**LAB5\_assignment**

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | 신민선 | Department: | 미디어학과 |
| Student ID: | 201723307 | Room Number: | 팔달관 333호 |
| Due Date: | (March 30, 11:30 ) | | |

**II. Exercises (31 points)**

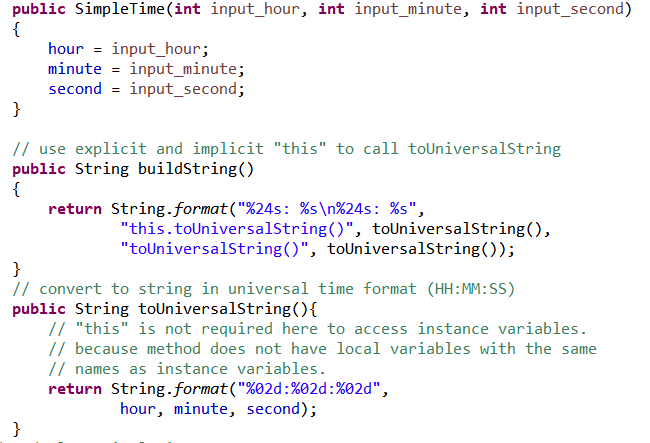
1.

A. ‘this’ reference is reference valuable and indicate own instance.

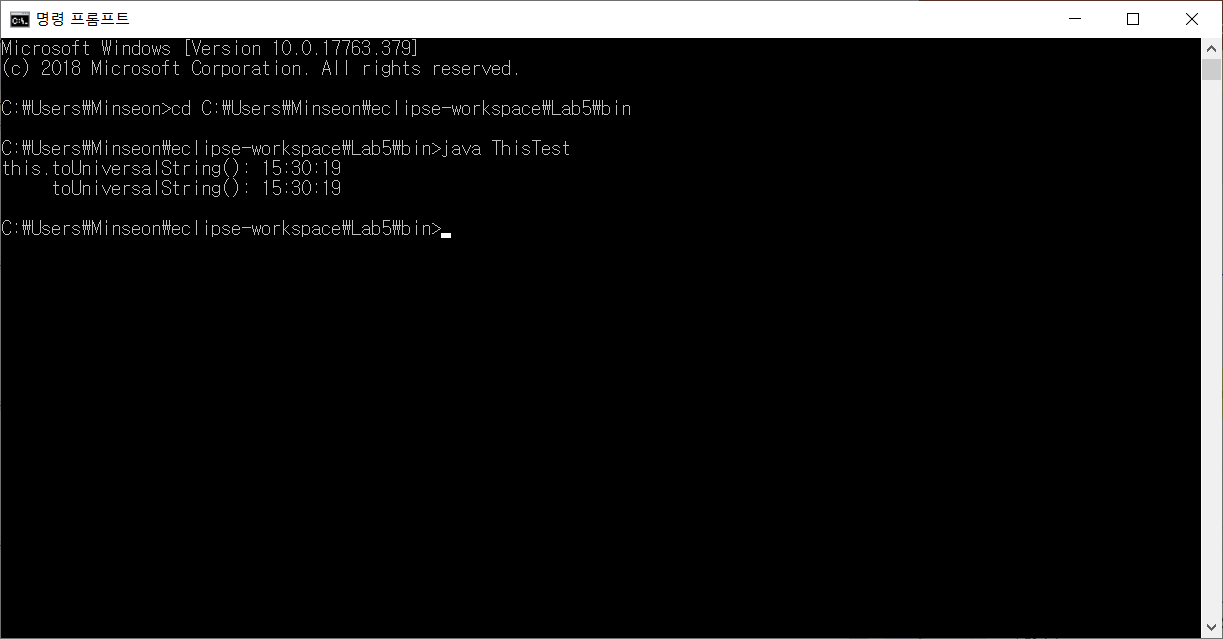
in this FIG.8.4 code, universalString() method is located in simpleTime class and buildString() method (that return string to this. To universalString and to universalString) is also located same area. So they indicate same method in same instance.

B.

