객체지향프로그래밍 10 주차

제출일	2023 년 11 월 14 일	학과	컴퓨터공학과
과목	객체지향프로그래밍 01	학번	2022112701
담당교수	한인 교수님	이름	안소희

□ 실습문제 1

□ 문제분석

- DiscountPolicy를 인터페이스로 만들고 이를 상속받은 FixedDiscountPolicy, PercentDiscountPolicy 두 클래스를 구현해야 한다. FixedDiscountPolicy에서 getPrice 메소드는 dDiscountPrice 만큼 할인된 값을 반환한다. PercentDiscountPolicy에서 getPrice 메소드는 dDiscountPrice %만큼 할인된 값을 반환한다.

□ 프로그램 설계 및 알고리즘

먼저 기존에 추상클래스였던 DiscountPolicy 를 인터페이스로 바꾸어 작성하기 위해 멤버 변수와 생성자를 없앴다. 또한 extends 가 아닌 인터페이스의 추상메소드를 구현한다는 의미의 implements 를 사용하여 상속받는다. 이 방식으로 인터페이스 DiscountPolicy 를 상속받은 FixedDiscountPolicy, PercentDiscountPolicy 클래스에서 추상메소드인 getPrice 메소드를 구현해준다. 이때 이것을 오버라이딩이라고 한다. 퍼센트 할인이 실수로 받을 수 있으므로 double 형을 사용하였고 돈을 원 단위로 나타낼 시 정수로 나타내어야하므로 int 로 형변환을 시켰다. 또한 100%초과 할인은 불가능하므로 100 초과 입력시 퍼센트할인을 선택하면 다시 입력받도록 하였다. 무한루프로 입력을 받기 때문에 가격에 음수를 입력하면 실행 중지하도록 하였다.

□ 소스코드 및 주석

(Discountpolicy)

```
package week10_1;

interface DiscountPolicy {//인터페이스 작성
   abstract double getPrice(double dBeforePrice);//할인 후 가격을 구하는
추상메소드
}
```

(FixedDiscountPolicy)

```
package week10_1;
```

```java
public class FixedDiscountPolicy implements DiscountPolicy {//인터페이스 DiscountPolicy 의 추상메소드를 구현하므로 implements 작
    private double dDiscountPrice;//할인값 멤버변수 선언

    public FixedDiscountPolicy(double dDiscountPrice) {//할인 값을 매개변수로 받는 생성자
        this.dDiscountPrice = dDiscountPrice;
    }

    @Override
    public double getPrice(double dBeforePrice) {//할인 전의 값을 매개변수로 받아 할인값을 뺀 값을 반환
       double dResult = dBeforePrice - dDiscountPrice;
         return dResult;
    }
}
```

(PercentDiscountPolicy)

```java
package week10_1;

public class PercentDiscountPolicy implements DiscountPolicy {
    private double dDiscountPercent;//할인 값을 멤버 변수로 선언

    public PercentDiscountPolicy(double dDiscountPercent) {//할인 값을 매개변수로 받는 생성자 작성
        this.dDiscountPercent = dDiscountPercent;
    }

    @Override
    public double getPrice(double dBeforePrice) {//이 전의 값을 매개 변수로 받아 할인 값을 100 으로 나눠 이 전의 가격에 곱해 뺄 값을 구한 후 할인 계산 한 값을 반

       double discountAmount, dResult;
       discountAmount = dBeforePrice * (dDiscountPercent / 100);
       dResult = dBeforePrice - discountAmount;
         return dResult;
    }
}
```

