객체지향프로그래밍 - 7주차 실습 답안지

성명: 조우연 학과: 수학과 학번: 201921195

※ 본 실습활동지에 대한 보고서나 코드를 작성함에 있어 교재나 강의노트를 제외한 다른 자료로부터 일부 또는 전체를 복사하였습니까? 예( ) 아니오(O)

1. 프로그램 수행결과 및 근거 (주어진 소스코드를 사용한다.)

-프로그램 실행결과

- 근거

먼저, employees 배열은 Employee타입의 배열이다. 또한 SalariedEmployee, Hourly Employee, CommissionEmployee, Manager는 Employee를 상속받은 subclass이다. 따라서 Employee타입의 배열인 employees의 원소로 각각의 subtype 변수인 SalariedEmployee, Hourly Employee, CommissionEmployee, Manager들로 assign할 수 있다. 각각의 class type에 따라 각각 알맞은 constructor를 실행하여 값을 초기화해준다. 이때 contructor에 넘겨주는 값 중, name과 ssn은 모든 Employee의 subclass에 공통적으로 사용되므로, super class인 Employee의 constructor에 정의된 대로 초기화해준다.

currentEmployee.getEarnings()를 실행하면 각각의 class type에 따라 각각의 class에 정의된 getEarnings()를 사용하게 된다. 즉, Manager타입의 변수는 Manager 클래스 안에 정의된 getEarnings()를 사용하고, HourlyEmployee타입 변수는 HourlyEmployee 클래스 안에 정의된 getEarnings()를 사용하는 overriding이 일어난다. 따라서 각각의 클래스 타입의 변수에 따라 각각의 클래스 안에 정의된 getEarnings()메소드를 실행하여 위와 같은 결과가 출력된다. (PayrollSystemTest안에 반복문에서 실행되는 if(currentEmployee instanceof Manager) 조건문에 의해 Manager타입의 변수는 추가적으로 Manager 클래스 안에 정의된 setBonus(100)을 추가로 실행하여 getEarnings()를 실행한다.)

결론적으로 Employee는 super class이므로 Employee타입의 배열 안에는 이에 대한 하위 클래스들을 모두 assign할 수 있고, 또한 Employee는 abstract 클래스이므로, polymorphism을 제공함으로써 일반적으로 처리할 수 있고, polymorphism성질에 의해 superclass에 정의된 메소드를 subclass에서 overriding하여 사용할 수 있다.

2. 변경된 Manager class의 getEarnings를 실행한 결과 및 이유

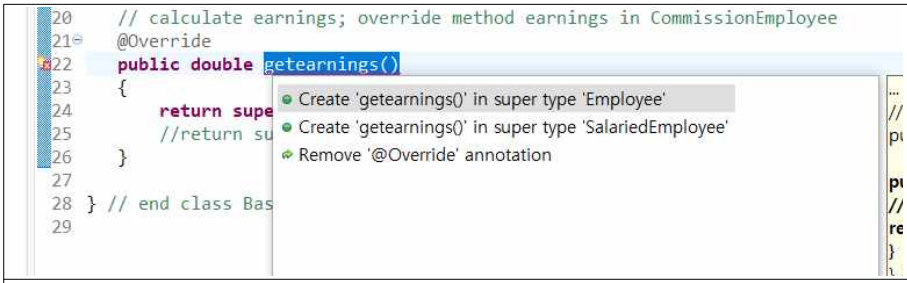
-컴파일 결과

- 이유

super class에 정의된 private데이터나, 메소드는 subclass에서 사용하거나 접근할 수 없다. 따라서 weeklySalary 변수는 Manager클래스의 super class인 SalariedEmployee 클래스 안에서 private으로 정의된 변수이므로, Manager클래스에서 따로 weeklySalary변수를 정의하지 않는다면, Manager클래스에서 사용할 수 없다.

