Title: Basketball Stats Tracker

Name: Aswinth Sinnathamby

Course: Data Structures and Object-Oriented Programming

Table of Contents:

1.Project description:

The Basketball Stats Tracker was developed to operate as a comprehensive system for recording and analyzing basketball players and game statistics. It helps coaches, players, and analysts track performance trends, make strategic decisions, and monitor improvements over time. For example, a coach can input stats for players after a game, such as points scored and rebounds, and use the application to identify high-performing players and areas for improvement. Game-specific stats, such as total fouls and turnovers, can also be monitored to gauge team performance.

2.Program features with Screenshots:

a) Adding Player Stats: Users can input player details (name, points, assists, rebounds, blocks, etc.) based on their position (Guard, Forward, Center) using specific methods like addStats(points, assists, rebounds, blocks).

Center position:

```
public class Main {  wembythegoat *
    public static void main(String[] arg) {  wembythegoat *

        CenterStats player = new CenterStats( name: "Jokic",  points: 30,  assists: 10,  rebounds: 10,  turnovers: 3,  blocks: 2);

        player.addStats( points: 10,  assists: 3,  rebounds: 3,  turnovers: 2,  blocks: 1);

        player.displayStats();
```

```
Main
C:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=56547:C:\Pr
Player: Jokic
Points: 40
Assists: 13
Rebounds: 13
Blocks: 3
Block Rate: 0.3

Process finished with exit code 0
```

```
/**
 * Adds the provided stats to the player's existing totals.
 * @param points The number of points scored by the player.
 * @param assists The number of assists made.
 * @param rebounds The number of rebounds secured.
 * @param turnovers The number of turnovers committed.
 * @param blocks The number of blocks recorded.
 */
public void addStats(int points, int assists, int rebounds, int turnovers, int blocks) {  1 usage  wembythegoat
    if (points < 0 || assists < 0 || rebounds < 0 || turnovers < 0 || blocks < 0) {
        throw new IllegalArgumentException("Stats cannot be negative.");
    }
    super.addStats(points, assists, rebounds);
    this.blocks += blocks;
}
```

Calculates the player's block rate per game:

```java
/**
 * Calculates the player's block rate per game.
 * This metric represents the average number of blocks made by the player over a given number of games.
 * @param totalGames The total number of games played. Must be zero or a positive integer.
 * @return The block rate
 */
public double calculateBlockRate(int totalGames) {  5 usages  wembythegoat
    if (totalGames == 0) {
        return 0;
    }
    return (double) blocks / totalGames;
}
```

```java
public class Main {  wembythegoat *
    public static void main(String[] arg) {  wembythegoat *
        CenterStats player = new CenterStats( name: "Shaquille O'Neal",  points: 20,  assists: 8,  rebounds: 12,  turnovers: 3,  blocks: 5);

        double blockRate = player.calculateBlockRate( totalGames: 10);
        System.out.println("Block Rate: " + blockRate);
```

```
Main
C:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=58
Block Rate: 0.5

Process finished with exit code 0
```

Guard position:

```java
public class Main {  wembythegoat *
    public static void main(String[] arg) {  wembythegoat *

        GuardStats player = new GuardStats( name: "Steph curry",  points: 30,  assists: 10,  rebounds: 5,  turnovers: 3,  steals: 2);

        player.addStats( points: 20,  assists: 3,  steals: 3);

        player.displayStats();
```

```
Main ×
C:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=
Player: Steph curry
Points: 50
Assists: 13
Steal: 5
Assist-to-turnover Ratio: 2.1666666666666665

Process finished with exit code 0
```

```java
    /**
     * Updates the player's statistics by adding new game stats.
     * @param points The number of points scored in the game.
     * @param assists The number of assists recorded.
     * @param steals The number of steals made.
     */
    public void addStats(int points, int assists, int steals) {  5 usages  wembythegoat
        if (points < 0 || assists < 0 || rebounds < 0 || turnovers < 0) {
            throw new IllegalArgumentException("Stats cannot be negative.");
        }
        super.addStats(points, assists,  rebounds: 0);
        this.steals += steals;
    }
```

