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

**// ObjectTest.java**

**import** java.lang.reflect.\*;

**public** **class** ObjectTest {

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

**int** count = 0;

Class c = Class.*forName*("java.lang.Object");

Method[]m = c.getDeclaredMethods();

**for** (Method mi:m) {

System.***out***.println(mi.getName());

count++;

}

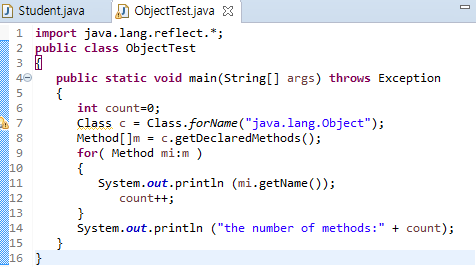
System.***out***.println("the number of methods:" + count);

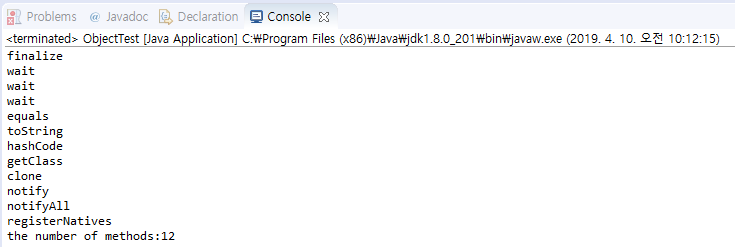
}

}

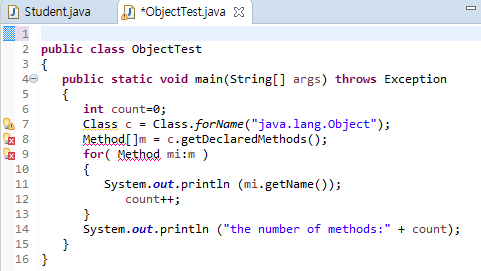
**Question**: If you delete line 1, what happens?

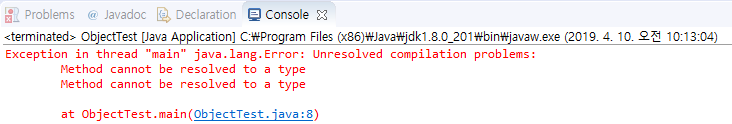
**Before:**

****

****

**After:**

****

****

* **Line 8의 Method []m 부분에서의 Method 클래스는 java.lang.reflect에 포함된 클래스이지만 이런 “import java.lang.reflect.\* 구문을 삭제함으로써 Method 클래스가 import 하지 못하여 컴파일 오류가 생겼다.**

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

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

**// Student.java**

**public** **class** Student {

String name;

**int** ID;

**public** Student(String name, **int** ID) {

**this**.name = name;

**this**.ID = ID;

}

**public** String toString() { // line 10

**return** "Student name: " + name + " and ID: " + ID;

}

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

Student S1 = **new** Student("Kim", 101);

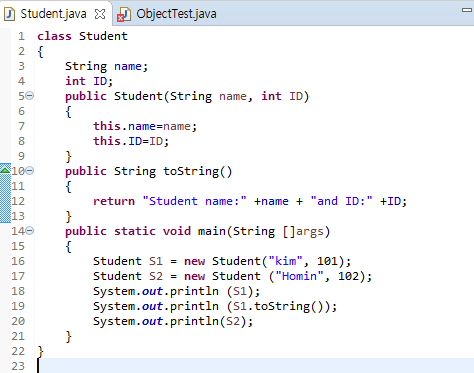
Student S2 = **new** Student("Homin", 102);

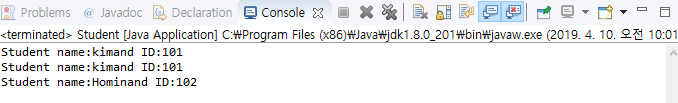
System.***out***.println(S1);

System.***out***.println(S1.toString());

System.***out***.println(S2);

Screenshot:





}

}

**Question(b): From Question(a), 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.

**L1: public String toString()**

**L2: {**

**L3: return getclass().getName()+ ”@” +Integer.toHexString(hashCode());**

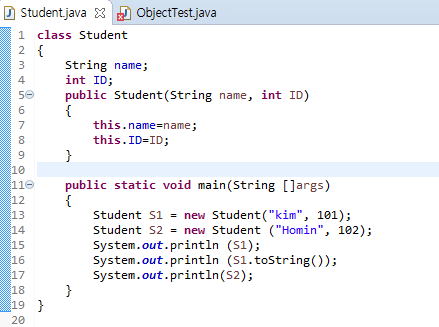
**L4: }**

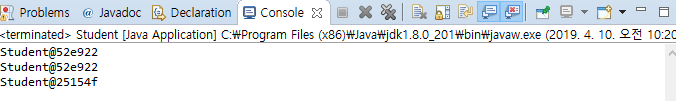
// a) getclass(): method of “Object” class

