CSCI 2110 Data Structures and Algorithms
Lab 4
Release Date: Oct 4th
Due Date: Oct 11th
23h55 (five minutes to midnight)

Unordered Lists

Introduction

The objective of this lab is to help you get familiar with the Unordered List Data Structure and the concept of developing data structures in layers.

Marking Scheme

Each exercise carries 10 points.
Working code, Outputs included, Efficient, Good basic comments included: 10/10
No comments or poor commenting: subtract one point
Unnecessarily inefficient: subtract one point
No outputs and/or not all test cases covered: subtract up to two points
Code not working: subtract up to six points depending upon how many methods are incorrect. TAs can and will test cases beyond those shown in sample inputs, and may test methods independently.

Your final score will be scaled down to a value out of 10. For example, if there are three exercises and you score 9/10, 10/10 and 8/10 on the three exercises, your total is 27/30 or 9/10.

Error checking: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.

Submission: All submissions are through Brightspace. Log on dal.ca/brightspace using your Dal NetId.

What to submit:
Submit one ZIP file containing all source code (files with .java suffixes) and a text documents containing sample outputs. For each exercise you will minimally have to submit a demo Class with a main method to be tested and your test code, sample outputs.

Your final submission should include the following files: *List.java, Node.java, LinkedList.java, Expense.java,ExpenseList.java, ExpenseListDemo.java, Expenses.txt, GeographyQuiz.java ,CountryCapitals.txt, Exercise1out.txt, Exercise2out.txt*

You MUST SUBMIT .java files that are readable by your TAs. If you submit files that are unreadable such as .class, you will lose points. Please additionally comment out package specifiers.

Your code must compile and follow the naming conventions in this document. Failure to do so may result in a score of 0.

Exercise0 (Review – no marks)
Download the example code/files provided along with this week's lab document. You will need the following files to complete your work:

Node.java (Generic Node Class)
LinkedList.java (Generic Linked List Class)
List.java (Generic Unordered List Class)

Expense.java
ExpenseList.java (Sample application that uses the Unordered List Class)
ExpenseListDemo.java
Expenses.txt (Sample text file for input in Exercise0 & Exercise1)

Read and test the code provided. Run the ExpenseListDemo.java program, passing the Expenses.txt file as input, and check the results. You should see the following screen dialog:

```
Enter the filename to read from: Expenses.txt
September Expenses
Max expense: $360.01
Min expense: $4.51
```

Exercise1 (Completing the ExpenseList class)

Start Here:
For this exercise, you will complete the *avgExpense*, *totalExpense* and *amountSpentOn* methods of the ExpenseList.java file.

- avgExpense():double calculates the average expenses from the input file
- totalExpense(): double calculates the total expenses from the input file
- amountSpentOn(String): double calculates the amount spent on an category passed in as a string
    - For example, amountSpentOn("Groceries") will return the total amount spent on items denoted as groceries

Uncomment the print statements at the end of *ExpenseListDemo.java* file. Run the program again with Expenses.txt as the input file. You should see the following output:

```
Enter the filename to read from: Expenses.txt
September Expenses
Max expense: $360.01
Min expense: $4.51
Average expense: $76.635
The amount spent on groceries: $95.13
Total expense: $919.62
```

Input/output

Your program should accept input in the following format:

- Filename entered by the user

Your program should provide output in the following format:

- Report the Min, Max, Average expenses from the text file
- Report the amount spent on groceries
- Report the total expense

2 Student Generated Test Cases

- Uncomment the print statements at the end of *ExpenseListDemo.java*
    - Run your code
    - Save the output
- Add one more test case by adding a new print statement to *ExpenseListDemo.java*
    - Calculate the amount spent on Pizza
    - Save the output as Exercise1out.txt

Sample inputs and outputs:

Exercise1.in	*Exercise1.out*
Enter the filename to read from: Expenses.txt	September Expenses Max expense: $360.01 Min expense: $4.51 Average expense: $76.635 The amount spent on groceries: $95.13 Total expense: $919.62

Exercise 2: Geography Quiz

For this exercise you will write a small program to implement a geography quiz game that might be used to memorize the capital cities of countries.

Start Here:

Complete your work in the *main()* of the *GeographyQuiz.java* starter code. You will need the *CountryCapitals.txt* file to complete your work. This file has a country name on one line, followed by its capital on the next:

Afghanistan
Kabul
Albania
Tirana
...
Etc.

Your program should **accept input from a user** (specifying the name of an input file) and store the countries and capitals as items in an unordered list.

Your program should randomly select a country *or* a capital city, and ask a question that prompts a user to match a country to a capital city or a capital city to a country. It should then search the unordered list and

report whether the answer supplied by a user is correct or incorrect. At the end of the program, display a statistic describing how well the player performed. Your program must **accept input from a user** playing the game. Here's a sample screen dialog:

```
Welcome to the Country-Capital Quiz
Play? Yes
What is the capital of Canada?
Ottawa
Correct. Play? Yes
What country has Vienna as its capital?
Austria
Correct. Play? Yes
What is the capital of Albania?
Algiers
Incorrect. The correct answer is Tirana. Play? No
Game over.
Game Stats:
Questions played: 3; Correct answers: 2; Score: 66.67%
```

The following objects may be useful for randomly choosing an item in the list:

- Random r = new Random()
- r.nextInt(listsize)

Input/output

Your program should accept input in the following format:

- File name to read from
- User commands as outlined in the above example
    - You may ignore case sensitivity for "yes" and "no"

Your program should provide output in the following format:

- Report if the user guessed the correct capital or country
- Report incorrect answers
- State when the game has ended
- State the game stats as outlined in the example above

2 Student Generated Test Cases

Play two rounds with your game. Demonstrate that your game can perform the following:

- Read from file
- Ask the user if they want to play
- Ask a capital question
- Ask a country question
- Report incorrect
- Report correct
- Display game stats

Save the output as *Exercise2out.txt*