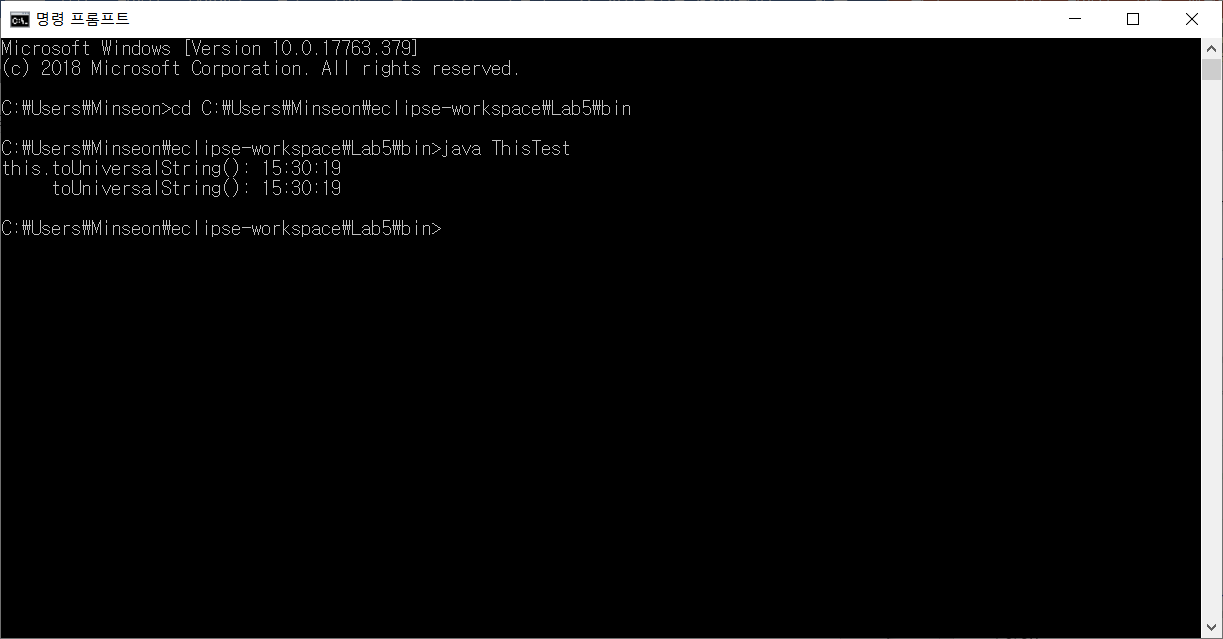
Code\_capture)