// b)getName(): method of “Class” class

// c)Integer.toHexString(): method of Integer wrapper class

// d)hashCode(): method of the “**Object**” Class





* 처음코드에서는 새로제작한 toString메소드가 오버라이딩 되어서 처음과 같은 형태로 출력되었지만 task2에서는 Object 클래스의 toString 메소드가 사용되어서 (클래스이름)@(객체 Hex값) 으로 출력되었다.

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

**// Test.java**

**import** java.util.\*;

**public** **class** Test {

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

String s = **new** String("Kim");

System.***out***.println(s); // line 6

Integer i = **new** Integer(10);

System.***out***.println(i); // line 9

ArrayList l = **new** ArrayList();

L.add("A");

L.add("B");

System.***out***.println(l); // line 14

Test t = **new** Test();

System.***out***.println(t); // line 17

}

}

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

**Yes. Object Class의 toString 메소드를 오버라이드 한다**

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

**Yes. Object Class의 toString 메소드를 오버라이드 한다**

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

**Yes. Object Class의 toString 메소드를 오버라이드 한다**

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

아니다. toString()클래스는 딱히Test 클래스에 새로 작성되어 있찌 않기 때문에 사실 오버라이드가 됐다고 보기는 어렵다.

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

**// Test.java**

**public** **class** Test {

**int** i;

**public** Test(**int** i) {

**this**.i = i;

}

**public** **int** hashCode() {

**return** i;

}

**public** String toString() {

**return** hashCode() + "";

}

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

Test t1 = **new** Test(10);

Test t2 = **new** Test(100);

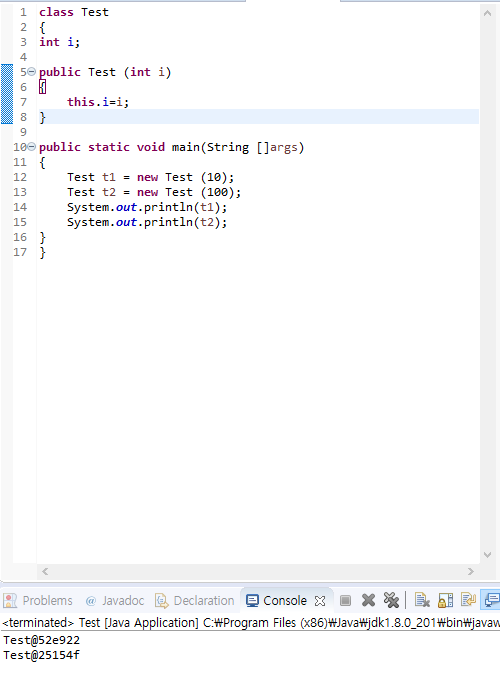
System.***out***.println(t1);

System.***out***.println(t2);

}

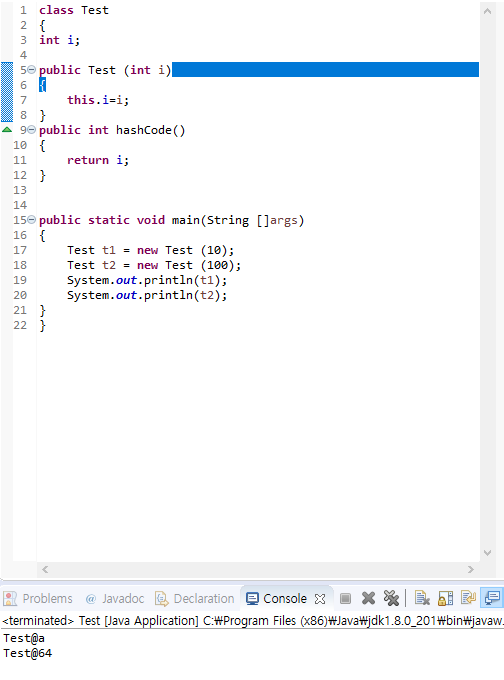
}

1. comment Line **8-14(hashCode() & toString())** and see the output. Why you get this output?



