Computer Science COMP-2120 - Fall 2020

Assignment 3

Due: End of Wednesday, December 2, 2020

****

Note: Submit your assignment as a **single zip file** on Blackboard. The zip file should be named as Assignment1_*StudentId*.zip, in which replace *StudentId* with your university student number. The zip file should include only one text file and several Java files, i.e. files with the extension name ".java". DO NOT include any compiled class file. For every problem, follow the name of the files as instructed. For problem 2, put your answers in a text (txt) file.

****

Problem 1. (10 points)

Consider an interface

```
public interface NumberFormatter {
      String format(int n);
}
```

Provide four classes that implement this interface as follows: A class `DefaultFormatter` that formats an integer in the usual way. A class `DecimalSeparatorFormatter` that formats an integer with decimal separators; for example, one million as `1,000,000`. A class `AccountingFormatter` that formats negative numbers with parentheses; for example, `-1` as `(1)`. A class `BaseFormatter` that formats the number in base `n`, where `n` is any number between `2` and `16`, that is provided in the constructor.

Also, provide a tester class, `FormatTester`, that tests above classes using various integer values read from an input file, `Numbers.txt`, and creates an output file, `FormattedNumbers.txt`, and add the formatted numbers into it. For instance if the input file contains:

5 -10000 1000000 36

Then output file, based on the methods you call for these numbers, can contain something like:

Default Format: 5 -10000 1000000 36
Decimal Format: 5 -10,000 1,000,000 36
Accounting Format: 5 (10000) 1000000 36
Base 8 Format: 5 -23420 3641100 44
Base 2 Format: 101 -10011100010000 11110100001001000000 100100

Problem 2. (10 points)

The CSV (or comma-separated values) format is commonly used for tabular data. Each table row is a line, with columns separated by commas. Items may be enclosed in quotation marks, and they must be if they contain commas or quotation marks. Quotation marks inside quoted fields are doubled. Here is a line with four fields:

```
1729, San Francisco, “Hello, World”, “He asked: “”Where are you going?”””
```

Implement a class `CSVReader` that reads a CSV file, and provide the following methods:

```
int numberOfRows()                  // Returns the number of lines in the CSV file
int numberOfFields(int row)         // Returns the number of fields in a particular row
String field(int row, int column)   // Returns the field in a particular row and column
```

Then, test your class with the tester class, provided, and with the two CSV files, provided.

Problem 3. (10 points)

Supply a class `Person` that implements the `Comparable` interface. Compare persons by their names. Then in a tester class, `PersonTester`, ask the user to input ten names and generate ten Person objects, and then using the `compareTo` method, determine and print the first and last person among them.

Problem 4. (10 points)

Write a Java class, `FirstLetterMap`, that reads all words from an input file and add them to a map whose keys are the first letters of the words and values are sets of words that start with that same letter. Then print out the word sets in alphabetical order. Sample input files are provided.

Sample input file:

She took down a jar from one of the shelves as she passed; it was labelled 'ORANGE MARMALADE', but to her great disappointment it was empty.

Output:

```
a: [a, as]
b: [but]
d: [disappointment, down]
e: [empty]
f: [from]
g: [great]
h: [her]
i: [it]
j: [jar]
l: [labelled]
m: [marmalade]
o: [of, one, orange]
p: [passed]
s: [she, shelves]
t: [the, to, took]
w: [was]
```

Problem 5. (10 points)

Suppose you are implementing a utility class, `ArrayListUtil`, in which you provide some utility methods to apply on ArrayLists with various types. Provide a static method that reverses the elements of a generic ArrayList. Provide another static method that returns the reverse of a generic ArrayList, without modifying the original list.

Problem 6. (10 points)

Complete the code inside the Java file below, ListUtil.java, such that this class supplies a utility method to reverse the entries in a linked list. Then, test your code using the tester class, ReverseTester.java, given below.

ListUtil.java

```java
import java.util.LinkedList;

/**
   This class supplies a utility method to reverse the entries in a linked list.
*/
public class ListUtil
{
   /**
      Reverses the elements in a linked list
      @param strings the linked list to reverse
   */
   public static void reverse(LinkedList<String> strings)
   {
      // Complete this static method based on its JavaDoc comment.
   }
}
```

ReverseTester.java

```java
import java.util.LinkedList;

/**
   A test program to reverse the entries in a linked list.
*/
public class ReverseTester
{
   public static void main(String[] args)
   {
      LinkedList<String> employeeNames = new LinkedList<>();
      employeeNames.addLast("Dick");
      employeeNames.addLast("Harry");
      employeeNames.addLast("Romeo");
      employeeNames.addLast("Tom");

      ListUtil.reverse(employeeNames);
      System.out.println(employeeNames);
      System.out.println("Expected: [Tom, Romeo, Harry, Dick]");
   }
}
```