3. Manager class의 getEarnings 메소드 이름을 getearnings로 변경한 후 실행한 결과와 이유

- 수행결과



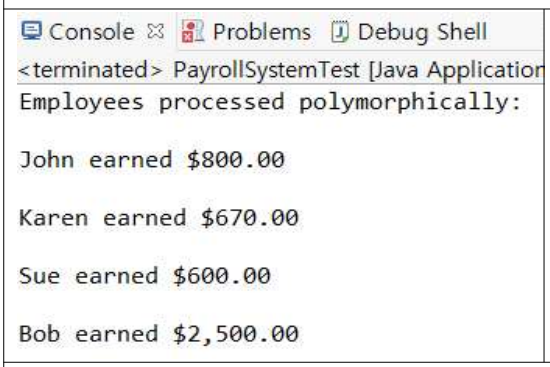
- 이유

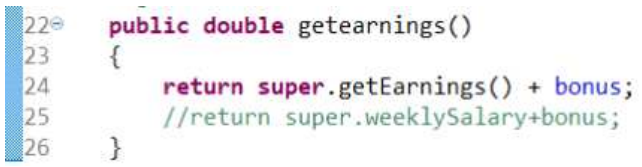
Manager class에 getEarnings() 메소드가 존재하지 않기 때문에, getEarnings는 Manager 클래스 안에서 overriding되지 않고, Manager class의 super class인 SalariedEmployee 클래스에 정의되어 있는 getEarnings 메소드를 실행함으로써 overriding된다. Manager클래스의 constructor를 통해 weeklySalary가 2500으로 초기화되었으므로 SalariedEmployee클래스에 정의된 getEarnings 메소드를 실행시켜 얻은 반환값도 2500이 되어 출력된다.

추가적으로 Manager클래스의 getearningsmethod에 아래 캡쳐본과 같이 에러메세지가 표시되는데, 이는 @Override표시를 통해 getearningsmethod가 Manager의 상위 클래스에 있는 메소드를 재정의했다는 것을 확인하기 위함인데, Manager의 상위 클래스에는 getEarningsmethod가 존재하지만, getearningsmethod는 존재하지 않으므로 에러메세지를 표시해준 것이다.

4. getearningsmethod에서 @Override를 삭제한 후 실행결과 및 이유, @Override의 역할

- 수행결과





- 이유 및 @Override 역할

문제 3번과 동일하게 Manager class에 getEarnings() 메소드가 존재하지 않기 때문에, getEarnings는 Manager 클래스 안에서 overriding되지 않고, Manager class의 super class인 SalariedEmployee 클래스에 정의되어 있는 getEarnings 메소드를 실행함으로써 overriding된다. Manager클래스의 constructor를 통해 weeklySalary가 2500으로 초기화되었으므로 SalariedEmployee클래스에 정의된 getEarnings 메소드를 실행시켜 얻은 반환값도 2500이 되어 출력된다. 다만, Manager클래스의 getearningsmethod에서 @Override를 삭제함으로써 Manager클래스의 getearningsmethod에서 에러메세지는 사

라졌다. 이는 getearnings 메소드가 상위 클래스에 있는 메소드를 overriding한 메소드임을 알려주지 않았기 때문에, 프로그램이 getearnings 메소드를 상위클래스에서 재정의된 메소드로 인식하지 않고, getearnings 메소드 자체를 그냥 새로운 메소드로 인식했기 때문이다. 따라서 @Override는 메소드가 Override된 메소드인지 확인해주는 역할을 한다고 볼 수 있다. 즉, subclass의 상위클래스에 있는 메소드를 재정의한 것이 맞는지(메소드이름의 철자 확인 등)를 확인하고, 맞지 않으면 에러메세지를 표시하는 역할을 한다. 추가적으로 @Override를 표시한 getearnings를 main함수에서 실행하면 getearnings는 오버라이딩된 메소드가 아니므로 컴파일에러를 낸다.

5. 문제5번에 주어진 코드를 main함수의 끝에 추가한 후 컴파일 결과 및 이유

- 수행결과

```
<terminated> PayrollSystemTest [Java Application] C:₩Program Files₩Java₩openjdk-11.0.2_windows-x
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The method setBonus(double) is undefined for the type Employee

        at PayrollSystemTest.main(PayrollSystemTest.java:35)
```

- 이유

컴파일에러가 발생한다. employees 배열은 Employee type의 배열이다. 따라서 employees[3]에 Employee클래스의 하위 클래스타입의 변수들을 assign할 수 있지만, 하위클래스의 메소드를 가져와 사용할 수는 없다. employees[3]자체는 Employee타입이므로, employees[3]은 Employee 클래스 내부에 정의된 메소드만 호출할 수 있다. Employee클래스 내부에는 setBonus()메소드가 정의되어있지 않으므로 컴파일에러가 발생한다.