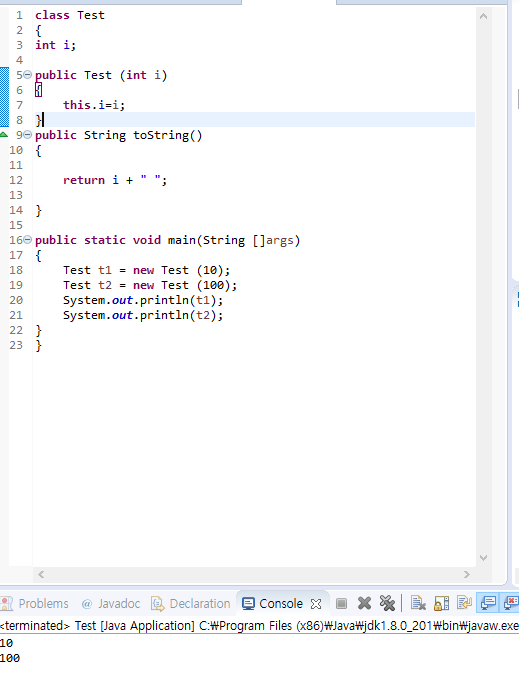
* hashCode() 메소와 toString()메소드가 주석처리가 되어버리면서 오버라이드 되지 않았기 때문에 Object 클래스에 있는 기본 hashCode() 메소드와 toString() 메소드가 호출되었다.

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



* 사용자가 제작했던 toString()메소드가 주석 처리되면서 오버라이드 되지 않았기 때문에 Object 클래스에 있는 기본 toString() 메소드가 호출되었다. 하지만 hashCode의 메소드는 정상적으로 오버라이드 되었기 때문에 , toString 메소드의 반환값 중에서 hashCode를 이용하는 부분은 인자값 10과 100이 16진수로 바뀐 a와 64가 대신 들어가는 것이다.

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



* **toString() 메소드가 정상적으로 오버라이드 되어서 t1, t2 인자에 들어간 10, 100이 추가된다.**

**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.**

**// Student.java**

**public** **class** Student {

String name;

**int** ID;

**public** Student (String name, **int** ID) {

**this**.name = name;

**this**.ID = ID;

}

**public** **boolean** equals(Object obj) { //line 10

**try** {

String name1 = **this**.name;

**int** ID1 = **this**.ID;

Student s = (Student)obj;

String name2 = s.name;

**int** ID2 = s.ID;

**if** (name1.equals(name2) && (ID1 == ID2)) {

**return** **true**;

} **else** {

**return** **false**;

}

} **catch** (ClassCastException e) {

**return** **false**;

} **catch** (NullPointerException e) {

**return** **false**;

}

} //line 29

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

Student S1 = **new** Student("Kim", 101);

Student S2 = **new** Student("Homin", 102);

Student S3 = **new** Student("Homin", 102);

Student S4 = S1;

System.***out***.println(S1.equals(S2)); //line 38

System.***out***.println(S1.equals(S3)); //line 39

System.***out***.println(S1.equals(S4)); //line 40

System.***out***.println(S1.equals("Kim")); //line 41

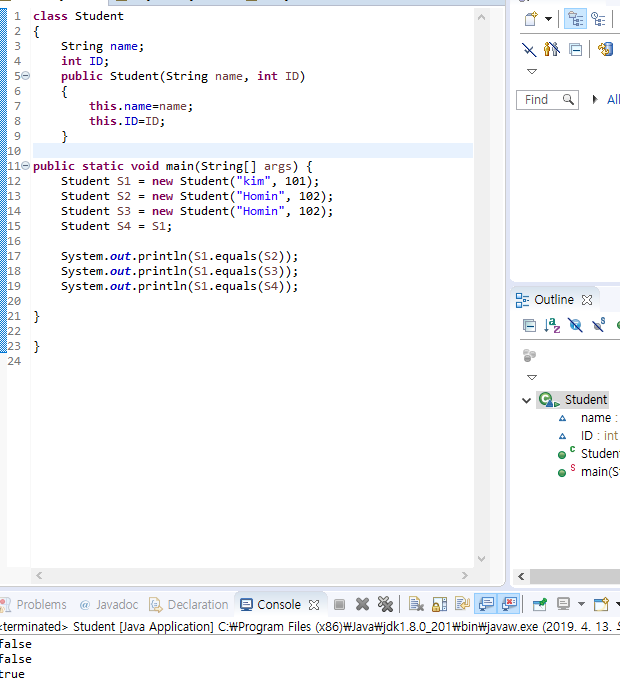
System.***out***.println(S1.equals(**null**)); //line 42

