CSCI 2110 Data Structures and Algorithms

Assignment No. 2

Date Given: Tuesday, October 9, 2018

Date Due: Wednesday, October 24, 2018, 11.55 p.m. (5 minutes to midnight)

This assignment has two exercises, one on unordered lists and one on ordered lists. Download the zip file alongside the assignment link. It contains all the necessary files for you to proceed with this assignment.

EXERCISE 1: (65 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.

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 may want to study Exercise 0 of Lab 4 for a quick review.

The objective of this assignment is to perform analytics on NHL (National Hockey League) stats data and derive some interesting results. The nhlstats.txt file provided to you in the folder contains read a text file containing important statistics on each NHL player in the current 2018-19 hockey season. The text file is arranged as shown below. The meaning (legend) of each row is given below (not in the text file):

Williams, Justin	→ Player name (last name, first name)
CAR	→ Team (CAR is Carolina Hurricanes)
3	→ GP (Games Played)
0	→ G (Goals scored)
3	→ A (Assists)
3	→ P (Points)
3	→ +/- Rating
0	→ PPG (Power Play Goals)
0	→ PPP (Power Play Points)
1.00	→ PTSG (Points Per Game)
0	→ SHG (Short handed goals)
0	→ SHP (Short handed points)
0	→ GWG (Game Winning Goals)
2	→ PIM (Penalty Minutes)
10	→ SOG (Shots on Goal)
0.0	→ SHP (Shooting Percentage)
16:39	→ ATOI (Average Time on Ice)

The nhlstats.txt file contains the raw scores of 60 top NHL players arranged as shown above.

Now, from these numbers, hockey statisticians calculate interesting information, and that is what you are required to do in this exercise. It essentially entails building an unordered list of the record of each player and searching for interesting data.

Follow these steps.

1. First create a class called PlayerRecord.java that has ***all the 17 instance variables*** for one player (Name, Team, GP, G, A, …, ATOI).

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

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

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

      ……
}
```

In this class, include the following methods in addition to the constructor. You may add other methods if necessary.

1. Who is the player who scored the most GWG (game winning goals)? : This method displays the player's name and his team's name. ***Note: If more than one player has the largest GWG, display all the players and their teams.***
2. Who is the player who spent the most time on ice (largest ATOI)?: This method displays the player's name and his team's name. ***Note: If more than one player has the largest ATOI, display all the players and their teams.***
3. Who is the most aggressive player? : This method that displays the name of the player who had the maximum number of penalty minutes and his team's name. ***Same note as above 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 (SOG) and his team name. ***Same note as above applies.***
5. Which player provided the most assists? : This method displays the name of the player who has the most number of assists and his team name. ***Same note as above applies.***

3. Next create a class Team that holds the team name, total number of game winning goals and the total number of penalty minutes of players belonging to this team. Add appropriate constructor, get and set methods.

```
public class Team
{
      private String name;
      private int totalGWG;
      private int totalPIM;

      ……
}
```

4. Next create a class TeamStats that holds an unordered list of Team objects.

```
public class TeamStats
{
      private List<Team> teamList;
```

```
        ......
}
```

In this class, include the following methods in addition to the constructor. You may add other methods if necessary.

1. 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). ***If more than one team has the same total penalty minutes, display the names of all such teams.***
2. 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 above applies.***

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

```
import java.io.*;
import java.util.Scanner;
public class NHLStatsDemo
{
     public static void main(String[] args) throws IOException
     {
          ...
     }
}
```

Note: When you read data from the file, each line is read as a String. 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 the highest game winning goals and their teams:
…
Players with the highest average time on ice and their teams:
…



…
Teams that had the most number of penalty minutes:
….
Teams that had the most number of game winning goals:
….
```

NOTE: You can use acronyms such as CAR, TOR for team names. If you want the full list of abbreviations for each team, consult
https://en.wikipedia.org/wiki/Template:NHL_team_abbreviations

Submit a zip file containing the following:
Node.java, LinkedList.java, List.java, PlayerRecord.java, PlayerStats.java, Team.java, TeamStats.java, NHLStatsDemo.java, nhlstats.txt and nhlstatsoutput.txt.

EXERCISE 2: (35 POINTS)
This exercise is on the two-finger walking algorithm for ordered lists.
The zip file provided to you has 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.

```
//appropriate imports
public class MergeLists
{
      public static void main(String[] args) throws IOException
      {
            …
      }
}
```

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 and display 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.

You can create two small files containing random names yourself.

Submit MergeLists.java, the two input files and the result output in your zip file.