MLB Analysis App

McMaster University Software Engineering

Software Engineering 2XB3

Software Engineering Practice and Experience: Binding Theory to Practice

L01 Group 8

Team Member: Tianzheng Mai (400143042), Linqi Jiang (400144709), Junhong Chen (400213422), Collin Kan (400186808), Richard Liu (400196797)

MLB Analysis Application

Table of Contents

MLB Analysis Application

Revision Page

Date	Version	Description	Author
01/29/2020	<1.0>	Project Proposal of Document	Group 8
02/18/2020	<1.1>	Requirement Specification	Group 8
03/30/2020	<1.2>	Algorithm Added	Group 8
04/05/2020	<1.3>	Application Interface Created	Group 8
04/10/2020	<1.4>	Final Revision	Group 8

MLB Analysis Application

Commitment

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of the Conduct of the Professional Engineers of Ontario.

Submitted By Tianzheng Mai 400143042

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of the Conduct of the Professional Engineers of Ontario.

Submitted By Linqi Jiang 400144709

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of the Conduct of the Professional Engineers of Ontario.

Submitted By Junhong Chen 400213422

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of the Conduct of the Professional Engineers of Ontario.

Submitted By Collin Kan 400186808

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of the Conduct of the Professional Engineers of Ontario.

Submitted By Richard Liu 400196797

MLB Analysis Application

Team Responsibilities and Contributions

Team Member:	Role:	Contribution:	Comments:
Richard Liu	Subject Matter Expert	● Developing the theme and research data of the project ● Managing and leading the presentation ● Helping with coding in sorting and searching ● Designing requirement specification document	● Project sponsor ● Team Leader
Collin Kan	Software Developer	● Designing sorting and searching algorithm ● Helping with requirement specification document	● Made a great contribution on implementing searching and sorting algorithms.
Tianzheng Mai	Technical Lead	● Sponsor project proposal ● Writing the technical report and testing the correctness of code ● Helping with designing application interface	● Responsible for technical problem and design documentation ● Helping with coding in project

MLB Analysis Application

Junhong Chen	Software Developer	● Designing the application interface of the project ● Summarizing every group meeting ● Helping with requirement specification	● Designing the interface of the application
Linqi Jiang	Project Manager	● Designing the application interface of the project ● Helping with project proposal	● Managing the processes of the project ● Designing interface

MLB Analysis Application

1. Introduction:

Project Abstract:

Over the years, baseball has become one of the most popular sports in North America, with the MLB being the forefront of the sport, but the fans of the MLB do not have a portable searching application to get information of their favorite players and teams. The MLB Analysis Application concentrates on helping fans and the league view the performance of one or more specific teams and players more efficiently by sorting and searching the data present.

1.1 Objective

MLB Analyze provides a statistical analysis tool with various sorting, searching and graphing algorithms for the MLB (Major League Baseball), based on the historical data of all players and teams between 1871 and 2015. Not only can this application show individual performance, it can also rank the baseball players on their batting success for each season they played in.

1.2 Definitions, Acronyms, Abbreviations:

When analyzing baseball statistics, there are a variety of different acronyms and abbreviations that are used to represent different statistics. Below there will be a list of relevant abbreviations and what they stand for:

- MLB: Major League Baseball
- G: Games Played
- AB: At Bats
- R: Runs
- H: Hits
- 2B: Doubles
- 3B: Triples
- HR: Home Runs
- RBI: Runs Batted In
- SB: Stolen Bases
- CS: Caught Stealing
- BB: Base on Balls
- IBB: Intentional Base on Balls
- HBP: Hit by Pitch
- SH: Sacrifice Hits
- SF: Sacrifice Flies
- GIDP: Grounded into Double Play

MLB Analysis Application

Using the above statistics, a formula was created to calculate the relative contribution that each player made during a season, called RCon. RCon was calculated using the following formula, and is a measure of the expected number of bases contributed by the player along with the contribution due to runs scored for each plate appearance.

RCon = ((H-2B-3B-HR)+2*2B+3*3B+4*HR+BB+IBB+HBP+SB+0.5*(SH+SF)+R+RBI-CS-2*GIDP)/(AB+BB+IBB+HBP+SH+SF)

When analyzing RCon, a higher score means that the player has a greater contribution per plate appearance. If a player does not have a plate appearance in a season, their RCon is scored as -100.