Calculates the assist-to-turnover ratio for a player:

```java
/**
 * Calculates the assist-to-turnover ratio for a player.
 * This metric represents how efficiently a player distributes the ball
 * while minimizing turnovers.
 * @param turnovers The number of turnovers made by the player.
 * @return The assist-to-turnover ratio.
 */
public double calculateAssistToTurnoverRatio(int turnovers) {  7 usages  wembythegoat
    if (turnovers == 0) {
        return 0;
    }

    if (turnovers < 0) {
        throw new IllegalArgumentException("Turnovers cannot be negative.");
    }

    return (double) assists / turnovers;
}
```


```java
/**
 * Calculates the assist-to-turnover ratio for a player.
 * This metric represents how efficiently a player distributes the ball
 * while minimizing turnovers.
 * @param turnovers The number of turnovers made by the player.
 * @return The assist-to-turnover ratio.
 */
public double calculateAssistToTurnoverRatio(int turnovers) {  7 usages  wembythegoat
    if (turnovers == 0) {
        return 0;
    }

    if (turnovers < 0) {
        throw new IllegalArgumentException("Turnovers cannot be negative.");
    }

    return (double) assists / turnovers;
}
```

Calculates efficiency based on field goal percentage, rebounds, and assists:

```java
/**
 * Calculates efficiency based on field goal percentage, rebounds, and assists.
 * @param fieldGoalMade Number of successful shots made.
 * @param fieldGoalAttempted Number of total attempts.
 * @return Efficiency rating.
 */
public double calculateEfficiency(int fieldGoalMade, int fieldGoalAttempted) {  5 usages  wembythegoat
    if (fieldGoalAttempted == 0) {
        return 0;
    }
    double fgPercentage = ((double) fieldGoalMade / fieldGoalAttempted) * 100;
    return (fgPercentage + super.getRebounds() + super.getAssists()) / 3;
}
```

```java
public class Main {  wembythegoat *
    public static void main(String[] arg) {  wembythegoat *
        ForwardStats player = new ForwardStats( name: "LeBron James",  points: 25,  assists: 7,  rebounds: 10,  turnovers: 4,  fieldGoalPercentage: 54);

        // Calculate and display efficiency
        double efficiency = player.calculateEfficiency( fieldGoalMade: 9,  fieldGoalAttempted: 15);
        System.out.println("Player Efficiency: " + efficiency);
```

```
un  Main ×
C:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=58394:C:\Pr
Player Efficiency: 25.666666666666668

Process finished with exit code 0
```

b) Tracking Game Stats: Users can input general game information (date, location, total score) and add individual or team stats for the game.

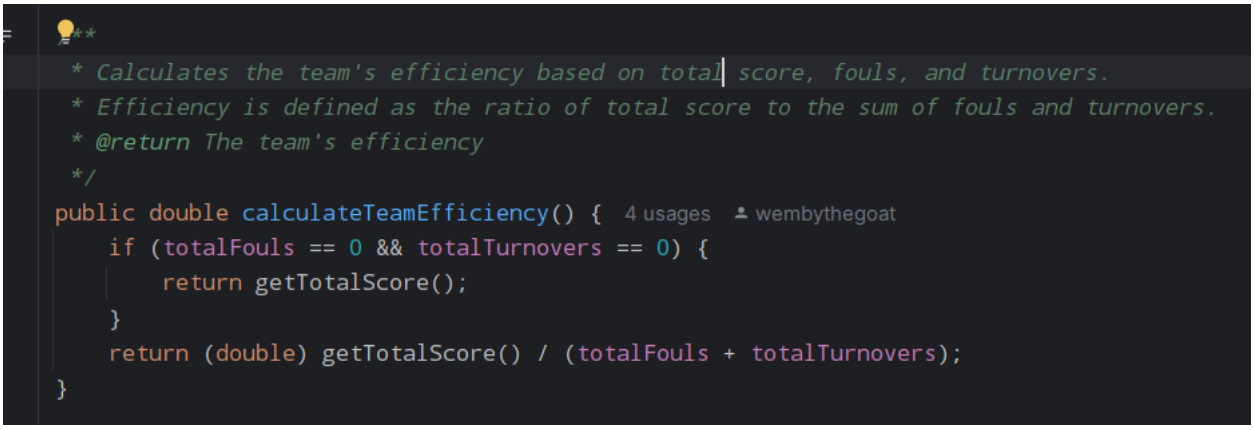
Adds the provided points to the team's total score:

```java
/**
 * Adds the provided points to the team's total score.
 * @param points The number of points to be added to the total score.
 */
public void addScore(int points) {  4 usages  wembythegoat
    if (points < 0) {
        throw new IllegalArgumentException("Points cannot be negative");
    }
    totalScore += points;
}
```

