

객체지향프로그래밍 -6주차 실습 활동지 (2019년 10월 8일)

성명: 안창희 학과: 소프트웨어 학번: 201723272 실습실명: 318호

※ 본 실습활동지를 작성함에 있어 다른 학생의 문서로부터 일부 또는 전체를 복사하였습니까
예() 아니오(O) (복사 하였다면 예에 체크하고 아니라면 아니오에 체크하시오)

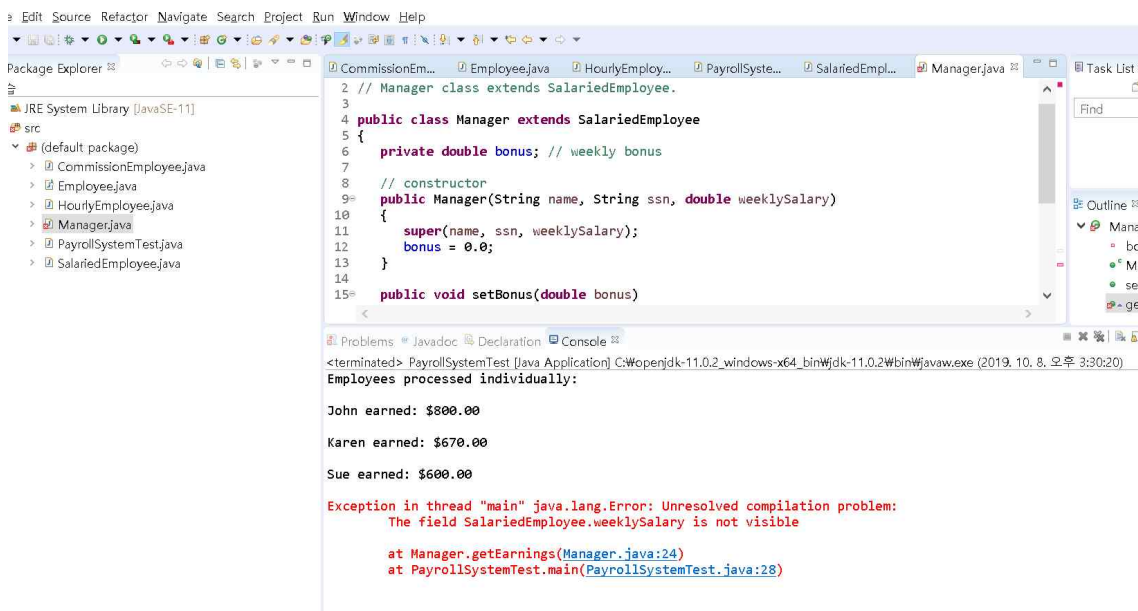
II. Review (9점)

1. Manager class의 getEarnings 메소드 본문을 아래와 같이 변경하여 컴파일해보시오. 어떤 결과가 나오는가? (컴파일 결과 및 설명 포함) (1점)

return weeklySalary + bonus;

```
18  
19  
20 // calculate earnings; override method earnings in CommissionEmployee  
21 @Override  
22 public double getEarnings()  
23 {  
24     return weeklySalary + bonus;  
25 }  
26  
27 } // end class BasePlusCommissionEmployee
```