}

}

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.**

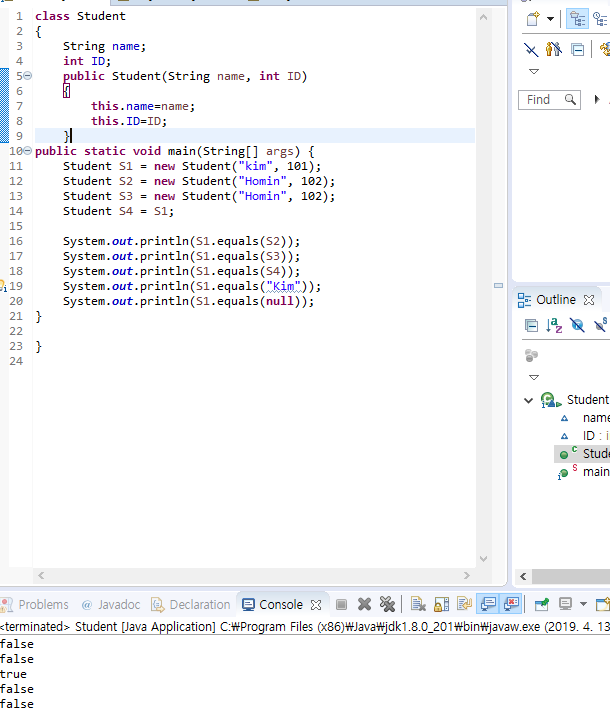
****

**-> Object 클래스에 정의되어 있는 equals 메소드는 일반적으로 주소값을 비교하게 됩니다.**

**1번째는 s1과 s2는 문자열 값, 객체 주소 값또한 다르고 2번째는 s1과 s3의 객체 주소 값이 다르기 때문에 false가 출력됩니다.**

**하지만 3번의 경우는 s4=s1을 선언했으므로 객체가 아예 동일하기 때문에 주소값이 같아서 True가 출력됩니다.**

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

****

* **아니요 error 나지 않습니다.**

**1,2,3가 저런식으로 출력되는 것은 위의 이유와 같고 4번째는 S1과 Strung kim은 Student 와 String으로 클래스가 달라서 예외가 발생하고 5번째는 S1과 Null은 S1.equals(null)에서 NullPointer 예외가 발생하기 때문에 둘다 false로 에러없이 정상 출력됩니다.**

**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?**

**public** **boolean** equals(Object obj) {

**try** {

Student s = (Student)obj;

**if** (name.equals(s.name) && (ID == s.ID)) {

**return** **true**;

} **else** {

**return** **false**;

}

} **catch** (ClassCastException e) {

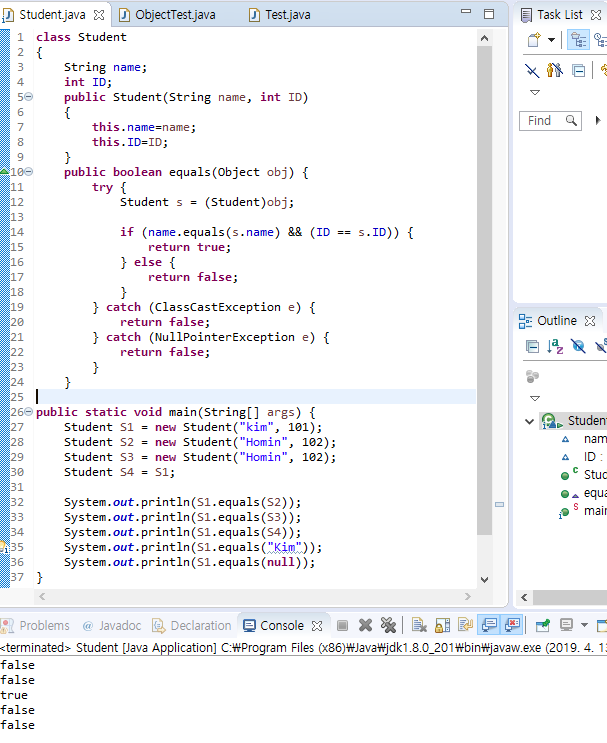
**return** **false**;

} **catch** (NullPointerException e) {

**return** **false**;

}

}



* Yes 똑같이 출력됩니다. 바꾸기 전은 새로 변수나 객체 선언 이후에 s1객체의 필드값들과 인자로 받은 obj 객체를 형변환한 s객체의 필드값을 새로만든 변수나 객체에 저장한 값들을 비교한 것이다 바꾼 후에 단순히 s1의 name과 ID 의 형변환된 obj 객체의 필드값들을 비교하는 것이 같기 때문이다.