1.3 Final Product

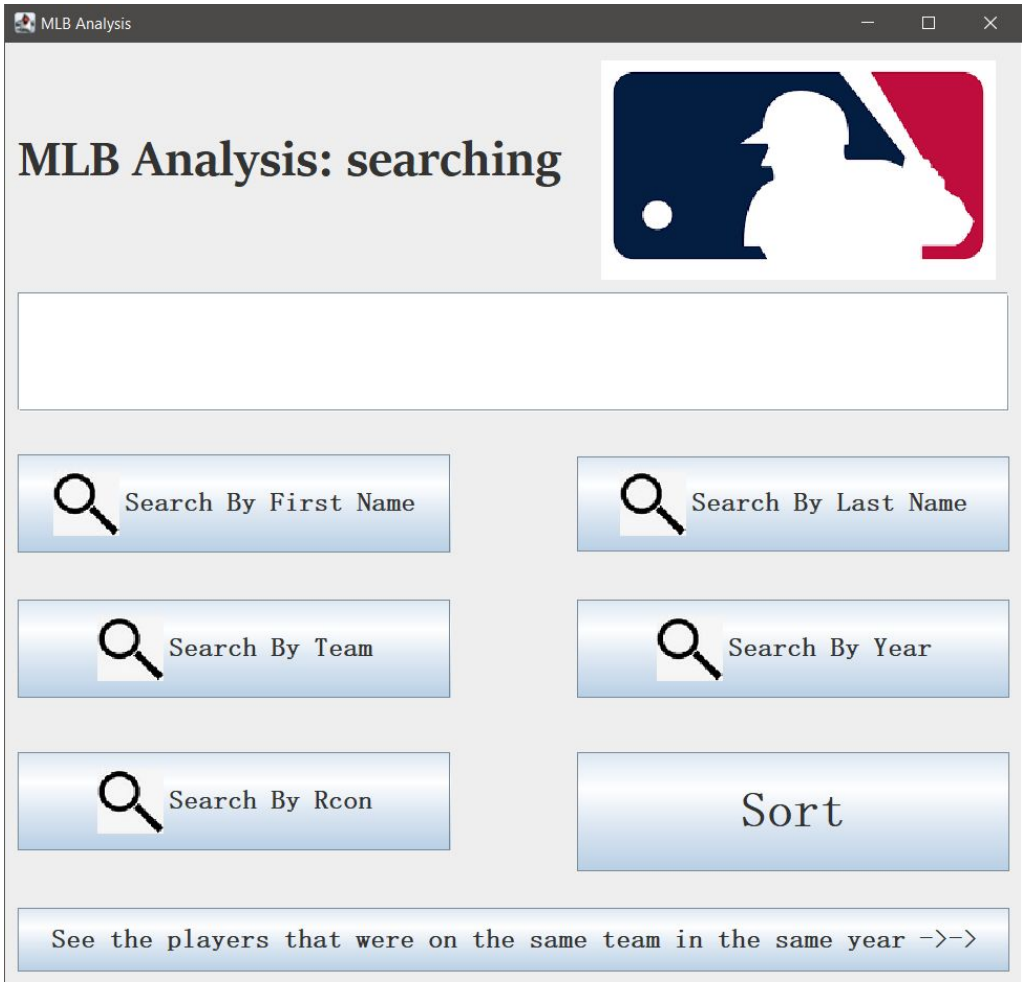


Figure 1: The searching interface of MLB Analyze Application

MLB Analysis Application

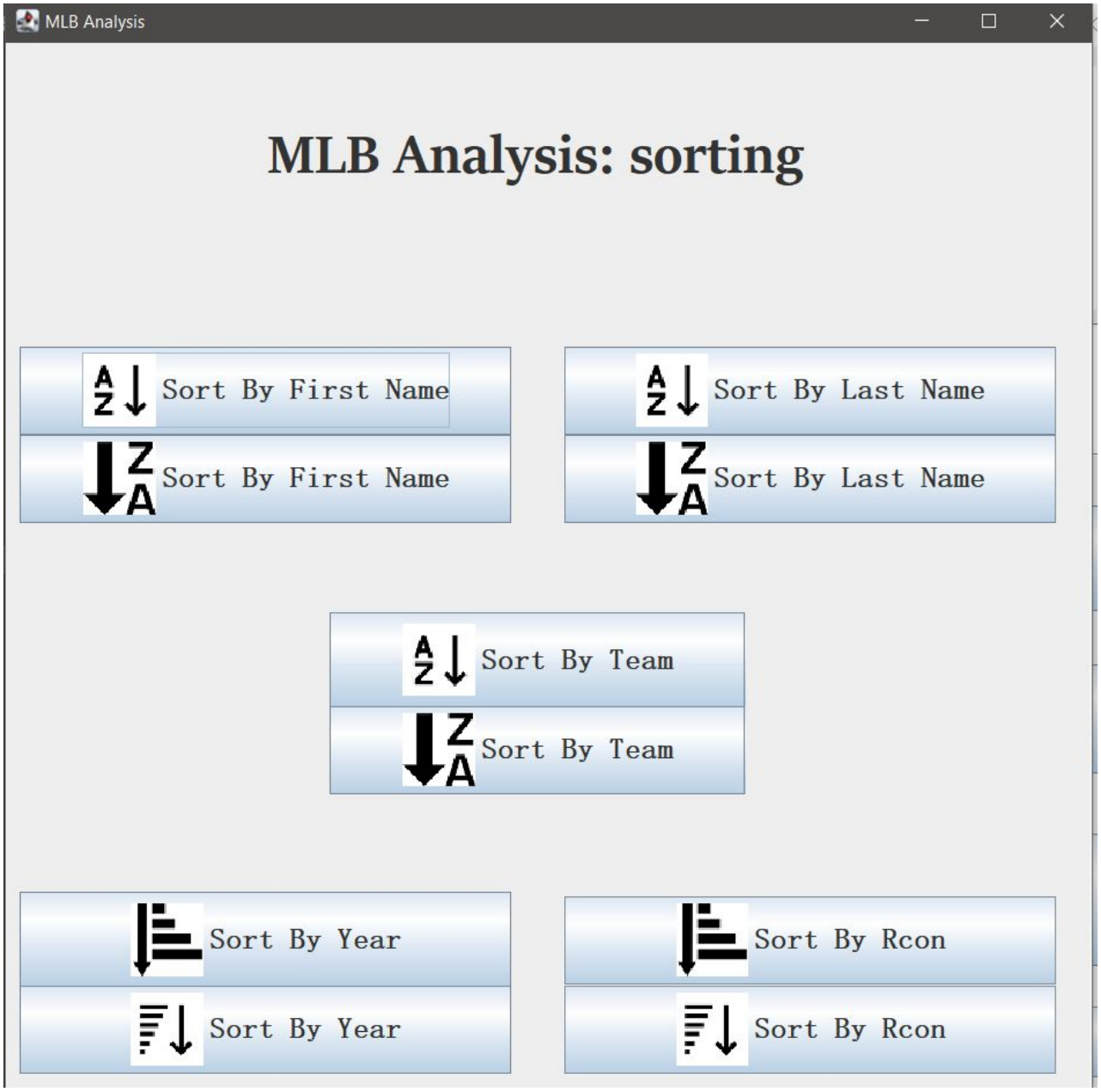


Figure 2: The sorting interface of MLB Analyze Application

MLB Analysis Application

