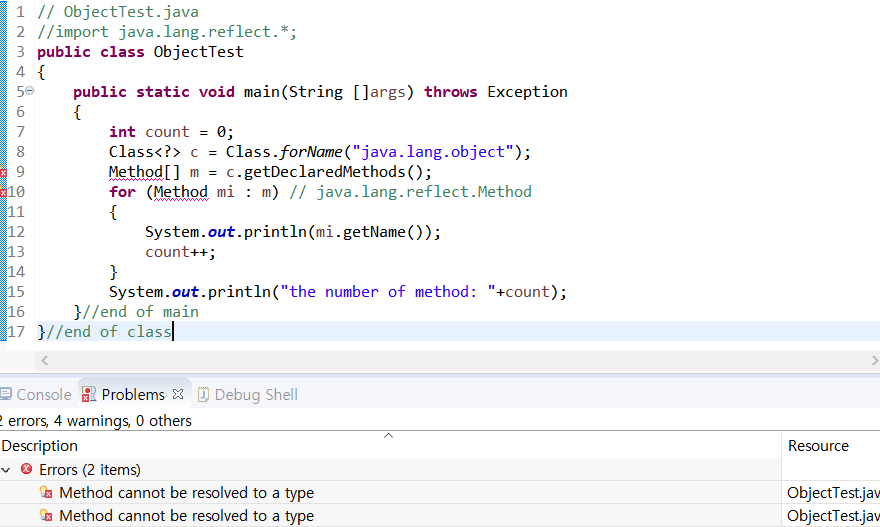
**LAB6\_assignment**

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

**Problem 1(): Use of “Class” class and reflection API.**

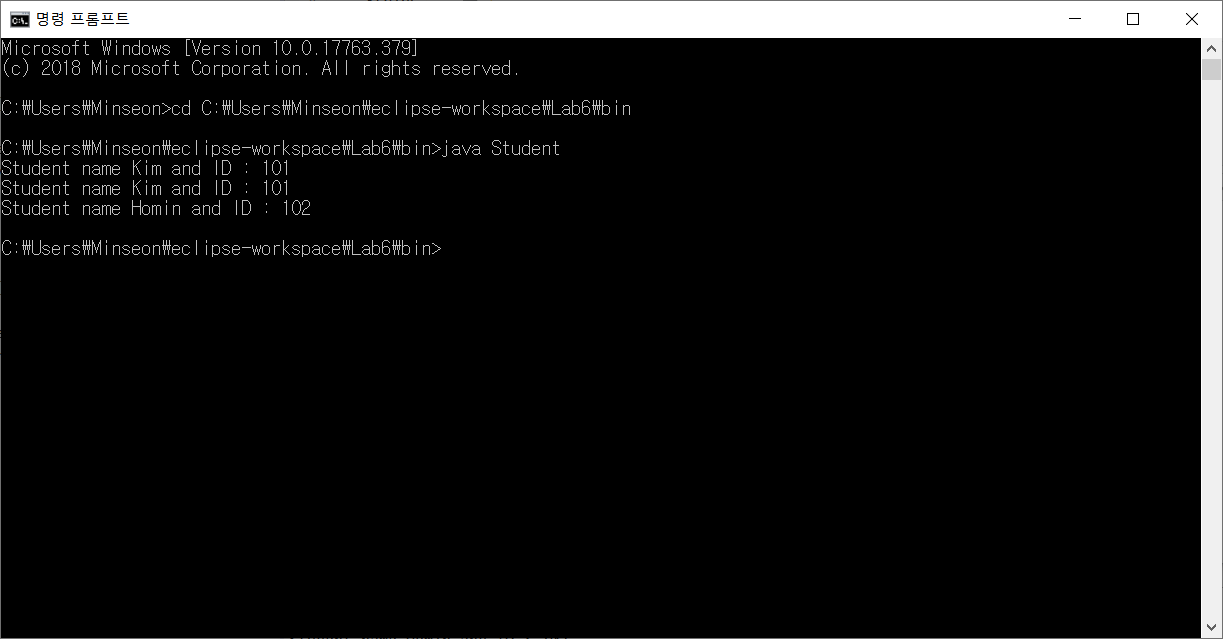
**Question :** If you delete line 1, what happens? (1pt)



**If I delete line 1 : “import java.lang.reflect”, then Error line has come. Because they can’t find ‘Method’ class and in java.lang.reflect, they have method class.**

