< JAVA PROGRAMMING LAB -6장: 상속 응용하기 >

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과

**1. Circle 클래스를 상속받아서 피자를 나타내는 Pizza 클래스를 작성하고 여러 가지 타입의 피자 객체를 생성한다. 피자들의 면적을 비교해보자.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목



\*

**class** Circle{

 **private** **double** radius; // 접근자나 설정자가 와야 변수를 접근할 수 있게 함

 **private** String color="red";

 **public** Circle() {

 radius=1.0;

 }

 **public** Circle(**double** radius) {

 **super**();

 **this**.radius=radius;

 }

 **public** Circle(**double** radius,String color) {

 **this**.radius=radius;

 **this**.color=color;

 }

 **public** **double** getRadius() { // 반지름 가져오기

 **return** radius;

 }

 **public** String getColor() { // 색 가져오기

 **return** color;

 }

 **public** **void** setRadius(**double** radius) { // 반지름 세팅하기

 **this**.radius=radius;

 }

 **public** **void** setColor(String color) { // 색 세팅하기

 **this**.color=color;

 }

 **public** String toString() { // 객체를 출력할 때 호출됨

 **return** "Circle[radius=+radius+,color=+color+]";

 }

 **public** **double** getArea() { // 면적 가져오기

 **return** radius\*radius\*Math.***PI***;

 }

}

**public** **class** Pizza **extends** Circle{

 **private** String type;

 **private** **int** price;

 **public** Pizza() { // 불고기피자 생성자

 **super**(10); // Circle 생성자 부르기 - 반지름만 전달 (색은 디폴트)

 type="firemeat";

 price=20000;

 System.***out***.println("<"+type+">");

 System.***out***.println("피자 가격: "+price);

 System.***out***.println("피자 색: "+getColor());

 System.***out***.println("피자의 반지름: "+getRadius());

 System.***out***.println("피자의 면적: "+getArea());

 System.***out***.println("---------------------------------------");

 }

 **public** Pizza(String type, **int** price) { // 다른 피자 생성자

 **super**(12,"brown-red"); // Circle 생성자 부르기 - 반지름하고 색까지 전달

 **if**(type.equals("pepperoni")) {} // 디폴트값 사용

 **else** **if**(type.equals("cheesebacon")) {

 setRadius(15);

 setColor("light-brown-red");

 }

 **else** **if**(type.equals("combination")) {

 setRadius(13);

 setColor("red-black");

 }

 **this**.type=type;

 **this**.price=price;

 System.***out***.println("<"+type+">");

 System.***out***.println("피자 가격: "+price);

 System.***out***.println("피자 색: "+getColor());

 System.***out***.println("피자의 반지름: "+getRadius());

 System.***out***.println("피자의 면적: "+getArea());

 System.***out***.println("---------------------------------------");

 }

 **public** **void** Compare(Pizza obj1,Pizza obj2,Pizza obj3) {

 **double**[] area=**new** **double**[4];

 area[0]=getArea();

 area[1]=obj1.getArea();

 area[2]=obj2.getArea();

 area[3]=obj3.getArea();

 String[] types=**new** String[4];

 types[0]=type;

 types[1]=obj1.type;

 types[2]=obj2.type;

 types[3]=obj3.type;

 String tmp;

 **for**(**int** i=0;i<3;i++) {

 **for**(**int** j=i;j<4;j++) {

 **if**(area[i]>area[j]) {

 tmp=types[i];

 types[i]=types[j];

 types[j]=tmp;}

 }

 }

 System.***out***.print("피자의 면적 비교: ");

 **for**(**int** i=0;i<4;i++) {

 **if**(i==3)

 System.***out***.print(types[i]);

 **else**

 System.***out***.print(types[i]+" <= ");

 }

 }

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

 // **TODO** Auto-generated method stub

 Pizza[] parray = **new** Pizza[4];

 parray[0]=**new** Pizza(); // 피자 객체 생성

 parray[1]=**new** Pizza("pepperoni",24000); // 피자 객체 생성

 parray[2]=**new** Pizza("cheesebacon",25000); // 피자 객체 생성

 parray[3]=**new** Pizza("combination",24500); // 피자 객체 생성

 parray[0].Compare(parray[1],parray[2],parray[3]);

