CSCI 2110 Data Structures and Algorithms
Laboratory No. 4
Week of 7 October

Due: Sunday 13 October
23h55 (five minutes to midnight)

Unordered Lists

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 a 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. Deadline for submission: Sunday 13 October 23h55 (five minutes before midnight).

What to submit:
Submit one ZIP file containing all source code (files with .java suffixes) and text documents containing sample outputs. For each exercise you will minimally have to submit a demo class with a main method, classes implementing the required generic structures, and sample outputs. If you wish, you may combine your test outputs into a single file called Outputs.txt via cut-and-paste or a similar method. Your final submission should include the following files: Node.java, LinkedList.java, List.java, Expense.java, ExpenseList.java, ExpenseListDemo.java, Student.java, StudentList.java, StudentListDemo.java, GeographyQuiz.java, Outputs.txt *or* labelled test outputs for each exercise.

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.

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)
StudentRecords.txt (Sample text file for input in Exercise2)
CountriesCapitals.txt (Sample text file for input in Exercise3)

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)
For this exercise, you will complete the *avgExpense*, *totalExpense* and *amountSpentOn* methods of the ExpenseList.java file. Uncomment the print statements at the end 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
```

Exercise2 (StudentList)
For this exercise you will develop a program that can store and manage information about students taking courses at a university.
First create a Student class. Call it Student.java. It will be similar to Expense.java. Student Objects should have the following fields: *StudentID*, *FirstName*, *LastName*, *Email*, *Major*, *Faculty*. Add appropriate methods and build out upon this framework as required..
Next create a StudentList class. Call it StudentList.java. It will be similar to ExpenseList.java. StudentList Objects should have a field that is an unordered List of type Student, and the following methods:

- `public void addRecord(Student s)`: Add a student record to the list.
- `public void deleteRecord(int ID)`: Delete a student record with the specified ID number.
- `public void displayMajors(String major)`: Display records of all the students taking a specified major.

- `public void displayFaculty(String faculty)`: Display records of all students belonging to a particular faculty.

- `public void displayName(String lName)`: Display records of all students with the specified last name.

- `public Student searchID(int ID)`: Search for a student's record given an ID number.

You may add other methods as necessary.

Finally, create a demo program. Call it StudentListDemo.java. It will be similar to ExpenseListDemo.java. Your program should **accept input from a user** (specifying the name of an input file) and read a file formatted like the one below (see also: StudentRecords.txt). Your program should create an unordered list of Student Objects, and demonstrate all the methods that you have developed (you may hard code these tests). Each line of the input file will consist of a Student ID number, First name, Last name, email address, Major and Faculty separated by whitespace.

```
201820 Kate West kwest@email.com Music Arts
201821 Julie McLain jmclain@email.com  Finance Business
201822 Tom Elrich telrich@email.com Sculpture Arts
201823 Mark Smith msmith@email.com Biology Science
201824 Ian Foster ifoster@email.com Physics Science
201825 Zhilei Wang zwang@email.com Finance Business
201826 Matt Knight mknight@email.com Music Arts
201827 John Smith jsmith@email.com CS ComputerScience
201828 Craig Cambell ccambell@email.com Linguistics Humanities
201829 Mike Williams mwilliams@email.com Music Arts
201830 Jane Reid jreid@email.com Physics Science
```

Note: One sample output which shows the results for all the methods that you have developed is sufficient.

A sample output is given below:

```
Enter the filename to read from: StudentRecords.txt
The Student List contains the following entries:
200132 Jane Reid jreid@email.com Music Arts
200131 Mike Williams mwilliams@email.com Linguistics Humanities
200130 Steve Edwards sedwards@email.com Biology Science
200129 Craig Cambell ccambell@email.com Music Arts
200128 Allison Page apage@email.com History Humanities
200127 John Smith jsmith@email.com Sculpture Arts
200126 Karen Weaver jkweaver@email.com Music Arts
200125 Matt Knight mknight@email.com Finance Business
200124 Jen Foster jfoster@email.com Physics Science
200123 Mark Smith msmith@email.com Biology Science
200122 Tom Erlich terlich@email.com Sculpture Arts
200121 Julie McLain jmclain@email.com Finance Business
200120 Kate West kwest@email.com Music Arts

These students are majoring in Music:
200132 Jane Reid jreid@email.com Music Arts
200129 Craig Cambell ccambell@email.com Music Arts
200126 Karen Weaver jkweaver@email.com Music Arts
200120 Kate West kwest@email.com Music Arts
```

```
These students are studying in the faculty of Science:
200130 Steve Edwards sedwards@email.com Biology Science
200124 Jen Foster jfoster@email.com Physics Science
200123 Mark Smith msmith@email.com Biology Science

Find the record for the student with ID '200128':
200128 Allison Page apage@email.com History Humanities

Find the record for the student with the name 'Mike Williams':
200131 Mike Williams mwilliams@email.com Linguistics Humanities

Find the record for the student with the email 'apage@email.com':
200128 Allison Page apage@email.com History Humanities

Remove 5 students from the Student List...
The Student List now contains the following entries:
200127 John Smith jsmith@email.com Sculpture Arts
200126 Karen Weaver jkweaver@email.com Music Arts
200125 Matt Knight mknight@email.com Finance Business
200124 Jen Foster jfoster@email.com Physics Science
200123 Mark Smith msmith@email.com Biology Science
200122 Tom Erlich terlich@email.com Sculpture Arts
200121 Julie McLain jmclain@email.com Finance Business
200120 Kate West kwest@email.com Music Arts
```

Exercise3 (Geography Quiz)

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

Note: One sample output which shows the results similar to the above is sufficient.