CS424/524 Programming Languages — Programming Assignment #1

Due: At beginning of class on Thursday, Feb 20th — (40 points)

Problem Statement

You are to develop a program in the Go programming language to read Baseball Player objects from an input file into a list/collection. You are to store the players in a collection in such a way that when they are displayed to the screen, they are in sorted order by lastname (then firstname to break name ordering ties).

Each line of the data file will contain one player's name and stats.

firstname lastname plateappearances atbats singles doubles triples homeruns walks hitbypitch

It is possible there are errors in the data lines. The errors you need to check for are that each numeric value is valid, and that all 10 values are on the line. Read until the end of the file is found.

Operations/Computations:

- Compute batting average: this is the sum of the hits (singles, doubles, triples, home runs) divided by the number of at-bats.
- Compute slugging percentage. The slugging percentage is a weighted average. It is computed by:

$$slugging = \frac{singles + 2 * doubles + 3 * triples + 4 * homeruns}{number\ of\ at\ bats}$$

- Compute the on base percentage. OBP is the sum of all hits, walks and hit-by-pitch divided by the number of plate appearances.
- Compute the team's overall batting average.

Summary of Operation

- Prompt the user for the input file name. DO NOT hardcode file names into your program.
- Open input file
- Read each player and add them to your List.
- Write a report summary line to the screen.
- Write each item from the list formatted to the screen, along with any other output required by the assignment
- Write an error report at the end of the report.

TURN IN:

Please turn in a print out of each of your program files. Submit the electronic version of your project on CANVAS. Make sure your name and the name of the file is in the comments at the top of each program file, along with the system you tested your program on. It must work on at least version 1.11 (11 is installed on some of our PCs and 12 is installed on the linux systems.

NOTE:

I am not teaching the Go language in class. It is up to you to give yourself enough time to explore and learn the language in order to complete this assignment.

The Go language tools can be downloaded from https://golang.org/

There is a tutorial for learning the basics of Go at: https://tour.golang.org/

Other Requirements

- Your program must be well-commented. Comment all variables, functions and remember to have a section of comments at the top of your program that includes your name, date, course section and a description of what your program does.
- Use good variable names.
- Think about data objects and what they contain. Even though GO is not fully object-oriented (as in inheritance) we should be using good design practices.
- Use good and consistent naming conventions for data members.
- Use proper code indentation to make sure your program is easy to read and understand.

REFERENCES

I am using the online baseball reference (www.baseball-reference.com/players) to look up batting statistics. Please note that the on-base percentage I get is slightly off of the actual percentages due to not using quite ALL of the plate appearance data in my calculations.

Sample Execution

```
Welcome to the player statistics calculator test program.  I am going to
read players from an input data file.  You will tell me the name of
your input file.  I will store all of the players in a list,
compute each player's averages and then write the resulting team report to
your output file.


Enter the name of your input file:  playerinput.txt

. . .   ← note player data will display here (see samples below)


End of Program 1
```

Sample Input File and Corresponding Output Data
sample input text file

```
Hank Aaron     13941 12364 2294 624  98 755 1402 32
Chipper Jones  10614  8984 1671 549  38 468 1512 18
Ty Cobb        13099 11434 3053 724 295 117 1249 94
Jonny Bench     8674  7658 1254 381  24 389  891 19
Tony Gwynn     10232  9288 2378 543  85 135  434 24
John Smoltz     1167   948  118  26   2   5   79  3
```

Note – there should be NO blank lines after the last line of data in an input file.

Sample output

```
BASEBALL TEAM REPORT --- 6 PLAYERS FOUND IN FILE
OVERALL BATTING AVERAGE is 0.290

    PLAYER NAME      :    AVERAGE  SLUGGING   ONBASE%
-----------------------------------------------------
         Aaron, Hank :     0.305    0.555      0.373
        Bench, Jonny :     0.267    0.476      0.341
            Cobb, Ty :     0.366    0.512      0.422
         Gwynn, Tony :     0.338    0.459      0.352
      Jones, Chipper :     0.303    0.529      0.401
        Smoltz, John :     0.159    0.207      0.200


--- (see example below for format of error report)
```

Sample File with errors in it.

Sample input text file

```
chipper jones 10614 8984 1671 549 38 468 1530
hank aaron 13941 12364 2294 624 98 755 1434
error example1 100 100 100
error example2 10614 8984 1671 5x9 38 468 1530
```

Sample output

```
BASEBALL TEAM REPORT --- 2 PLAYERS FOUND IN FILE
OVERALL BATTING AVERAGE is 0.304

    PLAYER NAME      :    AVERAGE  SLUGGING   ONBASE%
-------------------------------------------------------
         Aaron, Hank :     0.305    0.555      0.373
      Jones, Chipper :     0.303    0.529      0.401

----- 2 ERROR LINES FOUND IN INPUT DATA ----

 line  3: Line contains not enough data.
 line  4: Line contains invalid numeric data.
```

General Grading Rubric (out of 40 points)

I provide this as a general guideline for how each area of a program is to be rated for grading purposes.

Performance Element	5 - Excellent	4 – Very good	3 - Good	2 - Limited	1 - Inadequate
Specifications ***(20 points)***	Program runs & meets all specifications	Runs, gets correct answers, output displayed correctly, meets most other specifications	Runs, gets correct answers, output is not displayed correctly or other specifications not met	Runs, get some correct results, does not meet most specifications	Program does not run, runs but gets no results or mostly wrong results
Readability ***(5 points)***	Code is well organized and easy to follow	Most of the code is well organized & easy to read	Parts of the code are easy to read but the organization is not good	Code is readable if the reader knows what it is supposed to be doing	Code is poorly organized and very difficult to read
Documentation (i.e. comments) ***(5 points)***	Documentation is clearly written & explains what the program as a whole should do; comment blocks introduce each function	Documentation is brief but helpful, includes header information for functions	Documentation consists of embedded comments and some header information for functions	Little or no documentation other than obvious embedded comments	Little or no documentation
Software Architecture ***(10 points)***	Displays excellent information hiding and modularity	Displays good information hiding and modularity	Displays some information hiding and modularity	Displays some information hiding or modularity but not both	Little or no structure