Figure 3: Example of searching by year 2014

MLB Analysis Application

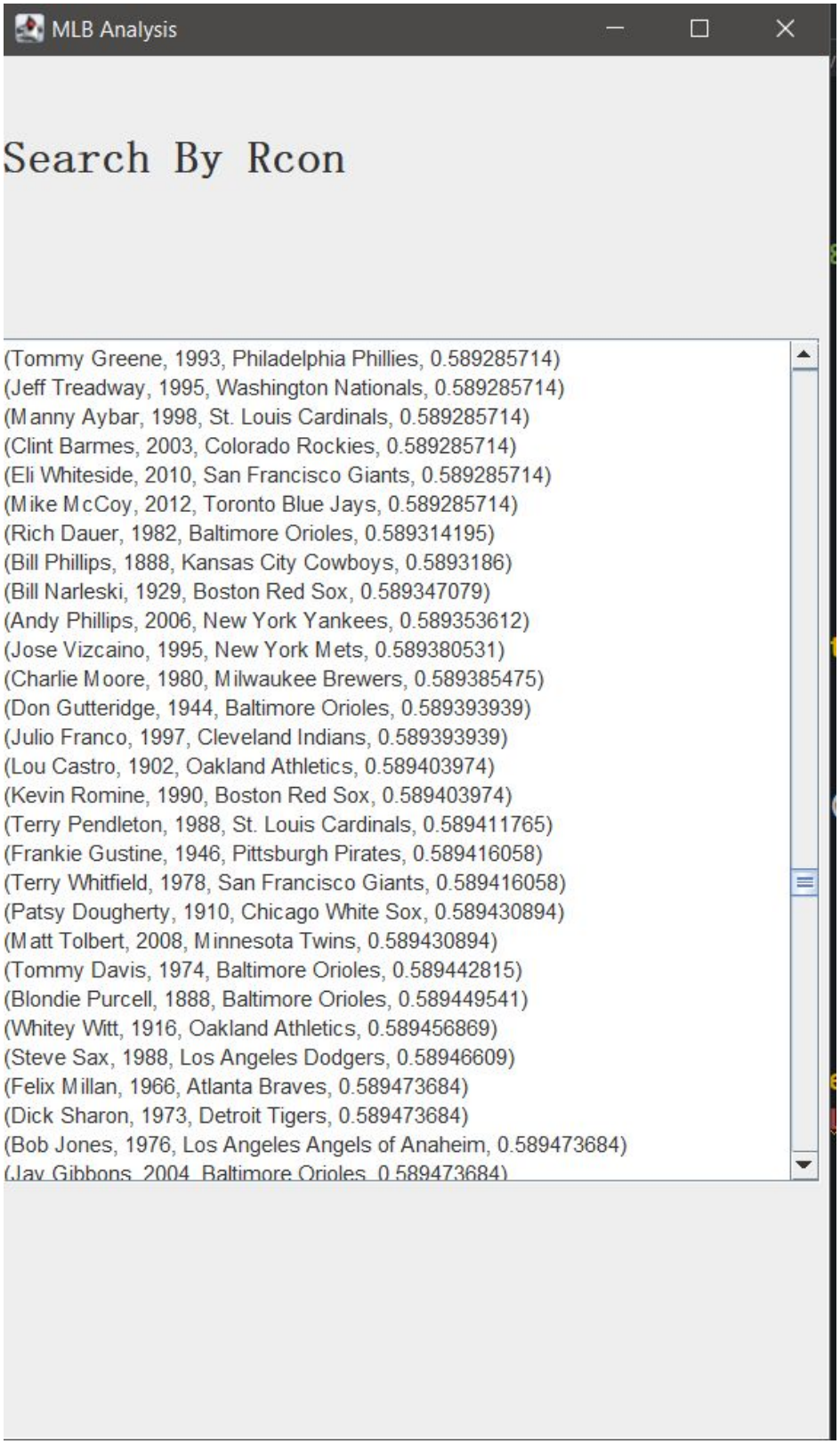


Figure 4: Example of searching by Rcon

MLB Analysis Application

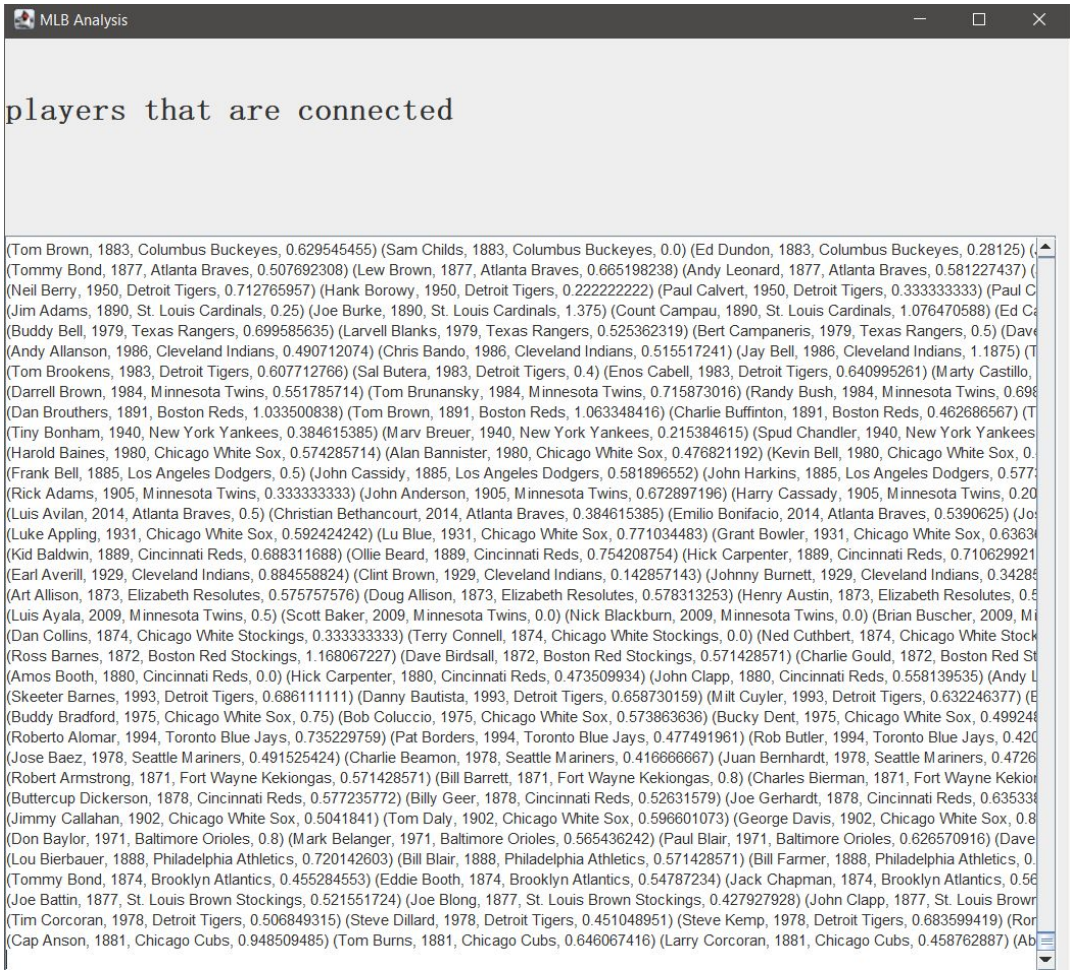


Figure 5: Graph Algorithm of MLB Analysis Application

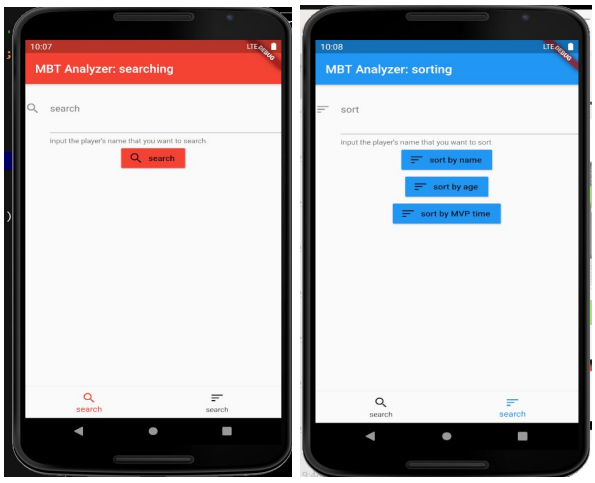


Figure 6:flutter searching and sorting page (Actually this user interface is not used because we failed to import file from eclipse into android studio , and ended up with a simple java GUI written by eclipse).

