COMP 1011: Advance Java Programming

Assignment #1

Due Date: (Mid-night – 11.59 pm) 4th Feb. **Marks/Weightage: 30/7.5%**
End Date: (Mid-night – 11.59 pm) 7th Feb. with 20% deduction/penalty. No exceptions.

Purpose: The purpose of this assignment is to:

- Practice the use of Collections, Generics and Regular Expressions

References: Learning materials for chapters 14, 16 and 20 of textbook, and other internet references (if any).

IDE: eclipse Java Enterprise Developer edition – 2022-12/2023-03 and Windows 10/11 OS.

Step1: At the start of eclipse, you must name your Eclipse workspace according to the following rule:

YourFullName_COMP1011SectionNumber_A01

For Example: **JohnSmith_COMP1011Sec002_A01 (if your section is Sec002)**

Step 2: And after that, you must name your Eclipse project according to the following rule:

YourFullName_COMP1011SectionNumber

For Example: **JohnSmith_COMP1011Sec002**

Step 3: And your package should be named as follows:

YourFullName_SectionNumber_Ex01

For Example: **johnsmith_sec002_ex01**

Submission/Upload Instructions:

After you complete, run and test your code, you need to do the following:

a) Close the eclipse, go to your **workspace** folder which you created in Step 1.
b) Zip it up. You should get a zip file like this– **JohnSmith_COMP1011Sec002_A02.zip. You should only be submitting it in the .zip file format (and not .rar or any other format)**
c) Upload this zip file using the **Assignment-01** link in blackboard.

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character for the first word and uppercase for every other word
- *classes* start with an *uppercase* character of every word
- **packages** use only *lowercase* characters
- *methods* start with a *lowercase* character for the first word and uppercase for every other word

Note: Late submissions are accepted until up to 3 days past due date with 20% deductions. After that no submission will be considered.

Exercise 01: This is based on the HashMap and extension to app – WordTypeCount.java (refer code examples-based on Collections posted on Blackboard) demonstrated in the class. It gives the following output. *[5 marks]*

```
Enter a string:
This is a a demo demo demo of of of of collections in java

Map contains:
Key             Value
a                   2
collections          1
demo                3
in                  1
is                  1
java                1
of                  4
this                1

size: 8
isEmpty: false
```

You need to enhance the above example to find and display the duplicate words i.e. those repeated minimum and maximum number of times. So, based on the sample input string above, output should be:

a – 2
of -- 4

Exercise 02: *[15 marks]*

Write an overloaded generic method which prints a double array, string array and character array as shown below:

Sample arrays are:

Double [ ] grades = { 56.7, 89.23, 45.56, 88.40, 90.56 }
String[ ] product = { “iphone”, “Galaxy”, “Pixel”, “Nokia”}
Char[ ] status = { ‘y’, ‘n’, ‘i’, ‘d’, ‘e’ }

Overloaded methods definitions:

public static <T> void printArray(T[] inputArray) { …} // version 1.0
public static <T> void printArray(T[] inputArray, int startIndex) { …} // version 2.0
Add validation for startIndex such as it can not be negative or higher than the size of the array

public static <T> void printArray(T[] inputArray, int startIndex, int endIndex) { …} // version 3.0
Add validations such as both the indices should be non-negative and startIndex should be smaller than endIndex.

Exercise 03: *[10 marks]*

Define a generic queue (FIFO) class (store it in file – DataQueue.java). Add appropriate field variable(s) required, constructors, methods to add an element (enqueue), delete an element (dequeue) and to print elements

Using this generic class, you should be able to process the following in the driver class – DataQueueTest.java):

a) Queue of Strings
b) Queue of Stock values – type double

Evaluation/Rubric:

Functionality	
(a) Correct implementation of business requirements, code logic, functionality etc.	90%
(b) Correct implementation of exception handling and intended results	5%
(c) Comments, correct naming of variables, methods, classes, etc.	2.5%
(d) Friendly input/output	2.5%
Total	100%