Programming Languages (CS524)

Programming Assignment I

Feb 20, 2020

Prepared By : Pooja Khanal , A25288740

Includes :

- **Software Description Document**
- **Software Test Plan Document**

Table of contents

Software Description Document

1.System Overview

A program is developed in GO programming language to read Baseball Player objects from an input file into a list/collection. The players are stored in 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 tiles) . Each line of the data file will contain one player's name and stats.

{ firstname lastname plateappearances atbats singles doubles triples homeruns walks hitbypitch}

Also, It is possible there are errors in the data lines. The errors are checked for that each numeric value is valid, and that all 10 values are on the line. The program reads until the end of the file to read for players.

2.Referenced Documents

[1] Programming Assignment I , CS-424/524 Spring 2020

[2] https://tour.golang.org/

3. Concepts of Execution

3.1. Operations/Computations

batting average: this is the sum of the hits (singles, doubles, triples, home runs) divided by the number of at-bats.

slugging percentage The slugging percentage is a weighted average. It is computed by:

slugging = (*singles*+2**doubles*+3**triples*+4*h*omeruns*)/*number of at bats*

on base percentage OBP is the sum of all hits, walks and hit-by-pitch divided by the number of plate appearances.

Team's overall batting average is also computed.

3.2. User input and Prompts

The user is asked for **inputfilename, Y/N** to declare whether to write the output to file or not, and **outputfilename** if to be written.

4.Abstract Data type (ADT)

4.1. Player structure

The player structure has the following attributes.
-{firstname lastname plateappearances atbats singles doubles triples homeruns walks hitbypitch}
Each player is added to the list of players after reading from file, one by one

5.Code Outline

List of Functions used

1. **`func (p Player) getBattingAverage() float64`**
   a. `Purpose: Calculate the batting average of each player.Calculation mentioned above`
   b. `Argument : Player object`
   c. `Return Type : float64`

2. **`func (p Player) getSluggingPercentage() float64`**
   a. `Purpose: Calculate the slugging percentage of each player. Calculation mentioned above`
   b. `Argument : Player object`
   c. `Return Type : float64`

3. **`func (p Player) getOnBasePercentage() float64`**
   a. `Purpose: Calculate the Base percentage of each player. Calculation mentioned above`
   b. `Argument : Player object`
   c. `Return Type : float64`

4. **`func getPlayer(s []string, lineCount int) (Player, error)`**
   a. `Purpose: //get the details of the player and create an instance of each Player Structure`
   b. `Argument : splitter and number of lines`
   c. `Return Type : Player, error`

5. **`func checkErrors(s []string, lineCount int) error`**
   a. `check for errors in the line,`

b. Line contains not enough data
c. line %d: Line contains invalid numeric data
d. Argument: splitter and number of lines
e. Return type : error

6. **func getPlayers(fileName string) ([]Player, []error)**
   a. Read from the file and append each player to the Player Structure
   b. Argument: filename
   c. Player structure and Error Structure Array

7. **func printErrorReport(errors []error)**
   a. Purpose: print the overall error report
   b. Argument: Error Structure
   c. Return type: void

8. **func printPlayerReport(players []Player)**
   a. Purpose: print the overall player report with the tabular structure
   b. Argument: Player Structure
   c. Return type: void

9. **func handlePrintToFile(players []Player, errors []error)**
   a. Purpose: /Output stream handler to write to file
   b. Argument: Player and Error Structure
   c. Return Type : void

10. **func main()** {
    a. Purpose: Main function to run the program. This is implemented in source.cpp
    b. Arguments: Not Applicable
    c. Return type : Integer after successful completion of program
    d. Contains the test abstraction and calls the required test function for running the application.

Software Test Plan Document

1. Test Procedures

a. **Purpose:** To test the validity of software and to test if the software is acceptable to the market or not. For this program, various test cases are created

b. **Procedure followed for Testing and Testing Strategies**

i. **Black box testing**

1. The input file was processed and seen if the output was written correctly to the output file or not.
2. Below mentioned test cases were run.

ii. **White box testing**

1. Modular approach was taken and each function was tested to see if it works or not. For such, the return type and return values were also compared.
2. After all the modules (functions) were tested, all the modules were integrated into one program and tested.

