# CS21120 Assignment
# Single-Elimination Style Competition

Ethan Swain
220029553 – ets6

Undergraduate Student
Computer Science
Aberystwyth University

October/November 2023

# Contents

# 1 Introduction

This report describes the comprehensive implementation of the Single-Elimination Style Competition program, as outlined in the assignment brief[1]. This includes the design, development, and testing of five tasks essential to the functioning of the program.

The program is executed via a command-line menu, prompting the user to input team information, player details, and the scores for both the group and knockout stages.

Accompanying this report are several screenshots, providing evidence of testing and illustrating any challenges encountered during development.

[1] *CS21120 Assignment (Online),*
**https://blackboard.aber.ac.uk/ultra/courses/_46440_1/outline/file/_2658783_1**
Accessed: 30th October 2023
Note: Restricted Access (Aberystwyth University Blackboard)

# 1 Task 1: The Player Class

The implementation of the *Player* class was a straightforward process, incorporating error handling for invalid player position. Initialising the player names and position within the constructor was a simple task, and the getters efficiently retrieved the name and position of the players.

Overall, no significant difficulties were encountered during the implementation of this class.

# 2 Task 2: The Team Class

Implementing the *Team* class went smoothly. I stored the players in an *ArrayList* structure due to its dynamic size allocation, allowing for flexible growth of the array with the addition of more players. Incorporating the *IPlayer* interface allowed for the retrieval of players names and positions, ensuring the *addPlayer()* method allocated one player to each position. I did encounter a minor issue when testing my program, where players were allowed to be allocated in the same position. I resolved this issue by ensuring I was implementing the *IPlayer* interface throughout the class, and not the *Player* class.

## 2.1 Worst Case Big-O Time Complexities

- *addPlayer()*
  - O(n) – Uses a linear search to check if a player with the same position exists in the team.
- *getPlayerInPosition()*
  - O(n) – Iterates through the list to find a player in a specified position.
- *getPlayers()*
  - O(1) – Returns array of players.
- *getName()*
  - O(1) – Returns the name of the team.

# 3 Task 3: Constructing the Match Tree

This task was more challenging compared to the previous tasks, particularly in the implementation of the *MatchTree* class. The *TreeNode* class represents a single node within the tree structure and was straightforward to implement, whereas the *MatchTree* class is more complex as it manages the entire tournament structure.

## 3.1 Testing the Code

To test the functionality of the *TreeNode class*, I utilised the *TreeNodeTests* testing class provided in the assignment code. As my code had already passed all the methods in the *TeamTests* testing class, the methods in the *TreeNodeTests* testing class were easy to pass.

As the task only required the *getRoot()* method to be implemented in full and the constructor to be written, I couldn't yet test the full functionality of the *MatchTree* class. However, I could test it by using the methods within the *testMatchTreeConstruction* class. Initially, I created a binary tree structure in a separate program, generating a basic tree representation[2]; module materials helped me to code this[3]. I then integrated this tree structure into the *MatchTree* class, making appropriate modifications.

## 3.2 Worst Case Big-O Time Complexity for the Constructor

The constructor in the *MatchTree* class has a worst-case Big-O time complexity of O(n log n), where n is the number of teams. This is due to the binary tree construction process, which divides the teams into subgroups until the tree is fully formed. The time complexity is logarithmic as the splitting halves at each step.

[2] *Simple Implementation of a Binary Search Tree (Online),*
**https://github.com/dtfiedler/java-binary-tree/blob/master/src/BinaryTree.java**
Accessed 2nd November 2023

[3] *CS21120 Workshop 4: Phone Book Manager (Online),*
**https://blackboard.aber.ac.uk/ultra/courses/_46440_1/outline/edit/document/_2649036_1?courseId=_46440_1&view=content**
Accessed: 31st October 2023
Note: Restricted Access (Aberystwyth University Blackboard)

# 4 Task 4: Retrieving and Scoring Matches

Task 4 involved the completion of the *MatchTree* class by implementing the *getNextMatch()* and *setScore()* methods, as well as the *Main* class for manual testing. This task had more significant challenges, and although the class functioned as intended, I encountered numerous difficulties attempting to pass the *MatchTreeTests* testing class.