실행전 컴파일 오류



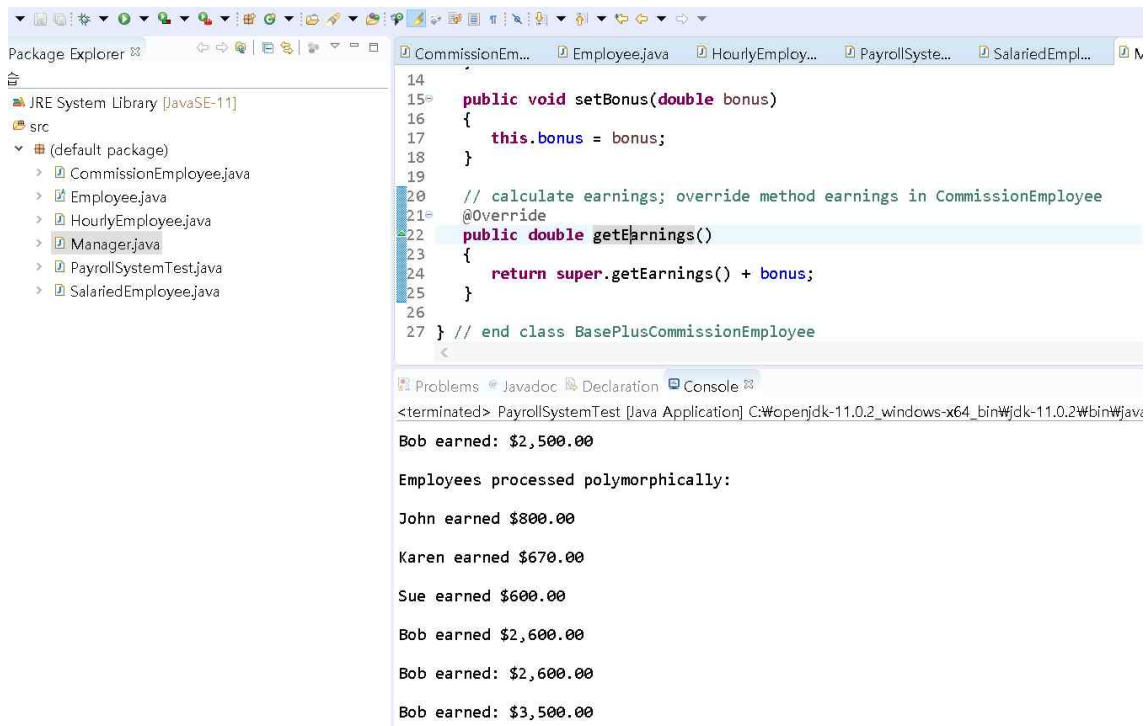
실행후 컴파일 오류

Manager 클래스에서는 public class Manager extends SalariedEmployee 라는 클래스 정의를 통해 super 클래스를 확장하는 입장이다 하지만 super 클래스에서 private double weeklySalary 라는 변수 선언을 통해 접근제어자 설정을 하였으므로 weeklySalary라는 변수에는 SalariedEmployee 안에서만 접근이 가능하다. 따라서 Manager 클래스에서 return weeklySalary + bonus; 라고 사용을 하여도 컴파일 에러가 뜰 수밖에 없다고 생각한다.

2. Manager class의 getEarnings 메소드의 이름을 getearnings으로 변경한 후 실행해 보시오. 어떤 결과가 나오는가? 왜 그런지 이유를 설명하시오. (수행결과 및 설명 포함) (1점)

```
16 {
17     this.bonus = bonus;
18 }
19
20 // calculate earnings; override method earnings in CommissionEmployee
21 @Override
22 public double getearnings()
23 {
24     return super.getEarnings() + bonus;
25 }
```

실행전 오류



```
<terminated> PayrollSystemTest [Java Application] C:\wopenjdk-11.0.2_windows-x64_bin\jdk-11.0.2\bin\java
Bob earned: $2,500.00

Employees processed polymorphically:

John earned $800.00

Karen earned $670.00

Sue earned $600.00

Bob earned $2,600.00

Bob earned: $2,600.00

Bob earned: $3,500.00
```

정상 실행결과

```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
src
  (default package)
    CommissionEmployee.java
    Employee.java
    HourlyEmployee.java
    Manager.java
    PayrollSystemTest.java
    SalariedEmployee.java
CommissionEm... Employee.java HourlyEmploy... PayrollSyste... Salarie...
55 }
56
57     System.out.printf("%s %s: $%,.2f%n%n",
58         manager.getName(),
59         "earned", employees[3].getEarnings());
60
61 Manager m = (Manager) employees[3];
62 m.setBonus(1000.0);
63     System.out.printf("%s %s: $%,.2f%n%n",
64         manager.getName(),
65         "earned", employees[3].getEarnings());
66
67 } // end main
68 } // end class PayrollSystemTest
Problems Javadoc Declaration Console
<terminated> PayrollSystemTest [Java Application] C:\openjdk-11.0.2_windows-x64_bin\jdk
Bob earned: $2,500.00

Employees processed polymorphically:

John earned $800.00
Karen earned $670.00
Sue earned $600.00
Bob earned $2,500.00
Bob earned: $2,500.00
Bob earned: $2,500.00
```

변경 후 실행결과

Manager class의 getEarnings 메소드의 이름을 getearnings으로 변경할 경우 PayrollSystemTest.java에서 printf를 통해 호출하는 클래스 변수 manager인 **"Bob"** 에게 적용되는 getEarnings() 메소드는 정상 실행결과에선 Manager 클래스의 getEarnings() 메소드가 호출되어 실행되기 때문에 초기에 Salary로 설정된 2500.0 에 Bonus 값을 더한 2600달러가 코드의 53번째 line으로부터 출력되고,