 }

}

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

Circle클래스의 필드는 radius와 color가 있다. Color은 “red”로 초기화되어있다. 이후 생성자가 3개나 나오는데, main메소드에서 이 클래스를 몇 개의 인수로 인스턴스화할지에 따라 실행되는 생성자가 다르다. 매개변수 한 개로 받을 때는 반지름만 수정하고, 두 개를 받으면 반지름과 색을 수정하고 하나도 안 받으면 반지름을 1.0으로 초기화하겠다. 그리고 필드들의 접근 지정자가 private이니 설정자와 접근자를 만들었다. 메소드 중 toString이 보이는데, 이는 객체를 출력할 때 자동으로 호출되는 메소드이다. 이 메소드는 object클래스의 toString을 오버라이딩한 것인데, 오버라이딩하지 않았으면 객체의 주소가 출력된다. GetArea는 원의 넓이를 구해서 리턴하는 메소드이다. 다음은 Pizza클래스를 살펴보겠다. 이 클래스는 Circle클래스를 상속받기에, Circle을 인스턴스화할 필요없이 Circle의 메소드를 호출할 수 있다. 이 클래스도 생성자가 2개인데 만약 인수로 아무 것도 오지 않으면 기본 피자인줄 알고 불고기 피자를 만들 것이다. 불고기 피자일때의 생성자를 먼저 보자. 피자의 반지름 10을 Circle클래스로 보내서 radius를 수정하고, Pizza클래스 필드를 수정한다. 이후 더 할 일이 없으니 피자의 정보를 출력한다. Circle클래스에서 만든 여러 접근자를 이용한다. 이번엔 매개변수를 2개 받는 Pizza생성자를 살펴보겠다. 인수를 보냈다는건 기본이 아닌 피자를 출력해야 된다고 간주하였다. 일단 super메소드를 이용해서 radius를 12로 만들고 색을 brown-red로 만들었다. 이후 받은 문자열이 ‘페페로니’이면 radius와 색을 저장한 그대로 이어 나가기로 했다. 그렇지 않고 ‘치즈베이컨’이면 설정자를 이용해서 반지름과 색을 15, 연갈색+빨강으로 수정하였다. 그렇지 않고 ‘콤비네이션’이면 설정자를 이용해서 반지름과 색을 13, 검빨색으로 수정하였다. 받은 price는 필드에 저장하고 받은 type도 필드에 저장하겠다. 이후 더 할 일이 없으니 피자의 상세 정보를 출력하였다. 이 작업 역시 접근자를 이용한다. 마지막으로 Compare메소드를 살펴보겠다. 이 메소드는 객체 3개를 전달받는다. 현재 객체와 다른 객체 3개를 비교하기 위해 전달받은 것이다. 일단 각 객체의 면적을 실수형 배열 요소에 각각 저장한다. 그리고 각 객체의 type필드를 스트링형 배열 요소에 각각 저장한다. 그리고 스트링형인 임시 변수 tmp를 만든다. 다음은 for문을 중첩으로 사용해서 만든 오름차순 소스 블록이다. i=0부터 배열의 크기-1만큼 반복하고 중첩으로 j=i부터 배열의 크기만큼 반복한다. 면적 배열 인덱스i의 값을 기준으로 인덱스 j의 값들과 비교해서 인덱스i의 값이 더 크면, 타입 배열 인덱스i의 값과 타입 배열 인덱스j의 값을 바꾸는 작업을 반복하는 것이다. 마지막으로 오름차순한 것을 기반으로 타입 배열 요소를 차례로 출력하면, 면적에 따라 부등호로 피자의 크기가 출력되겠다. 이번엔 main메소드를 살펴보겠다. Pizza클래스를 객체화하는데, 객체를 여러 개 만들어야 하니 객체 배열을 만들었다. 이후 객체 배열의 요소에 각각 객체를 생성하였다. 이러면 생성자로 데이터들이 가고, 생성자가 동작을 해서 객체를 초기화하는 것은 물론이고 객체의 필드들을 출력하겠다. 마지막으로 객체 배열 요소 하나를 기준으로 Compare메소드를 호출하는데, 인수로 다른 객체들을 보내면 메소드에서 알아서 크기 비교를 하고 출력까지 하는 결과를 낳는다.

◇ 제목

◇ 제목

◇ 프로그램 설명

< JAVA PROGRAMMING LAB -6장: 상속 응용하기 >

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과

**2. Person 클래스를 상속받아서 Student 클래스와 Employee클래스를 정의한다. 이후 “steve jobs”가 이름인 사람을 찾아 보자.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목



**class** Person{

 **private** String name;

 **private** **int** id;

 **public** **void** setname(String name) {**this**.name=name;} // 설정자

 **public** **void** setid(**int** id) {**this**.id=id;}

 **public** String getname() {**return** name;} // 접근자

 **public** **int** getid() {**return** id;}

 **public** **void** printInfo() {}

}

**class** Student **extends** Person{

 **private** **double** gpa;

 **private** **int** credits;

 Student(String name,**int** id,**double** gpa,**int** credits){

 setname(name);

 setid(id);