## 4.1 Implementation

The *MatchTree* class constructs a binary tree structure representing the knockout tournament. The *getNextMatch()* method recursively traverses the tree to find the next match to be played. It then checks if the current node has both child nodes with their teams set before returning the node for the next match. The *setScore()* method assigns the score for the current match, checking the scores of the two child nodes to determine the team that goes to the next round. I then created the *Main* class which I initially used to see the output the tree using the *TreePrinter* class, providing a visual representation based on user-entered scores for each team.

## 4.2 Problems Encountered

The primary issue I encountered was getting the *MatchTreeTests* testing class to pass the *testEightTeams()* method. The issue appears in the *getNextMatchNode()* method due to traversing the tree in a depth-first manner instead of a breadth-first manner, returning the parent node instead of the third child node. I attempted to address this by incorporating a *Queue* structure[4] to enqueue nodes during the traversal, checking that the current node satisfies the condition for the next match. However, I unsuccessfully implemented this without inadvertently causing other testing methods to fail.

## 4.3 Testing

I carried out testing through a combination of manual and automated methods utilising the *Main, MatchTreeTests* and *MatchTreeConstructionTests* classes. Automated testing was useful as I could continuously run the test classes to see where my code was encountering errors. However, this doesn't cover all scenarios so I carried out manual testing entering different inputs when running the *Main* class. This ensured that error checking was working, as well as the correct output of the tree. Detailed screenshots of testing using the *Main class* are provided in Section 4.4.

[4] *Breadth-First Search Algorithm in Java (Online),* **https://www.baeldung.com/java-breadth-first-search#:~:text=The%20idea%20behind%20the%20BFS,first%20node%20from%20the%20queue**
Accessed: 31st October 2023

## 4.4 Screenshots

### 4.4.1 Testing Two Teams

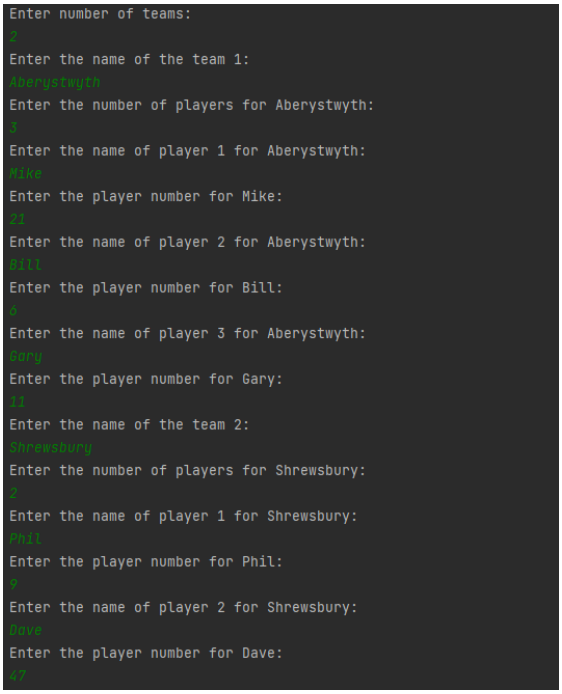


Figure 1.1: User-Input for Two Teams – Correct Format for Names of Teams, and Player Details.

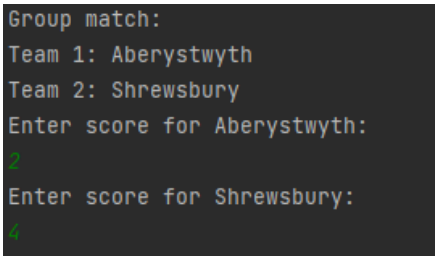


Figure 1.2: Group Match Stage – User-Input for Team Scores.