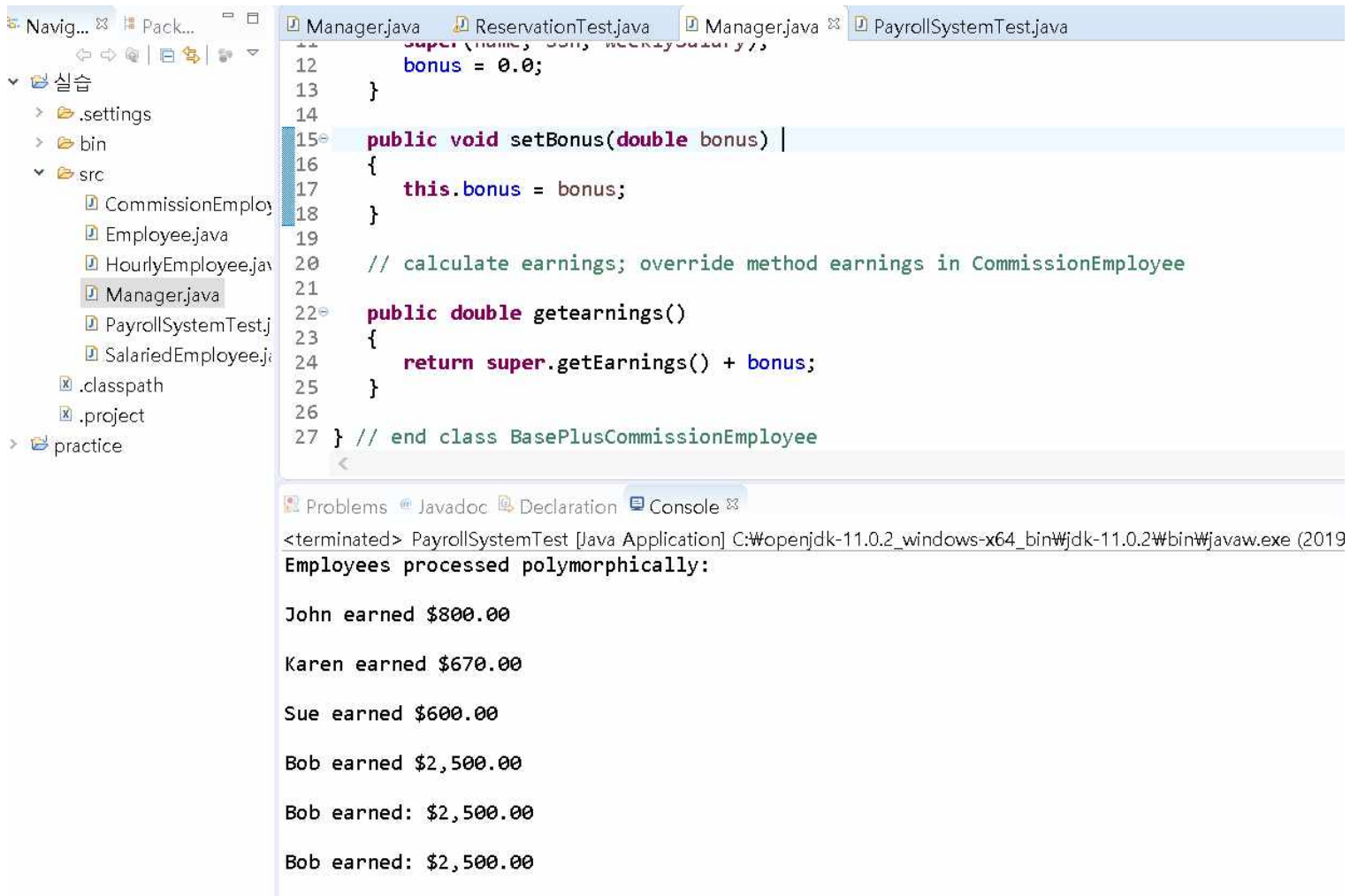
57번째 line에서도 같은 결과인 2600달러가 출력되며 65번째 line에서는 Bonus가 1000으로 설정된 후 다시 getEarnings() 메소드를 호출시켜 출력된 값인 3500 달러가 출력되게 된다.

하지만 Manager 클래스 변수 m을 외부에서 실행시켜서 Bonus 값을 1000으로 설정하든, 아니면 50번째 line처럼 setBonus를 하던 변경 후에 호출되는 getEarning은 SalariedEmployee클래스의 getEarning만이 호출되므로, 일관되게 2500달러로 주급이 출력되게 되는 것이다.

```
37     employees[3] = manager;
38
39     System.out.printf("Employees processed polymorphically:%n%n");
40
41     // generically process each element in array employees
42     for (Employee currentEmployee : employees)
43     {
44
45         // determine whether element is a Manager
46         if (currentEmployee instanceof Manager)
47         {
48             // downcast Employee reference to Manager reference
49             Manager employee = (Manager) currentEmployee;
50             employee.setBonus(100.0);
51         }
52
53         System.out.printf("%s earned $%,.2f%n%n", currentEmployee.getName(),
54             currentEmployee.getEarnings());
55     }
56
57     System.out.printf("%s %s: $%,.2f%n%n",
58         manager.getName(),
59         "earned", employees[3].getEarnings());
60
61 Manager m = (Manager) employees[3];
62 m.setBonus(1000.0);
63     System.out.printf("%s %s: $%,.2f%n%n",
64         manager.getName(),
65         "earned", employees[3].getEarnings());
66
67 } // end main
68 } // end class PayrollSystemTest
```