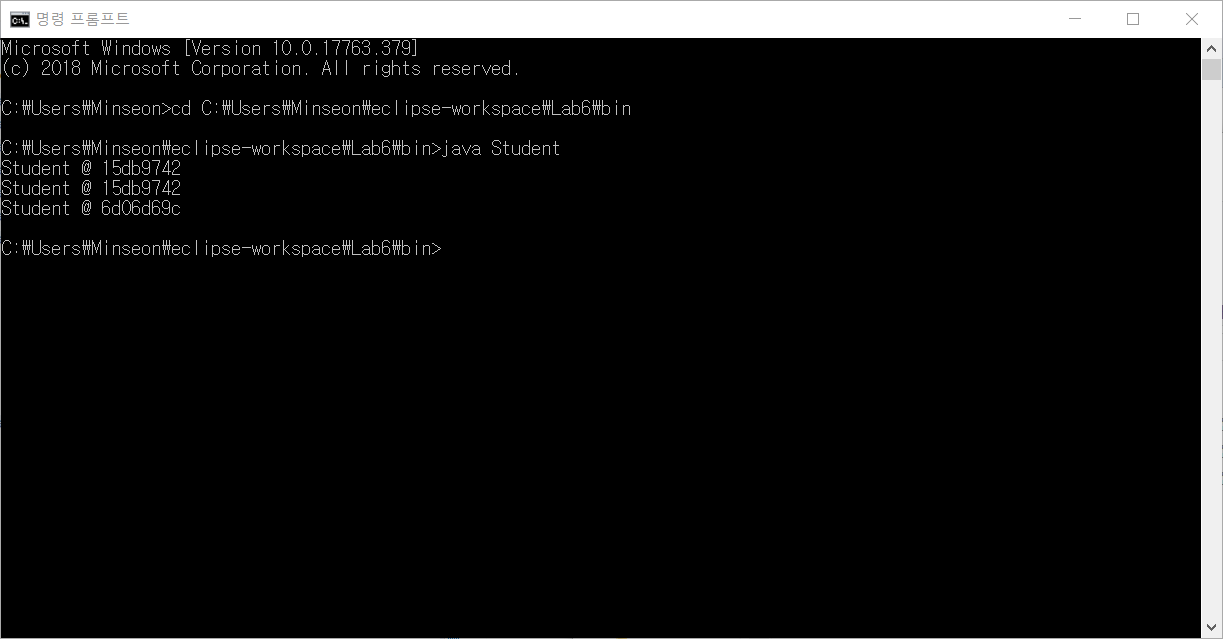
**Problem 2 : toString** () method of “**Object**” class ()

**Question(a):** run the following code and take the screenshot (1pt)



**Question(b):** **comment Line 10-12(toString() function) and run the program. Show the screen shoot. Why the screen shot is different from Question (a)? Explain your answer.**

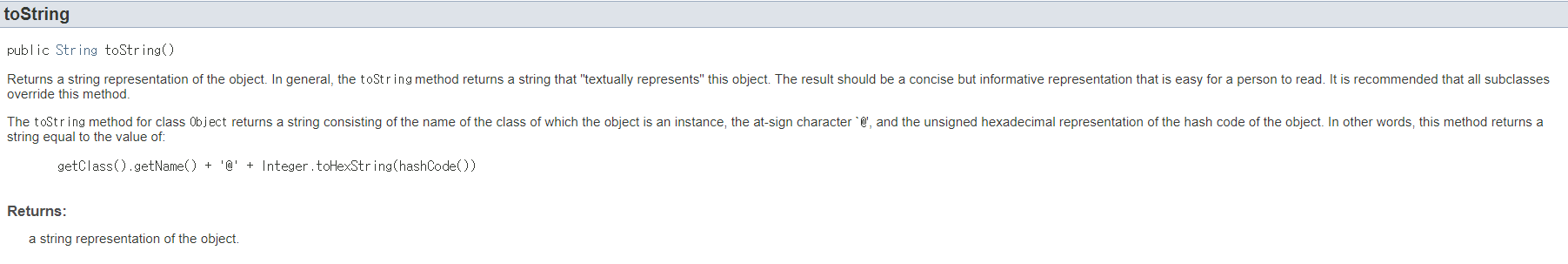
**Hint: the** standard implementation of **toString()** method in the **Object** class is as follows (3pt).



It is different from Question(a).