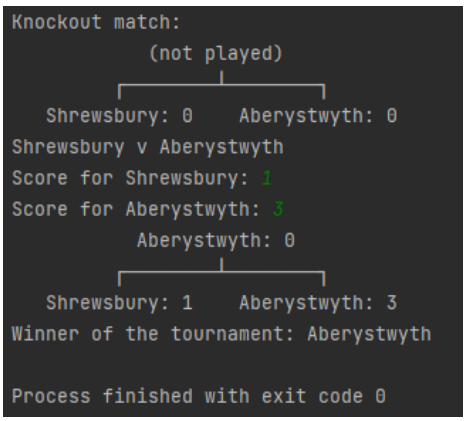


Figure 1.3: Knockout Stage – User-Input for Team Scores until Tournament Winner is Output.

### 4.4.2 Testing Four Teams

```
Enter number of teams:
4
Enter the name of the team 1:
Aberystwyth
Enter the number of players for Aberystwyth:
0
Enter the name of the team 2:
Shrewsbury
Enter the number of players for Shrewsbury:
0
Enter the name of the team 3:
Bangor
Enter the number of players for Bangor:
0
Enter the name of the team 4:
Cardiff
Enter the number of players for Cardiff:
0
```

Figure 2.1: User-Input for Four Teams – Correct Format for Names of Teams, and Player Details.

```
Group match:
Team 1: Bangor
Team 2: Cardiff
Enter score for Bangor:
2
Enter score for Cardiff:
3
Group match:
Team 1: Bangor
Team 2: Aberystwyth
Enter score for Bangor:
1
Enter score for Aberystwyth:
1
Group match:
Team 1: Bangor
Team 2: Shrewsbury
Enter score for Bangor:
4
Enter score for Shrewsbury:
2
```

```
Group match:
Team 1: Cardiff
Team 2: Aberystwyth
Enter score for Cardiff:
2
Enter score for Aberystwyth:
3
Group match:
Team 1: Cardiff
Team 2: Shrewsbury
Enter score for Cardiff:
3
Enter score for Shrewsbury:
1
Group match:
Team 1: Aberystwyth
Team 2: Shrewsbury
Enter score for Aberystwyth:
0
Enter score for Shrewsbury:
1
```

Figures 2.2.1; 2.2.2: Group Match Stage – User-Input for Team Scores.

```
Knockout match:
          (not played)
          ┌──────────┐
   Cardiff: 0       Bangor: 0
Cardiff v Bangor
Score for Cardiff: 0
Score for Bangor: 2
          Bangor: 0
          ┌──────┐
  Cardiff: 0     Bangor: 2
Winner of the tournament: Bangor
```

Figure 2.3: Knockout Stage – User-Input for Team Scores until Tournament Winner is Output.

### 4.4.3 Testing Eight Teams

```
Enter number of teams:
8
Enter the name of the team 1:
Aberystwyth
Enter the number of players for Aberystwyth:
0
Enter the name of the team 2:
Shrewsbury
Enter the number of players for Shrewsbury:
0
Enter the name of the team 3:
Bangor
Enter the number of players for Bangor:
0
Enter the name of the team 4:
Cardiff
Enter the number of players for Cardiff:
0
```

```
Enter the name of the team 5:
Swansea
Enter the number of players for Swansea:
0
Enter the name of the team 6:
Liverpool
Enter the number of players for Liverpool:
0
Enter the name of the team 7:
Nottingham
Enter the number of players for Nottingham:
0
Enter the name of the team 8:
Luton
Enter the number of players for Luton:
0
```