Line 설명의 이해를 돕기위한 그림 첨부

3. 2번에서 수정한 `getearnings` 메소드에서 `@Override`를 삭제한 후 실행해 보시오. 어떤 결과가 나오는가? 왜 그렇게 나오는지 설명하고 `@Override`의 역할에 대해 설명하시오. (실행 결과 및 설명 포함) (1점)



The screenshot shows an IDE with a project named 'practice'. The 'src' folder contains several Java files, including 'Manager.java'. The 'Manager.java' file is open, showing the following code:

```
12     bonus = 0.0;
13 }
14
15 public void setBonus(double bonus) {
16 {
17     this.bonus = bonus;
18 }
19
20 // calculate earnings; override method earnings in CommissionEmployee
21
22 public double getearnings()
23 {
24     return super.getEarnings() + bonus;
25 }
26
27 } // end class BasePlusCommissionEmployee
```

The console output shows the results of the program execution:

```
<terminated> PayrollSystemTest [Java Application] C:\Wopenjdk-11.0.2_windows-x64_bin\jdk-11.0.2\bin\javaw.exe (2019
Employees processed polymorphically:

John earned $800.00

Karen earned $670.00

Sue earned $600.00

Bob earned $2,500.00

Bob earned: $2,500.00

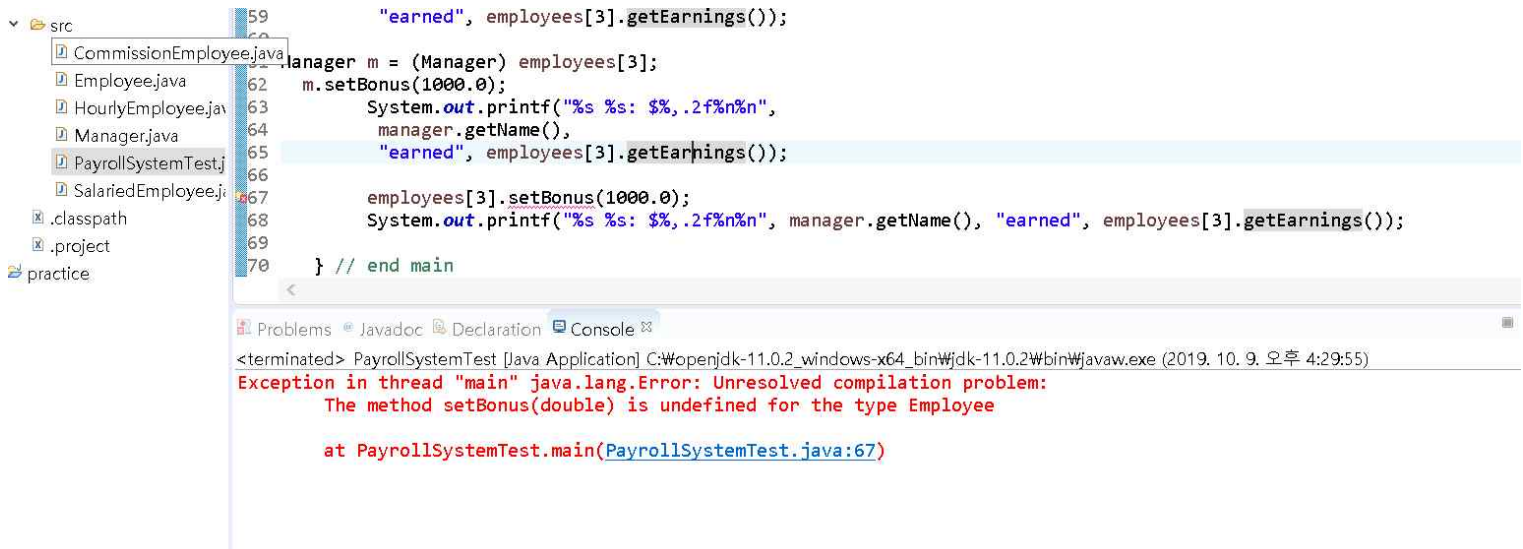
Bob earned: $2,500.00
```

