Technical Merit: 100
Documentation: 95
Quality of code: 100
Program runs: 90

48.25

Brendan Doucette

Project 2 Final Writeup

The motivation to do a database like this is because I am a big basketball fan. I thought it would be fun to create a program where you can enter certain stats of a player or team and compare the stats of those teams and players with others. This program can work with anything basketball related (NBA, college, overseas, personal stats, etc.). Users can edit certain players or team whenever they want because every variable has a getter and setter with it.

This program involves 4 classes. A stats, players, team, and roster class. The player class extends the stats class, and the roster class extends the team class. The team class includes the teams name (City and name), total wins and losses, total players, and total points scored and allowed. There are getters and setters for each of these variables. The class also finds the win/loss percentage by taking the wins and dividing it by the wins plus the losses (wins / (wins + losses)). The class also finds the teams points scored per game by taking the total points and dividing it by the wins plus the losses (total games) (total points / (wins + losses)). The class does the same thing for points allowed by the same method. The team class also has a print team method that prints the teams wins, losses, total players, win/loss percentage, total points per game, and points allowed per game. Lastly, there is a compare teams method that uses the print team method to print the two teams being compared side-by-side so that the user can assess the given numbers and compare different aspects of the team.

The stats class includes double variables for total points, rebounds, assists, steals, blocks, games played, and turnovers. There are getter and setters for each of these variables. There are also methods to find the per game stats. This is found by taking the stat in question and dividing it by the games played. This can be used to find points, rebounds, assists, steals, blocks, and turnovers per game. This class should be used to hold an individual player's stats.

The Player class extends the stats class. This class has String variables for players name and position. This class has getters and setters for both of those variables. There are two printing methods in this class. One method only prints out the players name and position. The other methods prints the players name, position, and per game stats for each category from stats. There is also a compare players method that uses the print player stats method. The compare players method prints the stats of two players side by side for the user to compare their stats. There is also a method that puts the players name, position, per game stats, and games played into a string array to make it easier to add a player into a roster.

The Roster class extends the team class. This class has a integer variable for roster size and a String matrix variable for the actual roster. Both constructors set up the matrix by putting labels in the first row. These labels are Name, Position, PPG (points per game), RPG (rebounds per game), APG (assists per game), SPG (steals per game), BPG (blocks per game), TOPG (turnovers per game), and games. The argument constructor uses an integer to create the roster size based on that user given integer. There is an add player method that accepts a player and spot corresponding to the row the player will be added in. This method uses the ArrayStats method of a player that puts the stats of a player into a string array so that each variable of a player can be taken out of that array and put into the correct corresponding spot on the array. There is also a print roster method that prints out the entire

matrix of the roster. The compare rosters method uses the print roster method to print both the rosters in question side by side for comparing.

The program can be used to compare different players, compare rosters, compare teams, or even just keeping a database of a single players stats or teams' stats. The system can be used by having the user enter in their own numbers, having the system ask the user to enter certain stats for a team or player, or having the system read a file and entering the data from a file into a team or player. In my code I had the system first ask the user for the team name and roster size. Then, depending on how big the roster was, I would ask the user to enter in the players name, position, and stat totals. Then, the system would add that player into the roster. The system would loop through this a certain amount of times depending on how big the roster was the user entered. After this the user would be asked to enter in the teams wins, losses, total points, and total points allowed. After the user entered these numbers in for two teams, I had the program compare the two teams that the user enters and compare the two rosters. I also entered in my own players into the system and had the system compare the two players.