MLB Analysis Application

2. Implementation

2.1 Implementation Tool:

The searching, sorting, and graphing algorithms and all related modules of the MLB Analysis Application are implemented in Java using the Eclipse IDE.

2.2 Classes/Modules Description and uses relationship:

Class	Roles	Methods	Purposes
Player	ADT(Player)	• Player(String playerid, String teamid, String year, String rcon) • getPlayerId() • getTeamId() • getYear() • getRcon() • getFranId() • getFirst() • getLast() • getTeam() • setFranId(String franid) • setFirst(String first) • setLast(String last) • setTeam(String team) • toString()	An abstract data type containing players information (including a constructor, getters and setters): player's id, team's id, year, rcon
HashMapLP	Hashmap Data Structure	• HashMapLP() • HashMapLP (int m) • put(String key, String val) • get(String key) • resize(int cap) • hash(String key)	Constructing HashMapLP data structure including put, get, resize and hash
CSVReader	Class for Reading data from database	• readData() • readBatting() • readMaster() • readTeams()	Implementing the BufferReader, FileReader to read the data from the

MLB Analysis Application

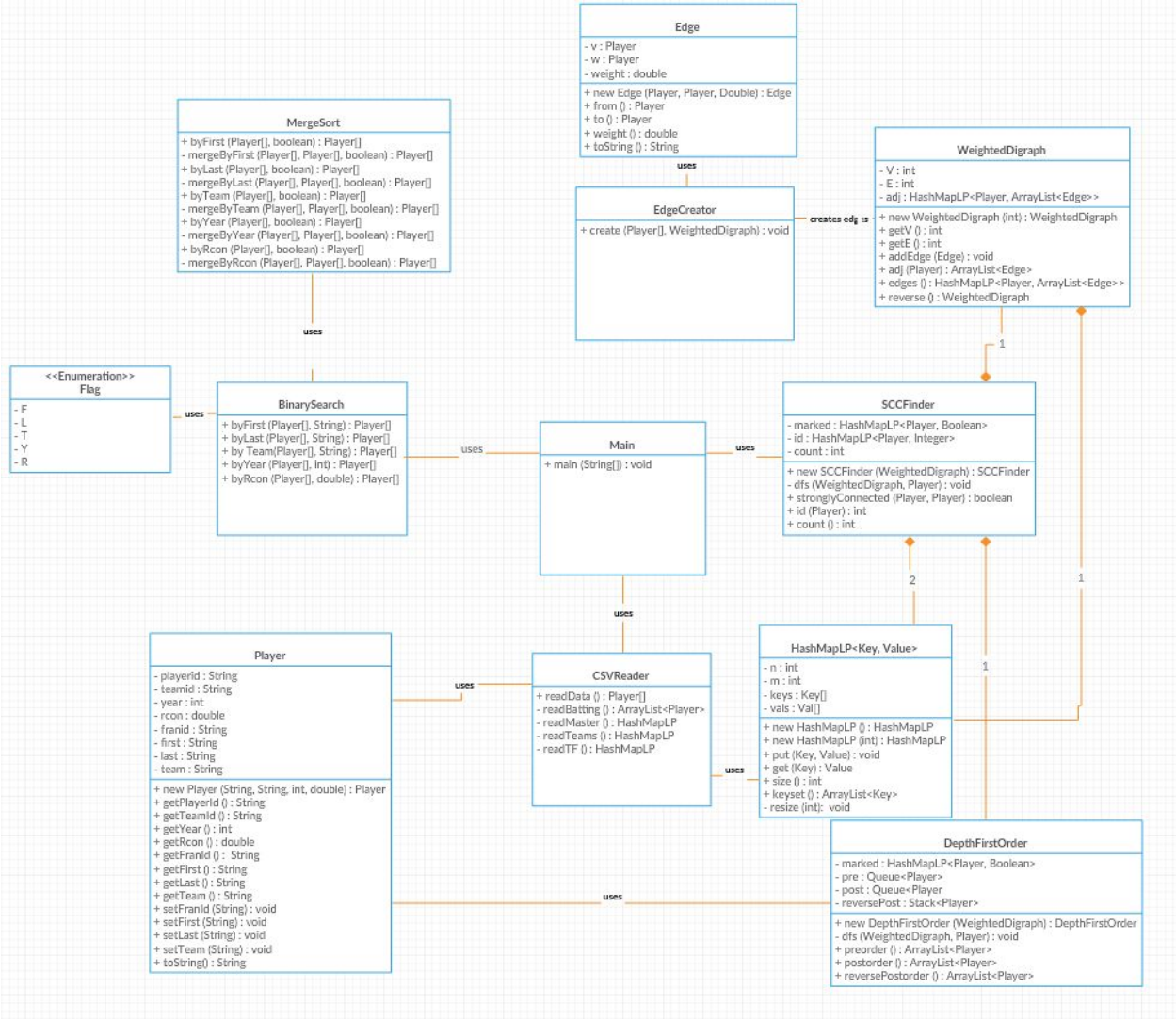
		• readTF()	dataset
Mergesort	Merge Sort Algorithm	• byFirst(Player[] input, boolean asc) • mergeByFirst(Player[] a, Player[] b, boolean asc) • byLast(Player[] input, boolean asc) • mergeByLast(Player[] a, Player[] b, boolean asc) • byTeam(Player[] input, boolean asc) • mergeByTeam(Player[] a, Player[] b, boolean asc) • byYear(Player[] input, boolean asc) • mergeByYear(Player[] a, Player[] b, boolean asc) • byRcon(Player[] input, boolean asc) • mergeByRcon(Player[] a, Player[] b, boolean asc)	Implementing the Merge sort algorithm to sort all the players depending on first name, last name, team, year and Rcon.
BinarySearch	Binary Search Algorithm	• byFirst(Player[] in, String first) • byLast(Player[] in, String last) • byTeam(Player[] in, String team) • byYear(Player[] in, int year) • byRcon(Player[] in, double Rcon)	Implementing the Binary search algorithm to search for a specific player or team by first name, last name, team name, year or Rcon.
DepthFirstOrder	Performs a depth first order search	• DepthFirstOrder (WeightedDigraph G) • dfs (WeightedDigraph G, Player v)	Performs a depth first search on a weighted directed graph