Calculates the team's efficiency based on total score, fouls, and turnovers:

```
public class Main {  wembythegoat *
    public static void main(String[] arg) {  wembythegoat *
        TeamGameStats teamGameStats = new TeamGameStats( gameDate: "2020-05-10", location: "MetLife Stadium", totalScore: 100, totalFouls: 15, totalTurnovers: 21);

        double efficiency = teamGameStats.calculateTeamEfficiency();

        System.out.println("Team Efficiency: " + efficiency);
```

```
Run  Main
C:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=56866:C:\Program Files\JetB
Team Efficiency: 2.7777777777777777

Process finished with exit code 0
```



```
/**
 * Calculates the team's efficiency based on total score, fouls, and turnovers.
 * Efficiency is defined as the ratio of total score to the sum of fouls and turnovers.
 * @return The team's efficiency
 */
public double calculateTeamEfficiency() {  4 usages  wembythegoat
    if (totalFouls == 0 && totalTurnovers == 0) {
        return getTotalScore();
    }
    return (double) getTotalScore() / (totalFouls + totalTurnovers);
}
```

Adds the provided points and turnovers to the team's game statistics. Also increases the total fouls randomly within a range of 0 to 4. Also displays the stats:


```java
/**
 * Adds a player's statistics to the list of player stats for the game.
 * @param player The PlayerStats object representing the player's game stats.
 */
public void addPlayerStats(PlayerStats player) {  4 usages  wembythegoat
    if (player == null) {
        throw new IllegalArgumentException("You can't add a null player");
    }
    playerStats.add(player);
}

@Override  wembythegoat
public void displayStats() {
    System.out.println("Game Player Stats: ");
    for (PlayerStats player : playerStats) {
        player.displayStats();
        System.out.println("--------------------");
    }
}
```

d) Sorting and Filtering: Players are sorted by points using Comparable and can be sorted by rebounds or assists using a Comparator.

Compares this player's assist-to-turnover ratio with another player:

```java
public class Main {  wembythegoat *
    public static void main(String[] arg) {  wembythegoat *
        // Create two player objects (assuming subclasses like GuardStats)
        PlayerStats player1 = new GuardStats( name: "Chris Paul",  points: 15,  assists: 12,  rebounds: 4,  turnovers: 3,  steals: 1);
        PlayerStats player2 = new GuardStats( name: "John Stockton",  points: 10,  assists: 14,  rebounds: 5,  turnovers: 2,  steals: 0);

        // Compare the players based on assist-to-turnover ratio
        int comparisonResult = player1.compareTo(player2);

        // Display comparison result
        if (comparisonResult < 0) {
            System.out.println(player1.getName() + " has a lower assist-to-turnover ratio than " + player2.getName());
        } else if (comparisonResult > 0) {
            System.out.println(player1.getName() + " has a higher assist-to-turnover ratio than " + player2.getName());
        } else {
            System.out.println(player1.getName() + " and " + player2.getName() + " have equal assist-to-turnover ratios.");
        }
```

```
Main
:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=57126:
hris Paul has a lower assist-to-turnover ratio than John Stockton

rocess finished with exit code 0
```

```java
    /**
     * Compares this player's assist-to-turnover ratio with another player's.
     * @param o the object to be compared.
     * @return A negative integer if this player's ratio is lower, zero if the ratios are equal
     */
    @Override  wembythegoat
    public int compareTo(PlayerStats o) {
        return Double.compare(calculateAssistToTurnoverRatio(turnovers), calculateAssistToTurnoverRatio(o.turnovers));

    }
```