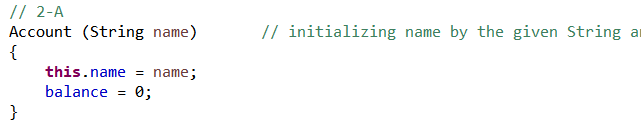
**Cmd capture\_before(use ‘this’)**

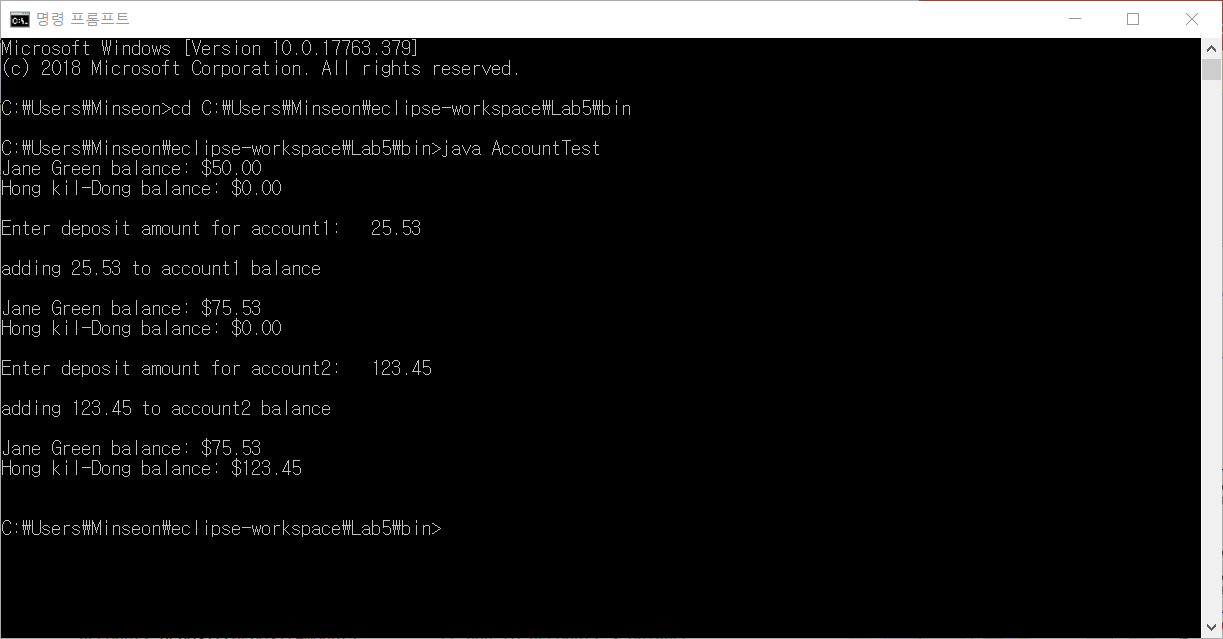
****

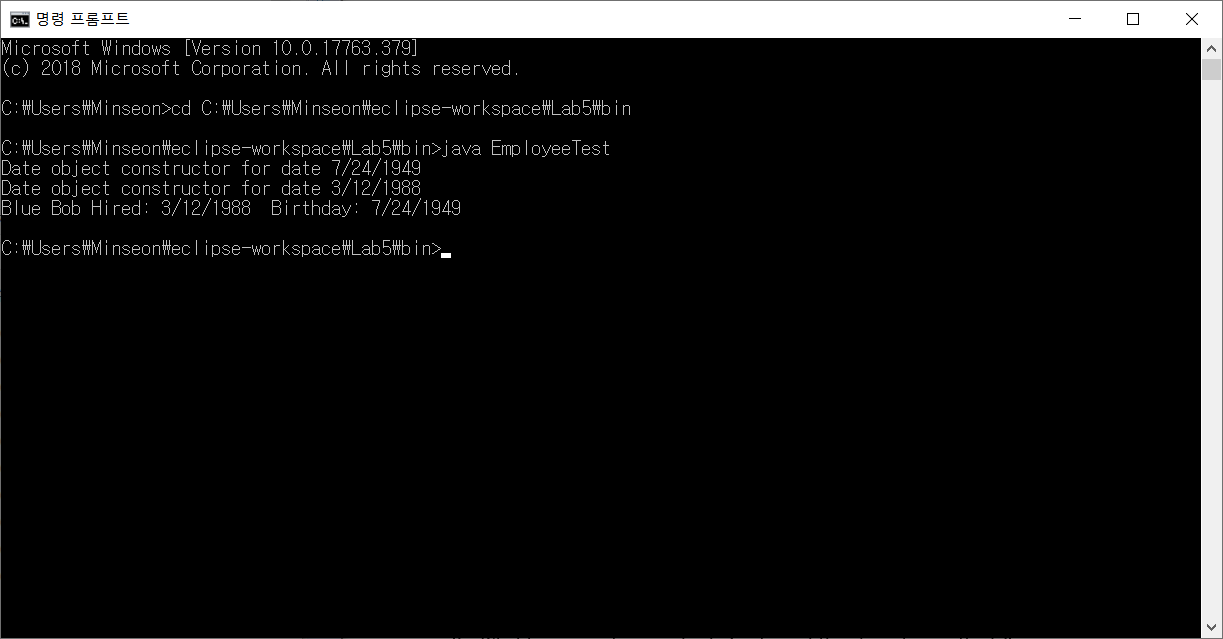
Cmd capture\_after (no use ‘this’)