@Override를 지운후 실행한 결과

출력 결과는 문제2)에서 나온 결과와 동일하며, 문제2)에서 컴파일 하기 이전에는 이클립스가 `@Override`를 감지하여 `getearnings()` 메소드가 상위 클래스의 메소드를 `override` 하는 것을 감지 못했기 때문에 오류라고 감지하고 빨간 줄이 그려져 있었지만, `@Override`를 삭제하고 이름이 `getearnings()`로 메소드를 바꾼 경우에는 아예 `Manager` 클래스에서만 처음 정의하는 메소드로 인식하기 때문에 이클립스가 오류를 감지하지 않는 것이 차이점이다. 문제2)에서와 마찬가지로 Bob의 earning은 `SalariedEmployee` 클래스의 `getEarning()` 메소드가 적용되어 2500 으로 세 번 일관되게 출력된다.

4. 다음은 employee[3]에 저장된 Manager object의 bonus를 증가시킨 후 정보를 출력하는 코드이다. PayrollSystemTest class의 main함수에 다음과 같은 코드를 넣은 후 컴파일시켜 보시오. 무슨 문제가 발생하며 왜 그런지 설명하시오. (컴파일 결과 및 설명 포함) (1.5점)

```
employees[3].setBonus(1000.0);
System.out.printf("%s %s: $%,.2f%n%n", manager.getName(), "earned",
employees[3].getEarnings());
```



```
59         "earned", employees[3].getEarnings());
60     }
61     Manager m = (Manager) employees[3];
62     m.setBonus(1000.0);
63     System.out.printf("%s %s: $%,.2f%n%n",
64         manager.getName(),
65         "earned", employees[3].getEarnings());
66
67     employees[3].setBonus(1000.0);
68     System.out.printf("%s %s: $%,.2f%n%n", manager.getName(), "earned", employees[3].getEarnings());
69
70 } // end main
```

Problems | Javadoc | Declaration | Console

<terminated> PayrollSystemTest [Java Application] C:\openjdk-11.0.2_windows-x64_bin\jdk-11.0.2\bin\javaw.exe (2019. 10. 9. 오후 4:29:55)

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:67)

원래의 PayrollSystemTest.java 코드에 주어진 코드를 추가시켜서 실행시켜 보았다. 제대로 작동하던 Line 61~65의 코드와는 다르게 새로 추가한 Line 67~68의 코드는 에러를 일으켰으며 Employee 형에게 setBonus라는 메소드는 정의되지 않았다는 에러문이 출력된다. setBonus는 Manager 형에게만 적용되는 메소드 이므로 컴파일 에러가 나는 것이다.

employee 변수안에 Manager 형 객체가 들어있다고 해도 틀이 Employee 였기 때문에 getBonus라는 메소드를 적용시키기 위해선 위의 선언처럼 Manager m =(Manager) employees[3] 이렇게 강제 형변환 후 m.setBonus(1000.0) 라고 적용시키는 방법이 적합하다.

5. 4번 문제의 문제를 해결하여 bonus가 반영된 급여가 출력될 수 있도록 코드를 수정해보시오. (해결부분 코드 포함) (1.5점)



```
67     Manager m2 = (Manager) employees[3];
68     m2.setBonus(1000.0);
69     System.out.printf("%s %s: $%,.2f%n%n", manager.getName(), "earned", employees[3].getEarnings());
70
71 } // end main
72 } // end class PayrollSystemTest
```

Problems | Javadoc | Declaration | Console

<terminated> PayrollSystemTest [Java Application] C:\openjdk-11.0.2_windows-x64_bin\jdk-11.0.2\bin\javaw.exe (2019. 10. 9. 오후 4:41:05)

John earned \$800.00

Karen earned \$670.00

Sue earned \$600.00

Bob earned \$2,600.00

Bob earned: \$2,600.00

Bob earned: \$3,500.00

Bob earned: \$3,500.00

코드를 수정한 후 출력한 결과 => 밑에 3500달러의 earning이 하나 더 추가되었다.

6. Employee class modifier에서 다음과 같이 abstract를 없애고 public class Employee 또한 getEarnings 함수를 다음과 같이 변경한다.

```
public double getEarnings() { return 0.0; }
```

이 프로그램을 수행하면 결과가 어떻게 나오나? 기존 프로그램과 비교하시오. (설명만 포함) (1점)

기존 프로그램과의 출력 결과는 동일하다. Employee 클래스와 getEarnings 메소드는 abstract가 쓰여져서 실제로 구현되는 일 없이 선언만 되어있는 상태였고, 하위 클래스에서 모두 재정의가 되어있는 상태였다. 따라서 main에서도 Employee 클래스나 Employee 클래스에 적용되는 getEarnings 메소드는 사용되고 있지 않았었고, 결과에 반영되는 일 또한 없었다. 따라서 abstract문을 없애고 실제 클래스와 메소드로 정의해놓아도, main 코드상에서 구현되어있지 않기 때문에 결과에 반영되는 일이 없으며, 모두 하위에서 재정의 되어있으므로 오류 또한 발생될 일이 없다.