 **this**.gpa=gpa;

 **this**.credits=credits;

 }

 @Override

 **public** **void** printInfo() {

 **boolean** decide=getname().equals("steve jobs");

 **if**(decide==**true**) {

 System.***out***.println("< steve jobs를 찾았습니다 >");

 System.***out***.print("id:"+getid()+", ");

 System.***out***.print("gpa:"+gpa+", ");

 System.***out***.print("credits:"+credits+"\n");}

 }

}

**class** Employee **extends** Person{

 **private** **int** salary;

 **private** **int** years;

 Employee(String name,**int** id,**int** salary,**int** years){

 setname(name);

 setid(id);

 **this**.salary=salary;

 **this**.years=years;

 }

 @Override

 **public** **void** printInfo() {

 **boolean** decide=getname().equals("steve jobs");

 **if**(decide==**true**) {

 System.***out***.println("< steve jobs를 찾았습니다 >");

 System.***out***.print("id:"+getid()+", ");

 System.***out***.print("salary:"+salary+", ");

 System.***out***.print("years:"+years+"\n");}

 }

}

**public** **class** people{

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

 Person[] parray=**new** Person[4];

 parray[0]=**new** Student("steve jobs",1,3.9,30);

 parray[1]=**new** Employee("Jeff Bezos",2,10000,10);

 parray[2]=**new** Employee("Bill Gates",3,20000,15);

 parray[3]=**new** Employee("steve jobs",4,30000,20);

 **for**(**int** i=0;i<4;i++)

 parray[i].printInfo();

 }

}

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 프로그램 설명

Person클래스부터 살펴보겠다. 필드는 스트링형 변수 name과 정수형 변수 id가 있다. 이후 접근자와 설정자를 만들었다. 클래스의 필드를 직접적으로 가져다 쓰는 것은 위험하기 때문에, 이렇게 필드의 접근 지정자를 private로 만들고 접근자와 설정자를 호출하는 것이 이상적이다.이후 나오는 printInfo메소드는 현재 몸체가 정의되지 않았으니, Person클래스를 상속받는 클래스에서 오버라이딩할 것을 암시한다.

이젠 Student클래스이다. 이 클래스는 Person클래스를 상속받는다. Person의 필드가 private상태기에 필드는 상속 못 받는다. 그래서 매개변수를 받아도 설정자를 이용해서 그냥 부모 클래스의 필드에 저장하고, 접근자를 이용해서 필드를 빼네오는게 최선이겠다. 설정자를 쓰지 않으면 부모 클래스에서 생성자를 만들고 호출해서 클래스를 초기화하는 것도 방법이 될 수 있다. 소스를 보면 Student클래스의 필드는 gpa와 credits이고, 생성자를 통해서 부모에게 필드도 보내고 본인의 필드에 저장도 한다. 이후 오버라이딩한 printInfo를 살펴보면 먼저 decide라는 변수를 만드는 모습이 보인다. 접근자를 통해서 name 스트링 변수를 전달받은 뒤 “steve jobs”와 스트링 변수가 같으면 true가 변수에 저장되고, 아니면 false가 변수에 저장된다. 스트링 이해를 제대로 하기 위해 한 번 자세히 설명해보겠다. 먼저 접근자를 통해서 (이름 문자열을 문자형 배열 형태로 만든) 스트링 객체의 주소가 전달된다. 이후 스트링 객체의 주소를 통해 해당 객체의 equals메소드를 호출하면, 호출할 때 보낸 인수 데이터와 해당 객체가 만든 문자형 배열의 값을 비교해서 같으면 true, 다르면 false를 호출하게 된다.

만약 decide가 true면 “steve jobs”를 찾았다고 출력한 뒤, 객체의 필드를 호출해서 스티브 잡스의 정보를 출력하게 된다.

마지막으로 employee클래스이다. 사실 이 클래스는 student클래스와 필드만 다르지 다른 내용은 모두 동일하므로 설명을 생략하겠다. 필드가 다르므로 만약 decide가 true면 객체의 salary,years필드를 출력하게 된다.

이 모든걸 실행하기 위해 main메소드에 소스가 담겨 있다. Main에서는 일단 부모 객체를 저장할 수 있는 크기가 4인 배열을 만들고, 부모 클래스 참조 변수 parray가 이 배열을 가리키게 하였다. 그리고 각 배열의 요소에 객체를 생성해서 저장하였다. 각 객체는 서로 다른 인물들이 되겠고 Student,Employee클래스를 이용하였다.

마지막으로 for문을 이용해서 각 객체의 printInfo메소드를 호출함으로써 스티브 잡스와 객체 의 네임 필드가 같은지 아닌지를 조사하면서 마무리 된다.