6. 5번 문제를 해결하여 bonus가 반영된 급여가 출력되도록 하는 코드

- 수정된 코드

```
Manager manager = (Manager) employees[3];
manager.setBonus(1000.0);

System.out.printf("%s %s: $%,.2f%n%n", employees[3].getName(),"earned",employees[3].getEarnings());
```

- bonus가 반영된 급여 출력결과

```
Bob earned: $3,500.00
```

7. Employee class modifier에서 abstract를 없애고, getEarnings함수를 문제7번에 주어진대로 변경하여 수행한 결과 및 기존 프로그램과의 비교

- 실행결과	- 기존 프로그램과 비교
<terminated> PayrollSystemTest [Java Application Employees processed polymorphically: John earned $800.00 Karen earned $670.00 Sue earned $600.00 Bob earned $2,600.00	기존 프로그램(1번에서 실행한 프로그램)실행결과와 동일한 결과가 나온다. 기존 프로그램에서 Employee class의 modifier에서 abstract를 없애고, getEarnings 함수도 7번에서 주어진대로 변경하면, getEarnings함수는 dafault값을 리턴하는 함수가 된다. 하지만 Employee함수를 상위 클래스로 둔 subclass안에 getEarnings를 각각의 클래스에 맞게 재정의하였으므로, overriding에 의해 각 클래스에 정의된 getEarnings 메소드가 실행되면서 기존과 같은 결과를 출력하게 된다.

8. 7번처럼 변경하기 전과 변경한 후에 8번에서 주어진 코드를 main()함수에 추가해 실행 후 차이설명

7번처럼 변경하기 전에 Employee e = new Employee("Kildong", "000-00-0000");를 main함수에 추가하고 실행시키면 컴파일 에러를 발생시키고, 7번처럼 변경한 후에 Employee e = new Employee("Kildong", "000-00-0000");를 main함수에 추가하고 실행시키면 컴파일 에러를 발생시키지 않는다. 전자의 상황이 컴파일에러를 발생시키는 것은 abstract 클래스인 Employee는 변수로 사용할 수는 있지만, 객체를 생성하지는 못하기 때문이다. (abstract class는 아직 구현되지 않은 메소드 등이 선언되어있을 수 있기 때문이다.) 반면에 후자의 상황이 컴파일 에러를 발생시키지 않는 것은 Employee클래스가 abstract가 아니므로, getEarnings함수 또한 완성시켜주었으므로(default를 반환하는 함수), Employee는 concrete한 class가 된다. 따라서 모든 메소드들이 구현되어있으므로 Employee를 변수로만 사용하는 것이 아니라, 객체를 생성할 수도 있게 되어서 컴파일 에러가 발생하지 않는다.

9. 각 constructor에 하나의 출력문을 추가하여 상속을 하는 경우 constructor들이 일반적으로 어떤 순서로 실행되는지 확인 후 설명

- 실행결과	- 설명
<terminated> PayrollSystemTest [Java Application] C:¥ This is Employee's constructor This is SalariedEmployee's constructor This is Employee's constructor This is HourlyEmployee's constructor This is Employee's constructor This is CommissionEmployee's constructor This is Employee's constructor This is SalariedEmployee's constructor This is Manager's constructor Employees processed polymorphically: John earned $800.00 Karen earned $670.00 Sue earned $600.00 Bob earned $2,600.00	왼쪽 캡쳐본의 출력결과를 보면, subclass의 constructor가 실행되기 전에 superclass의 constructor가 먼저 실행되는 것을 알 수 있다. 특히 Manager 클래스는 SalariedEmployee를 superclass로 가지고 있고, SalariedEmployee 클래스는 Employee클래스를 superclass로 가지고 있으므로, 가장 상위 클래스인 Employee 클래스의 constructor를 가장 먼저 실행하고, SalariedEmployee의 constructor를 그 다음으로 실행하고, Manager의 constructor를 마지막으로 사용한다.

10. BasePlusCommissionEmployee클래스에 판매액에 따른 커미션 외에 기본 base salary가 소득에 추가되게끔 하는 getEarnings메소드 코드

- BasePlusCommissionEmployee class에 getEarnings 메소드를 채운 코드

```
30⊖    @Override
31     public double getEarnings()
32     {
33         return getCommissionRate() * getGrossSales() + getBaseSalary();
34     }
```