Compares player's efficiency:

```java
public class Main {  wembythegoat *
    public static void main(String[] arg) {  wembythegoat *
        ForwardStats player1 = new ForwardStats( name: "Kevin Durant",  points: 25,  assists: 6,  rebounds: 9,  turnovers: 3,  fieldGoalPercentage: 50);
        ForwardStats player2 = new ForwardStats( name: "Giannis Antetokounmpo",  points: 30,  assists: 5,  rebounds: 12,  turnovers: 4,  fieldGoalPercentage: 52);

        int comparisonResult = player1.compareTo(player2);

        if (comparisonResult < 0) {
            System.out.println(player1.getName() + " has lower efficiency than " + player2.getName());
        } else if (comparisonResult > 0) {
            System.out.println(player1.getName() + " has higher efficiency than " + player2.getName());
        } else {
            System.out.println(player1.getName() + " and " + player2.getName() + " have equal efficiency.");
        }
```

```
Run    Main
C:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=57284:C:\Program Files\JetBrains\IntelliJ
Kevin Durant has lower efficiency than Giannis Antetokounmpo
```

```java
/**
 * Compares this player's assist-to-turnover ratio with another player's.
 * @param o the object to be compared.
 * @return A negative integer if this player's ratio is lower, zero if the ratios are equal
 */
@Override  wembythegoat
public int compareTo(PlayerStats o) {
    return Double.compare(calculateAssistToTurnoverRatio(turnovers), calculateAssistToTurnoverRatio(o.turnovers));

}
```

Compares this center player's block count with another center player:

```
public class Main {  1 wembythegoat *
    public static void main(String[] arg) {  wembythegoat *
        CenterStats player1 = new CenterStats( name: "Shaquille O'Neal",  points: 20,  assists: 8,  rebounds: 12,  turnovers: 3,  blocks: 5);
        CenterStats player2 = new CenterStats( name: "Dwight Howard",  points: 18,  assists: 6,  rebounds: 10,  turnovers: 4,  blocks: 7);

        // Compare the players based on their block count
        int comparisonResult = player1.compareTo(player2);

        // Display comparison result
        if (comparisonResult < 0) {
            System.out.println(player1.getName() + " has fewer blocks than " + player2.getName());
        } else if (comparisonResult > 0) {
            System.out.println(player1.getName() + " has more blocks than " + player2.getName());
        } else {
            System.out.println(player1.getName() + " and " + player2.getName() + " have the same number of blocks.");
        }

        //List<PlayerStats> players = new ArrayList<>();
```

```
un   Main ×

C:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=57535:C:\Program Files
Shaquille O'Neal has fewer blocks than Dwight Howard

Process finished with exit code 0
```

```
/**
 * Compares this center player's block count with another center player's.
 * @param o the object to be compared.
 * @return A negative integer if this player has fewer blocks,
 *         zero if both players have the same number of blocks,
 *         a positive integer if this player has more blocks.
 */
@Override  wembythegoat
public int compareTo(PlayerStats o) {
    return blocks - ((CenterStats) o).blocks;
}
```

e) Displaying Stats: Stats for players and games are displayed using polymorphic methods like displayStats(), which provide customized outputs for different positions (e.g., highlighting assists for Guards).

```java
@Override  wembythegoat
public void displayStats() {
    System.out.println("Player: " + getName());
    System.out.println("Points: " + getPoints());
    System.out.println("Assists: " + getAssists());
    System.out.println("Steal: " + steals);
    System.out.println("Assist-to-turnover Ratio: " + calculateAssistToTurnoverRatio(turnovers));
}
```

```java
public class Main {  wembythegoat *
    public static void main(String[] arg) {  wembythegoat *

        GuardStats player = new GuardStats( name: "Steph curry",  points: 30,  assists: 10,  rebounds: 5,  turnovers: 3,  steals: 2);

        player.addStats( points: 20,  assists: 3,  steals: 3);

        player.displayStats();
```

```
un   Main ×

C:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=
Player: Steph curry
Points: 50
Assists: 13
Steal: 5
Assist-to-turnover Ratio: 2.1666666666666665

Process finished with exit code 0
```

