CS 250 - Fall 2021 - Homework 04

Objectives

The objectives of this assignment are:

- To give additional practice with array and looping constructs.
- To provide extra practice with String methods.
- Testing your understanding of abstract data types
- To give additional practice with List, Set, Stack, Queue, Map.
- To give additional practice with parsing inputs and exception handling.

Part 1. Non-programming Questions (50 points)

Q1. (12 points) Fill in the blank with the correct answer:

1. The size of a _____ structure is fixed, while the size of a _____ structure may change during the course of execution.

a. static, abstract

b. dynamic, static

c. static, dynamic

d. None of the above

2. The _____ operation adds an element to the top of the stack, while the _____ operation removes the top element from the stack.

a. push, get

b. push, pop

c. add, pop

d. peek, pop

3. An object that contains a variable that refers to an object of the same class is a _____.

a. link

b. this

c. self-referential object

d. self

4. The _____ data structure is used to manage the method call and returns in a computer program.

a. stack

b. array

c. vector

d. list

Q2. (20 points) Given a LinkedList variable below:

LinkedList<Character> charLL = new LinkedList<>();
Write code to complete the following sequence of tasks:
a. Add three characters 'A', 'C', 'E' to the linked list charLL.
b. Assume charLL contains the three characters after the previous operations, write code to insert 'B' and 'D' into charLL so that 'B' is between 'A' and 'C', 'D' is between 'C' and 'E'.
c. Write code to print the characters in the linked list charLL in the reversed order (so, the sequence E, D, C, B, A should be displayed on the Console).

Q3. (18 pts) Complete the following empty method:

```
/*
 This method receives a List of Integer parameter (list) and returns another List object that
    contains no duplicated elements from the list.
*/
public static List<Integer> removeDuplicates(List<Integer> list) {
        //TODO: add code below
}
```

Complete the above method to return a new List of Integers that contains no duplicated elements from the parameter list. For example, if you define a List<Integer> object as follows:
List<Integer> list = Arrays.asList(2, 3, 1, 3, 5, 4, 6, 3, 5);
Then you call removeDuplicates() with list, it should return a List<Integer> object with values { 2, 3, 1, 5, 4, 6 }. **Note**: The duplicated elements of 3 and 5 are not included in the result list.

Part 2. Programming Question (50 points + bonus 10 points)

2.1 Problem Descriptions

Develop a Java application to read a text file containing students' ids, names, and grades into your program. Then, print different statistics of the students' grades such as min, max, mean, the percentage of students who got A, B, C, D, and F grades.

2.2 Problem Design and Implementation Guideline

2.2.1. Create the HW#4 Project and a student's grades text file:

First, you need to create a Java project HW4 on Eclipse. Then, follow the similar approach in Lab4 to create a text file named students_grades.txt under the src folder of your project. You can do that by right-clicking on the src item on the project explorer window. Then, choose New, File, enter students_grades.txt in the File name text box, and click the Finish button. A subwindow to edit the content of this file is opened.
This file contains the student records in separate rows. Each row stores one student's information, including id, name, and grades. The formats of id, name, and grades are integer, string, and double correspondingly. A comma separates the data fields of each student. **Copy the following contents to this students_grades.txt file and save it**.

```
847342, Kaitlin Goyette, 2.50
859403, Ward Cremin, 3.22
964535, Zachary Ziemann, 3.96
295666, Ryan Hyatt, 2.15
211686, Arjun Price, 3.50
593397, Fabiola Nikolaus, 1.99
552643, Mittie Tremblay, 2.05
128645, Ronaldo Bartoletti, 3.12
970343, Cristobal Marvin, 3.85
82617, Samir Muller, 2.96
```

2.2.2. Student class: You need to create a Student class to store each student's information read from the above file. This Student class is similar to the Student class on Lab 4. Code this class with the following requirements:
- This class has three instance variables: ID (int), name (String), grade (double)
- This class has the following methods:
+ Student(int id, String name, double grade): The constructor of this class initializes the instance variables with the parameters (assign parameters to corresponding instance variables)
+ Getter and Setter methods for each instance variables
+ String toString(): This method overrides the toString() method derived from the Object class. You should return a string including all student information, i.e., "id, name, grade".

2.2.3. Main application class: You should create a new class named StudentGrade. Write code inside the main() method to implement the following requirements:

- Define a dynamic data structure variable named arrStudents to store a collection of Student objects (**Hint**: you can use ArrayList, LinkedList, Vector, etc.)
- Write code that uses Java IO library to reads each line of the file students_grades.txt, parse intro multiple tokens, create the Student object and add it to the arrStudents. Remember to use try-catch to handle IO exceptions and parsing exceptions (could be thrown when parsing ID and grade). Also, you

should display an error message “Grade should be between 0.0 and 4.0” and exit if the grade in the file is not valid.

- Loop through the Students objects in and print the max, min, and average grade of all students
- Loop through the list of Students, calculate and print the percentage of students who have an A (their grade >= 3.6 and <= 4.0)
- Loop through the list of Students, calculate and print the percentage of students who have a B (their grade >= 3.2 and < 3.6)
- Loop through the list of Students, calculate and print the percentage of students who have a C (their grade >= 2.8 and < 3.2)
- Loop through the list of Students, calculate and print the percentage of students who have a D (their grade >= 2.4 and < 2.8)
- Loop through the list of Students, calculate and print the percentage of students who have an F (their grade < 2.4)

Below is the sample output of the program on the Console after execution:

```
Min  grade: 1.89
Max grade: 3.96
Avg  grade: 2.93
Percentage of students who got A: 20%
Percentage of students who got B: 20%
Percentage of students who got C: 20%
Percentage of students who got D: 10%
Percentage of students who got F: 30%
```

2.3 Submission Requirements

A. Source code: Write your **own** code

B. Program requirements: Your program can solve and meet all of the requirements described in 2.1 and 2.2. Do not write additional code that is not required.

C. Coding style: Your source code must be indented appropriately and contain adequate comments for class, methods, variables, etc.

- At the beginning of your Java class file, specify the following items:
  - Your name, the assignment number for this program, the creation date
  - The problem description that defines your application
  - Enumerate (list) the goals of your application
  - Enumerate (list) the inputs of your application
  - Enumerate (list) the outputs of your application
- Before each method in your class, specify the following items:
  - Specify the purpose of the corresponding method

    - Enumerate the possible inputs and outputs of the corresponding method
    - Write down the pseudocode to implement the corresponding method
- Others comments for variables, inside method implementation

2.4 Grading Rubrics

Your code in this part will be evaluated using the following grading scheme (50 points for completed functional source code + bonus 10 points for good code comments):

Points	Item
25	Correct class, variable, and method definition
10	Consistent coding style indented appropriately, such as indents, curly bracket positions, etc.
15	Have all required variables and methods
10	Good and helpful comments

Assignment Submission Instructions

1. Answer Part 1 questions in a WORD, PDF, or any other document application

2. Compress the whole Eclipse source code of your project in part 2 in zip format using the Export function

Submit two items above to D2L.