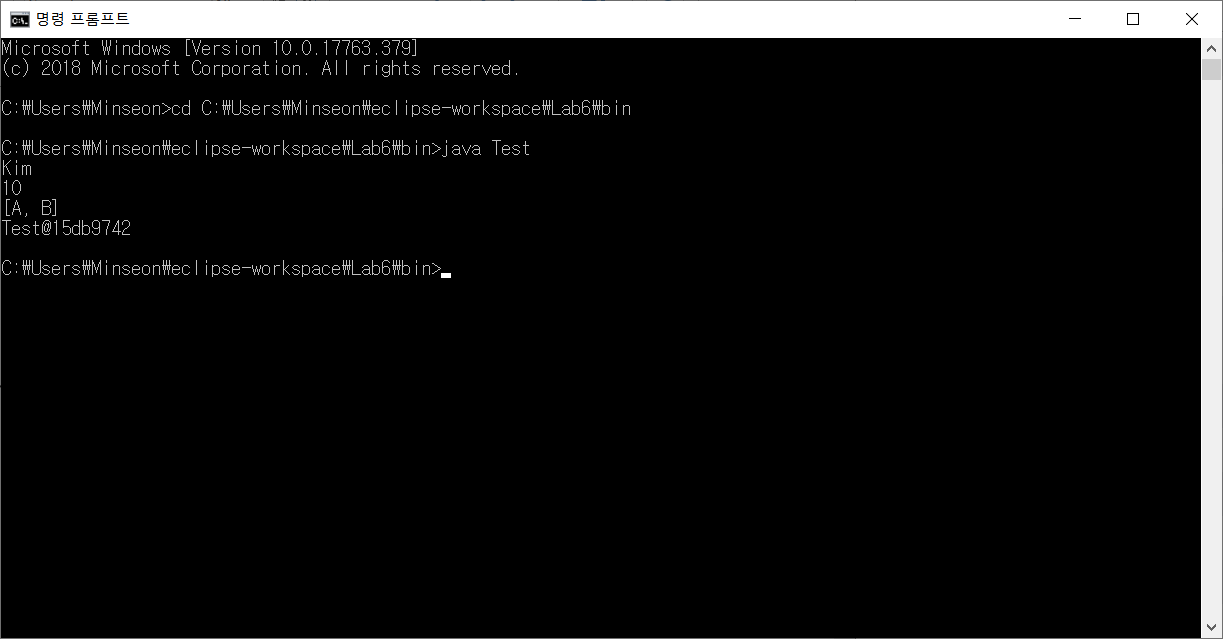
Compare of upper class, the line of “*toHexString(hashCode())*” is not printing ID number but printing unsigned hexadecimal representation of the hash code of the object.

In explanation of java.lang.object, Explain about toString() method about that.



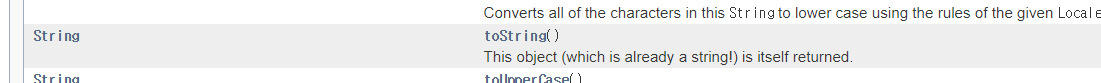
So that code represent class name. And change hexadecimal representation to integer numbers about location of object memory, and then return object value.

**Problem 3: toString()and hashCode() methods of Object class**



**Question(a): Is java.lang.String class override toString() method of “Object” class? Explain your reason.** Hint: refer line 6 **(2pt)**

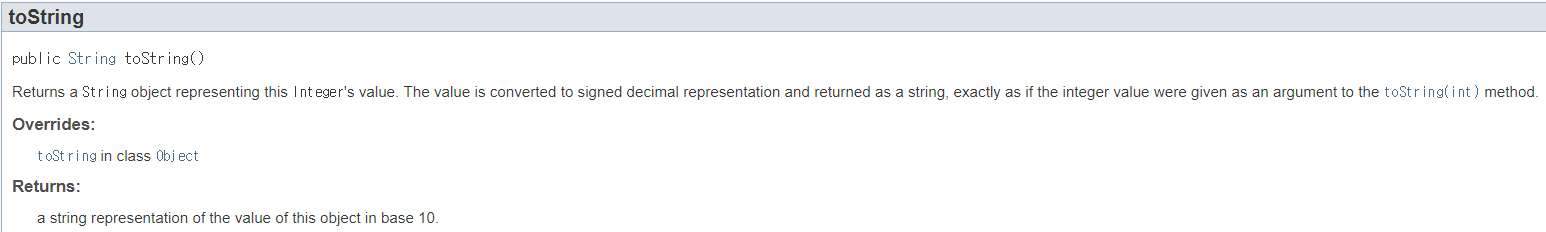
Java.lang.String class override toString() method of object class.



So, they print “KIM” not “String @######” (Not Enough explanation. You should give little more explanation about for your reason, so deduct 1pt)

**Question(b): Is java.lang.Integer class override toString() method of “Object” class? Explain your reason. Hint: refer line 9(2pt)**

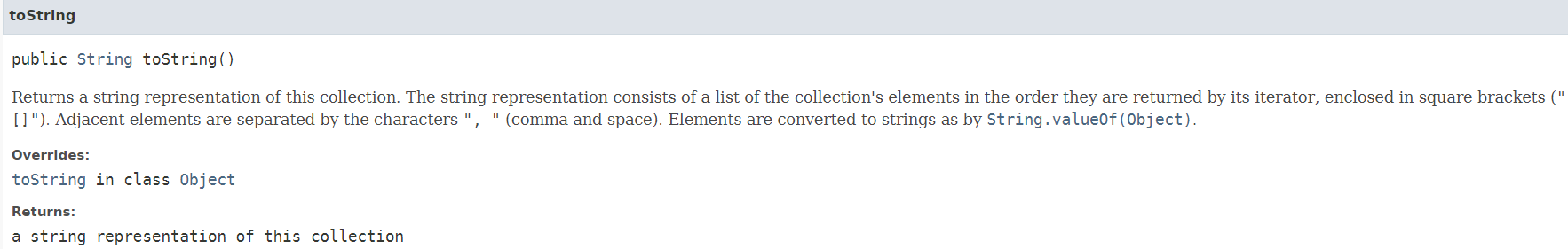
Java.lang.Integer class override toString() method of object class.