Figure 3.1.1; 3.1.2: User-Input for Eight Teams – Correct Format for Names of Teams, and Player Details.

```
Group match:
Team 1: Cardiff
Team 2: Liverpool
Enter score for Cardiff:
1
Enter score for Liverpool:
1
Group match:
Team 1: Cardiff
Team 2: Luton
Enter score for Cardiff:
0
Enter score for Luton:
3
Group match:
Team 1: Cardiff
Team 2: Swansea
Enter score for Cardiff:
2
Enter score for Swansea:
4
Group match:
Team 1: Liverpool
Team 2: Luton
Enter score for Liverpool:
3
Enter score for Luton:
1
```

```
Group match:
Team 1: Liverpool
Team 2: Swansea
Enter score for Liverpool:
0
Enter score for Swansea:
4
Group match:
Team 1: Luton
Team 2: Swansea
Enter score for Luton:
3
Enter score for Swansea:
2
Group match:
Team 1: Shrewsbury
Team 2: Bangor
Enter score for Shrewsbury:
2
Enter score for Bangor:
4
Group match:
Team 1: Shrewsbury
Team 2: Aberystwyth
Enter score for Shrewsbury:
0
Enter score for Aberystwyth:
1
```

```
Group match:
Team 1: Shrewsbury
Team 2: Nottingham
Enter score for Shrewsbury:
1
Enter score for Nottingham:
2
Group match:
Team 1: Bangor
Team 2: Aberystwyth
Enter score for Bangor:
3
Enter score for Aberystwyth:
3
Group match:
Team 1: Bangor
Team 2: Nottingham
Enter score for Bangor:
0
Enter score for Nottingham:
0
Group match:
Team 1: Aberystwyth
Team 2: Nottingham
Enter score for Aberystwyth:
1
Enter score for Nottingham:
3
```

Figure 3.2.1; 3.2.2; 3.2.3: Group Match Stage – User-Input for Team Scores.

