**OOP Lab 6**

|  |  |  |  |
| --- | --- | --- | --- |
| Name: |  | Department: |  |
| Student ID: |  | Room Number: |  |
| Due Date: | **April 13, 23 : 59** | | |

* **Submit your assignment using the following file format:**

LabNumber\_Name\_IdNo.zip (eg. Lab6\_Hongkildong\_2017.zip).

* This zip file will contain **two types of** files, namely:

1. **report file** with file format **“Report\_Lab number**” (eg. report\_6) to answer theory questions and to write the screen shot of your program.
2. Source code file that contains codes of classes to answer programming questions.

**Contents**

1. **Objec**t class defines the following most common methods shared by any class in java.
2. public **String** **toString**()
3. public native int **hashCod**e()
4. public boolean **equals(Object** o)
5. public final Class **getClass**()
6. **Class** class, **String** class, **ArrayList** Class and **Integer**(Wrapper Class) are also covered

**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?

**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);

}

}

**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

**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

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

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

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

**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?
2. commnet Line **12-14(toString())** and see the output. Why you get this output?
3. comment **8-10(hashCode())** and see the result. Why you get this output?

**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.**

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

**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**;

}

}

**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**;

}

}

**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**;

}

}