
# 객체지향프로그래밍 9주차

| 제출일 | 2023년 11월 08일 | 학과 | 컴퓨터공학과 |
|---|---|---|---|
| 과목 | 객체지향프로그래밍 01 | 학번 | 2022112701 |
| 담당교수 | 한인 교수님 | 이름 | 안소희 |

# □ 실습문제 1

## □ 문제분석

id 와 이름을 저장하는 Member 클래스를 상속받아 department 정보를 추가한 Crew 클래스를 만들고 이 정보를 출력하는 메소드를 각 클래스에 정의하는 것은 같지만 다른 점은 오버라이딩을 이용해 정보를 출력하는 것이다.

## □ 프로그램 설계 및 알고리즘

Member 클래스에 학번과 이름을 선언해주고 출력함수와 학번을 반환하는 함수를 선언한다. Crew 클래스는 Member 클래스를 상속받아 따로 학번과 이름을 선언하지 않고 super()함수를 통해 서브클래스에서 슈퍼클래스의 생성자를 호출한다. 그리고 추가기능으로 부서를 선언하고 생성할 때 이를 같이 생성하도록 한다. 또한 정보를 문자열로 받는 메소드를 선언한다. 메인 함수에서 일반 부원의 수와 집행부원의 수를 입력 받은 후 그 수만큼 객체 배열을 생성한다. 그 후 먼저 일반 부원의 학번을 입력 받는다. 만약 이전에 입력 받은 학번과 같다면 이미 존재한다는 에러 메세지가 출력된다. 그 후 다시 입력 받는다. (이름은 동명이인일 수 있으므로 생략) 그 후 입력 받은 학번과 이름을 Member 클래스의 객체를 배열의 원소이자 객체의 레퍼런스인 Members[i]에 할당한다. 그 후 집행부원의 정보를 입력 받는다. 동아리에는 일반 동아리원, 집행부원 두 분류로 나누어져 있으므로 일반 동아리원과 집행부원이 겹치지 않도록 이전과 같이 학번으로 중복을 확인한다. 그렇게 학번과 이름을 입력 받은 뒤 서브 클래스인 Crew 에서 추가한 기능인 부서를 입력 받는다. Member 와 Crew 의 정보를 문자열로 받는 로직은 다음과 같다. toString()메소드의 오버라이딩을 이용하기 위해 먼저 슈퍼클래스인 Member 에 toString 메소드를 만든 후 Crew 클래스에 메소드 이름, 리턴 타입, 매개변수 타입 및 개수를 같게 생성한다. 이를 통해 Crew 클래스의 toString 을 실행할 때 슈퍼클래스의 toString 을 무시하고 실행하도록 한다.

Crew 클래스의 toString 을 실행할 때 Crew->Member 클래스 순으로 메소드를 찾기 때문이다. 이때 처음 Member 함수에서 생성한 toString()메소드 또한 Object 클래스의 toString()메소드를 오버라이딩 한 것이다.