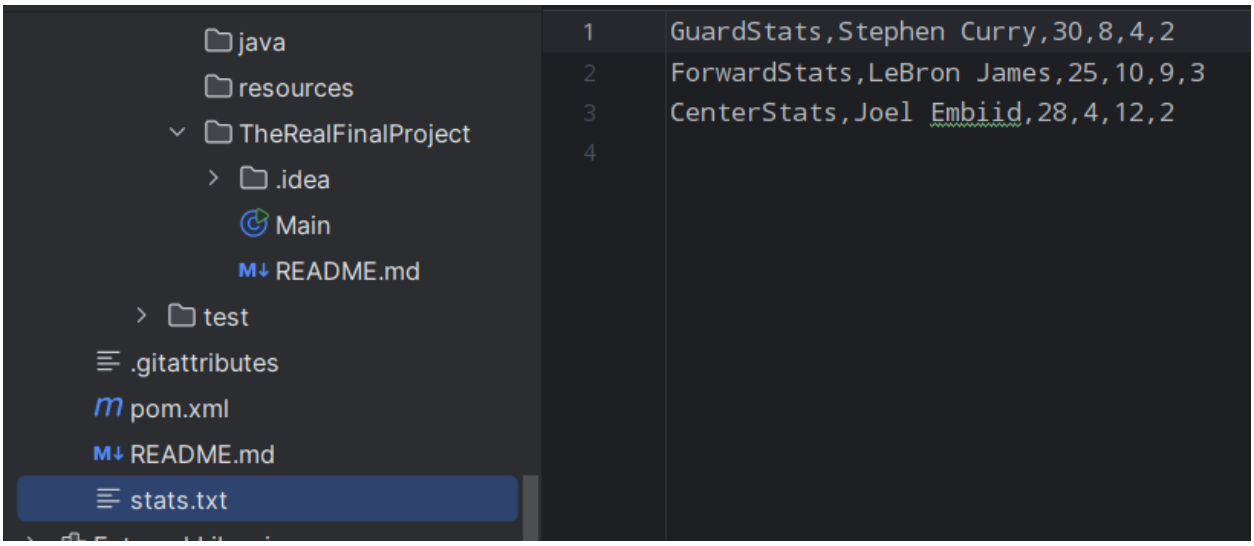
f) Data Persistence: Stats can be saved to and loaded from text files using StatsFileManager.

```
public class StatsFileManager {  no usages  wembythegoat

    private static final String FILE_NAME = "stats.txt";  4 usages

    public static void saveStats(List<PlayerStats> players) {  no usages  wembythegoat
        try (PrintWriter writer = new PrintWriter(new FileWriter(FILE_NAME))) {
            for (PlayerStats player : players) {
                String classType = player.getClass().getSimpleName(); // Get subclass type
                writer.println(classType + "," + player.getName() + "," + player.getPoints() + "," +
                        player.getAssists() + "," + player.getRebounds() + "," + player.getTurnovers());
            }
            System.out.println("Stats saved to " + FILE_NAME);
        } catch (IOException e) {
            System.err.println("Error saving player stats: " + e.getMessage());
        }
    }

    public static List<PlayerStats> loadStats() {  no usages  wembythegoat
        List<PlayerStats> players = new ArrayList<>();
        try (BufferedReader reader = new BufferedReader(new FileReader(FILE_NAME))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] data = line.split( regex: ",");
                if (data.length == 6) {
                    String classType = data[0];
                    String name = data[1];
                    int points = Integer.parseInt(data[2]);
                    int assists = Integer.parseInt(data[3]);
                    int rebounds = Integer.parseInt(data[4]);
                    int turnovers = Integer.parseInt(data[5]);

                    PlayerStats player;
                    switch (classType) {
```

```
                String classType = data[0];
                String name = data[1];
                int points = Integer.parseInt(data[2]);
                int assists = Integer.parseInt(data[3]);
                int rebounds = Integer.parseInt(data[4]);
                int turnovers = Integer.parseInt(data[5]);

                PlayerStats player;
                switch (classType) {
                    case "GuardStats":
                        player = new GuardStats(name, points, assists, rebounds, turnovers, steals: 0);
                        break;
                    case "ForwardStats":
                        player = new ForwardStats(name, points, assists, rebounds, turnovers, fieldGoalPercentage: 0.0);
                        break;
                    case "CenterStats":
                        player = new CenterStats(name, points, assists, rebounds, turnovers, blocks: 0);
                        break;
                    default:
                        throw new IllegalArgumentException("Unknown player type: " + classType);
                }
                players.add(player);
            }
        }
        System.out.println("Stats loaded from " + FILE_NAME);
    } catch (IOException e) {
        System.out.println("Error loading player stats: " + e.getMessage());
    }
    return players;
}
}
```

```
public class Main {  wembythegoat
    public static void main(String[] arg) {  wembythegoat
        List<PlayerStats> players = new ArrayList<>();
        players.add(new GuardStats( name: "Stephen Curry",  points: 30,  assists: 8,  rebounds: 4,  turnovers: 2,  steals: 2));
        players.add(new ForwardStats( name: "LeBron James",  points: 25,  assists: 10,  rebounds: 9,  turnovers: 3,  fieldGoalPercentage: 50.0));
        players.add(new CenterStats( name: "Joel Embiid",  points: 28,  assists: 4,  rebounds: 12,  turnovers: 2,  blocks: 5));

        StatsFileManager.saveStats(players);

        List<PlayerStats> loadedPlayers = StatsFileManager.loadStats();
        for (PlayerStats player : loadedPlayers) {
            System.out.println(player);
        }
```

```
Run   Main ×

C:\Users\sinna\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=5893
Stats saved to stats.txt
Stats loaded from stats.txt
PlayerStats{name='Stephen Curry', points=30, assists=8, rebounds=4, turnovers=2}
PlayerStats{name='LeBron James', points=25, assists=10, rebounds=9, turnovers=3}
PlayerStats{name='Joel Embiid', points=28, assists=4, rebounds=12, turnovers=2}

Process finished with exit code 0
```