MLB Analysis Application

		• preorder() • postorder() • reversePostorder()	
Edge	Edge class used for graphing algorithm	• Edge(Player v, Player w, double weight) • from() • to() • weight() • toString	Edges between nodes on a weighted directed graph
EdgeCreator	Creates all edges needed	• create(Player[] input, WeightedDigraph G)	Creates the edges between nodes on a weighted directed graph
Flag	Enum used for search type	• Flag()	Enumerator used to declare a search type
SCCFinder	Finds all strongly connected components in the graph	• SCCFinder (WeightedDigraph G) • dfs (WeightedDigraph G, Player v) • stronglyConnected (Player v, Player w) • id(Player v) • count()	Finds all the strongly connected components within the graph
WeightedDigraph	Creates a weighted directed graph used to find SCC's	• WeightedDigraph (int V) • getV() • getE() • addEdge(Edge e) • adj(Player V) • edges() • reverse()	Creates a weighted directed graph
Main	Function for getting expected output	• main(String[] args)	Runs the application when called

MLB Analysis Application

2.3 UML Diagram: Static representation of application classes and relationship between classes



2.4 Project Specification

Description of Interface(Public Entities and Private Entities)

Notes: Using '+' to represent a public method

Using '-' to represent a private method

Player:

Description:

Player class, used to represent a player in a given year.

MLB Analysis Application

State Variables:

- String playerid: The unique identification sequence given to each player
- String teamid: The unique identification sequence given to each team
- int year: The year that the player was playing in the MLB
- double rcon: The RCon score assigned depending on a player's performance in a given year
- String franid: The unique identification sequence given to each franchise
- String first: The first name of the player
- String last: The last name of the player
- String team: The name of the team

Interface:

- +Player(String playerid, String teamid, String year, String rcon):
  Initialize the player ADT

  transition: playerid, teamid, year, rcon := playerid, teamid, year, parsedouble(rcon)
  output: out := self
  exception: this.rcon= -100.00

- +getPlayerId()

  Return the value of the playerid
  output: out := playerid
  exception: None

- +getTeamId()

  Return the value of the teamid
  output: out := teamid
  exception: None

- +getYear()

  Return the value of the rcon
  output: out := rcon
  exception: None

- +getRcon()

  Return the value of the year
  output: out := year
  exception: None

MLB Analysis Application

- +getFranId()

  Return the value of the franid
  output: out := franid
  exception: None

- +getFirst()

  Return the value of player first name
  output: out := first
  exception: None

- +getLast()

  Return the value of player last name
  output: out := last
  exception: None

- +getTeam()

  Return the value of team
  output: out := team
  exception: None

- +setFranId(String franid)

  Set the value of franid
  transition: out := franid
  exception: None

- +setFirst(String first)

  Set the value of first
  transition: out := first
  exception: None

- +setLast(String last)

  Set the value of first
  transition: out := first
  exception: None

- +setTeam(String team)

  Set the value of team

transition: out := team
exception: None

- +toString()

  Return the string of all information of player
  output: out := the string of player with all information
  exception: None

+HashMapLP:

Description:

HashMapLP structure

State Variables:

- int n
- int m
- String[] keys
- String[] vals

Interface:

- +HashMapLP():
  Initialize the size of HashMapLP

  transition:= this(16)
  exception: None

- +HashMapLP(int m):

  Initialize the HashMapLP structure
  transition: m, keys, vals := m, String[m], String[m]
  output: out := self
  exception: None

- +put(String key, String val)

  Put value into keys and vals
  transition:= keys[i], vals[i]=key, val
  exception: None

- +get(String key)

  get the value from the key

output: if(this.keys[i].equals(key) out:= this.vals[i]
exception: None

- -resize(int cap)

  Resize HashMapLP structure
  transition: this.keys, this.vals, this.m := HashMapLP(cap).keys, HashMapLP(cap).vals, HashMapLP(cap).m
  exception: None

- -hash(String key)

  Setting key hashcode
  output: out:= key.hashCode()
  exception: None

+CSVReader:

Description:

CSVRead Class, used to read all players and team information from the database.