## □ 소스코드 및 주석

(week9_1)

```
package week9_1;

public class Member {
	private int iID;//학번과 이름을 필드로 가지는 Member 클래스
	private String sName;

	Member(){//기본 생성자
		iID = 0;
		sName = "";
	}
	Member(int id, String name){//학번과 이름을 매개변수로 가지는 생성자
		iID = id;
		sName = name;
	}

	public int getID(){//ID 를 반환, 메인함수에서 ID 중복을 확인할 때 사용한다.
		return iID;
	}
	//오버라이딩 본래 있던 object 클래스의 toString 오버라이딩
	public String toString() {//Member 정보(ID, 이름)를 출력한다.
	   return "Student ID: " + iID + "\nName: " + sName;
    }

}
package week9_1;

public class Crew extends Member{//Member 클래스 상속받음
```

```
	private String sDepartment;//부서 기능 추가


	Crew(){//기본 생성자 생성
		super();//super()를 통해 명시적으로 슈퍼 클래스의 생성자를 호출한다.
		sDepartment = "";
	}

	Crew(int id, String name, String Department) {//학번과 이름 부서를 매개변수로 한 생성자
		super(id,name);//super()를 통해 명시적으로 슈퍼 클래스의 생성자를 호출한다. 이때 매개변수를 기준으로 호출함
		sDepartment = Department;
	}
	//Member 클래스의 toString 메소드 오버라이딩
	public String toString() {//Crew 정보를 출력한다.
	   return super.toString() + "\nRole: " + sDepartment;//슈퍼클래스의 toString()을 호출하여 문자열로 정보를 받고 Crew 가 확장한 정보인 Department 를 출력한다.
    }
}

package week9_1;
import java.util.Scanner;
public class MemberEx {

	public static void main(String[] args) {

		int iMemberNum,iCrewNum;
		int iID;
		int i , j, k;
		String sName,sDp;
		Scanner oInDev;
		oInDev = new Scanner(System.in);//Scanner 객체 생성하여 입력받도록 함
		System.out.println("일반 동아리원 수를 입력하시오: ");
		iMemberNum = oInDev.nextInt();
		System.out.println("집행부 수를 입력하시오: ");
		iCrewNum = oInDev.nextInt();
```

```
            Member Members[];
            Members = new Member[iMemberNum];//입력 받은 일반 부원
수만큼 객체의 배열 원소 수 지정

            Crew Crews[];
            Crews = new Crew[iCrewNum];//입력 받은 집행부원 수만큼
객체의 배열 원소 지정

            System.out.println("---Input Members Information---
");//일반 부원 정보 입력
            for( i = 0 ; i < iMemberNum; i++) {

                  System.out.print("student ID:");//학번
                  while (!oInDev.hasNextInt()) {
                        System.out.println("ID 숫자를 입력하시오
");//숫자로 입력하도록 함
                  oInDev.next(); // 잘못된 입력값을 버린다
                  System.out.print("student ID:");//학번
             }
                  iID = oInDev.nextInt();
                  for( j = 0 ; j < i ; j++) {
                        if(Members[j].getID() == iID) {//만약
같으면 반복문 빠져나감
                              break;
                        }
                  }
                  if( i == j ) {//두 개의 값이 같다면 이전의
반복문에서 if 문이 실행되지 않은 것이므로 같은 학번이 존재하지 않다고 판단
                  System.out.print("Name: ");//이름
                  sName = oInDev.next();


                  Members[i] = new Member(iID,sName);//입력받은
학번과 이름을 매개변수로 하는 생성자 실행
                  System.out.println("");
                  }
                  else {
```

```
                        System.out.println("Error: ID already
exists. Please enter a different ID.");//만약 반복문에서의 if 문이
실행되었다면 같은 학번이 있던 것이므로 중복 메세지 출력
                        i--;//다시 입력받도록 함
                  }


            }
            System.out.println("---Input Crews Information---
");//집행 부원 정보 입력
            for(i = 0 ; i < iCrewNum; i++) {
                  System.out.print("student ID:");//학번
                  while (!oInDev.hasNextInt()) {
                  System.out.println("ID 숫자를 입력하시오 ");//숫자
입력받도록 에러메세지
                  oInDev.next(); // 잘못된 입력값을 버립니다.
                  System.out.print("student ID:");//학번
             }
                  iID = oInDev.nextInt();
                  for( k = 0 ; k < iMemberNum ; k++) {//일반
부원의 학번과의 중복 발생했는지 확인
                        if(Members[k].getID() == iID) {
                              break;
                        }
                  }
                  for( j = 0 ; j < i ; j++) {//집행 부원끼리 중복이
발생했는지 확인
                        if(Crews[j].getID() == iID) {
                              break;
                        }
                  }
                  if(i == j&&k==iMemberNum){//두 곳에서 전부 중복이
발생하지 않았다면
                  System.out.print("Name: ");//이름 입력받음
                  sName = oInDev.next();

                  System.out.print("Department: ");//부서 입력받음
                  sDp = oInDev.next();
```

```
                Crews[i] = new Crew(iID,sName,sDp);//학번, 이름,
부서를 매개변수로 하는 집행부원 객체 생성
                System.out.println("");
                }
                else {
                      System.out.println("Error: ID already
exists. Please enter a different ID.");//중복 메세지 출력

                      i--;//다시 입력 받음
                }
           }
           System.out.println("---Input Members---");//입력받은
일반 부원 전체 정보 출력
           for(i = 0 ; i < iMemberNum; i++) {
                String sResult =
Members[i].toString();//toString 메소드를 통해 문자열(출력 정보)을
받음
                System.out.println(sResult);//그 문자열을 출력한다
                System.out.println("");
           }
           System.out.println("---Input Crews---");//입력받은
집행부원 전체 정보 출력
           for(i = 0 ; i < iCrewNum; i++) {
                String sResult =
Crews[i].toString();//toString 메소드를 통해 문자열(출력 정보)을 받음
                System.out.println(sResult);//그 문자열을 출력한다
                System.out.println("");
           }

           oInDev.close();//사용자의 입력을 받는 oInDev 객체 닫음


     }

}
```