Below is an example that I entered in two show two teams. Below the two teams are two players that I entered and had compared. I only did two players for the two teams even though that isn't a full roster for the sake of space, but you can enter a roster of any size.

```
Enter the Team Name(City and Name):
Brooklyn Nets
How big is the roster?
2
Enter the players Last name:
Dinwiddie
Enter the players position:
PG
Enter total points:
1318
Enter total rebounds:
221
Enter total assists:
432
Enter total steals:
40
Enter total blocks:
21
Enter total games:
64
Enter total turnovers:
174

Enter the players Last name:
Irving
Enter the players position:
PG
Enter total points:
548
Enter total rebounds:
103
```

```
Enter the Team Name (City and Name):
NewYork Knicks
How big is the roster?
2
Enter the players name:
Randle
Enter the players position:
PF
Enter total points:
1248
Enter total rebounds:
622
Enter total assists:
198
Enter total steals:
51
Enter total blocks:
22
Enter total games:
64
Enter total turnovers:
193

Enter the players name:
Robinson
Enter the players position:
C
Enter total points:
590
Enter total rebounds:
428
```

```
Enter total assists:
128
Enter total steals:
27
Enter total blocks:
10
Enter total games:
20
Enter total turnovers:
52


How many wins do the Brooklyn Nets have?
30
How many loses does the Brooklyn Nets
have?
34
How many total points do the Brooklyn
Nets have?
7089
How many total points have the Brooklyn
Nets allowed?
7130

Name: Brooklyn Nets
 Wins: 30.0
 Losses: 34.0
 Win/Loss percentage: 0.469
 Roster: 15.0 players
 Total Points PG: 110.766 points per
game
 Points Allowed PG: 111.406 points
allowed per game


Brooklyn Nets
Name |  Position |  PPG |  RPG |  APG |
SPG |  BPG |  TOPG |  Games |
Dinwiddie |  PG |  20.59 |  3.45 |  6.75
|  0.62 |  0.33 |  2.72 |  64 |
Irving |  PG |  27.4 |  5.15 |  6.4 |
1.35 |  0.5 |  2.6 |  20 |
```

```
Name: Lebron James Position: SF PPG:
25.73 RPG: 7.88 APG: 10.6 SPG: 1.23 BPG:
0.5
```

```
Enter total assists:
36
Enter total steals:
52
Enter total blocks:
119
Enter total games:
61
Enter total turnovers:
37


How many wins do the NewYork Knicks
have?
21
How many loses does the NewYork Knicks
have?
45
How many total points do the NewYork
Knicks have?
6983
How many total points have the NewYork
Knicks allowed?
7409

Name: NewYork Knicks
 Wins: 21.0
 Losses: 45.0
 Win/Loss percentage: 0.318
 Roster: 15.0 players
 Total Points PG: 105.803 points per
game
 Points Allowed PG: 112.258 points
allowed per game


NewYork Knicks
Name |  Position |  PPG |  RPG |  APG |
SPG |  BPG |  TOPG |  Games |
Randle |  PF |  19.5 |  9.72 |  3.09 |
0.8 |  0.34 |  3.02 |  64 |
Robinson |  C |  9.67 |  7.02 |  0.59 |
0.85 |  1.95 |  0.61 |  61 |
```

```
Name: Luka Doncic Position: SG PPG:
28.69 RPG: 9.3 APG: 8.7 SPG: 1.09 BPG:
0.19
```

Other types of literature that are similar are databases on the internet like nba.com, basketball reference, espn, etc. I used basketball reference to add in the stats for these players. The goals this program accomplishes is that user can enter the basic basketball statistics of a team or player, and those stats can be used to compare different teams and players, or just keep as a database and add into as a season goes along. This program can be improved by adding in more advanced statistics like a players field goal percentage and a teams field goal percentage as a whole.

UML diagrams:

Team
-names: String -wins: double -losses: double -players: double -totalpoints: double -pointsallowed: double
+Team() +Team(newName: String, newWins: double, newLosses: double, totalPlayers: double, newTotalPoints: double, newPointsAllowed: double) +getName(): String +setName(n: String) +getWins(): double +setWins(w: double) +getLosses(): double +setLosses(l: double) +getWinpct(): double +getTotalPlayers(): double +setTotalPlayers(tp: double) +getTtlPts(): double +setTotalPts(tpt: double) +getPtsAllowed(): double +setPtsAllowed(pa: double) +getPtsPG(): double +getPtsAPG(): double +printTeam() +compareTeams(team2: Team)

Roster extends Team
-rosterSize: Int -roster: String[][]
+Roster() +Roster(newRosterSize: Int) +AddPlayer(newPlayer: Player, spot: Int) +printRoster() +compareRosters(team2: Roster)

Stats
-points: double -rebounds: double -assists: double -steals: double -blocks: double -games: double -turnovers: double
+Stats() +Stats(totalPts: double, totalRebs: double, totalAst: double, totalStl: double, totalBlks: double, gamesPlayed: double, totalTurns: double) +getPoints(): double +setPoints(p: double) +getRebounds(): double +setRebounds(r: double) +getAssists(): double +setAssists(a: double) +getSteals(): double +setSteals(s: double) +getBlocks(): double +setBlocks(b: double) +getGames(): double +setGames(g: double) +getTurnovers(): double +setTurns(t: double) +getPPG(): double +getAPG(): double +getRPG(): double +getBPG(): double +getSPG(): double +getTOPG(): double

Player extends Stats
-name: String -position: String
+Player() +Player(newName: String, newPosition: String) +getName(): String +setName(n: String) +getPosition(): String +setPosition(p: String) +printPlayer() +printPlayerStats() +ArrayStats(y: int): String +comparePlayer(p: Player)