Project Four Pseudocode

Andy Nguyen

CS – 2336.003

Smith

Main Class

Main

- Call hash maps for key file, away team, and home team
- **Read Key File**
- **Read Play by Play File**
- **Extract and Alphabetically Sort Stats of Team**
- **Display Stats of Team**
- Create an array list for each possible leader stat
- Use **Find Leaders** to create and then store each of those leader array lists in one big array list
- **Display Leaders**

Ask Input File Name

Parameters: Nothing

Return: string containing the file name

- Call a scanner object
- Ask for a file to be inputted by the user and store input into string

Read Key File

Parameters: scanner object, hash map for key file

Return: Nothing

- Create necessary objects for reading in files and call the “keyfile.txt”
- Using a scanner, use while loop to loop through each line of the file
    - Read in the type of stat associated with the code
    - While loop through the lines containing the code and stop until another type of stat has occurred
        - Store the types of code associated with the stat into the key file hash map

Read Play by Play File

Parameters: scanner object, hash map for home team, hash map for away team

Return: Nothing

- **Ask Input File Name**
- Create necessary objects for reading in files and call the file containing the play by play
- Using a scanner, use while loop to loop through each line of the file
    - Read in a line from the file
    - If the first character is A // if the player is on the away team
        - Read the name of the play and create a player object
        - If the name is not in the hash map // using hash map's contain method
            - Using the hash map created by using the key file, increment the stat using the get method from the hash map; the parameter being the parsed code from the file line
            - Store the name of the player and the player object into the away team hash map
        - Else if the name is already in the hash map // handles occurrences of the same player in play by play
            - Using the get hash method, get the player object associated with the name and increment the parsed stat
    - Else if the first character is H // if the player is on the home team
        - Read the name of the play and create a player object
        - If the name is not in the hash map // using hash map's contain method
            - Using the hash map created by using the key file, increment the stat using the get method from the hash map; the parameter being the parsed code from the file line
            - Store the name of the player and the player object into the home team hash map
        - Else if the name is already in the hash map // handles occurrences of the same player in play by play
            - Using the get hash method, get the player object associated with the name and increment the parsed stat

Extract and Alphabetically Sort Stats of Team

Parameters: hash map of whatever team

Return: an array list with sorted alphabetically

- Using the hash maps for each method, we will store each player object into an array list // extract the values from the hash map for sorting
    - For loop which loops through the array list // as we are adding items to array list, add them in alphabetically
        - If the array list is empty
            - Add the player object to array list
        - If the Compare To value of the player named contained within in the player object is greater than the Compare To value of the player's name of the player object already in the array list.

- Append to the end of the array list

Check List In Order (Least to Greatest or Greatest to Least)

Parameters: an array list containing extracted player object data from hash map, boolean value which determines to check from least to greatest or greatest to least, and a string representing the type of stat that needs to be sorted

Return: a Boolean saying whether the array list is in order or not

- For loop which loops through each element in the array list
    - If the Compare To value of the player objects **Which Stat to Get** is greater than the next element and the boolean value of checking least to greatest is true
        - Return false
    - If the Compare To value of the player objects **Which Stat to Get** is less than the next element and the boolean value of checking least to greatest is false
        - Return false
- Return true if the elements are in order

Sort Stats of Team

Parameters: array list containing player objects, boolean value which determines to check from least to greatest or greatest to least, and a string representing the type of stat that needs to be sorted

Return: an array list with sorted alphabetically

- While **Check List In Order** is false
    - Use Compare To method to compare adjacent player objects **Which Stat to Get**
    - If least to greatest boolean is true and the earlier element is greater than the later element
        - Save a copy of the earlier element
        - Delete the earlier element
        - Append the copy of the earlier element
    - If least to greatest boolean is false and the earlier element is less than the later element
        - Save a copy of the earlier element
        - Delete the earlier element
        - Append the copy of the earlier element
    - Break out of the loop if there is no element after the later element

Find Leaders

Parameters: array list of sorted player objects in either least to greatest or greatest to least

Return: an array list of stat array list leaders

- Get the first element's value and add to leader array list

- For loop to loop through each element in array list
    - If the value at index matches the first element
        - Add them on to leader array list
        - Increment count of 1st leaders
- If the value of 1st leaders is over or equal 3
    - Return the array list leaders
- Get the value of the element not equal to 1st leaders
- For loop to loop through each element in array list
    - If the value at index matches the first element
        - Add them on to leader array list
        - Increment count of 2nd leaders
- If the value of 1st leader + the 2nd leaders is over or equal 3
    - Return the array list leaders
- Get the value of the element not equal to the 2nd leaders or 1st leaders
- For loop to loop through each element in array list
    - If the value at index matches the first element
        - Add them on to leader array list
        - Increment count of 3rd leaders
- Return the array list leaders