So, they print “10” not “String @######” (deduct 1pt, same reason with q(a))

**Quetsion(c): Is java.util.ArrayList class override toString() method of “Object” class? Explain your reason. Hint: refer line 14(2pt)**

Java.lang.ArrayList class override toString() method of object class.



So, they print “[A, B]” not “String @######” (deduct 1pt, same reason with q(a), (b))

**Question(d) : Is Test class override toString() method of “Object” class? Explain your reason. Hint: refer line 17(2pt)**

Test class doesn’t override toString() method of object class.

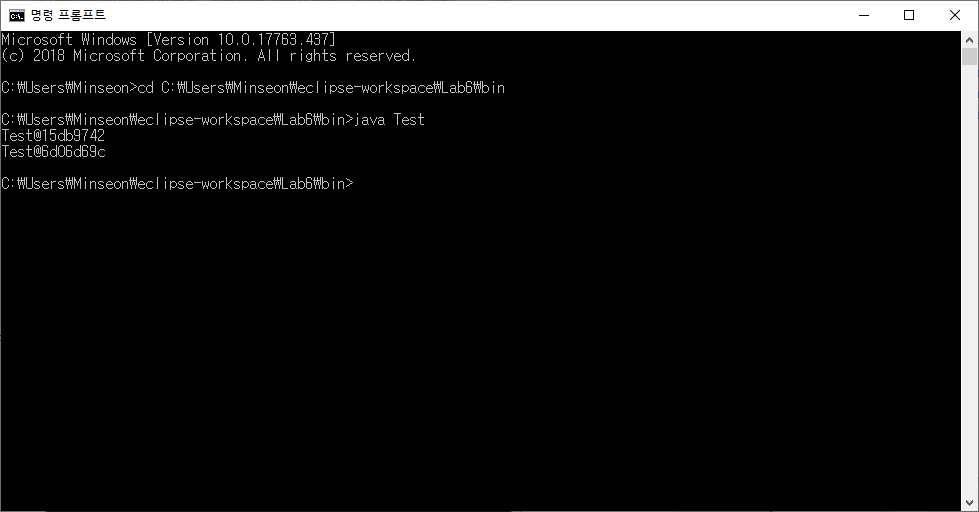
Because Test class made by us and we don’t override and put toString() method in Test class. So they can’t override.

So, they print “Test@15db9742” according to Object method like “**getClass().getName() + “@” + Integer.toHexString(hashCode())**”

**Program 4: Overriding toString()method by Overriding only hashcode()method**

1. comment Line **8-14(hashCode() & toString())** and see the output.

Why you get this output?



According to Object class in java, It will automatically runs that program.

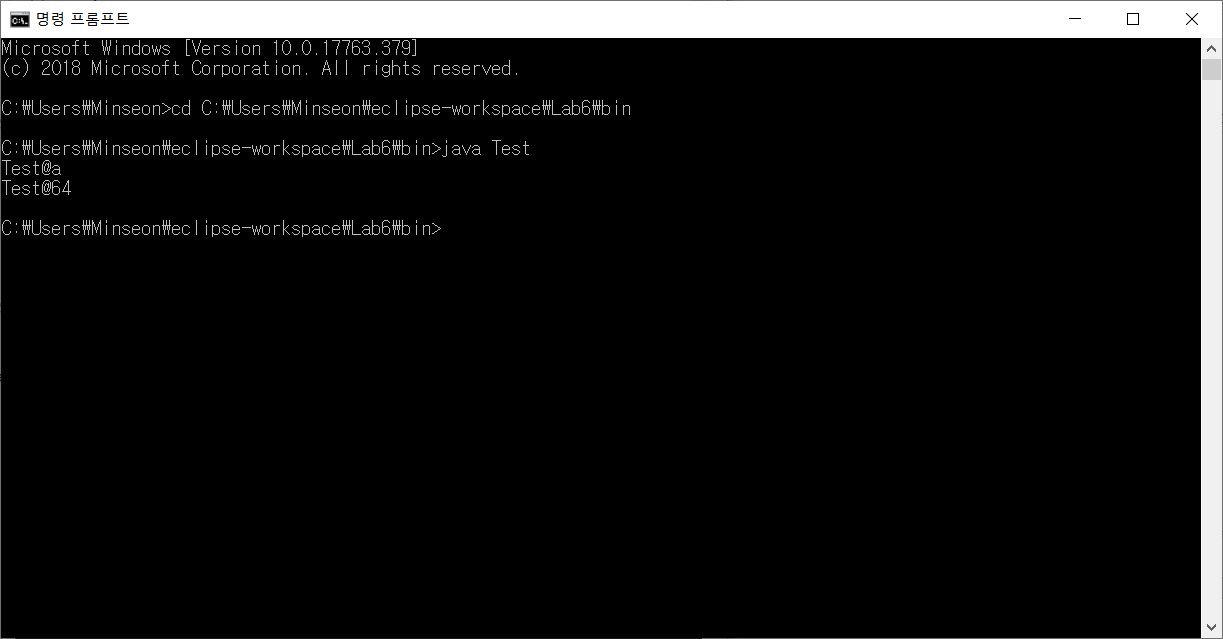
In Object class, they have toString method(). Because We don’t override this method, so the program automatically use toString method in object class. In definition of method, they return “**getClass().getName() + “@” + Integer.toHexString(hashCode())**” line. So we get @ + hashcode() in Hexadecimal number output.

