CSCI 2110 Data Structures and Algorithms - Fall 2017

Assignment No. 2
Date Given: Tuesday, October 10, 2017
Date Due: Wednesday, October 25, 2017, 11.55 p.m. (5 minutes to midnight)

EXERCISE 1: (60 POINTS)

This exercise is on the application of Unordered Lists. You will need the source codes for Node.java, LinkedList.java and List.java discussed in the lectures. You will also need the text file nhlstats.txt (these files are available to download along with the link to this assignment).

Before you begin this assignment, it will be useful to study the Expense.java, ExpenseList.java and ExpenseListDemo.java programs that were discussed in the lectures. You will also need knowledge of File Processing and String Tokenizer from CSCI 1101.

You are given the raw scores of various NHL players' regular hockey games in the file nhlstats.txt. The file has data organized in the following manner:

```
Name         Pos    Team   GP     G      A      PIM    SOG    GWG
Marchand     LW     BOS    45     18     18     27     91     5
Eberle       C      EDM    48     16     21     16     133    3
...
etc.
```

Note: ***The first row is not there in the actual nhlstats.txt file***. It just tells you what each column stands for.
The columns are as follows:
The first column is the name of the player.
Pos stands for Position. In this column, C means Center, LW means Left Wing, RW means Right Wing, LD means Left Defense, RD means Right Defense, and G means Goalie.
GP stands for Games Played.
G stands for Goals scored.
A stands for Assists.
PIM stands for Penalties In Minutes
SOG stands for Shots on Goal
GWG stands for Game Winning Goals.

So, for example, in the above file, Marchand plays Left Wing, belongs to team BOS (Boston Bruins), played 45 games during the season, scored 18 goals, had 18 assists, spent 27 penalty minutes, had 91 shots on goal and 5 game winning goals.

Now, from these numbers, hockey statisticians calculate interesting information, such as the following.

1. Player with the highest points. Points = Goals + Assists. For example, Marchand has 36 points and Eberle has 37 points.
2. Player with the maximum penalty points
3. Player with the most game winning goals
4. Etc.

Here is what your program should do. Follow these steps.

1. First create a class called PlayerRecord.java that has all the instance variables for one player (Name, Position, Team, GP, G, A, PIM, SOG, GWG). It may also be useful to add another instance variable P (points), which is just G+A.

```
public class PlayerRecord
{
….
}
```

2. Next create a class NHLStats holds an unordered list of PlayerRecord objects.

```
public class NHLStats
{
      private List<PlayerRecord> playerlist;

      ……
}
```

In this class, include the following methods, in addition to the constructor and other useful methods:

1. ***1.*** Who is the player with the highest points? : Method that displays the player's name with the maximum number of points and his team's name. ***Note: If more than one player has the largest number of points, display all the players and their teams.***
2. Who is the most aggressive player? : Method that displays the name of the player who had the maximum number of penalty minutes, his team name and his position. ***Same note as in 1 applies.***
3. Who is the most valuable player (MVP)? : Method that displays the name of the player who scored the most number of game winning goals and his team's name. ***Same note as in 1 applies.***
4. Who is the most promising player? : Method that displays the name of the player who took the most number of shots on goal and their team names. ***Same note as in 1 applies.***
5. Which team has the most penalty minutes? : Method that displays the name of the team that had the most penalty minutes (sum of all penalty minutes of all players in that team). ***Same note as in 1 applies.***
6. Which team has the most game winning goals? : Method that displays the name of the team that had the most number of game winning goals (sum of all GWGs for that team). ***Same note as in 1 applies.***

You may add other methods as needed.

Note: If you find it useful, you can create one more class called the Team class and store the attributes of each team. This will help you search and get answers specific to a team. You can also use an ArrayList within the NHLStats class to hold items belonging to a team, for example.

4. Write a client program NHLListDemo.java with the main method that reads the file nhlstats.txt and prints the following into another file nhlstatsoutput.txt.

Note: When you read data from the file, each line is read as a String. Use StringTokenizer to break it down into individual components.

Note that the input file has rows in which the items are delimited by tabs. So you need to use the StringTokenizer in a manner similar to the following:
token = new StringTokenizer(line, "\t");

Also to convert a String to an integer value, use Integer.parseInt(...).

Your output should be similar to the format given below:

```
NHL Results Summary

Players with highest points and their teams:
…
Most aggressive players, their teams and their positions:
…
Most valuable players and their teams:
…
Most promising players and their teams:
…
Teams that had the most number of penalty minutes:
….
Teams that had the most number of game winning goals:
….
```

Submit a zip file containing the following source codes:
PlayerRecord.java, NHLList.java, NHLListDemo.java, List.java, LinkedList.java, Node.java
and the output text file nhlstatsoutput.txt

EXERCISE 2: (40 POINTS)
This exercise is on the two-finger walking algorithm for ordered lists.
From the link alongside the lab, download the Ordered List class that was discussed in the lectures.

Step 1: Write a program that reads a list of names from a file and constructs an ordered list. For example, if the input text file is:
Shai
Tom
Jim
Aaron
Barbara
Beth
Fred
Jack
Jarred
Jill
Amar
Ralph
Jill
Hillary

it should create the following ordered list. Print the contents of the list:

[Aaron, Amar, Barbara, Beth, Fred, Hillary, Jack, Jarred, Jill, Jim, Ralph, Shai, Tom]

Note: Although Jill is repeated in the input list, the ordered list does not have repeated items.

Step 2: Modify the above program so that it reads two text files each containing a list of names and constructs two ordered lists.

Step 3: Add a method to the above program that takes as input two ordered lists and returns a third list that is a merger of the two ordered lists. ***Use the two-finger walking algorithm discussed in class.***

For example, if list1 is:

Amar	Boris	Charlie	Dan	Fujian	Inder	Travis

and list2 is:

Alex	Ben	Betty	Charlie	Dan	Pei	Travis	Zola	Zulu

the method should create the following new ordered list and return it:

Alex	Amar	Ben	Betty	Boris	Charlie	Dan	Fujian	Inder	Pei	Travis	Zola	Zulu

Note: The header of the method should be written as follows:

```
public static <T extends Comparable<T>> OrderedList<T> merge (OrderedList<T> list1,
    OrderedList<T> list2)
    {
            //your code here
    }
```

Your final code should be one java class file that reads two files containing random names, creates two ordered lists from these names, merges them and displays the three lists.