Display Leaders

Parameters: array list of array list of stat leaders

Return: Nothing

- For loop which loops through all possible types of leaders
    - Print out the leader category
    - For loop which loops through the 3 types of leaders (1st, 2nd, 3rd leaders)
        - For loop which loops through the players within the 1st/2nd/3rd
            - If the stat is a double type
                - Add the decimal formatted player stat value (perform this once per leader)
                - Append this player name next to stat
            - Else
                - Add the integer value of player stat value (perform this once per leader
                - Append this player name next to stat
            - If the player index is not equal to last element in array list
                - Add a comma to the output string
            - Else
                - Add a new line
    - Print out the stat line or 1st/2nd/3rd leaders

Display Stats of Team

Parameters: array list of alphabetical player objects, boolean value which determines if the team received is home or away

Return: Nothing

- If the boolean value is true
    - Print “HOME
- Else
    - Print “AWAY”
- For loop which loops through the array list
    - Print out the player object using the **To String** method from the class. // Formatting of output is done within **To String**

Player Class

Calculating Batting Average // this function calculates the batting average when it is necessary preventing stale data

Parameter: Nothing

Return: double value holding the batting average

- Create integer holding the at bat value and set it equal to sum of the strike out, out, and hit stats
- If at bat value is equal to 0
    - Return 0 // prevents divide by zero scenario
- Create double batting average value holding batting average and set equal to quotient of the player hit stat divided by the at bat value

Calculating On Base Percentage

Parameter: Nothing

Return: double value holding the on base percentage

- Create an integer holding the numerator of the on base percentage formula set equal to sum of hit, walk, and hit by pitch stats
- Create an integer holding the plate appearances (also known as the denominator of the formula) value set equal to hit, out, strike out, walk, hit by pitch, and sacrifice stat
- If numerator or denominator is equal to zero
    - Return 0
- Create double value set equal to the numerator of the on base percentage formula and divide by the casted double of the plate appearance

Which Stat to Get

Parameters: String which represents desired stat that the player class should get/return

Return: Nothing

- If the string is equal to H
    - Get player hit stats
    - Else if string is equal to O
        - Get player out stats
    - Else if string is equal to strikeout
        - Get player strikeout stats
    - Else if string is equal to W
        - Get player walk stats
    - Else if string is equal to P
        - Get player hit by pitch stats
    - Else if string is equal to S
        - Get player sacrifice stats

To String Method

Parameters: Nothing

Return: string which outputs player stats

- Call a string which contain all the necessary stats to display for each player

Test Cases

1) Have a play-by-play file have multiple instances of the same player and make sure stats are properly incremented and combined
2) Test the searching capability of hash map and make sure it can find a player's object utilizing the player's name associated with the player object
3) Test the searching capability of hash map and make sure it can find the associated stat if given a code from the play by play
4) Test the extraction of the values from the hash map by utilizing the for each and make sure the term object has its proper stats gathered from the play-by-play file
5) Test the parsing of the play-by-play file and make sure the At-Bat value is properly incremented when encountering an error code.
6) Check that all codes scanned from the key file have been put into the hash map and that their associated stat is properly paired
7) Check the proper differentiation of home players and away players when parsing the play-by-play file and making sure that the home and away players go into their respective hash maps
8) Test the hash map's ability to handle large play-by-play files with lots of players and stats
9) After finishing the processing of the play-by-play file, check the alphabetical sorting of the players in their respective teams when taking out values from the hash table
10) Check that the proper combination and incrementation of data when there is only one player but a bunch of stats and it also resulting in proper display of stats and leaders
11) Check the output of program in the off chance that there might not be any players on a home/away team and that there is proper display for the edge case

12) Check the output of league leaders when encountering a bunch of ties stat wise and by making sure the hash map is properly combining and incrementing data
13) Check the ability of the hash map by searching for players after finished processing the play-by-play file. For example, looking up players name and getting their player objects. This is useful for helping to gather necessary data to determine league leaders in certain stats
14) Test out the incrementation of a multitude of stats when there is a bunch of plate appearances of that player in the play-by-play
15) When finding leaders, after finishing extraction of the play-by-play, make sure when outputting values from hash map that they can be ordered least to greatest or greatest to least for any aspect of the player object to help with determine our leaders for the game