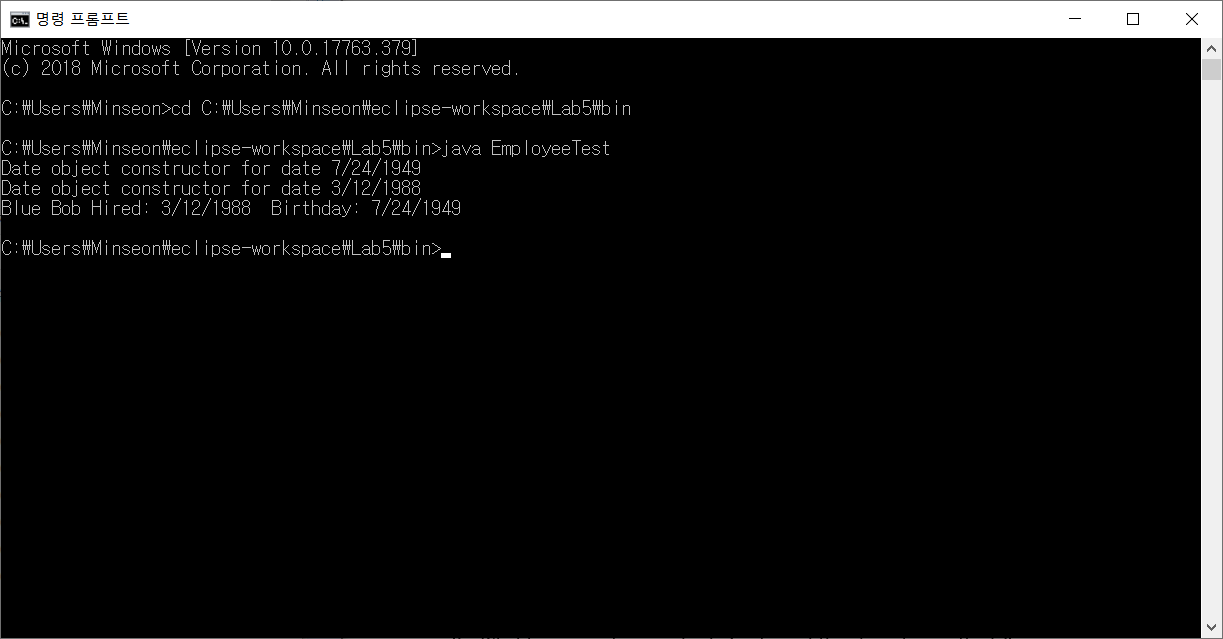


2. A.



 B.

3. A. Before & After cmd capture



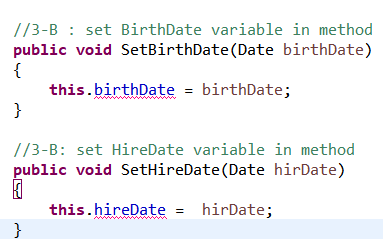
Reason: There are same result about before and after. Because of feature of ‘final’, they have same result. If you use this ‘final’ modifier, final variable can’t modify original(initialize) value and final method can’t override itself.

Therefore, the result is not changed.

B.

1. **What happens when you try to implement the above two methods?**

**First, I made two method to initialize two variables(SetBirthDate(), SetHireDate()).**



And then, remaining ‘final modifier’ in variables, I try to run the codes but failed.

Complier let me show the reason like that picture.



That means variable initialization using final variable can’t be done in method at runtime. So if we want to use that method, we have to erase ‘final’ or just initialize.

* 1. What result you get? Explain your reason.

**Public Employee**() { }

**When we use this constructor, we have to get this same Error.**

**Because of ‘final modifier’, the constructor must initialize ‘final’ variable only one time.**

* 1. **Find all places where polymorphism is used in Fig 10.1. (2 points).**



Because CommissionEmployee class is parent class and BasePlusCommissionEmployee class is child class, this line show the polymorphism.

* 1. **After line 34 in Fig 10.1, insert the following code and run the program. What happens? Why does that happen? (2 points)**

