Lab Assignment 1: Generics and Stack [40 Marks]

SE2205: Data Structures and Algorithms using Java – Fall 2021

Open Day: October 3, 2021; **Cut off time**: October 10, 2021, Sunday @11pm

Demo Days During Lab Time: Section 002: 12/Oct/Tuesday, Section 003 & 005: 14/Oct/Thursday, Section 004: 18/Oct/Monday

---

Adapted by Dr. Quazi Rahman (qrahman3@uwo.ca).

A. Rationale and Background

This is a group assignment. Each group can have two students. If there is any extra student in a certain lab section, that student can join any one of the groups and it will be the only group with 3 members. In the naming requirement you need to add the group members' usernames together when it says *usernames*. In this lab Assignment we will reuse the Lab Exercise 2 code and add a new method to it for Question 1 and then work with Stack for Question 2. You can have an extension up to three days with 10% penalty on each day.

B. Evaluation and Submission Instructions

Submit your Lab-Assignment online by carrying out the following instructions:

1. Create a Project with the following name: *usernames_LabAssignment1*
2. For this question create a package for each Question (LA1Q1, LA1Q2)
3. Use meaningful names for each class and the associated variables by following the general naming conventions.
4. For this question, use the static header and footer methods your created before.
5. Comments: **Writing short but clear comments for Lab Exercises is <u>mandatory</u>**. As long as the comments are clear, full credit will be given to the written comments.
6. Once the assignment is completed, go to your 'Assignments' folder. Select the project folder (e.g. *usernames_LabAssignment1*). Right-click to select 'Send to' 'Compressed zipped folder'. Make sure it has the same naming convention (e.g. *usernames_LabAssignment1*.zip). Upload this file on OWL as your submission. Both of the group members need to submit the same zipped folder.
7. You will be graded on this lab Assignment based on the comments, running code and your understanding of the code (demonstration). Comments: 5%, Running Code: 35%, Demonstration: 60%. You will do team-demonstration for this lab.

C. Lab Question

1. Question 1 [20 Marks]

This question is a repeat from Lab Exercise 1. Here, along with the header and footer methods, you need to add a static method. The Question: Create a data structure that will keep a record of key and values. Here the key will be the year of studies of a group of university students, and the values will be their first names. These students are the leaders in the University Student Council. Your task would

be to find out how many student leaders belong to a specific year and get a leader-representative from each year and print the names of those representatives.

(a) Define a generic class called Pair <Y, N> with the following specifications (see the class diagram below): [Hint: this code is given in the class handout]
- Two private data fields: key Y and value N.
- Constructor with both Y and N parameters.
- Getter and setter methods for both the data fields.

Pair <Y, N>
- key: Y - value: N
+ Pair(key: Y, value: N) + setKey (key: Y): void + setvalue (value: N): void +getKey():Y +getValue(): N

(b) Create a public static void header method (with void parameter list) with the following information (see the sample output):
Names: firstName1 and firstName2. (Hardcode it; meaning put the data manually)
Student Numbers: studentNumber1 and studentNumber2 (Hardcode it)
Goal of this project: Brief description of the project. (Hardcode it)

(c) Create a public static void footer method (with void parameter list) with the following Message (see the sample output):
This is timeOfDay on theDate.
Completion of Lab Assignment 1 is successful!
Good bye! yourFullName.

(d) Write a static method public static <Y, N> Pair[] getRep_yourFirstName(Pair[] pa) that will return a pair containing student representative along with the corresponding year, from each year from the Pair list. Hint: You can create two ArrayList<> lists with appropriate tags and populate those arrayLists based on the ArrayList method called c***ontains***[see slide #16, Unit 1-Part 2]. (example: if the name of the arrayList is **myArray**, then if myArray.contains(get the year from Pair object reference variable) is false, populate **myArray**)

(e) Define the driver method and do the following (Check the class handouts):
i) Call the header method.
ii) Declare an ArrayList type reference variable with Integer-tag and fill out the list with integer values 2, 3, 4, 3, 2, 2 with the aid of Array.asList(value1, value2..) method as shown below. These numbers will represent the year of studies of the student leaders, Note that both **ArrayList<E>** and **Arrays** classes are available in **java.util.*** package.
ArrayList<Integer> anyValidName = new ArrayList(Arrays.asList(2,3,….));
iii) Declare a second ArrayList type reference variable with String-tag and fill out the list with String values Harry, Lavender, Ron, Hermione, Luna and Vincent with the aid of Array.asList(..) method. These string values represent the names of the student leaders. (FYI: Based on both the Lists, Harry is in 2nd year, Lavender is in 3rd year and so on).
iv) Create an array of size ***anyValidName.***size() of Pair type reference variables.

v) Populate this Pair-array by the **key** and the corresponding **value** pairs using the two ArrayList<E> reference variables with the help of the getter methods (Slide 16, Unit 1-P2) from the ArrayList class.
vi) Now ask the user the following question: "From which academic year would you like to list the names of the leaders: "
vii) Validate that the user enters either 2 or 3 or 4 (see the sample output).
viii) Now based on the user's choice print the names of the leader(s) from that specific year.
ix) Keep asking the user till she/he does not want to continue.
x) When the user press 'n', your program will call the getRep_yourFirstName() method from main, and then based on the returned values, print the list of the student's rep from each year (see the sample output).
xi) Call the footer method.