## □ 결과 및 결과 분석

결과 화면

```
일반 동아리원 수를 입력하시오:
2
집행부 수를 입력하시오:
2
---Input Members Information---
student ID:jj
ID 숫자를 입력하시오
student ID:22
Name: Jane

student ID:22
Error: ID already exists. Please enter a different ID.
student ID:23
Name: Kelly

---Input Crews Information---
student ID:23
Error: ID already exists. Please enter a different ID.
student ID:24
Name: John
Department: president

student ID:asd
ID 숫자를 입력하시오
student ID:27
Name: Jully
Department: secretary
```

```
---Input Members---
Student ID: 22
Name: Jane

Student ID: 23
Name: Kelly

---Input Crews---
Student ID: 24
Name: John
Role: president

Student ID: 27
Name: Jully
Role: secretary
```

**결과 분석**

동아리원 2 명, 집행부 수 2 명으로 입력한다. 그 후 아이디 jj 를 입력한다. 그러면 에러메세지가 뜨고 숫자로 다시 입력받는다. 그 후 22 를 입력한다. 그 다음 아이디도 22 로 입력했을 경우 중복이므로 에러 메세지를 출력한다 그리고 다시 입력을 받고 Crew 멤버 정보를 입력한다. Crew 또한 Member 아이디와 중복되면 에러 메세지가 출력된다. 그리고 Department 를 입력 받고 Member와 Crew의 정보를 출력한다. 이 때 toString() 메소드 오버 라이딩을 이용해 정의했으므로 Member 정보를 출력할 때 Object 의 toString()이 아닌 Member 클래스의 toString 메서드가 실행되어 출력된다. 또한 Crew 의 정보를 출력할 때 Member 나 Object 의 toString()이 아닌 Crew 클래스의 toString 메서드가 실행되어 출력된다.

# □ 소감

Object 클래스가 가진 메소드 String toString()메소드, Boolean equals()메소드에 대해 조사하였다. toString 메소드는 Object객체에서 현 객체에 대한 문자열 표현을 리턴하는데 현 객체의 이름과 해시코드를 출력한다. 하지만 프로그래머들은 이 toString()으로 클래스의 변수와 같은 정보를 출력하고자 하기 때문에 이 메소드를 오버라이딩하여 클래스에 정의해서 이 클래스의 틀로 만든 인스턴스의 toString()메소드 호출 시 이름과 해시코드가 아닌 정보가 출력되도록 많이 쓰인다.

equals메소드는 비교연산자 '=='와 무엇이 다른지 아는게 중요하다. 먼저 ==는 피연산자들의 레퍼런스를 비교한다. 즉 객체의 주소값을 비교하는데 equals는 객체의 내용을 비

교한다. 그래서 보통 이를 오버라이딩하여 각 요소들이 같은지 비교하는 메소드를 정의하고 boolean으로 반환한다. 이때 오버로딩을 사용하지 않는 이유는 동적바인딩을 사용하지 못하기 때문이다. 이번 실습을 통해 오버라이딩에 대해 자세히 알 수 있었다.

# □ 실습문제 2

## □ 문제분석

DiscountPolicy클래스를 추상클래스로 정의한다. 그리고 이를 상속받은 FixedDiscountPolucy와 PercentDiscountPolicy클래스를 정의한다. dDiscountPrice변수는 할인되는 양을 의미하며 FixedDiscountPolicy에서 getPrice 메소드는 dDiscountPrice만큼 할인된 값을 반환한다. PercentDiscountPolicy에서 getPrice 메소드는 dDiscountPrice%만큼 할인된 값을 반환한다. 이때 멤버 접근 지정자와 extends, 업캐스팅, 오버라이딩, 입출력을 이용해야한다.

## □ 프로그램 설계 및 알고리즘

먼저 DiscountPolicy클래스를 추상클래스로 정의하여 상속받은 클래스에서 구현이 이루어지도록했다. 이 클래스에 멤버 변수 dDiscountPrice와 생성자, getPrice메소드를 정의한다. 그리고 FixedDiscountPolicy클래스에서 이를 상속받아 생성자를 만들고 getPrice 함수를 구현한다. 이때 생성자는 super를 통해 슈퍼클래스의 생성자를 호출한다. PercentDiscountPolicy클래스 또한 DiscountPolicy클래스를 상속받아 생성자를 만들고 getPrice함수를 구현한다. 이때 생성자는 super를 통해 슈퍼클래스의 생성자를 호출한다. 그리고 main함수가 있는 DiscountEx에서 실행한다. 먼저 무한루프를 이용하여 사용자에게 계속 입력을 받고 종료는 음수를 입력받아 시행한다. 만약 고정 금액을 선택했을 경우 DiscountPolicy의 레퍼런스로 FixedDiscountPolicy 객체를 가리키도록 한다. 그래서 할인 후 가격을 할당받을 때 FixedDiscountPolicy의 getPrice메소드를 실행하는 동적바인딩을 사용한다. PercentDiscountPolicy또한 위와 같다. 1또는 2를 선택하지 않을 경우 에러메세지를 출력한다. 또한 퍼센트할인 선택 시 할인 값이 100초과면 에러메세지를 출력하도록 했다. 메인에서 PercentDiscountPolicy와 FixedDiscountPolucy 클래스의 멤버함수를 호출하므로 접근지정자를 public으로 설정하고 슈퍼 클래스인 DiscountPolucy는 추상 클래스로 구현이 안되어있어 메인에서 호출할 일이 없기 때문에 상속받은 서브클래스가 접근할 수 있는 protected접근지정자를 사용하였다.