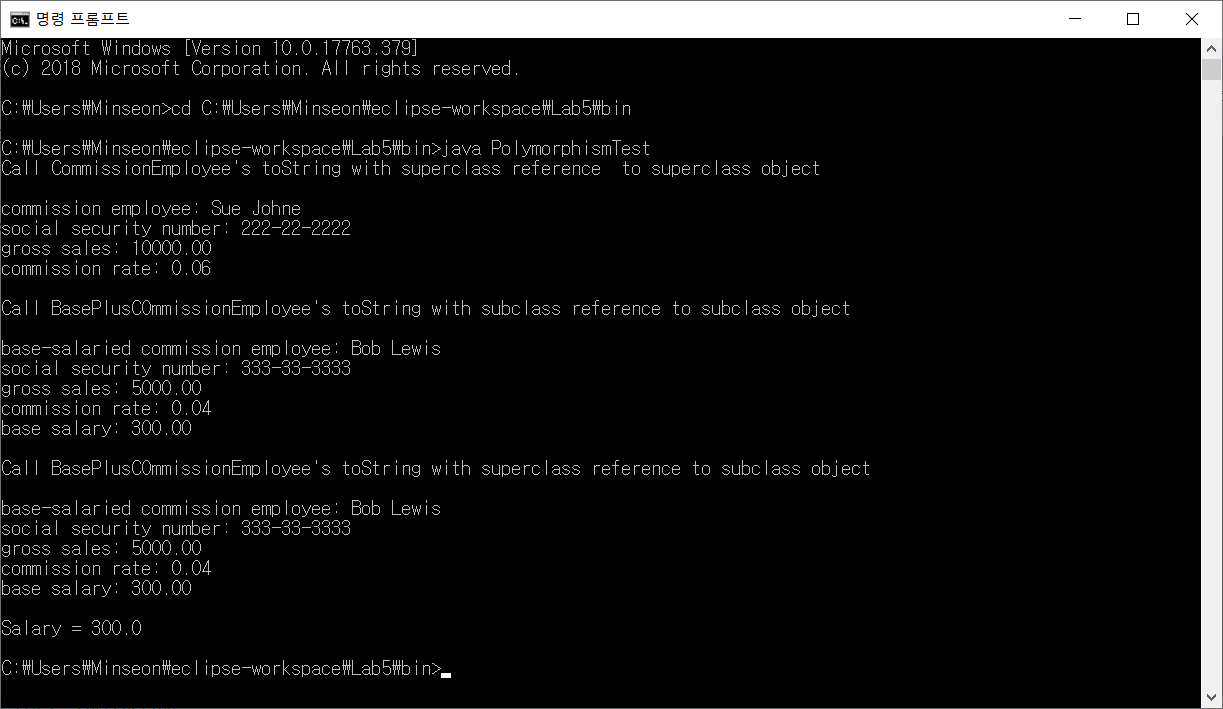


This is the Error.

As I said, BasePlusCommissionEmployee is child class and CommissionEmployee class is parent class in polymorphism. So parent class can’t use child class.

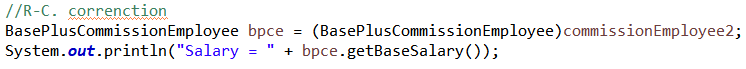
If you want to correct Error, you have to cast.(type)

* 1. **Replace the code in B by the following code and run the program. What is different between program in A and program B ? (2 points)**



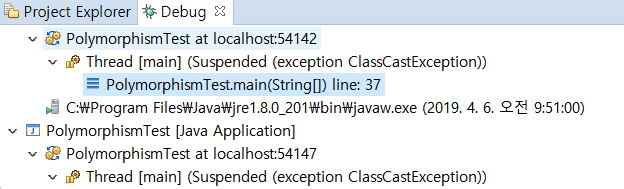
The program is normally running.

In A code, type of commissionEmployee2 is parent class and in B code, type of commissionEmployee2 is basePlusCommissionEmployee(child class) through casting. As a result,