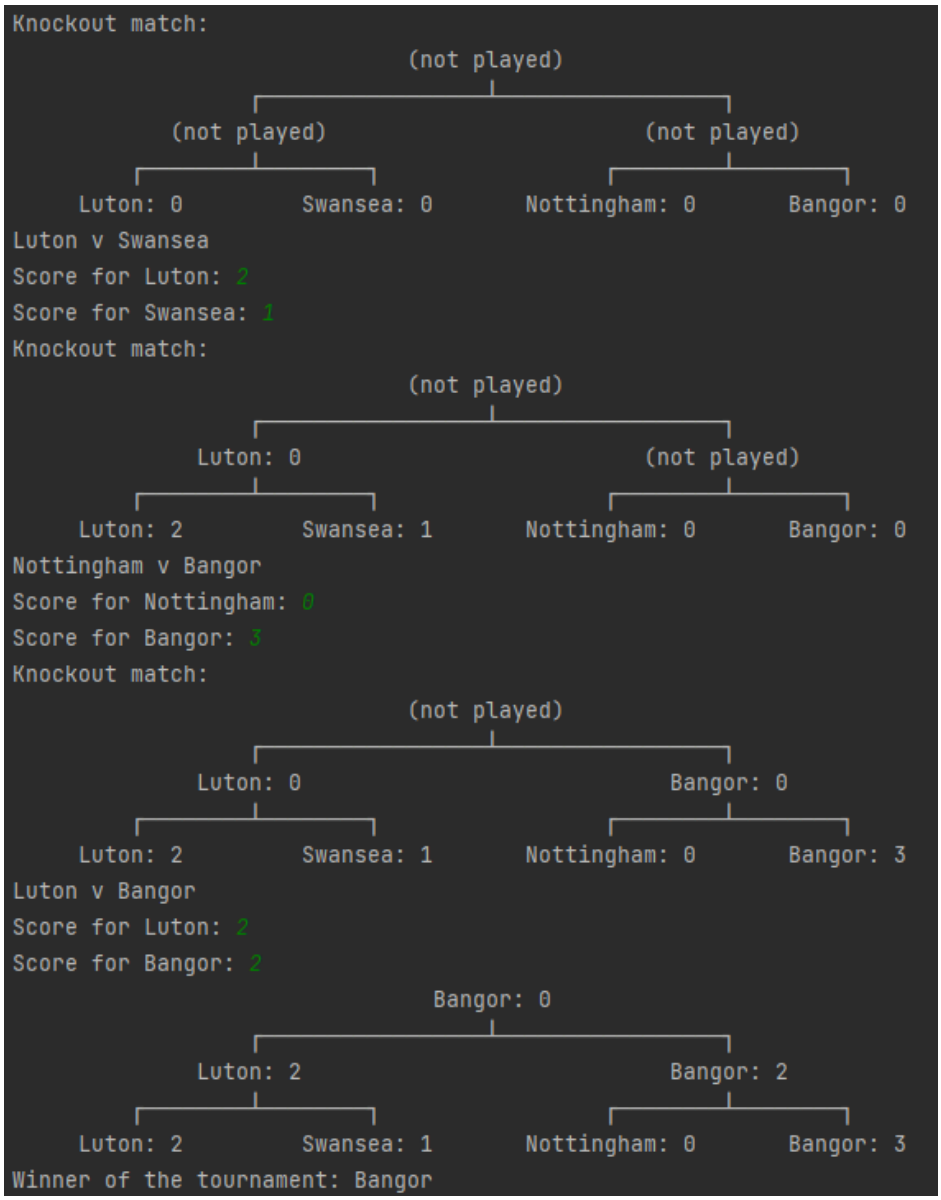


Figure 3.3: Knockout Stage – User-Input for Team Scores until Tournament Winner is Output.

### 4.4.4 Testing Sixteen Teams

```
Enter number of teams:
16
Enter the name of the team 1:
Aberystwyth
Enter the number of players for Aberystwyth:
0
Enter the name of the team 2:
Shrewsbury
Enter the number of players for Shrewsbury:
0
Enter the name of the team 3:
Bangor
Enter the number of players for Bangor:
0
Enter the name of the team 4:
Cardiff
Enter the number of players for Cardiff:
0
Enter the name of the team 5:
Swansea
Enter the number of players for Swansea:
0
Enter the name of the team 6:
Liverpool
Enter the number of players for Liverpool:
0
Enter the name of the team 7:
Nottingham
Enter the number of players for Nottingham:
0
Enter the name of the team 8:
Luton
Enter the number of players for Luton:
0
```

```
Enter the name of the team 9:
Telford
Enter the number of players for Telford:
0
Enter the name of the team 10:
Barmouth
Enter the number of players for Barmouth:
0
Enter the name of the team 11:
Chester
Enter the number of players for Chester:
0
Enter the name of the team 12:
Bristol
Enter the number of players for Bristol:
0
Enter the name of the team 13:
Aberdeen
Enter the number of players for Aberdeen:
0
Enter the name of the team 14:
Norwich
Enter the number of players for Norwich:
0
Enter the name of the team 15:
Manchester
Enter the number of players for Manchester:
0
Enter the name of the team 16:
Leeds
Enter the number of players for Leeds:
0
```