1. comment Line **12-14(toString())** and see the output. Why you get this output?

If we don’t have toString() method changing integer value to String value, we get toString() method from object class in java.

So they print about class object memory location.

“**getClass().getName() + “@” + Integer.toHexString(hashCode())**”

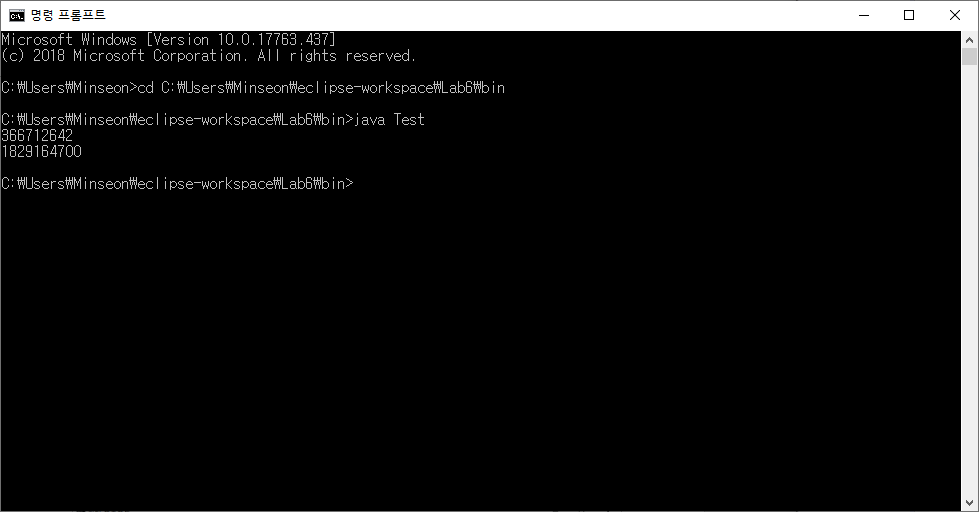


1. comment **8-10(hashCode())** and see the result. Why you get this output?

In toString() method, they return hashcode() method and string “ ”.

That hashcode method isn’t from Test class but Object class in java.

So we get output like this.



We get integer hashcode numbers that indicated location.

That numbers are addresses shown such integer numbers.

So we can’t read that information in the number’s location.

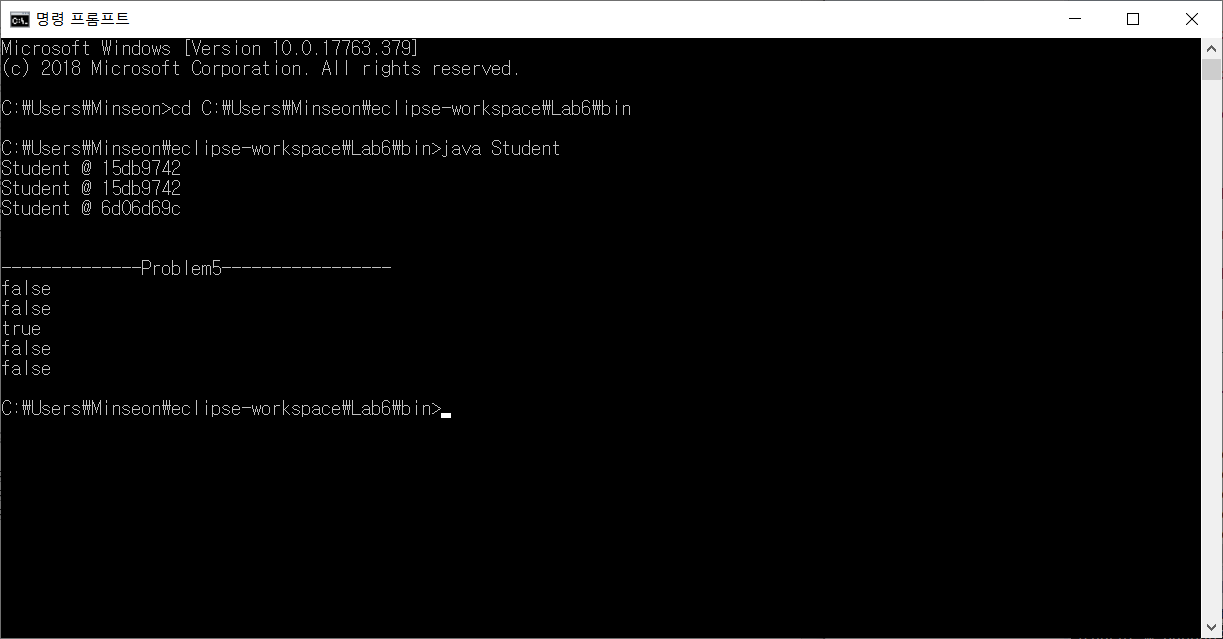
**Problem 5:** Difference between operator (= =) and **equals ()** method in “**Object**” Class

Remark 1: Assume that the meaning of equality is to comparing only names, only roll numbers or both.