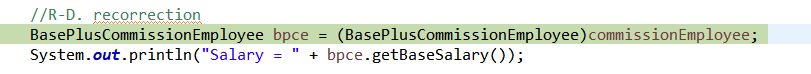


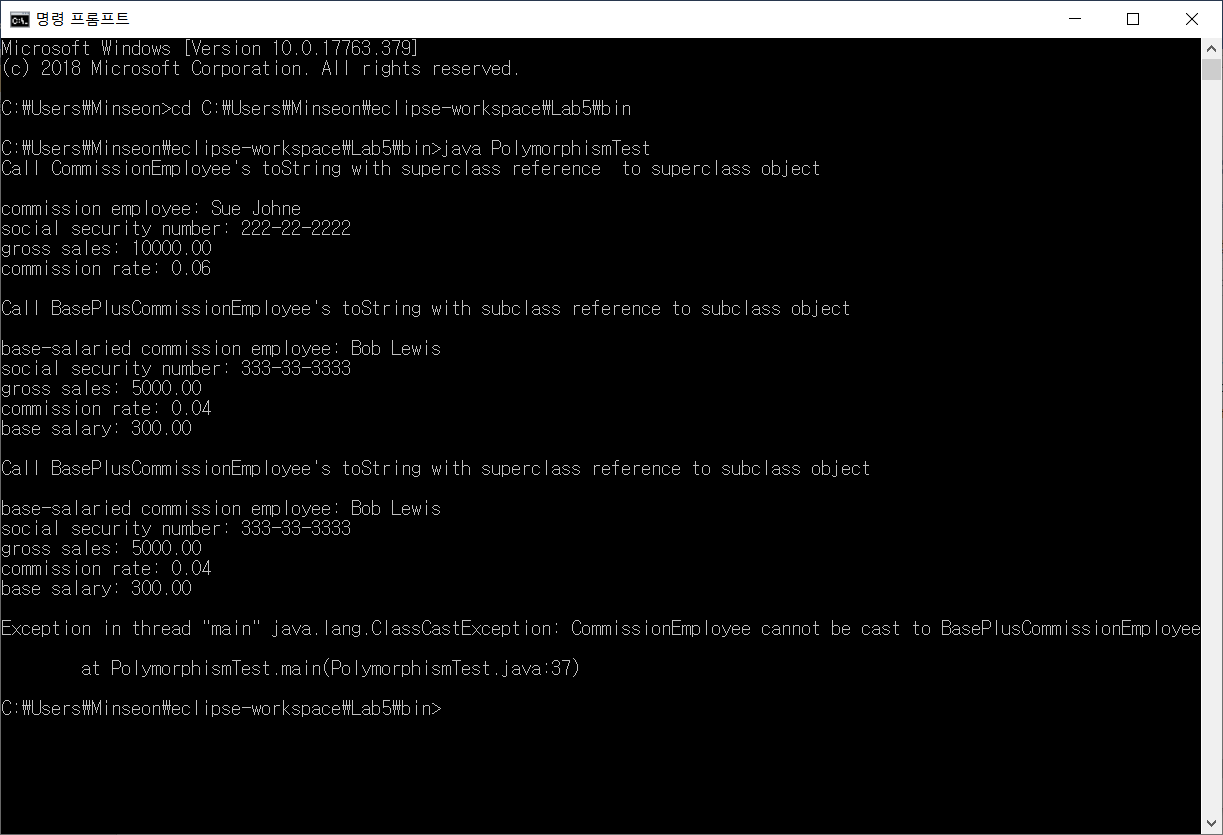
In this code, because type of between ‘bpce’ and ‘commissionEmployee2’ are same, so program don’t pop up any error.

* 1. **Replace the code in C by the following code and run the program. What happens? Why does it happen? (2 points)**

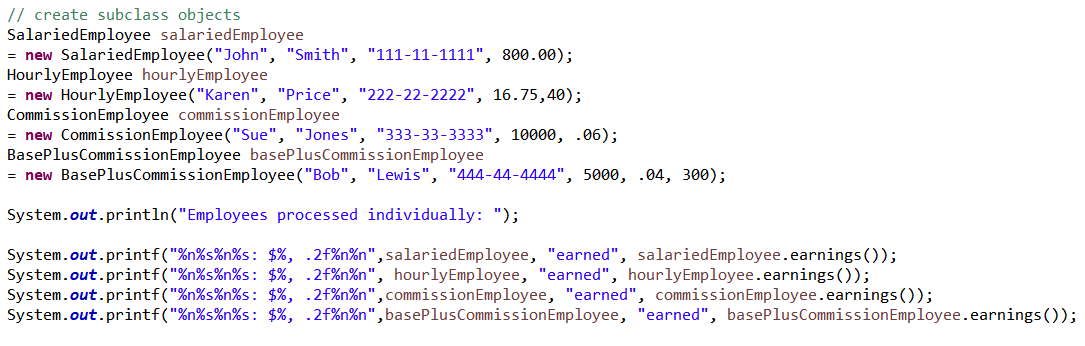


It has a Error.



You can’t assign a value to another instance because CommissionEmployee class has variables declared ‘final modifier’. 

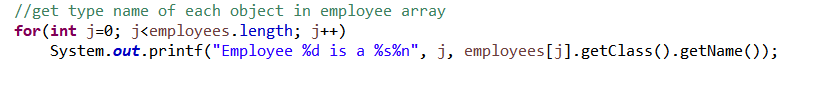
* 1. **The output of the program has two different parts. Explain the output of each part (2 points)**

First part is that they access each instance and print out result string. 

Second part is that makes an array(Employee[]) and outputs it through for each array statement. And add ‘ new base salary with 10% increase’ statement.