2. Test cases :

1. File without errors
2. File with all type of errors
3. File with one type of error
4. File with line having both type of error
5. Integers with long int (instead of string)
6. Alphabetically sorted or not

3. Tested Environments:

1. macbook pro 2015, go version go1.13.8 darwin/amd64

2. go version go1.11.5 windows/amd64

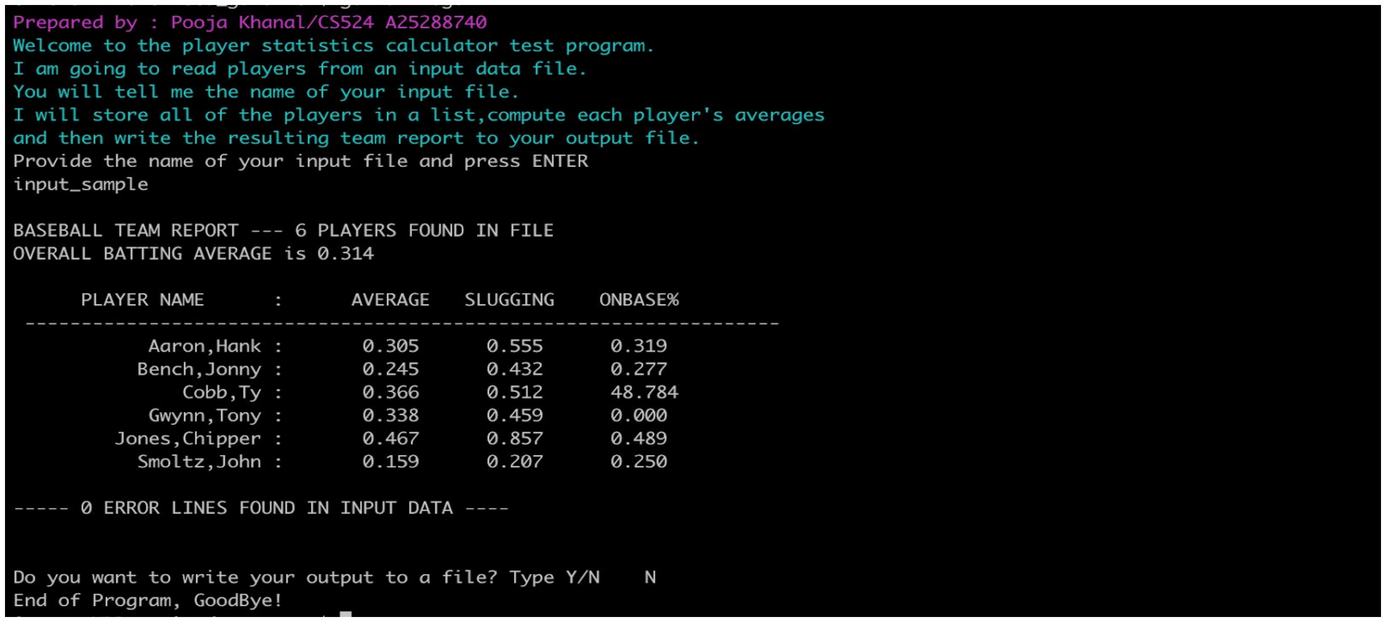
4. Sample Runs

```
Command to compile and build: go run *.go
```

a. Sample Input I

```
Hank Aaron 13941 12364 2294 624 98 755 1402 32

Chipper Jones 10614 8984 1671 2020 38 468 567 897

Ty Cobb 111 11434 3053 724 295 117 1249 94 889

Jonny Bench 8674 7658 1254 212 24 389 891 19

Tony Gwynn 2147483647 9288 2378 543 85 135 434 24

John Smoltz 1167 948 118 26 2 5 79 67 89
```

Output :