Sample output:

****
Names: firstName1 and firstName2.
Student Numbers: studentNumber1 and studentNumber2
Goal of this project: Brief description of the project.
****

From which year you are looking for the names of the student leaders: 1
Invalid entry! Enter a positive integer between 2 and 4: dsfsdfas
Invalid entry! Enter a positive integer between 2 and 4: 5
Invalid entry! Enter a positive integer between 2 and 4: f
Invalid entry! Enter a positive integer between 2 and 4: 2

We found 3 students from year 2 and they are:
Harry
Luna
Vincent

Do you want to continue? [y or n]: y
From which year you are looking for the names of the student leaders: 5
Invalid entry! Enter a positive integer between 2 and 4: r
Invalid entry! Enter a positive integer between 2 and 4: 3

We found 2 students from year 3 and they are:
Lavender
Hermione

Do you want to continue? [y or n]: n

Here is the list of the class-reps.....
2 - Harry

3 - Lavender
4 – Ron

****
This is timeOfDay on theDate.
Completion of Lab Assignment 1 is successful!
Good bye! yourFullName.
****

2. Question 2 [20 Marks]

Create a Stack-data structure and use that stack in the driver class.

(a) Define a generic class called *MyStack<E>* with the following specifications (see the class diagram below):

i) Create a private Array reference field of Object type, and a private integer type field to store the length of the array.
ii) Create a constructor with length parameter that will generate an Object-array size length for the Stack. (Assumption: The index [length – 1] is the top of the array)
iii) Create a member method called empty () to tests if the stack is empty or not
iv) Create a member method called peek () to check the data-item at the top of the stack without removing it from the stack.
v) Create a member method called pop () to remove the data item from the top of the stack and return that item to the called method.
vi) Create a member method called push() to push an item onto the top of the stack.
vii) Create a member method called search() that returns the 1-based position where a data item is on this stack. In this case, the topmost item on the stack is considered to be at distance 1. If the searched item is in the 3rd position from the top, this method will return 3. The equals method from the Object Class can be used to compare anyData to the items in the stack.
viii) Override the Object's toString() method so that it can print your stack content (Hint: you can return Arrays.toString(arrayName) from inside the toString() definition.)

[Hint: Check the following site (scroll down) for the description of the methods again: https://docs.oracle.com/javase/10/docs/api/java/util/Stack.html]

MyStack<E>
-objectArray: Object[] -length: int
+MyStack(length: int) +empty(): boolean +peek(): E +pop(): E +push(anyName: E): E +search (anyName: Object): int +toString(): String

Now define a driver class called *StackDemo0* with the following:

(b) Use the same header method from Q1 with different description of the Goal

(c) Define the driver method with the following specs:

i) Call the header method.
ii) Create an empty (length = 0) String stack using the class you have created in (a)
iii) Print the stack content with the help of the toString() method.
iv) creating a String array with 6 different values following this order: "studentNumber2", "lastName2", "firstName2", "studentNumber1", "lastName1", "firstName1"
v) Push the above array values to the stack you created in ii.
vi) Print the stack content with the help of the toString() method.
vii) Print the top of the stack value by peeking.
viii) Search for the first student number (e.g., 250922765) using the search method.
ix) Using the distance metric that you found from the search method store that number in a string variable
x) Convert the first and last characters of this string to number using Integer's parseInt method and then print the average of these two numbers on the screen. (Ask your instructor on this part if you have any confusion)

(f) Create a new public static void footer method with MyStack type reference vriable as a formal parameter and do the following (see the sample output):

i) Print a message "Team Member 1 Info…." on the screen.
ii) Pop the stack for three times.
iii) Repeat i) and ii).
iv) Print a message "Here is the status of the Stack..."
v) Print the stack content with the help of the toString() method.

Sample Output:

```
**********************************************************
Names: firstName1 and firstName2.
Student Numbers: studentNumber1 and studentNumber2
Goal of this project: Brief description of the project.
**********************************************************

Revisiting the concept of Stack.....
The stack is Empty! The stack: []
Pushing the String values to the stack....
The stack: [251896357, lastName2, firstName2, 250922765, lastName1, firstName1]
The value at the top is: firstName1
Searching for studentNumber1....
The item has been found in distance 3 with reference to the top
The first number in the student ID is 2
The last number in the student ID is 5
The average of these two numbers is: 3.5

Team Member 1 info.....
```

First Name: firstName1
Last Name: lastName1
Student Number: 250922765
Team Member 2 info.....
First Name: firstName2
Last Name: lastName2
Student Number:251896357
Here is the status of the Stack...
The stack is Empty! The stack: []
Goodbye!