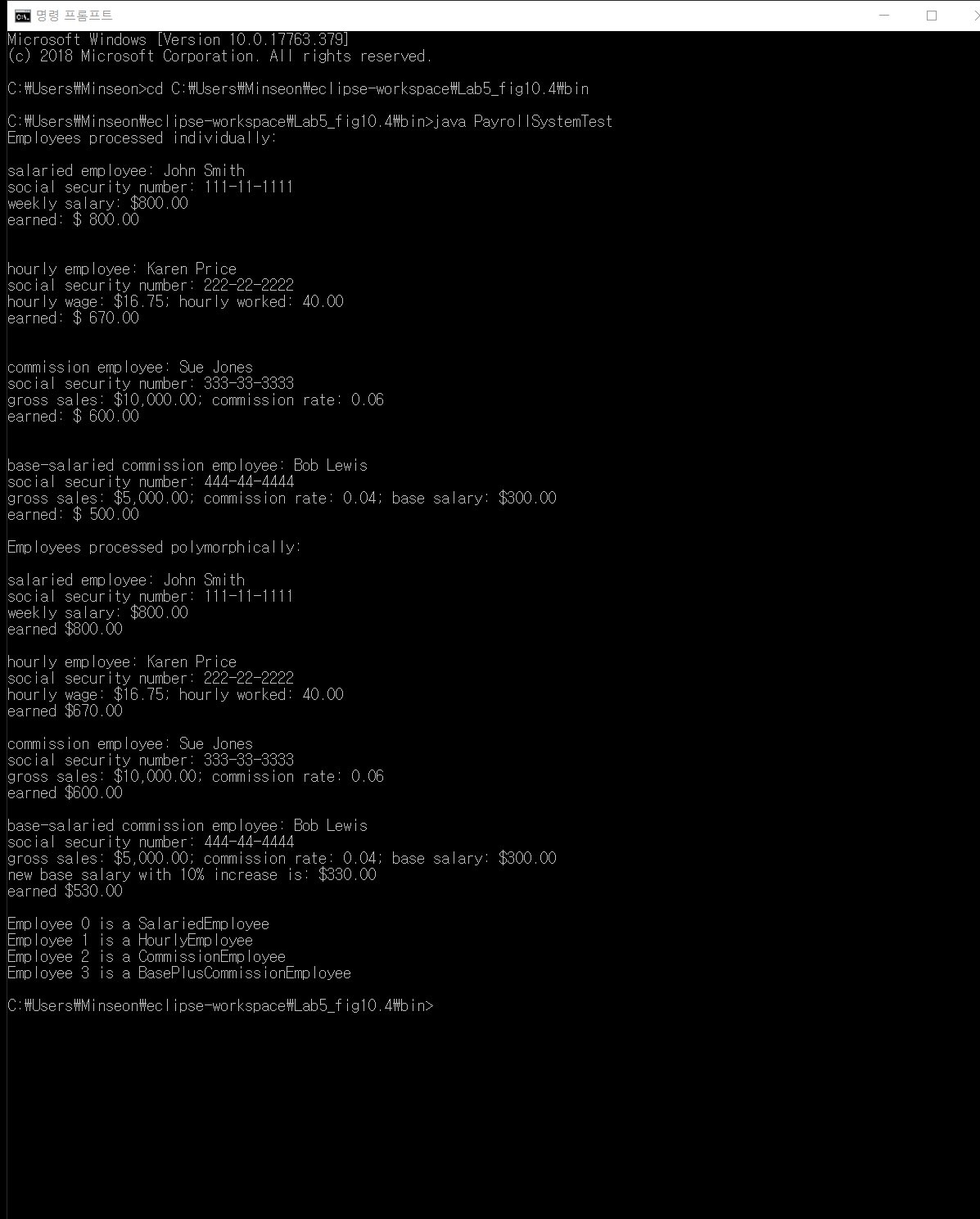


The third output statement accesses the getclass method of the 'object' class and prints the class name.