(DiscountEx)

```java
package week10_1;
import java.util.Scanner;
public class DiscountEx {

       public static void main(String[] args) {
```

```
            Scanner oInDev = new Scanner(System.in);
            while(true) {

                  System.out.println("상품의 가격을 입력하세요(종료시 음수 값
입력)");
                  int iPrice = oInDev.nextInt();
                  if(iPrice < 0) {//가격이 음수면 종료
                        break;
                  }
                  System.out.println("discountValue 설정 값을
입력하세요");//할인 값 입력받음
                  double dDiscount = oInDev.nextDouble();

                  System.out.println("할인 정책을 선택하세요(1. 고정 금액
할인, 2. 퍼센트 할인)");
                  int iChoice = oInDev.nextInt();
                  if(iChoice == 1) {
                        DiscountPolicy fixedPolicy = new
FixedDiscountPolicy(dDiscount);//업캐스팅
                    int discountedPrice1 =
(int)fixedPolicy.getPrice(iPrice);//상품의 본래 가격을 매개변수로 넣어 할인 후
가격을 얻음
                    System.out.println("고정 금액 할인 후 가격: " +
discountedPrice1);

                  }

                  else if(iChoice == 2){// PercentDiscountPolicy 를
이용한 할인 정책
                        if(dDiscount>100) {//100%를 초과할 때 에러
                              System.out.println("100%을 넘을 수
없습니다.");
                              continue;
                        }
                    DiscountPolicy percentPolicy = new
PercentDiscountPolicy(dDiscount);//업캐스팅
                    int discountedPrice2 =
(int)percentPolicy.getPrice(iPrice);//상품의 본래 가격을 매개변수로 넣어 할인
후 가격을 얻음
                    System.out.println("퍼센트 할인 후 가격: " +
discountedPrice2);
                  }
                  else {
                        System.out.println("번호를 잘못
입력하셨습니다");//1 또는 2 가 아닐 때
```

```
                    }
                }
                oInDev.close();

        }

}
```

□ 결과 및 결과 분석

결과 화면

```
상품의 가격을 입력하세요(종료시 음수 값 입력)
1000
discountValue 설정 값을 입력하세요
10
할인 정책을 선택하세요(1. 고정 금액 할인, 2. 퍼센트 할인)
1
고정 금액 할인 후 가격: 990
상품의 가격을 입력하세요(종료시 음수 값 입력)
1000
discountValue 설정 값을 입력하세요
10.2
할인 정책을 선택하세요(1. 고정 금액 할인, 2. 퍼센트 할인)
2
퍼센트 할인 후 가격: 898
상품의 가격을 입력하세요(종료시 음수 값 입력)
1000
discountValue 설정 값을 입력하세요
1000
할인 정책을 선택하세요(1. 고정 금액 할인, 2. 퍼센트 할인)
2
100%을 넘을 수 없습니다.
상품의 가격을 입력하세요(종료시 음수 값 입력)
-1
```

결과 분석

가격을 1000, 할인 값을 10 으로 입력한 뒤 1 을 선택하면 올바르게 할인이 되어 출력이 된 것을 볼 수 있고, 10.2 로 할인 값을 정한 뒤 퍼센트 할인으로 선택하면 실수 값으로 알맞게 계산한 뒤 정수로 결과를 나타내는 것을 볼 수 있다. 할인 값 1000 으로 입력하고 퍼센트 할인을 선택하면 에러 메세지가 뜨고 다시 가격 입력 문장이 뜨는 것을 볼 수 있다.

□ 소감

추상클래스와 인터페이스의 차이점은 먼저 구성하는 것들에 차이가 있다. 추상클래스는 멤버 변수와 일반 멤버 메소드가 있을 수 있고 추상 메소드가 한 개 이상 있으면 되고,

생성자를 가질 수 있다. 인터페이스는 모든 메소드가 추상 메소드이어야 하고 모든 변수가 static 이어야 한다. 즉 일반 변수는 가질 수 없고 상수만을 가져야 한다. 그리고 추상클래스에서 클래스가 상속 받을 때 extends 를 사용하지만 인터페이스에서 클래스가 상속 받을 때 implements 를 사용한다. 하지만 인터페이스에서 인터페이스가 상속 받을 때 추상 메소드를 구현 하지 않으므로 implements 가 아닌 extends 를 사용해야한다. 클래스는 상속이 하나 밖에 안되므로 추상클래스 또한 상속할 때 하나 밖에 안된다. 하지만 인터페이스는 여러 개 상속받을 수 있다. 추상클래스는 상속 관계를 타고 올라갔을 때 같은 조상 클래스를 상속하는데 기능까지 완벽히 똑같은 기능이 필요한 경우 사용하며 인터페이스는 다른 조상클래스를 상속하는데 같은 기능이 필요할 경우 많이 사용한다.

□ 실습문제 2

□ 문제분석

Member 와 Crew 의 정보를 입력 받고 equals()를 오버라이딩 하여 ID 를 기준으로 객체가 같은지 boolean 형태로 반환해야한다. 그 후 중복된 동아리원의 student 의 id 와 이름을 출력해야한다.