7. 6번처럼 변경하기 전과 변경한 후에 아래 코드를 main()함수에 추가하여 실행해보시오. 어떤 차이가 나며 왜 그런지 설명하시오. (설명만 포함) (1점)

```
Employee e = new Employee("Kildong", "000-00-0000");
```

변경한 후의 실행에선 아무런 이상이 없고 출력결과가 동일하지만 변경전의 코드에 위의 코드를 추가했을 경우 Cannot instantiate the type Employee 라는 오류메세지가 출력된다. 변경전의 Employee 클래스는 abstract하게 정의되어있을 뿐이지 객체를 생성할 수 없고 하위에서 재정의 하기 위한 클래스일 뿐이다. 하지만 이것을 이용해 객체를 역지로 생성하려고 하면 컴파일상의 에러가 발생할 수밖에 없다.

8. 각 constructor에 하나의 출력문을 추가하여 수행해본 후 상속을 하는 경우 constructor들이 일반적으로 어떤 순서로 실행되는지 설명하시오. (실행결과 화면 및 설명포함) (1점)

```
19      System.out.printf("%n%s %s: $%,.2f%n%n",
20                          salariedEmployee.getName(), "earned", salariedEmployee.getEarnings());
21      System.out.printf("%s %s: $%,.2f%n%n",
22                          hourlyEmployee.getName(), "earned", hourlyEmployee.getEarnings());
23      System.out.printf("%s %s: $%,.2f%n%n",
24                          commissionEmployee.getName(), "earned", commissionEmployee.getEarnings());
25      System.out.printf("%s %s: $%,.2f%n%n",
26                          manager.getName(),
27                          "earned", manager.getEarnings());
28
29      // create four-element Employee array
30      Employee[] employees = new Employee[4];
31
32      // initialize array with Employees
33      employees[0] = salariedEmployee;
34      employees[1] = hourlyEmployee;
35      employees[2] = commissionEmployee;
36      employees[3] = manager;
```

Problems | Javadoc | Declaration | Console

```
<terminated> PayrollSystemTest [Java Application] C:\wopenjdk-11.0.2_windows-x64_bin\jdk-11.0.2\bin\javaw.exe (2019. 10. 9)
Employee 생성합니다!
SalariedEmployee 생성합니다!
Employee 생성합니다!
HourlyEmployee 생성합니다!
Employee 생성합니다!
CommissionEmployee 생성합니다!
Employee 생성합니다!
SalariedEmployee 생성합니다!
Manager 생성합니다!
Employees processed individually:
```

John, Karen, Sue, Bob을 constructor로 생성하는 과정은 하나의 객체마다 최상위 클래스의 constructor를 가장먼저 거치고 다음의 상위클래스를 거치고 하는 순서대로 거치며 생성되는 것을 확인할 수 있었다.

John의 경우엔 Employee 생성 => SalariedEmployee 생성(확장)

Karen의 경우엔 Employee 생성 => HourlyEmployee 생성(확장)

Sue의 경우엔 Employee 생성 => ComissionEmployee 생성(확장)

Bob의 경우엔 Employee 생성 => SalariedEmployee 생성(1차확장) => Manager 생성(2차확장)

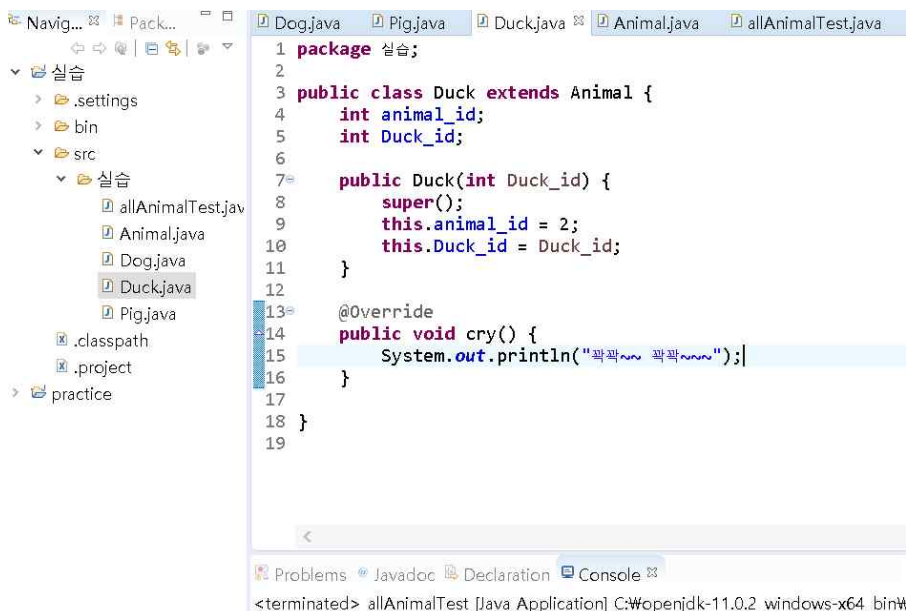
위와같이 constructor들의 실행순서가 보여진다고 할수 있다.