Remark 2: When we pass heterogeneous objects, avoid rising of Class Cast exception

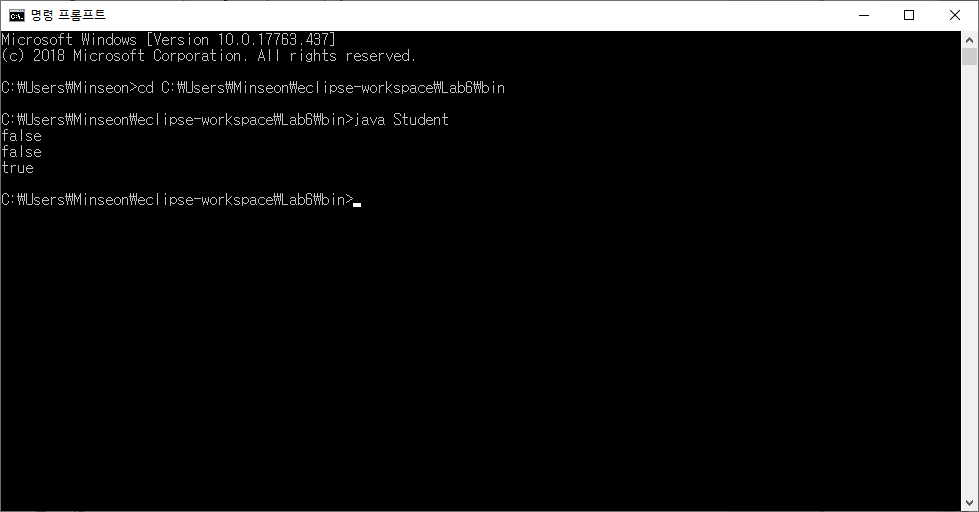
Remark 3: When we pass null argument, avoid rising of Null Pointer Exception

1. **Run the following Code and take screen shoot (1pt).**



1. **In the above code, comment Line 10-29 (equals()) and comment lines 41-42.**

**What is the output at line 38, 39 and 40? Why you get this output?**



This program compare of location address using hashcode() and equals().

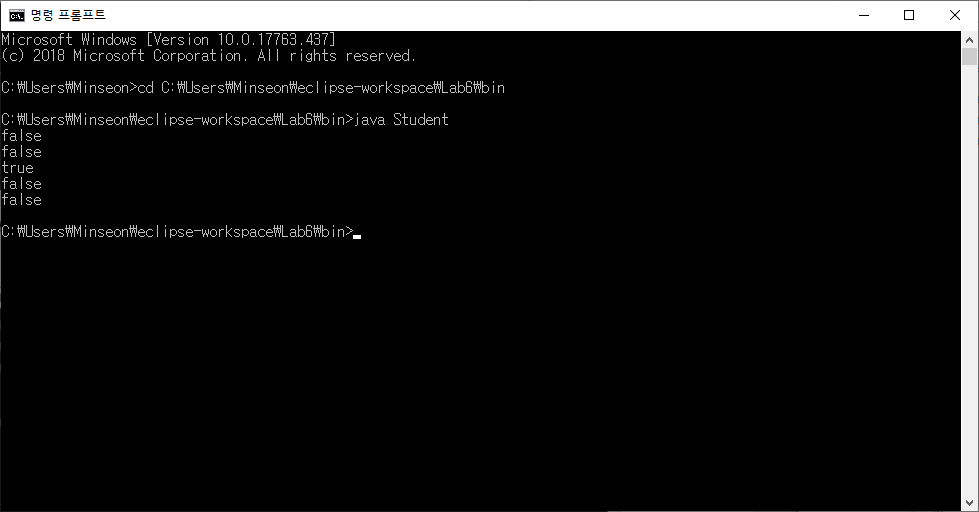
And equals() method also use hashcode, so if you want to make objects that have same key between each object, you have to set same memory location though hashcode.

First, about line 38, they compare of S1 and S2. But they are not same as ‘initialized value’ and location. So it return **false.**

Second, about line 39, they compare of S1 and S3. S1 and S3 has same values but location is different. So they are not same key(reference). So it return **false.**

Last, about line 40, they compare of S1 and S4. But , in line 37, S4 is set as same key(same reference) as S1. A.K.A, S1 and S4 are same key and value. So it return **True**.

1. **In the above code, comment only Line 10-29 (equals()). Is there error at line 41 and Line 42? Why? (3pt).**



There is not any compile Error.