**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?**

**public** **boolean** equals(Object obj) {

**if** (obj **instanceof** Student) {

Student s = (Student)obj;

**if** (name.equals(s.name) && (ID == s.ID)) {

**return** **true**;

} **else** {

**return** **false**;

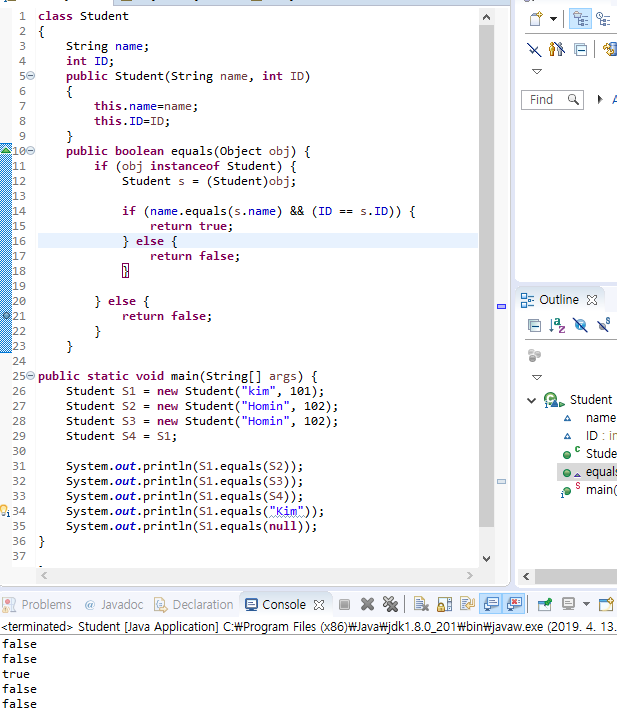
}

} **else** {

**return** **false**;

}

}

****

* **같다. 이유는 비록 예외처리가 없어졌지만 instanceof를 사용했기 때문이다.**

**1번과 2번의 경우는 단순히 이름과 ID가 다르고 4번5번은 조건문에 맞지 않기 때문에 false 가 출력되지만 3번의 경우는 s4객체와 s1객체가 같기 때문에 True가 나온다 따라서 Problem5의 결과와 같게 출력된다.**

**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?**

**public** **boolean** equals(Object obj) {

**if** (obj == **this**) {

**return** **true**;

}

**if** (obj **instanceof** Student) {

Student s = (Student)obj;

**if** (name.equals(s.name) && (ID == s.ID)) {

**return** **true**;

} **else** {

**return** **false**;

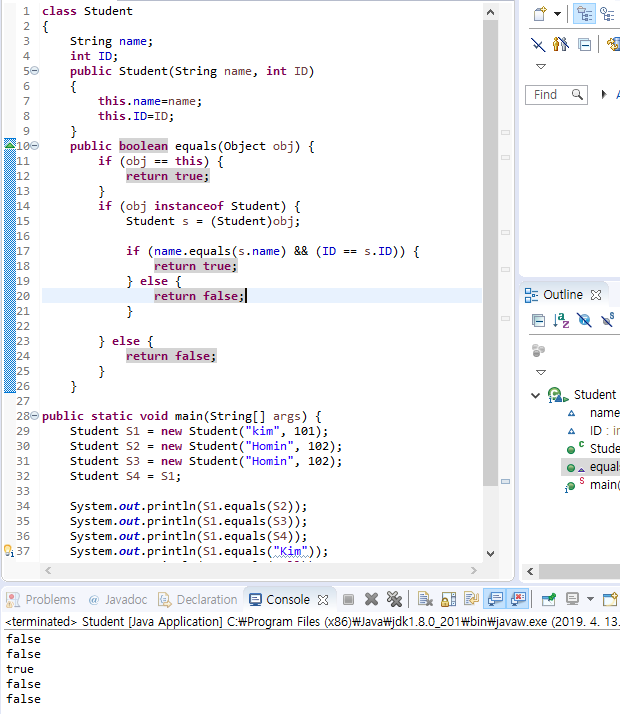
}

} **else** {

**return** **false**;

}

}



* 같다. 위의 구문은 객체 자체를 비교하기 때문에 3번의 경우는 True가 출력되고 나머지는 Problem7과 같은 이유로 false 출력된다. 그래서 결과가 같다.