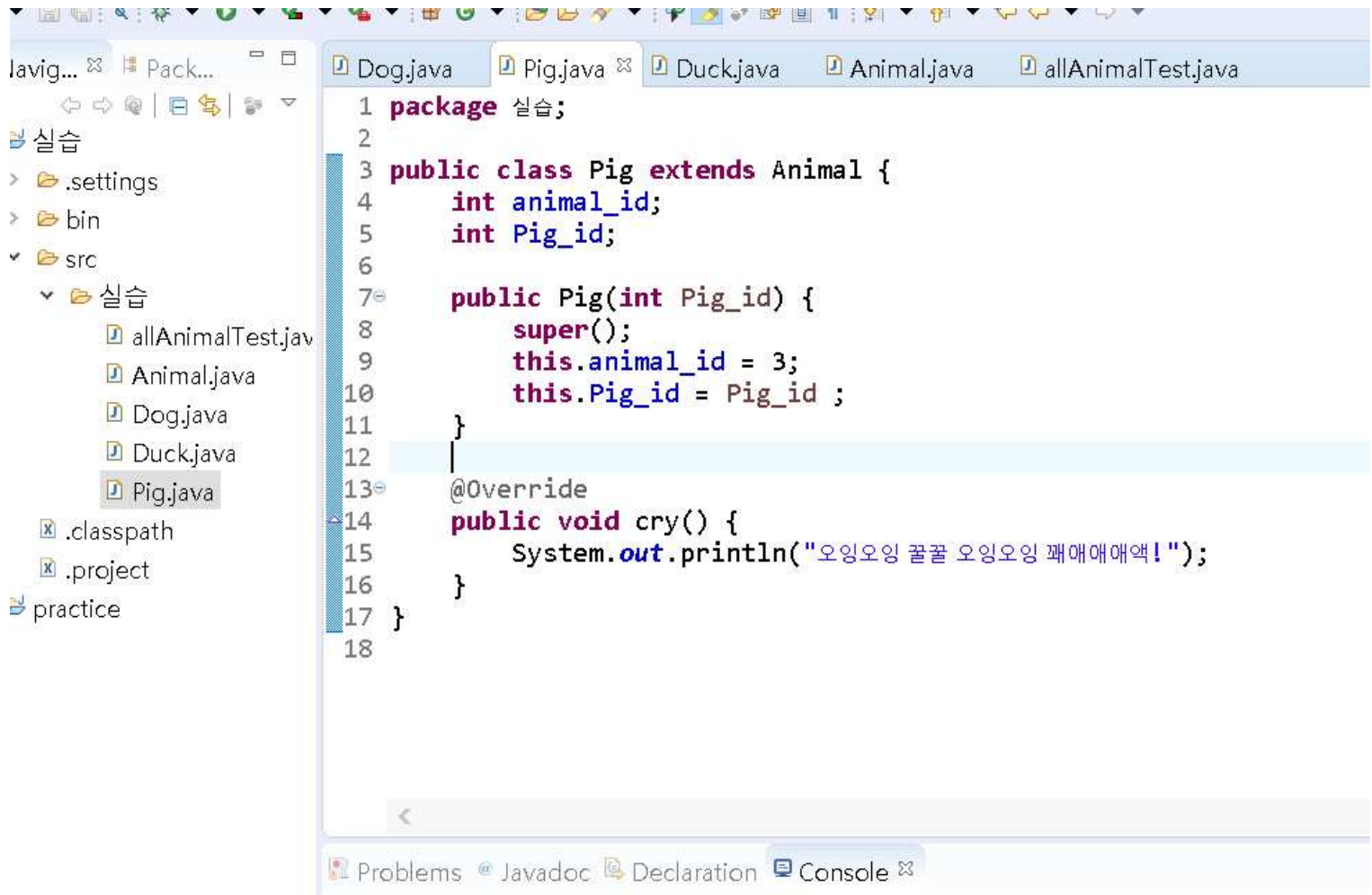
III. Exercises (6점)