Figure 4.1.1; 4.1.2: User-Input for Sixteen Teams – Correct Format for Names of Teams, and Player Details.

```
Group match:
Team 1: Bangor
Team 2: Bristol
Enter score for Bangor:
2
Enter score for Bristol:
1
Group match:
Team 1: Bangor
Team 2: Cardiff
Enter score for Bangor:
0
Enter score for Cardiff:
0
Group match:
Team 1: Bangor
Team 2: Aberdeen
Enter score for Bangor:
0
Enter score for Aberdeen:
1
Group match:
Team 1: Bristol
Team 2: Cardiff
Enter score for Bristol:
1
Enter score for Cardiff:
0
Group match:
Team 1: Bristol
Team 2: Aberdeen
Enter score for Bristol:
4
Enter score for Aberdeen:
1
```

```
Group match:
Team 1: Cardiff
Team 2: Aberdeen
Enter score for Cardiff:
3
Enter score for Aberdeen:
5
Group match:
Team 1: Swansea
Team 2: Aberystwyth
Enter score for Swansea:
1
Enter score for Aberystwyth:
2
Group match:
Team 1: Swansea
Team 2: Barmouth
Enter score for Swansea:
4
Enter score for Barmouth:
1
Group match:
Team 1: Swansea
Team 2: Chester
Enter score for Swansea:
4
Enter score for Chester:
0
Group match:
Team 1: Aberystwyth
Team 2: Barmouth
Enter score for Aberystwyth:
2
Enter score for Barmouth:
1
```

```
Group match:
Team 1: Aberystwyth
Team 2: Chester
Enter score for Aberystwyth:
0
Enter score for Chester:
3
Group match:
Team 1: Barmouth
Team 2: Chester
Enter score for Barmouth:
0
Enter score for Chester:
2
Group match:
Team 1: Leeds
Team 2: Nottingham
Enter score for Leeds:
2
Enter score for Nottingham:
0
Group match:
Team 1: Leeds
Team 2: Norwich
Enter score for Leeds:
4
Enter score for Norwich:
0
Group match:
Team 1: Leeds
Team 2: Manchester
Enter score for Leeds:
3
Enter score for Manchester:
0
```

```
Group match:
Team 1: Nottingham
Team 2: Norwich
Enter score for Nottingham:
4
Enter score for Norwich:
2
Group match:
Team 1: Nottingham
Team 2: Manchester
Enter score for Nottingham:
2
Enter score for Manchester:
1
Group match:
Team 1: Norwich
Team 2: Manchester
Enter score for Norwich:
4
Enter score for Manchester:
0
Group match:
Team 1: Shrewsbury
Team 2: Telford
Enter score for Shrewsbury:
2
Enter score for Telford:
0
Group match:
Team 1: Shrewsbury
Team 2: Liverpool
Enter score for Shrewsbury:
4
Enter score for Liverpool:
1
```

```
Group match:
Team 1: Shrewsbury
Team 2: Luton
Enter score for Shrewsbury:
3
Enter score for Luton:
1
Group match:
Team 1: Telford
Team 2: Liverpool
Enter score for Telford:
1
Enter score for Liverpool:
1
Group match:
Team 1: Telford
Team 2: Luton
Enter score for Telford:
0
Enter score for Luton:
2
Group match:
Team 1: Liverpool
Team 2: Luton
Enter score for Liverpool:
2
Enter score for Luton:
1
```

Figure 4.2.1; 4.2.2; 4.2.3; 4.2.4; 4.2.5: Group Match Stage – User-Input for Team Scores.