3. Challenges Faced:

During development, certain challenges were encountered:

- **Testing Console Output (**displayStats()**):** Using ByteArrayOutputStream was needed to verify printed stats. I remember using this for another past project, but I still needed to relearn a bit about this, and it was one of the hardest things for me in the project

- **Ensuring Correct Sorting Behavior**: Used **Comparable and Comparator** for ranking players efficiently. For some of the classes like GuardStats and ForwardStats, I had to really think about what stat i want to compare with each other.

- **Managing File Persistence**: Ensuring **data consistency** when saving/loading player stats. TextIO out of all the material learned this semester, is easily the hardest and complicated one for me. This was the only part of the project where I actively asked for help from my friends. I have always struggled with linereader because because at the beginning it really confused me. I also had to relook at how to make a switch case.

4. Learning Outcomes:

Throughout this project, the following skills were improved:

- **Java Object-Oriented Programming** (inheritance, polymorphism, method overriding)

- **Unit Testing** (writing JUnit tests for validation): I'm now way more comfortable and confident in my ability to create tests.

- **Sorting Algorithms** (implementing Comparable and Comparator for ranking players)

- **Exception Handling** (ensuring user input validation): Before I used to really forget about putting exceptions in methods because I would be way more focused on creating the method itself. But now, making the exception has become this automatic thing that I do first thing first every time I create a method.

- **Text File Management** (saving and loading player data efficiently): I'm now way more comfortable with TextIO. I wouldn't say it's the best aspect of my coding ability, but I would say that my ability in it has increased. I also comprehend its concept and the reason why we do this more.

5. Conclusion

The Basketball Stats Tracker successfully provides a structured system for tracking players and team performance. It leverages object-oriented principles for managing hierarchical player stats and integrates sorting and filtering mechanisms for analysis.

Overall, the project delivers an effective performance monitoring tool for coaches, players, and analysts.