java.util.Scanner;

public class Exercise02 {

    public static void main(String[] args) {
        Delivery walkingDelivery = new Walking();
        Delivery bicycleDelivery = new Bicycle();
        Delivery motorbikeDelivery = new Motorbike();

        Scanner input = new Scanner(System.in);
        System.out.print("거리를 입력하세요(단위: km): ");
        double distance = input.nextDouble();

        if(distance <= 2)
            walkingDelivery.startDelivery(distance);
        else if (distance <= 5)
            bicycleDelivery.startDelivery(distance);
        else
            motorbikeDelivery.startDelivery(distance);

        input.close();
    }

}
```

실행결과 예시 :

```
거리를 입력하세요(단위: km): 1.5
걸어서 배달을 시작합니다.
예상 시간: 18분
배달 비용: 750원
```

```
거리를 입력하세요(단위: km): 3.8
자전거로 배달을 시작합니다.
예상 시간: 15분
배달 비용: 1900원
```

```
거리를 입력하세요(단위: km): 15
오토바이로 배달을 시작합니다.
예상 시간: 23분
배달 비용: 8250원
```

```
거리를 입력하세요(단위: km): 20.5
유효 거리가 아닙니다. 해당 거리는 배달할 수 없습니다.
```

문제 3 (35점)

- 람다식을 활용하여 **Exercise03**에서 아래 연산을 구현하고 테스트하세요.
  - ➢ 필요한 인터페이스를 선택하여 람다식으로 아래 연산을 구현
    - **덧셈** : 두 숫자의 합을 반환
    - **제곱** : 한 숫자의 제곱을 반환
    - **평균** : 세 숫자의 평균을 반환
    - **짝수 검사** : 한 숫자가 짝수인지 검사
    - **max** : 두 숫자 중 큰 값 반환
    - **절댓값** : 한 숫자의 절댓값을 반환
    - **비트 반전** : 한 숫자의 비트를 반전
    - **비트 XOR 연산** : 두 숫자의 비트 XOR 연산
    - **거듭제곱** : 두 숫자를 받아 첫 번째 숫자를 두 번째 숫자만큼 거듭제곱하여 반환

- **클래스 명 : Exercise03**

문제 3 (35점)

- 클래스 명 : **Exercise03**

```java
@FunctionalInterface
interface Check {
    public boolean is(int a);
}

@FunctionalInterface
interface FuncA {
    public int calc(int a);
}

@FunctionalInterface
interface FuncAB {
    public int calc(int a, int b);
}

@FunctionalInterface
interface FuncABC {
    public double calc(int a, int b, int c);
}

public class Exercise03 {

    public static void main(String[] args) {
        // 덧셈 연산
        FuncAB add = (a, b) -> a + b;

        // 나머지 연산 구현

        // 덧셈 테스트
        System.out.println("덧셈 결과: " + add.calc(5, 6));

        // 나머지 연산 테스트

    }

}
```

실행결과 예시 :

```
덧셈 결과: 11
5의 제곱: 25
5, 10, 15의 평균: 10.0
5는 짝수인가요? false
6은 짝수인가요? true
5와 6의 최댓값: 6
-5의 절댓값: 5
5의 비트 반전: -6
5 XOR 8: 13
2의 10제곱: 1024
```