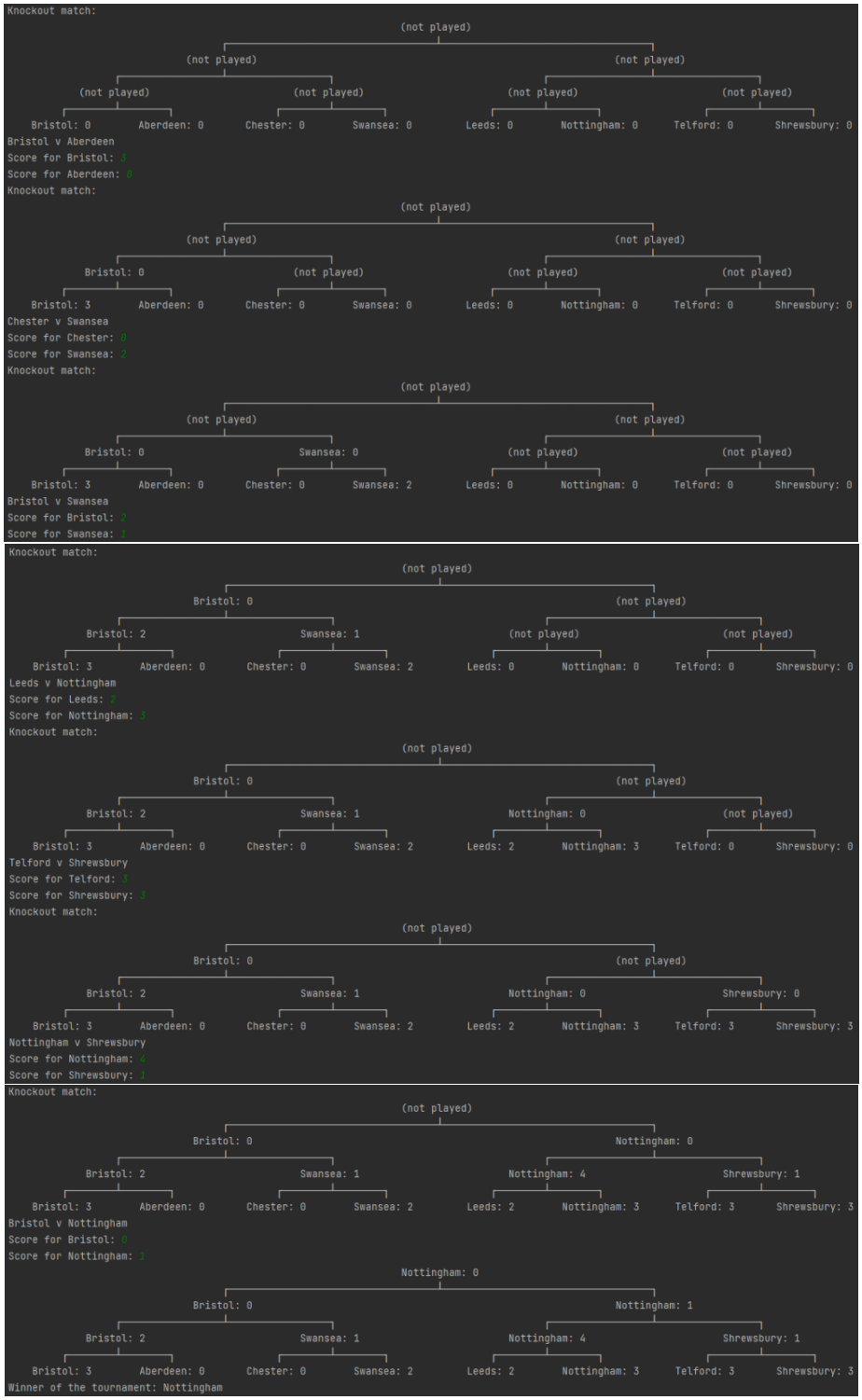


Figure 4.3: Knockout Stage – User-Input for Team Scores until Tournament Winner is Output.

### 4.4.5 Testing Seven Teams (odd number)

```
Enter number of teams:
7
Enter the name of the team 1:
Aberystwyth
Enter the number of players for Aberystwyth:
0
Enter the name of the team 2:
Shrewsbury
Enter the number of players for Shrewsbury:
0
Enter the name of the team 3:
Bangor
Enter the number of players for Bangor:
0
Enter the name of the team 4:
Cardiff
Enter the number of players for Cardiff:
0
Enter the name of the team 5:
Swansea
Enter the number of players for Swansea:
0
Enter the name of the team 6:
Liverpool
Enter the number of players for Liverpool:
0
Enter the name of the team 7:
Nottingham
Enter the number of players for Nottingham:
0
```

Figure 5.1: User-Input for Seven Teams – Correct Format for Names of Teams, and Player Details.

```
Group match:
Team 1: Cardiff
Team 2: Shrewsbury
Enter score for Cardiff:
1
Enter score for Shrewsbury:
4
Group match:
Team 1: Cardiff
Team 2: Swansea
Enter score for Cardiff:
3
Enter score for Swansea:
3
Group match:
Team 1: Cardiff
Team 2: Aberystwyth
Enter score for Cardiff:
2
Enter score for Aberystwyth:
4
Group match:
Team 1: Shrewsbury
Team 2: Swansea
Enter score for Shrewsbury:
2
Enter score for Swansea:
3
Group match:
Team 1: Shrewsbury
Team 2: Aberystwyth
Enter score for Shrewsbury:
1
Enter score for Aberystwyth:
1
```

```
Group match:
Team 1: Swansea
Team 2: Aberystwyth
Enter score for Swansea:
3
Enter score for Aberystwyth:
1
Group match:
Team 1: Nottingham
Team 2: Liverpool
Enter score for Nottingham:
0
Enter score for Liverpool:
1
Group match:
Team 1: Nottingham
Team 2: Bangor
Enter score for Nottingham:
0
Enter score for Bangor:
0
Group match:
Team 1: Liverpool
Team 2: Bangor
Enter score for Liverpool:
3
Enter score for Bangor:
4
```

Figure 5.2.1; 5.2.2: Group Match Stage – User-Input for Team Scores.