State Variables and State Invariant:

None

Interface:

- -readData()

  Read Data from the database
  output: out:= player[ ] players
  exception: IOException

- -readBatting()

  Read Batting from the database
  output: out:= ArrayList<Player> players
  exception: IOException

- -readBattingPost()

  Read Batting Post from the database
  output: out:= HashMapLP rcons
  exception: IOException

MLB Analysis Application

- -readMaster()

  Read Master from the database
  output: out:= HashMapLP names
  exception: IOException

- -readTeams()

  Read Teams from the database
  output: out:= HashMapLP franchise
  exception: IOException

- -readTF()

  Read TF from the database
  output: out:= HashMapLP teams
  exception: IOException

+Mergesort:

Description:

Mergesort Class, sorting and merging by player's first name, player's last name, year and rcon.

State Variables and State Invariant:

None

Interface:

- +byFirst()

  Sorting the players by their first name
  output: out:= mergeByFirst(byFirst(a, asc), byFirst(b, asc), asc)
  exception: None

- -mergeByFirst(Player[] a, Player[] b, boolean asc)

  Merging the players by their first name after sorting by their first name
  output: out:= Player[a.length + b.length]
  exception: None

- +byLast()

  Sorting the players by their last name

MLB Analysis Application

output: out:= mergeByLast(byLast(a, asc), byLast(b, asc), asc)
exception: None

- -mergeByLast(Player[] a, Player[] b, boolean asc)

  Merging the players by their first name after sorting by their first name
  output: out:= Player[a.length + b.length]
  exception: None

- +byTeam()

  Sorting the players by their Team
  output: out:= mergeByTeam(byTeam(a, asc), byTeam(b, asc), asc)
  exception: None

- -mergeByTeam(Player[] a, Player[] b, boolean asc)

  Merging the players by team after sorting by team
  output: out:= Player[a.length + b.length]
  exception: None

- +byYear()

  Sorting the players by year
  output: out:= mergeByYear(byYear(a, asc), byYear(b, asc), asc)
  exception: None

- -mergeByYear(Player[] a, Player[] b, boolean asc)

  Merging the players by year after sorting by year
  output: out:= Player[a.length + b.length]
  exception: None

- +byRcon()

  Sorting the players by Rcon
  output: out:= mergeByRcon(byRcon(a, asc), byRcon(b, asc), asc)
  exception: None

- -mergeByRcon(Player[] a, Player[] b, boolean asc)

  Merging the players by their Rcon after sorting by their Rcon
  output: out:= Player[a.length + b.length]
  exception: None

MLB Analysis Application

+BinarySearch:

Description:

Binary Search Class, used to search player by its first name, last name, Team, Year, or Rcon

State Variables and Invariant:

None

Public Interface:

- +byFirst(Player[] in, String first)

  Search the players by their first name
  Output: out:= in.contains(first)
  exception: None

- +byLast(Player[] in, String last)

  Search the players by their last name
  Output: out := in.contains(last)
  exception: None

- +byTeam(Player[] in, String team)

  Search the players by their team
  Output: out := in.contains(team)
  exception: None

- +byYear(Player[] in, int year)

  Search the player by its first name
  Output: out := in.contains(year)
  exception: None

- +byRcon(Player[] in, double Rcon)

  Search the players by Rcon
  Output: out := in.contains(Rcon)
  exception: None

MLB Analysis Application

+DepthFirstOrder:

Description:

Performs a depth first order search on a weighted digraph from one node to another

State Variables and Invariant:

- HashMapLP<Player, Boolean> marked
- Queue<Player> pre
- Queue<Player> post;
- Stack<Player> reversePost

Public Interface:

- +DepthFirstOrder(WeightedDigraph G)

  Initializes the DepthFirstOrder ADT
  exception: None

- -dfs(WeightedDigraph G, Player v)

  Performs a DFS on the graph starting at Player v
  exception: None
- +preorder()

  Returns all the nodes in preorder
  exception: None

- +postorder()

  Returns all the nodes in postorder
  exception: None

- +reversePostorder()

  Returns all the nodes in reverse postorder
  exception: None

+Edge:

Description:

An edge object to link two nodes together in a weighted digraph

State Variables and Invariant:

MLB Analysis Application

- Player v
- Player w
- double weight

Public Interface:

- +Edge(Player v, Player w, double weight)

  Initializes the Edge ADT
  transition: v, w, weight := v, w, weight
  exception: None

- +from()

  Returns the value of the beginning of the edge
  Output: out := v
  exception: None

- +to()

  Returns the value of the end of the edge
  Output: out := w
  exception: None

- +weight()

  Returns the weight of the edge
  Output: out := weight
  exception: None

- +toString()

  Converts the edge to a string
  Output: out := a string containing the beginning, end, and weight of the edge
  exception: None

+EdgeCreator:

Description:

Creates all the edges of the weighted digraph

State Variables and Invariant:

None

MLB Analysis Application

Public Interface:

- +create(Player[] input, WeightedDigraph G)