Let me explain about Result of 4th and 5th line.

Line 4th compare object S1 to string value ”kim”. As I said, equals method in Object class is that compare reference like address, so there are not same addresses between S1 and ‘kim’ even though they are same values. So it return **false.** Line 5th compare object S1 to null. Equals(null) means function that examine whether the value are empty or not. But in Object class, they compare about reference value. So they eventually return **false.**

But it is wrong.

If we want to compare values not address, we have to override equals() or use equals in String class.

**Problem 6: Replace the line 10-29 (equals()) in Problem 5 by the following code.**

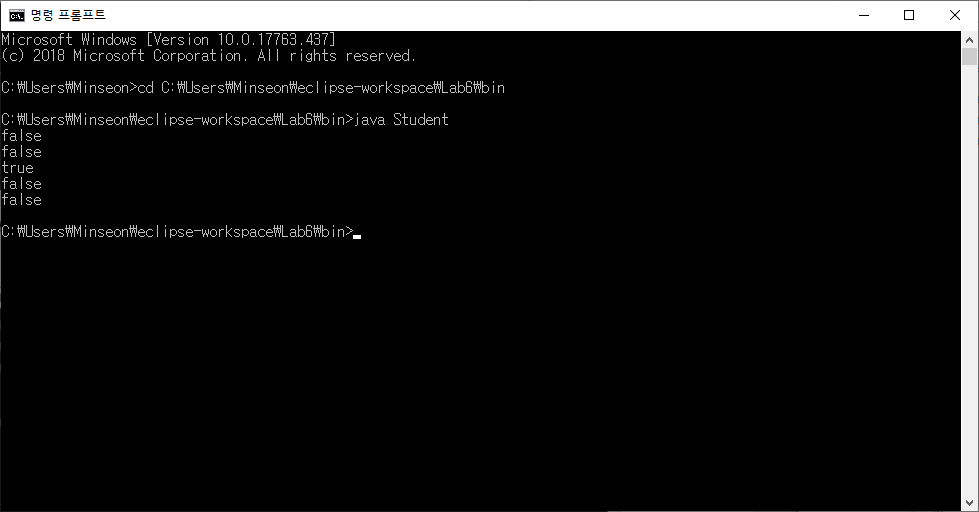
**Did you get the same result as in (a) in the problem 5?**

They are same results. No changes.

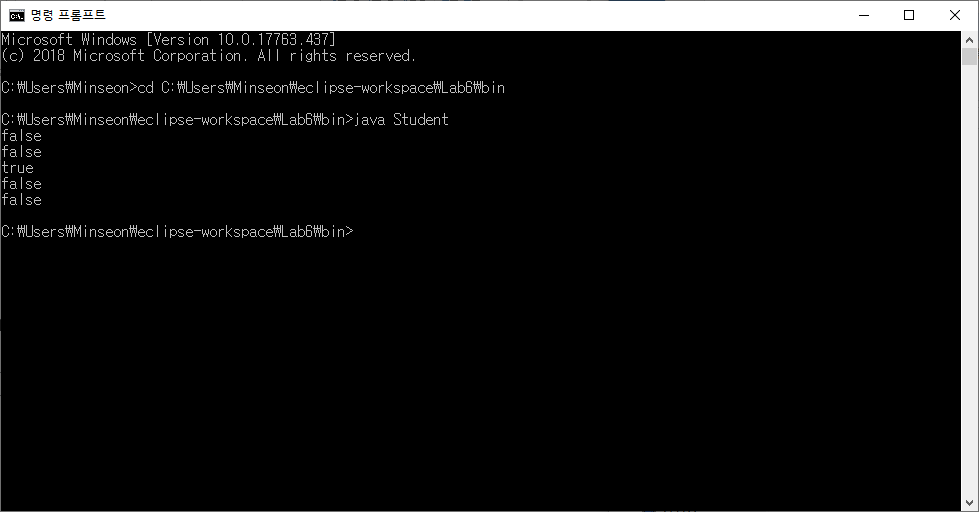
I will pick one example.

In this code, after overriding equals method, System.***out***.println(S1.equals("Kim"));” are not be changed. “kim” object is same value comparing S1, but number value is not same. So they are **false**.

