# HW6: Programming with lists

Due Wednesday Oct. 15 1:59PM through turnin
Total: 20pts

You may work on this assignment with one other student. A team of two members must practice pair programming. Pair programming "is a practice in which two programmers work side-by-side at one computer, continuously collaborating on the same design, algorithm, code, or test." Both members in a team must read the article All I really need to know about pair programming I learned in kindergarten. This article can be downloaded to any computer that is in the Marquette network.

## Preparation

The textbook includes a program, namely RefSortedList.java that implements a sorted list with a public method in pages 442-445 and reuses some public methods in RefUnsortedList.java in pages 437-440. You are recommended to study and understand the program before you attempt the assignment.

## Task

For this assignment, you are going to write a java program that uses linked structure to implement a list for keeping track of the top 10 players for a game system. The players are stored in the list in descending order based on their scores. A player has a name and a score.

The classes from the textbook you can reuse:
1. public class `LLNode`. The file `LLNode.java` is at page 204 in the textbook.

The classes that you will implement:
1. Class `Player` has two private, non-static instance variables: `name` and `score`. Assume the players' names are unique. The class has constructors, public non-static accessor and modifier methods that get and set the instance variables. This class should implements `Comparable` interface and compare on the `score`.

2. Class `TopTenPlayersList`. It has a private instance variable `private LLNode list`. You will implement the following methods.
    a. `public void addEntry(String name, double score).`
    It adds the given name and score to the list if (1) the player is not in the current list, or (2) the player is in the current list but has a lower score. For case 1, add the entry to the right place so that the players are ordered by their scores in descending order. For case 2, do the same as for case 1. In addition, delete the existing entry for the given name with the lower score.
    b. `public void removeFirst()` and `public void removeLast()` to remove the first player and last player in the list respectively.
    c. `public String toString()`
    It returns a string that contains the information of the top ten players, or all players if there are less than ten players available in the list.
    d. Other methods as needed or if you prefer.

## Test your design and implementation

You should try to exactly match the solution output for a given input.

A driver `TopTenPlayersRunner.java` is provided. This file includes a `main(String[] args)` that reads from a file and writes to the console. The input filename will be obtained via command line arguments. For example,

```
java  TopTenPlayersRunner scores.in
```

`scores.in` is the input file.

**Input:** A sample content of `scores.in` is as follows:

```
#game player score
1 David 97
2 Sarah 50
3 Joe 120
4 Peter 150
5 David 160
…
-1  nobody -1
```

The fields in each line are tab delimited. The record with -1 for the #game field is the sentinel. The program will stop reading from the input file here.

**Output:** Your program prints out the top ten players to the console when the #game is a product of 5 (such as 5, 10, 15, 20, …). If there are less than ten players in the list, print out all players. The content of the output is in the format of (#rank player score). For example, after game 5, the output for the above input is:

```
After game 5, the top players are:
1 David 160
2 Peter 150
3 Joe 120
4 Sarah 50
```

When the program sees the sentinel record, it calls the first and last record in your linked list first and then print out the updated top ten players to the console.

## Submission

Submit your java files including TopTenPlayersList.java, Player.java, and other java files you have programmed to turnin system. **Do not submit TopTenPlayersRunner.java and LLNode.java.** The provided two java files will be used to compile and run your submission. If your submission requires different versions of `TopTenPlayersRunners.java`, it won't pass compilation and tests.

You should use a single command to turnin all files and separate the files with space. For example:

```
turnin -c cosc2010-Ge -p TopTenPlayers TopTenPlayersList.java Player.java
```

You can submit multiple times but only the last submission will be saved by turnin. If you work in a team of two, make sure to write both names in the beginning of TopTenPlayersList.java as JavaDoc and always let the same user submit the files. The lower grade from two submissions will be used if both team members submit files.