## □ 소스코드 및 주석

(week9_2)

```
package week9_2;

public abstract class DiscountPolicy {//추상클래스
    protected double dDiscountPrice; // 할인액

    protected DiscountPolicy(double dDiscountPrice) {//할인값을 매개변수로 하는 생성자 생성
        this.dDiscountPrice = dDiscountPrice;//이 클래스의 변수에 매개변수로 받은 dDiscountPrice 할당
    }

    // getPrice 메소드를 추상 메소드로 정의
    protected abstract double getPrice(double dBeforePrice);
}
package week9_2;

public class FixedDiscountPolicy extends DiscountPolicy {

     public FixedDiscountPolicy(double dDiscountPrice) {//할인값을 매개변수로 하는 생성자
           super(dDiscountPrice);//슈퍼클래스 생성자 호출

     }
     //오버라이딩
     public double getPrice(double dBeforePrice) {//할인 이전의 가격을 매개변수로 받음
           double dResult;
           dResult = dBeforePrice - dDiscountPrice;
        return dResult;//할인 이후 가격 반환
    }

}
package week9_2;

public class PercentDiscountPolicy extends DiscountPolicy{
```

```
	public PercentDiscountPolicy(int dDiscountPrice)
{//할인값을 매개변수로 하는 생성자
		super(dDiscountPrice);//슈퍼클래스 생성자 호출

	}
	//오버라이딩
	public double getPrice(double dBeforePrice) {//할인 이전의
값을 매개변수로 받음
		double discountAmount,dResult;
		discountAmount = dBeforePrice *
(double)(dDiscountPrice / 100);
		dResult = dBeforePrice - discountAmount;

	   return dResult;//할인 이후의 가격 반환

	}

}
package week9_2;
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");
			int iDiscount = oInDev.nextInt();

			System.out.println("할인 정책을 선택하세요(1. 고정
금액 할인, 2. 퍼센트 할인)");
			int iChoice = oInDev.nextInt();
			if(iChoice == 1) {
```

```
                    DiscountPolicy fixedPolicy = new
FixedDiscountPolicy(iDiscount);//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(iDiscount>100) {//100%를 초과할 때 에러
                    System.out.println("100%을 넘을 수
없습니다.");
                    continue;
                }
                DiscountPolicy percentPolicy = new
PercentDiscountPolicy(iDiscount);//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)
30000
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)
30000
discountValue 설정 값을 입력하세요
10
할인 정책을 선택하세요(1. 고정 금액 할인, 2. 퍼센트 할인)
2
퍼센트 할인 후 가격: 27000
상품의 가격을 입력하세요(종료시 음수 값 입력)
30000
discountValue 설정 값을 입력하세요
1000
할인 정책을 선택하세요(1. 고정 금액 할인, 2. 퍼센트 할인)
1
고정 금액 할인 후 가격: 29000
상품의 가격을 입력하세요(종료시 음수 값 입력)
-1
```

**결과 분석**

30000 원을 입력하고 1000 으로 값을 설정한 다음 퍼센트할인을 선택하여 에러 메세지가 떴다. 그 후 다시 입력 받았다. 그 다음으로 다시 30000 원 입력 후 10 으로 할인 값을 정하고 퍼센트 할인을 하여 제대로 결과가 나온 것을 볼 수 있다. 또한 30000 원 입력 후 할인 값을 1000 으로 하여 고정 금액 할인을 선택했을 때 29000 원으로 알맞은 결과가 나오고 -1 입력 시 종료가 된 것을 알 수 있다. 이때 getPrice 메소드를 호출할 때 슈퍼클래스인 DiscountPolicy 의 메소드가 아닌 각각의 오버로딩된 서브클래스의 메소드가 동적바인딩되어 알맞게 호출된 것을 알 수 있다.

## □ 소감

추상 클래스가 어떤 것인지는 알았지만 활용하는 방법을 몰랐는데 아직 많은 것을 하지 않았지만 적용하는 법을 배운 거 같아 좋았다. 동적 바인딩이 언제 되는지 헷갈렸는데 이번 실습을 통해 업캐스팅된 클래스가

오버라이딩 메소드를 호출할 때 슈퍼클래스의 메소드가 아닌 자신의 클래스(서브 클래스)의 메소드를 호출한다는 것을 알 수 있었다.