□ 프로그램 설계 및 알고리즘

먼저 Member 수와 Crew 수를 입력받고 그 수만큼 배열을 생성하여 Member 는 ID, name 를 입력받고 Crew 는 이와 더불어 Department 를 입력받는다. 그 후 생성자를 호출해 객체를 생성한다. ID 를 기준으로 같은 동아리원인지 판단하기 위해 Member 클래스에 equals()를 오버라이딩하여 ID 를 기준으로 같은지 판단하도록 한다. 같으면 true 다르면 false 를 반환하고 중복 동아리 원을 출력하도록한다. 그 후 Member, Crew 전체 정보를 출력한다.

□ 소스코드 및 주석

(Member)

```
package week10_2;

public class Member {
    private int iID;
    private String sName;

    Member() {//매개변수 없는 생성자
        iID = 0;
        sName = "";
    }
```

```
    Member(int id, String name) {//ID와 name을 매개변수로 하는 생성자
        iID = id;
        sName = name;
    }

    public int getID() {//ID 반환 메소드
        return iID;
    }
    public String getName() {//Name 반환 메소드
        return sName;
    }

    // ID를 기준으로 같은지 확인한다.
    @Override
    public boolean equals(Object obj) {//매개변수를 object로 지정

        Member member = (Member) obj;//다운캐스팅
        boolean bResult;

        bResult = (iID == member.iID);//둘이 같은지 비교연산자로 확인 후
결과를 boolean 형태로 반환하여 bResult에 저장

        if(bResult==true) {//결과가 true
            return true;//true 반환
        }
        else {//아니면
            return false;//false 반환
        }


    }


    @Override
    public String toString() {//toString()오버라이딩 현재 객체의 정보를
스트링형태로 반환
        return "Student ID: " + iID + "\nName: " + sName;
    }
}
```

(Crew)

```
package week10_2;

public class Crew extends Member {
    private String sDepartment;

    Crew() {//매개변수 없는 생성자
```

```
        super();
        sDepartment = "";
    }

    Crew(int id, String name, String department) {//id, name,
department 가 매개변수인 생성자
        super(id, name);
        sDepartment = department;
    }


    @Override
    public String toString() {//toString()오버라이딩하여 객체 정보 스트링으로
반환
        return super.toString() + "\nDepartment: " + sDepartment;
    }
}
```

(MemberEx)

```
package week10_2;
import java.util.Scanner;


public class MemberEx {

    public static void main(String[] args) {
        int iMemberNum, iCrewNum;
        int iID;
        int i, j;
        String sName, sDp;
        Scanner oInDev = new Scanner(System.in);

        System.out.println("일반 동아리원 수를 입력하시오: ");
        iMemberNum = oInDev.nextInt();
        System.out.println("집행부 수를 입력하시오: ");
        iCrewNum = oInDev.nextInt();

        Member Members[] = new Member[iMemberNum];//Member 수 크기의
Member 객체 배열 생성
        Crew Crews[] = new Crew[iCrewNum];////Crew 수 크기의 Crew 객체 배열
생성

        System.out.println("---Input Members Information---");

        for (i = 0; i < iMemberNum; i++) {
            System.out.print("student ID:");
            iID = oInDev.nextInt();//ID 입력 받음

            System.out.print("Name:");
```

```
            sName = oInDev.next();//이름 입력 받음
            Members[i] = new Member(iID, sName);//입력 받은 정보로 생성자
호출하여 객체 생성
            System.out.println("");
        }

        System.out.println("---Input Crews Information---");

        for (i = 0; i < iCrewNum; i++) {
             System.out.print("student ID:");
            iID = oInDev.nextInt();//ID 입력 받음

            System.out.print("Name:");
            sName = oInDev.next();//name 입력 받음

            System.out.print("Department:");
            sDp = oInDev.next();//Department 입력 받음
            Crews[i] = new Crew(iID, sName, sDp);//위의 정보들로 생성자
호출하여 Crew 객체 생성
            System.out.println("");
        }
        oInDev.close();

        // Check for duplicates
        for (i = 0; i < iMemberNum; i++) {
            for (j = i + 1; j < iMemberNum; j++) {
                if (Members[i].equals(Members[j])) {
                    System.out.println("Duplicate Member
found:");//Member 끼리의 중복 찾음
                    System.out.println(Members[i].toString());
                    System.out.println("");
                    System.out.println(Members[j].toString());
                    System.out.println("");
                }
            }
        }
        for (i = 0; i < iCrewNum; i++) {
            for (j = i + 1; j < iCrewNum; j++) {
                if (Crews[i].equals(Crews[j])) {
                    System.out.println("Duplicate Crew
found:");//Crew 끼리 중복 찾음
                    System.out.println(Crews[i].toString());
                    System.out.println("");
                    System.out.println(Crews[j].toString());
                    System.out.println("");
                }
            }
        }

        for (i = 0; i < iMemberNum; i++) {
            for (j = 0; j < iCrewNum ; j++) {
```

```
                if (Members[i].equals(Crews[j])) {
                    System.out.println("Duplicate Member&Crew
found:");//Member 와 Crew 사이의 중복 찾음
                    System.out.println(Members[i].toString());
                    System.out.println("");
                    System.out.println(Crews[j].toString());
                    System.out.println("");
                }
            }
        }

        System.out.println("---Input Members---");//정보 출력
        for (i = 0; i < iMemberNum; i++) {
            System.out.println(Members[i].toString());//Member 의 메소드인
toString()호출하여 정보 출력
            System.out.println("");
        }

        System.out.println("---Input Crews---");
        for (i = 0; i < iCrewNum; i++) {
            System.out.println(Crews[i].toString());//Crew 의 메소드인
toString()호출하여 정보 출력
            System.out.println("");
        }


    }
}
```