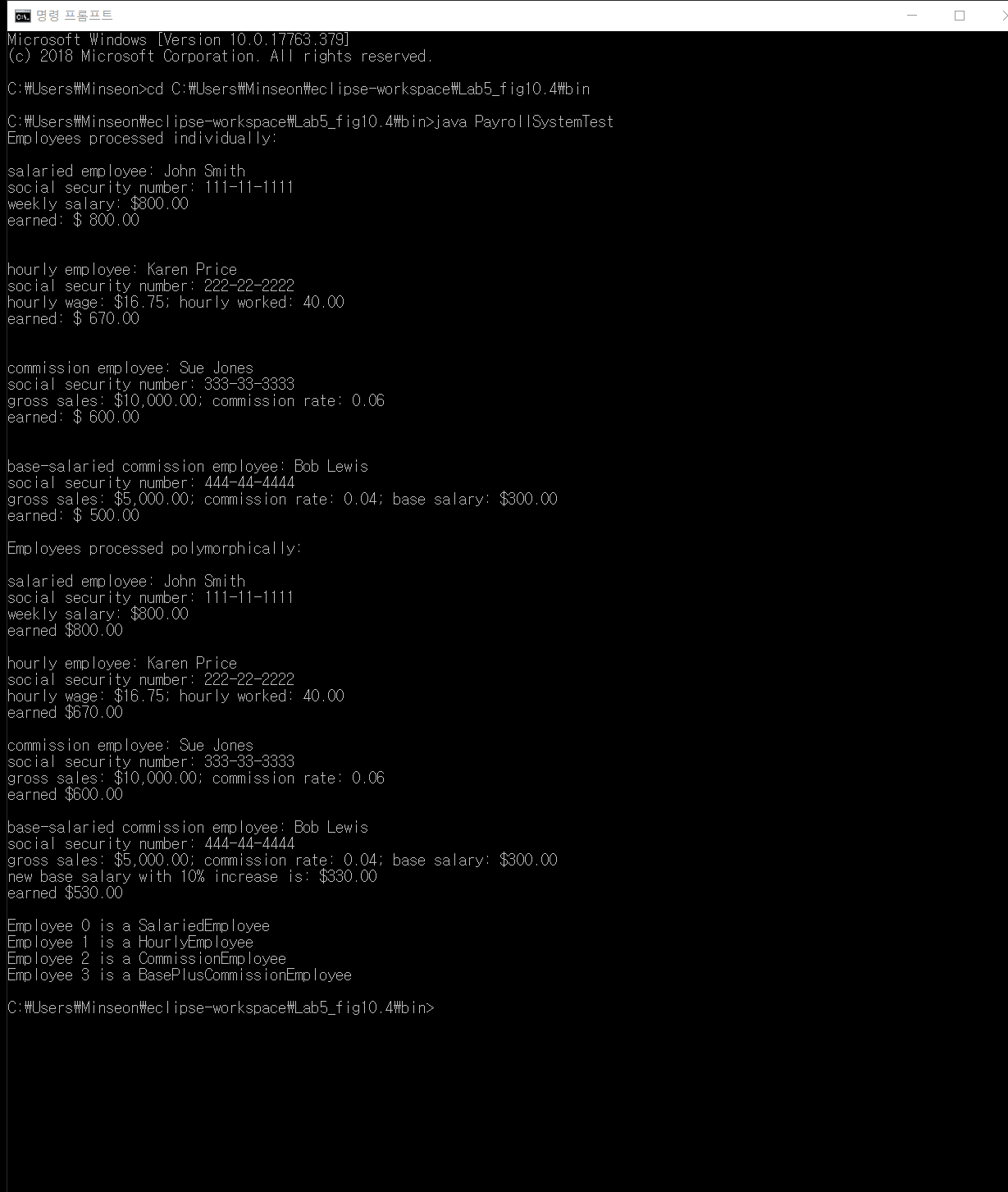


* 1. **Change the code in Employee class as follows,**

**Before screenshot**



**After screenshot**



**Before modification, there is a ‘abstract’ class named Employee and one ‘abstract’ method in the ‘abstract’ class. After modification from ‘abstract’ to ‘non-abstract’, Normally,** the abstract method was implemented in another class. Therefore it was executed without problems

**As a result, there is no change in result**

* 1. **What is the difference between the two? Explain the reason (2 points)**

Before changing, There is a compiler error.



**After changing, there is no error.**



**Reason :**

**Before changing(‘abstract), we can’t make an instance for abstract class.**

**But after changing (delete abstract) we can make an instance for save information. So now, we don’t have any error.**

1. **a&b.**

**Modify the program by adding the following subclasses of the Employee class (see Fig.10.2 and Fig.10.10) (2pt).**

(check java code file)

First, I put three class in these project, Commission, Hourly and BasePlus.

And for each class, I add getPaymentAmount method and modify earning method. Because payable is interface class, and that condition must have getPayementAmount method.



**(code will attach in code file, please check folder.)**

* 1. Define a class of each animal including at least **one method** as follows.

