**Linked List Class** (Generic Type)

Function print linked list

Pass In: linked list, output file

display first 3 nodes (top 3 players)

Pass Out: nothing

Endfunction

Function sort linked list (8 sort functions – 1 for each stat)

Pass In: linked list

sort linked list in descending order recursively

Pass Out: sorted linked list

Endfunction

Function add node

Pass In: node

tail of linked list points to new added node

Pass out: nothing

Endfunction

Function delete node

Pass In: node index

Count up to node in linked list

Previous node points to node after node to be deleted

Set node to be deleted to null

Pass out: nothing

Endfunction

**Node Class**

Accessors and Mutators for all stats

**Player Class**

Function calculate batting average

Pass In: number of hits, number of at-bats

CALCULATE: batting average := number of hits / number of at-bats

Pass Out: batting average

Endfunction

Function calculate on-base percentage

Pass In: number of hits, number of walks, number of hit-by-pitch, number of plate appearances

CALCULATE: on-base percentage := (number of hits + number of walks + number of hit-by-pitch) / number of plate appearances

Pass Out: on-base percentage

Endfunction

**Main Class**

Prompt user for input and output filename

call: sort players alphabetically function

FOR every player in array

display player name and all stats (tab b/w each field)

call: Sort function for specific stat

Display linked list

(Repeat sort->display for each stat)

Function read in input file

Pass In: scanner variable of input filename

create linked list

create array for players

WHILE input file has next line

Read in line to string

Store player in array

ENDWHILE

FOR every player in player array

create generic node of 9 types (one for each stat and player name)

CAALCULATE: each stat of player

SET: each stat to designated type in node

add node to linked list

ENDFOR

Pass Out: nothing

Endfunction

Function sort players alphabetically

Pass In: array of players

DO

FOR each player in array

IF ascii value of first letter of player’s name < ascii value of first letter of second player’s name

swap player positions

ELSE

continue

WHILE a swap is made

END DO-WHILE

Pass Out: nothing

Endfunction