□ 결과 및 결과 분석

결과 화면

```
일반 동아리원 수를 입력하시오:
3
집행부 수를 입력하시오:
3
---Input Members Information---
student ID:2022
Name:Herry

student ID:2022
Name:Lilly

student ID:2023
Name:kane

---Input Crews Information---
student ID:2023
Name:kane
Department:PRESIDENT

student ID:2023
Name:Lilly
Department:SECRETARY

student ID:2024
Name:Herry
Department:VICE_PRESIDENT
```

```
Duplicate Member found:
Student ID: 2022
Name: Herry

Student ID: 2022
Name: Lilly

Duplicate Crew found:
Student ID: 2023
Name: kane
Department: PRESIDENT

Student ID: 2023
Name: Lilly
Department: SECRETARY

Duplicate Member&Crew found:
Student ID: 2023
Name: kane

Student ID: 2023
Name: kane
Department: PRESIDENT

Duplicate Member&Crew found:
Student ID: 2023
Name: kane

Student ID: 2023
Name: Lilly
Department: SECRETARY
```

```
---Input Members---
Student ID: 2022
Name: Herry

Student ID: 2022
Name: Lilly

Student ID: 2023
Name: kane

---Input Crews---
Student ID: 2023
Name: kane
Department: PRESIDENT

Student ID: 2023
Name: Lilly
Department: SECRETARY

Student ID: 2024
Name: Herry
Department: VICE_PRESIDENT
```

결과 분석

Member에서 2022로 ID를 지정한 동아리원 2명 Member, Crew사이에서 2023으로 ID를 지정한 2명, Crew에서 2023로 ID 를 지정한 2 명의 정보 입력 후 결과를 보니 각각 2 명씩 중복되었다고 알맞게 출력되었다. 같은 ID 에 name 과 Department가 달라도 중복으로 출력하는 것을 통해 ID를 기준으로 중복으로 판단하는 것을 알 수 있다.

□ 소감

equals()메소드에서 obj 를 매개변수로 받기 때문에 호출 시 자동 업캐스팅이 일어나고 Member member = (Member) obj;을 통해 명시적으로 다운캐스팅이 일어난다. 이를 사용하지 않고 매개변수로 특정한 타입을 받는다면 해당 클래스 혹은 그 하위 클래스들만 비교할 수 있다. 또한 매개변수가 달라져 오버라이딩이 아닌 오버로딩이 된다. 만약 오버로딩을 사용한다면 동적바인딩이 실행되지 않기 때문에 예기치 않게 Object.equals()가 호출되면서 원치 않은 결과를 얻을 수 있다는 단점이 있다.