Case1) problem5 result



Case2) problem 6 result



**Problem 7: Replace the line 10-29 (equals()) in Problem 5 by the following code.**

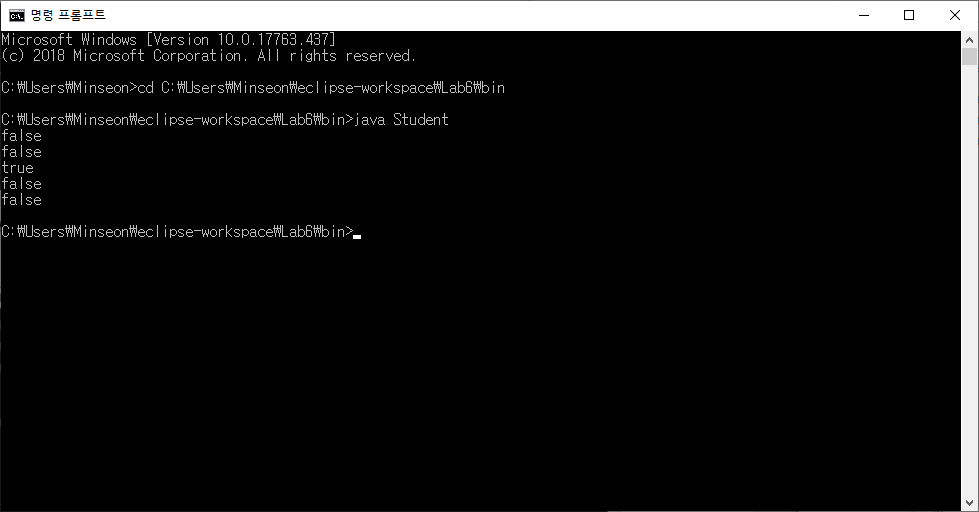
**Did you get the same result as in (a) in problem 5?**

They are same results. No changes.

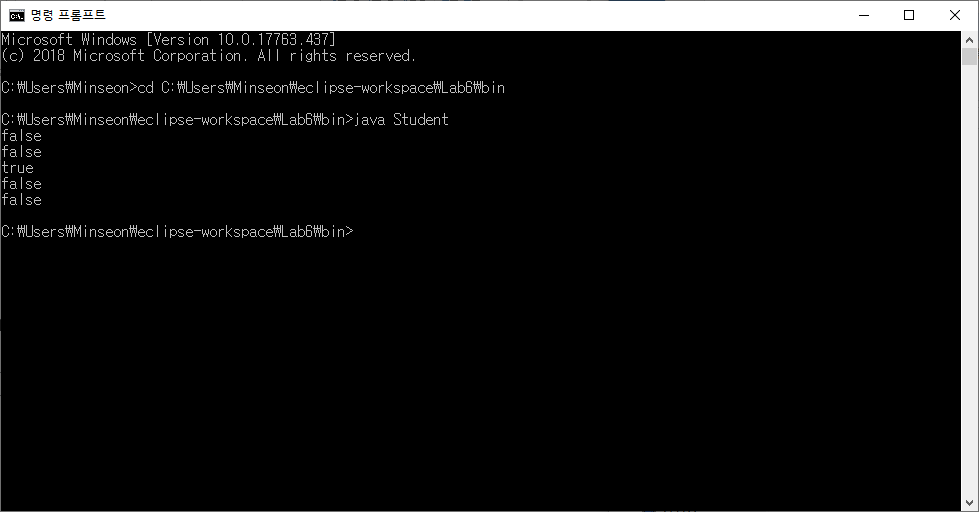
In this code, first through **if** (obj **instanceof** Student) this line, it can judge whether they need to cast or not.

But this code as same as problem6. So they can’t change.

Case1) problem5 result



Case2) problem 6 result



**Problem 8: Replace the line 10-29 (equals()) in Problem 5 the following code.**

**Did you get the same result as in (a) in problem 5?**

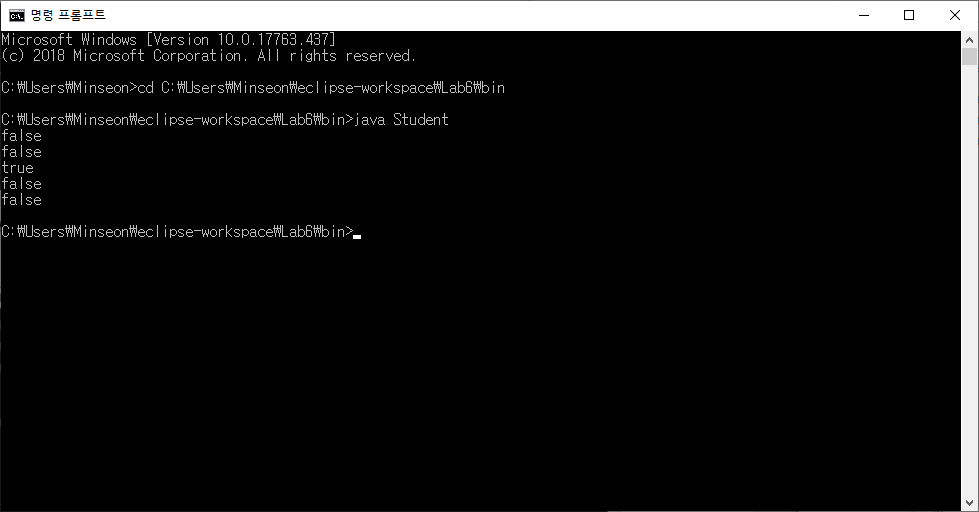
They are same results. No changes.

In this code, first judge that object are same object or not.

And then same as problem 7.

Therefore, problem 8 are not changed.

Case1) problem5 result



Case2) problem 6 result