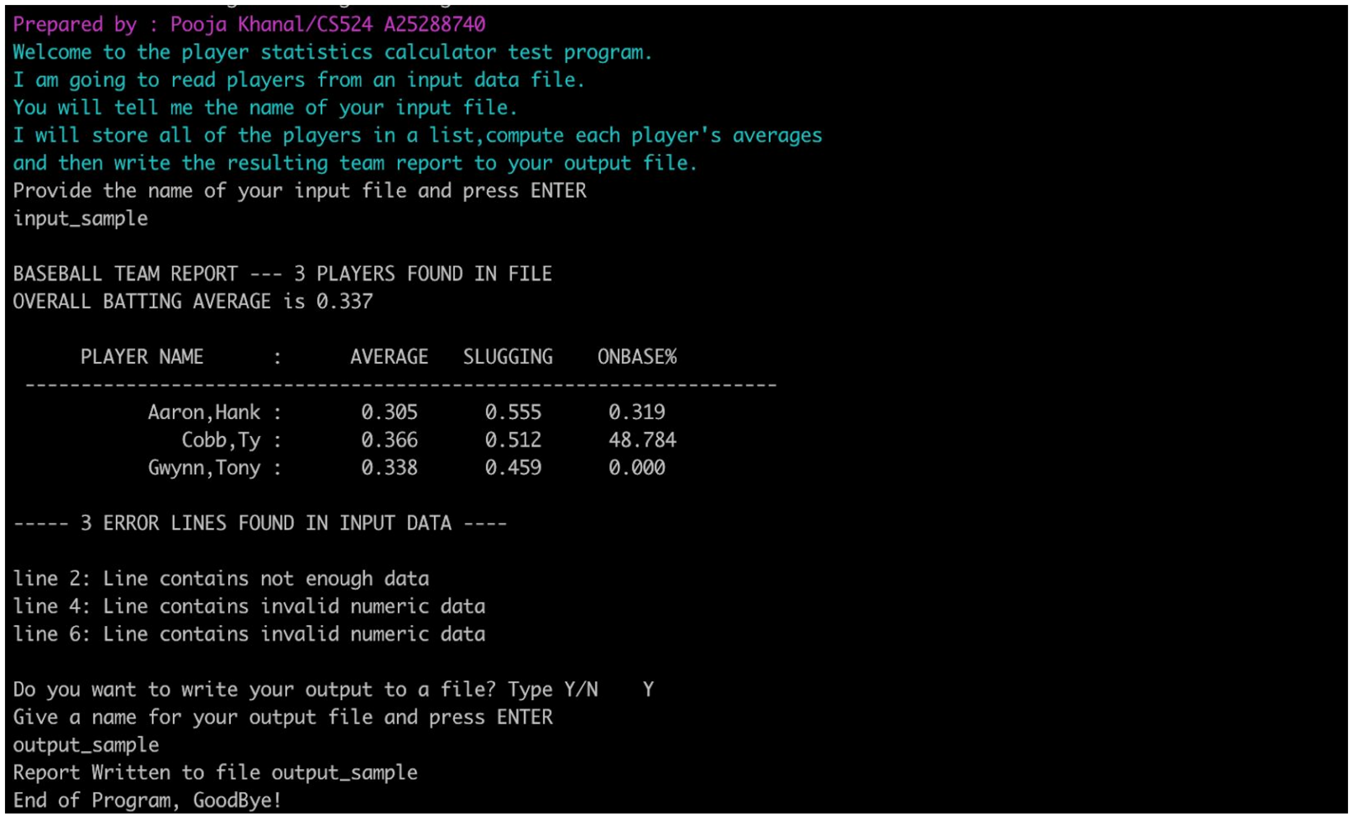


```
Prepared by : Pooja Khanal/CS524 A25288740
Welcome to the player statistics calculator test program.
I am going to read players from an input data file.
You will tell me the name of your input file.
I will store all of the players in a list,compute each player's averages
and then write the resulting team report to your output file.
Provide the name of your input file and press ENTER
input_sample

BASEBALL TEAM REPORT --- 6 PLAYERS FOUND IN FILE
OVERALL BATTING AVERAGE is 0.314

      PLAYER NAME      :      AVERAGE   SLUGGING    ONBASE%
 ----------------------------------------------------------------
            Aaron,Hank :       0.305      0.555      0.319
           Bench,Jonny :       0.245      0.432      0.277
               Cobb,Ty :       0.366      0.512      48.784
            Gwynn,Tony :       0.338      0.459      0.000
         Jones,Chipper :       0.467      0.857      0.489
           Smoltz,John :       0.159      0.207      0.250

----- 0 ERROR LINES FOUND IN INPUT DATA ----


Do you want to write your output to a file? Type Y/N    N
End of Program, GoodBye!
```

b. Sample Input II

```
Hank Aaron 13941 12364 2294 624 98 755 1402 32

Chipper Jones 10614 8984 1671 -2.345 38 468 1512

Ty Cobb 111 11434 3053 724 295 117 1249 94 889

Jonny Bench 8674 7658 1254 string 24 389 891 19

Tony Gwynn 2147483647 9288 2378 543 85 135 434 24

John Smoltz 1167777777777777777777 948 118 26 2 5 79 67 89
```

Output:



```
Prepared by : Pooja Khanal/CS524 A25288740
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.
Provide the name of your input file and press ENTER
input_sample

BASEBALL TEAM REPORT --- 3 PLAYERS FOUND IN FILE
OVERALL BATTING AVERAGE is 0.337

      PLAYER NAME      :      AVERAGE   SLUGGING    ONBASE%
 -------------------------------------------------------------------
            Aaron,Hank :       0.305      0.555      0.319
               Cobb,Ty :       0.366      0.512      48.784
            Gwynn,Tony :       0.338      0.459      0.000

----- 3 ERROR LINES FOUND IN INPUT DATA ----

line 2: Line contains not enough data
line 4: Line contains invalid numeric data
line 6: Line contains invalid numeric data

Do you want to write your output to a file? Type Y/N    Y
Give a name for your output file and press ENTER
output_sample
Report Written to file output_sample
End of Program, GoodBye!
```

5.Assumptions

1. The file format (input) is always consistent.
2. There is no new line at the end of the input file.
3. Round of to 3 decimal places for all float values.