**OOP Lab 10**

|  |  |
| --- | --- |
| Due Date: | **Nov 22,** 23 : 59 (Friday) |

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

LabNumber\_RoomNumber\_StudentName\_Student\_ID.zip

Example: Lab10\_328\_Hongkildong\_201620505.zip

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

1. Report file with file format “**Report\_Lab number**” (eg. report\_10) 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.

**Objectives**

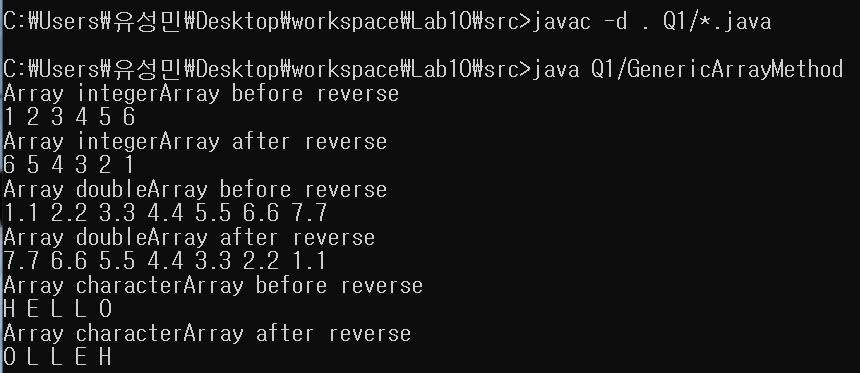
* **How to write a generic class**
* **How to write a generic method**
* **How to create an object of a generic class**
* **How to invoke a generic method**

**Exercises**

1. Complete the given partial code (see the attached code)( 10pt)

The program has two generic methods, namely: ***reverseArray*** () and printArray(). reverseArray() method reverses the order of elements of an array with different types. PrintArray() display different types of array before calling ***reverseArray****() and*  after calling ***reverseArray* (**) using a generic method. This is similar to the code in Fig**. 20.3.**

**Hence, Complete the program by filling the missing parts of the given source code.**



2.  **Complete the given partial code (see the attached code)(10pt).**

**In Fig.20.3,** the program has ***printArray* (**) generic method. Assume that you want to add another *generic printArray* () method that overloads the existing generic *printArray* () method which takes two additional **integer arguments**, namely ***lowSubscript*** and ***highSubscript***. A call to this method prints only the designated portion of the array. **Validate *lowSubscript* and *higSubscript***. If either of them is out of range, the overloaded *printArray* () method should throw an ***InvalidSubscriptException***; otherwise, *printArray* () should return the **number of elements printed**. Then modify *main* () to use both versions of *printArray* () on *integerArray*, *doubleArray* and *characterArray*. Test all capabilities of both versions of *printArray()*.

Hence, c**omplete the program by filling the missing parts of the given source code.**

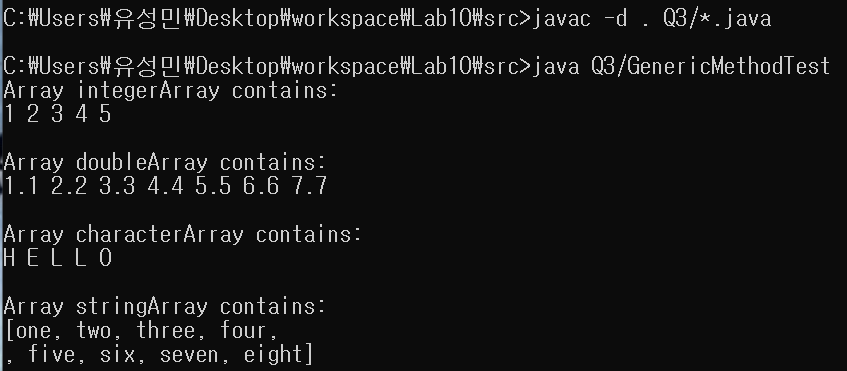
텍스트, 신문, 스크린샷이(가) 표시된 사진

자동 생성된 설명

3. Complete the given partial code (see the attached code)(10pt).

In Fig.20.3, the program has ***printArray* (**) generic method. Assume that you want to add another *non-generic printArray* () method. This method specifically prints an array of ***Strings* in the following** tabular format.  **Complete the program by filling the missing parts of the given source code.**

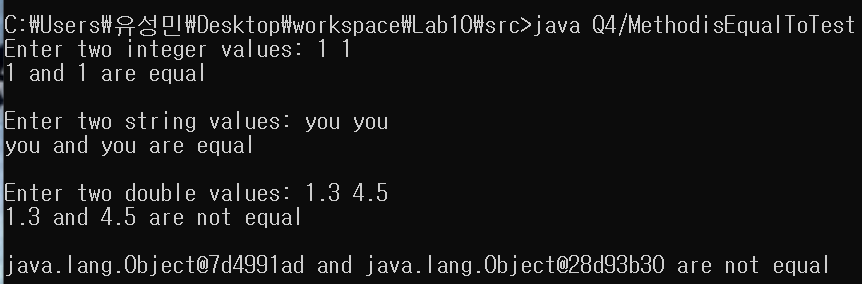
|  |
| --- |
| Array stringArray contains:  One two three four  Five six seven eight |



4. Complete the given partial code (see the attached code)(10pt).

A generic **isEqualTo**() method compares its two arguments with the **equals () method** and returns true if they are equal and false otherwise. You can calls the generic **isEqualTo()** method with a variety of built-in types, such as Integers, Strings, Doubles and Object.

a) **Complete the program by filling the missing parts of the given source code.**

**b)**  What result do you get when you attempt to run this program? **Include the result in your report file**

**c)**  Write your reason for question in **b**? **Include your reason in the report file.**

**=>**

**As you can see the isEqualTo() method, I get two parameters and then compare them with equals method. If I used “==” instead of equals method, I will get different result. In case of comparing integer variables, ‘==’ works well to compare two variables. Because integer is primitive type variables. But in case of String, String is Object, so it compares String’s address. So result of comparing all strings that I input says not equal. (because addresses are not same). But by using equals method at String object, we can compare the input variables correctly.**

5**. Write your own Program (10pt)**

**Write a generic class called “Pair” which has two type parameters, namely F and S.**

* This class has two fields. The type of the first field is **F** and the type of the second filed is **S**.
* Add get() and set() methods for the first field of the pair.
* Add get() and set() methods for the second field of the pair
* In order to test the generic **Pair** class, write test class that contains the main () method.
* The input and output format of the program is given below.

**Input:**

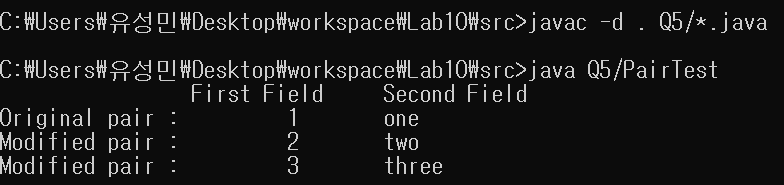
**First Field Second Field**

1. **one**
2. **Second**
3. **Third**

**Output:**

Original pair: (1, One)

Modified pair: (2, Second)

****