가) 각 동물의 클래스를 정의하고 다음과 같은 울음소리를 내는 method cry()를 정의하여야 한다. 단, 울음소리를 내는 작업은 울음소리에 해당하는 문자열을 출력하는 것으로 대체한다. (3점)

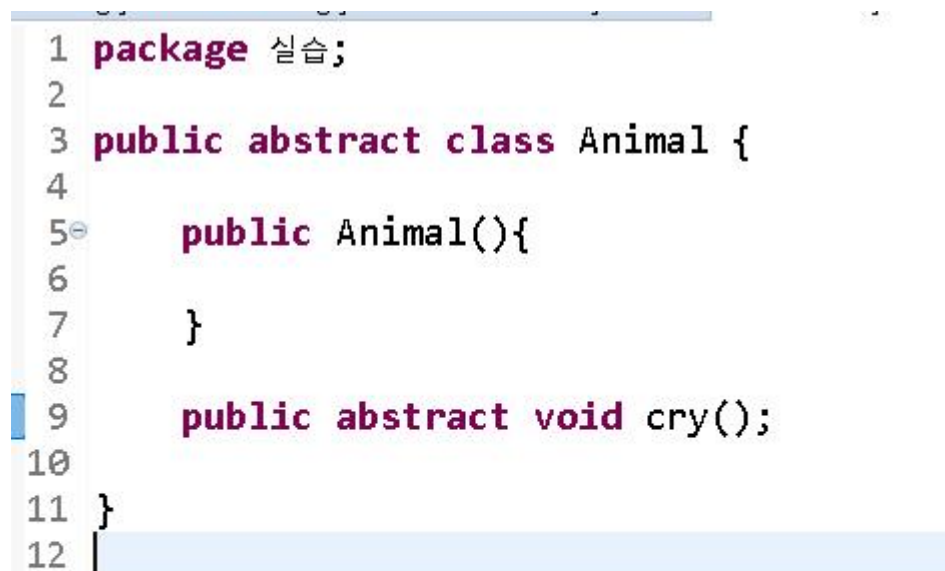
```
void cry() { ... }
```

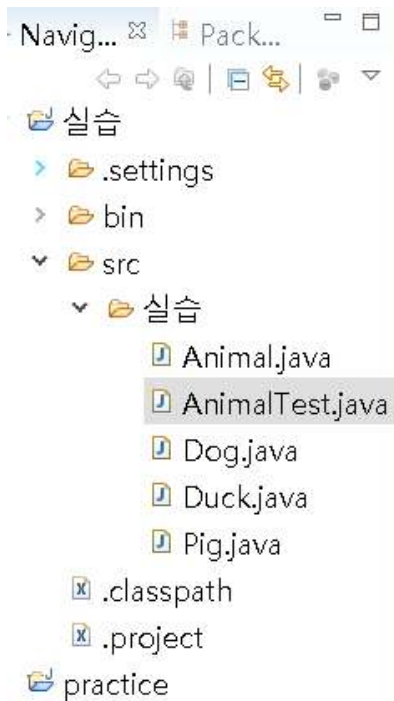
```
1 package 실습;
2
3 public class Dog extends Animal {
4     int animal_id;
5     int Dog_id;
6
7     public Dog(int Dog_id) {
8         super();
9         this.animal_id = 1;
10        this.Dog_id=Dog_id;
11    }
12
13    @Override
14    public void cry() {
15        System.out.println("멍멍!! 왈왈 크르르르 멍멍!!");
16    }
17
18 }
```





나) 세 종류의 동물의 공통 superclass Animal을 정의하고 abstract method cry()를 정의하시오. 각 종류의 동물을 적어도 하나 이상 생성하여 Animal 배열에 저장한 후 배열에 저장된 각 동물의 울음소리 를 순서대로 출력하는 AnimalTest class를 작성하여 실행하시오. (3점)





```
Dog.java  Pig.java  Duck.java  Animal.java  AnimalTest.java
1 package 실습;
2
3 public class AnimalTest {
4
5     public static void main(String[] args) {
6
7         Dog dog = new Dog(1);
8         Duck duck = new Duck(1);
9         Pig pig = new Pig(1);
10
11         Animal[] Animals = new Animal[3];
12
13         Animals[0] = dog;
14         Animals[1] = duck;
15         Animals[2] = pig;
16
17         for(Animal current : Animals) {
18             current.cry();
19         }
20
21
22
```

Problems Javadoc Declaration Console

<terminated> AnimalTest [Java Application] C:\w\openjdk-11.0.2_windows-x64_bin\bin\java.exe
멍멍!! 왈왈 크르르르 멍멍!!
꽹꽹~~ 짹꽹~~
오잉오잉 꿀꿀 오잉오잉 짹애애애!