Assignment 4

Objectives

- Practice with searching and sorting algorithms
- Exposure to Java Collection<E> interface

  http://docs.oracle.com/javase/7/docs/api/java/util/List.html
  http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html
  http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html
  http://docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html
- Exposure to Java Collections class

  http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html

Introduction

This assignment has two parts:

- implement a variety of sort and search methods
- implement static methods that perform sorting and searching of List<T> objects

Part I

NOTE: In this part of the assignment, you CANNOT use the static methods built into the `Collections` class. You can and should be using the `List, LinkedList` and `ArrayList` classes and their instance methods but NOT the built in sort – you are making your own ☺.
Implement the blank method stubs in `ListOperations.java`. You will notice two methods are implemented for you. They give you an example of the use of the `get(num)` method and `listIterator()` in `List` interface as well as provide you some useful functionality to print your list while you debug your methods.
Use `TesterPart1.java` to test your methods. You will notice the tester will run all tests and provide a tally of the number of tests passed at the end of the program execution. This allows you to work on them in any order. Your output will look similar to this to start:

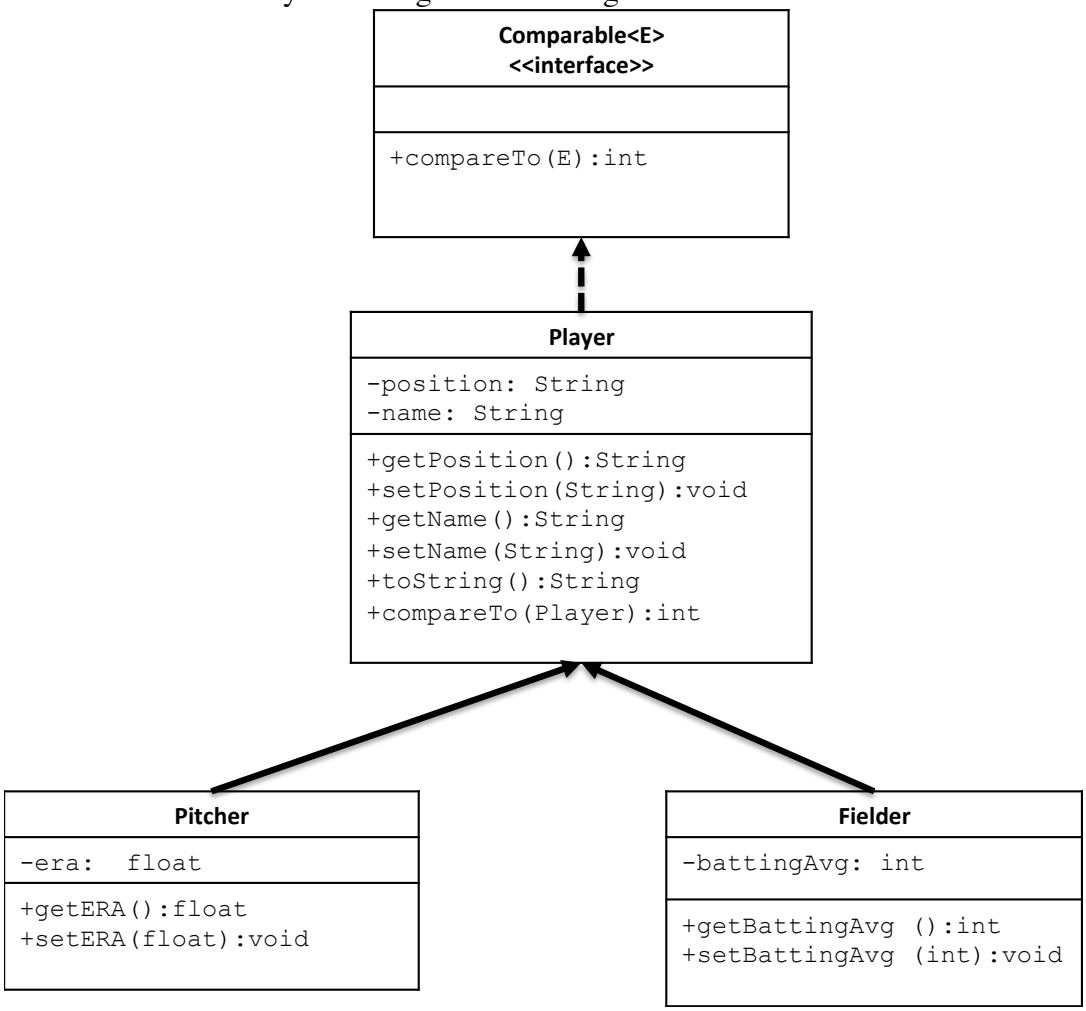
```
Testing bubblesort with get
Failed test: 0 at line 69
Failed test: 1 at line 70
Testing bubblesort with iterator
Failed test: 2 at line 94
Failed test: 3 at line 95
Testing quicksort
Failed test: 4 at line 119
Failed test: 5 at line 120
Testing sequential search with get
Failed test: 6 at line 143
Failed test: 7 at line 144
Testing sequential search with iterator
Failed test: 8 at line 167
Failed test: 9 at line 168
Testing binary search with get
Failed test: 10 at line 191
Failed test: 11 at line 192
Passed this many tests: 0
```

Part II

Make sure you have the following files in the same directory for this part of the assignment. The three `.class` files provide is necessary for the tester to work.

```
Fielder.java
fielderData.txt
ListOperations.java
Pitcher.java
pitcherData.txt
Player.java
PlayerAnalysis.java
PlayerAnalysisSoln.class
BAVGComparator.class
ERAComparator.class
PositionComparator.class
TesterPart2.java
```

In this part of the assignment you are to write methods that will process raw baseball player stats given to you in text files (`fielderData.txt` and `pitcherData.txt`). The data must be processed and an object created for each line entry following this class diagram:



Use `TesterPart2.java` to test your methods.

Step 1:

You may choose to override the `toString()` method in `Fielder` and `Pitcher` to give more information about the `era` and `battingAvg` when you print objects of those types.

Step 2:

You need to complete the implementation of `compareTo` in the `Player` class. The tester tests the implementation of this method first. The correctness of the methods you implement in Step 3 will depend on it.

Step 3:

For this part of the assignment you are welcome to use any of the methods provided in the `Collections` class (http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html). Implement the four methods in `PlayerAnalysis.java` tested by `TesterPart1.java`. You are welcome to add any private helper methods you see necessary. The last method maybe trickier but the first three are straightforward with the first two being mostly complete.

Submission

Submit only your `ListOperations.java`, `Player.java`, `Fielder.java`, `Pitcher.java` and `PlayerAnalysis.java` and any other files that your classes depend on via conneX. **Please be sure you submit your assignment, not just save a draft**. Submit as many times as you want, your last submission will be the one graded.

A reminder that it is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement their own solution.

We will be using plagiarism detection software on your assignment submissions.

Grading

If you submit something that does not compile, you will receive a grade of 0 for the assignment. It is your responsibility to make sure you submit the correct files.

Requirement	Marks
Passes part 1 tests	12
Passes part 2 tests	11
Total	**23**