**OOP Lab 10**

|  |  |  |  |
| --- | --- | --- | --- |
| Name: |  | Department: |  |
| Student ID: |  | Room Number: |  |
| Due Date: | **May 18,** 23 : 59 (Friday) | | |

* Submit your assignment using the following file format:

LabNumber\_StudentName.zip (eg. Lab10\_Hongkildong.zip).

* This 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).

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).**

**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.**

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

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).

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.**

5**. Write your own Program**

**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)