  Creates all the edges in the graph
  exception: None

+Flag:

Description:

Creates all the enum objects used

State Variables and Invariant:

F, L, T, Y, R

+SCCFinder:

Description:

Finds all strongly connected components in a weighted directed graph

State Variables and Invariant:

- HashMapLP<Player, Boolean> marked;
- HashMapLP<Player, Integer>id;
- int count;

Public Interface:

- +SCCFinder(WeightedDigraph G)

  Finds all strongly connected components in the graph
  exception: None

- -dfs(WeightedDigraph G, Player v)

  Performs a DFS on the graph starting at Player v
  exception: None

- +stronglyConnected(Player v, Player w)

  Checks if two players are strongly connected
  Output: out := id(v) == id(w)
  exception: None

MLB Analysis Application

- +id(Player v)

  Returns the component id of the component with Player v
  exception: None

- +count()

  Returns the number of strongly connected components
  Output: out := count
  exception: None

+WeightedDigraph:

Description:

Creates a weighted directed graph ADT

State Variables and Invariant:

- int V
- int E
- HashMapLP<Player, ArrayList<Edge>> adj

Public Interface:

- +WeightedDigraph(int V)

  Constructs a weighted digraph
  transition: V, E := V, 0
  exception: None

- +getV

  Returns the number of vertices in the graph
  Output: out := V
  exception: None

- +getE

  Returns the number of edges in the graph
  Output: out := E
  exception: None

- +addEdge(Edge e)

MLB Analysis Application

Adds an edge to the graph
exception: None

- +adj(Player V)

  Returns all nodes adjacent to V
  exception: None

- +edges()

  Returns all edges in the graph
  exception: None

- +reverse()

  Returns a weighted directed graph will all edge directions reversed
  exception: None

+Main:

Description:

Runs the program by calling all necessary methods to produce the working application

State Variables:

None

Public Interface:

- +main(String[] args)

  Runs the application
  exception: IOException

3. The view of the use relationship

Trace back to requirements in each class interface

3.1 User Interface

Our MLB Analyzer App provides 4 different searching methods in the homepage. The users can decide to search by Rcon, first name , last name or year.And each method can be shown in ascending or non ascending view. There is a sort button

in the homepage which will jump to sorting page if the user clicks on it, it also contains four ways to sort:by rcon, by year , by first name and by last name.

3.2 External Interface

The GUI may not fit the mobile application. In the first we tried to use flutter to design the UI but we failed to import the file from eclipse to android studio. As a result we created a simple java GUI as the User Interface, it still has functions which are required but it can not work on mobile phones.

3.3 Software Interface

As specified by the client, the model of the product is written in Java, which holds the logic of the function and the states of MLB analyze application. However application to IOS and android system is failed because of the time constraints and lack of learning material of flutter.

3.4 **Hardware Interface**

The system must have at least 300 megabytes of memory and users can interact with the product by clicking the button in the GUI screen. The running time for sorting and searching methods are around 1 second.

4. Internal Review/evaluation of the MLB Analysis Application

4.1 Meeting Functional Requirements

- Functionality: Efficiently applied mergesort, binary search and digraph algorithm and return the expected result.
- System: compatible in Windows.
- Timing Complexity log(N)

4.2 Meeting Non-Functional Requirements

1. Software Quality:

- Availability: MLB Analyze provides baseball statistics for the users.
- Maintainability: The software is maintained to ensure that it functions properly when it is available to be used.
- Portability: The users can access MLB Analyze in Windows

- Accuracy of Results: The application will be designed to output the expected messages based on the user's requirement. The application will output a table about all the details and the results of different games and players. By comparing the information of different objects, the users will know how players rank with respect to each other based on RCon.
- Performance: The design processes a considerable dataset and uses a variety of efficient algorithms such as quicksort, binary search in Java to produce the output by minimizing run time and memory usage.
- Integrity: The MLB analysis will ensure the privacy of the users and not expose confidential information without explicit consent.

2. Performance Requirement

   The goal for the response time is less than 0.5 seconds

3. Safety and Security Requirement

   MLB Analyze does not have any security requirements as its job is to analyze the statistics of MLB players and provide searching capabilities for the users based on input. Any users can use it without any additional permission and privileges.

4.3 Changes during the development process

There were some key changes during the development. The software interfaces were changed to be functional on the Windows operating system because the application is more compatible with this software.

4.4 Future Changes

In the future, the MLB Analyze Application will be updated with more players' and teams' information, so that the users can view and keep track of players in real time.

5. References

GitLab Link:

https://gitlab.cas.mcmaster.ca/se_2xb3_lab_l01_group8/mlb-analyze

IEEE SRS 2009:

MLB Analysis Application

IEEE Recommended Practice for Software Requirements Specifications, Dec-2009. [Online]. Available: http://www.cse.msu.edu/~cse870/IEEEXplore-SRS-template.pdf. [Accessed: 05-Mar-2020].

Sample SRS:

K. Varvoutas, "Gephi SRS," *Software Requirements Specification for Gephi*, Feb-2017. [Online]. Available: https://gephi.org/users/gephi_srs_document.pdf. [Accessed: 05-Mar-2020].