Figure 5.3: Knockout Stage – User-Input for Team Scores until Tournament Winner is Output.

### 4.4.6 Testing Invalid User-Input

```
Enter number of teams:
Test
Number of teams must be an integer. Try again
Enter number of teams:
!!!
Number of teams must be an integer. Try again
Enter number of teams:
5
Enter the name of the team 1:
```

Figure 6.1: Incorrect User-Input – Non-Integer Format for Number of Teams.

```
Enter the name of the team 1:
Aberystwyth
Enter the number of players for Aberystwyth:
Test
Number of players must be an integer. Try again
Enter the number of players for Aberystwyth:
11
Enter the name of player 1 for Aberystwyth:
```

Figure 6.2: Incorrect User-Input – Non-Integer Format for Number of Players.

```
Enter the name of player 1 for Aberystwyth:
Mike
Enter the player number for Mike:
42
Enter the name of player 2 for Aberystwyth:
Brian
Enter the player number for Brian:
42
A player with the same number already exists. Try again
Enter the player number for Brian:
```

Figure 6.3: Incorrect User-Input – Entering Two Players with the Same Number.

```
Group match:
Team 1: Aberystwyth
Team 2: Shrewsbury
Enter score for Aberystwyth:
Test
Score must be an integer. Try again
Enter score for Aberystwyth:
1
Enter score for Shrewsbury:
```

Figure 6.4: Incorrect User-Input – Non-Integer Format for Score.

# 5 The Group Stage

This task involved the implementation of the *Group* and *GroupMatch* classes, allowing for the management of matches and teams within a group. The primary objective was to divide each group of four teams down to the two top teams, decided by the number of wins. This required generating random groups[5], creating matches, keeping track of match results, and calculating points.

## 5.1 Problems Encountered

Whilst implementing the classes, I encountered several issues that were easily resolved when examining through my code. One issue was that the score was not being calculated correctly, producing a tree where all the nodes at the lowest depth displayed 0 and the program would then crash when proceeding the next level. This was resolved by fixing logical errors within the *Group* class, ensuring the score entered by the user was correctly being stored in the *ArrayList* structure. Additionally, I encountered an error in the *Main* class where the teams were not being randomised correctly, to resolve this I added a *for* loop to copy the teams in the randomised array to a new array.

## 5.2 Generating the Sequence of Matches

The sequence of matches was generated in the *Group* class using a nested loop iterating through the list of teams. This guaranteed that each team played against every other in the group exactly once. Matches were created using the *GroupMatch* class, storing the resulting matches in the 'matches' list of the *Group* class.

## 5.3 Tie Resolution Approach

Within the *calculatePoints* method of the *GroupMatch* class, teams were awarded 1 point for a draw. Sorting the teams in the table in the *Group* class was done based on each team's total points. If two teams had identical scores, tiebreakers were resolved by randomly selecting a team to progress to the next stage.

## 5.4 Calculation of Required Matches

To calculate the number of matches required for a given number of teams (where 'n' is the number of teams) in a group to play each other once, I applied the formula:

- (n * (n - 1)) / 2

[5] *Getting Random Numbers in Java (Online)*
**https://stackoverflow.com/questions/5887709/getting-random-numbers-in-java**
Accessed: 8th November 2023

# 6 Self-Evaluation

I believe that I have completed this assignment to a high-level, creating robust and functional code, as well as going beyond the assignment specification by creating my own testing class. Although I encountered many challenges during the development process, I found the *MatchTree* class particularly difficult due to the method for finding the next match to be played and creating the tree structure. Attempting to resolve the *testEightTeams()* method within the *MatchTreeTests* testing class was especially difficult due to having to receive the correct depth of the tree. On the other hand, I did well in creating the *Main* class, ensuring that it met all the functional requirements as well as ensuring it was user-friendly and contained error-handling.

As my code fully functions, contains JavaDoc and my own testing class, and my report is detailed, I believe I should be awarded a grade of 85%.