Java Manual

1.) Bug Report

I don't believe I have any bugs

2.) Feature Report

All Features implemented, nothing special added.

3.) Description of the data structures/classes

Classes description

- Tournament class - Base class

Keeping track of stats and dictates the winner of the tournament.

- Round - Base Class

Keep track of the starting player from the coinflip and input validation.

- Player - Base Class

Have strategies stored, such as the characteristics of players' color, total pieces, if it is human, score, win, and track of all pieces.

- Human - Derived from player class

User validation. Size validation, letter validation, and number validation.

- Computer - Derived from player class

Finds and players strategy.

- BoardView - Composite class of Board, Round, Player, Human, Computer, and Tournament

It displays and manipulates everything relating to the game's logic, from the board, movement, validation, coin flip, tournament winner, score, and serialization.

- Board - Base class

Validates if a piece can move to specific positions. Look for win conditions. Modifies board to move pieces. Finds pieces on the board and looks for positions to be played on the board. Converts coordinate between readable human text for the user and converted coordinates for my game to read correctly.

Data Structures

Player: p_pieces - held all the current player's pieces on the board

p_newpos - held all the positions that specific piece could go

allStrats - held all the strategies that were found.

Board: visitedPieces—used for my win condition logic. I held all the pieces my function passed through to find if the total number of pieces on the board equaled the total number of pieces counted.

Algorithms

DFS (Depth First Search) - I used a simpler version of DFS for my win condition. The logic behind my win condition was a recursive function. I would find a piece on the board that matched the current player's color in the recursive function. From there, I would look around the piece and see if other similar colored pieces exist around the current piece. If so, go onto this piece and see if similar pieces exist next to it. However, it would also store pieces that have already been visited. After branching out, the total number of visited pieces is compared to the total number on the board. If the totals are equal, the win condition is true; however, if it doesn't, it is false.

4.) Log

April 8: Converting C++ code into Java

I translated my tournament class from C++ into Java, aiming to familiarize myself with the Language and grasp its nuances. One notable difference I encountered was that Java requires header files to be consolidated into a single file, which posed a unique challenge.

Time Spent: 2 hours

April 10: Continue to convert C++ code into Java.

The next class I tackled was my Round. While there were similarities with C++, one of the most notable differences was the need in Java to initialize a Random Object for my coinflip function to work. I then proceeded to create my Player class, although I set everything up without the computer/helper functions, another key difference from C++.

Time spent: 2 hours

April 15: Finish converting base game

Finish implementing the rest of the code, including Computer, Human, Board, and BoardView classes, and start with the main difference between vectors and List and vectors to have a difference in bounds checking. Some of my C++ could sometimes go out of bounds in its vectors, but Java was very strict with the limitations of going outside the array. So, additional boundaries were added to my movement validations.

Time Spent: 2 hours.

Another issue arose when selecting a piece because the character conversion in its math is different. So, instead, I changed my approach using a switch statement because Strings in Java have a function to get specific characters in a string. From this, I would compare the characters and then return a number appropriate for the board to parse through.

Time Spent: 1 hour.

April 18: Learning Andriod Studio

Watched half of the video

https://www.youtube.com/watch?v=tZvjSl9dswg&t=6191s&ab_channel=CalebCurry

Time Spent: 2 hours.

Playing around with Andriod Studio

Time Spent: 2 Hours

April 20: Continuing to learn Andriod Studio

I finished the video and started some test applications to play around with the components and activities in Android Studio.

Time Spent: 2 hours.

Playing around with Andriod Studio

Time Spent: 2 Hours

April 22: Starting converting my Java code into an Andriod Studio.

I developed the main activity, where the game will start as a New or serialized game. I'm going to finish the new game before starting with the serialization. I made an onClick event handler on New Game to open a new " load game activity." I figured the best way to make a board was dynamically changing the Table layout. The information on the board is given from the board class. Depending on the information from the board class, it changes the piece's color.

Time spent: 2 hours.

April 23: Setting up the piece onClick

The buttons on the boards have onClick Listen for validatePiece to see whether or not the click button corresponds to the current player. If the button is clicked correctly, it will turn green; otherwise, it isn't valid.

Time Spent: 1 hour

The issue with the piece onClick before is that I need to check if the first piece clicked is valid and if the second piece clicked is a valid place to move to. However, this isn't possible because this onClick only checks one piece and can't combine and validate these selections. I changed to having a setOnClickListener to create an onClick event handler within it. This allows me to check for the number of clicks and use this to see if the selected piece is correct. Once the piece is chosen correctly, check through all the moves for the piece to see if the second selection is possible from the board. If valid, play through the selection.

Time Spent: 2 hours

April 25: Making move piece more userfriendly and swap player

Since I got movement setup, I decided to work on a swap player. I used my round's logic to flip a coin and decide on a winner. Coinflip started with two buttons, Tails and Heads, depending on which one chose was the response to the coin flip. If correct, the human player is assigned blue pieces, and the other is white. Implemented swap to my validation selection movement. Realize that the game moves too fast for the user to understand. If the player selects the first valid movement, it'll automatically play it out and immediately do the computer's move.

Time spent: 2 hours

I changed my movement to move off an onClick button from continue. So, my previous code was used to validate piece selection and movement, but the continuation now advances the game for the computer or human player.

Time Spent: 2 hours

April 27: Implementing help & Log

I added a help button to my game XML. When pressed, the button will text text information to the Text view below the board in yellow. The strategy is found through functions from the board and stored in Player, with pick strategies returning the strategy to use from all the ones found.

Time Spent: 30 minutes

I have added a vertically scrolling text view at the bottom of the app to show all previous moves made by the player and computer. Every time the continue button is pressed, the log is updated with information about the selected piece and the chosen movement. If it's the computer's turn, the entire strategy is saved instead.

Time Spent: 1.5 hours

April 26: Change helps to have all movement and display score

I updated the help to display all possible pieces and movements. Once the help button is clicked, the information is shown in the log text view, and this is done by storing all the pieces from the function in the board class. All pieces are stored in a list and then run through a for loop to display all possible places to move for that piece. I also implemented a button to display all the log information if the user wants to see the log after using the help function.

Time spent: 1.5 hours.

The score is updated after every round finishes. This happens when a win condition is met, and the information is stored in a text view called stats at the bottom left of the app. All information is updated in the player, interpreted, and run through displayScore to update the view correctly.

Time Spent: 1 hour

April 27: Announce round winner and issue with thwart

When using the help function in the game, a thwart would appear when it was a winning move. I reprogrammed my entire thwart strategy to see if I could find an issue. I was still getting the same problem, although I confirmed my logic was correct. I realized my help function was passing two human players through its parameters instead of one computer and one human.

Time Spent: 2 hours

We need to implement a round-winner announcement and figure out who won. I redeveloped my tournament class since it hasn't been used before. Depending on the information given, the endRound will decide who won that round. The information will then be displayed on a Text View in a constraint layout, asking if the user wants to play another round. Depending on the response, it either starts a new round or displays the winner of the round based on the total number of times from the more significant player. If a rematch happens, the player with the most wins will be first, or a coin flip will occur if the score is tied.

Time Spent: 2 hours

April 28: Serialization implemented

I tried to implement serialization by adding Write/Read and permissions in Android Manifest XML, although this doesn't work anymore since the implementation is now deprecated and not allowed.

Time Spent: 2 hours

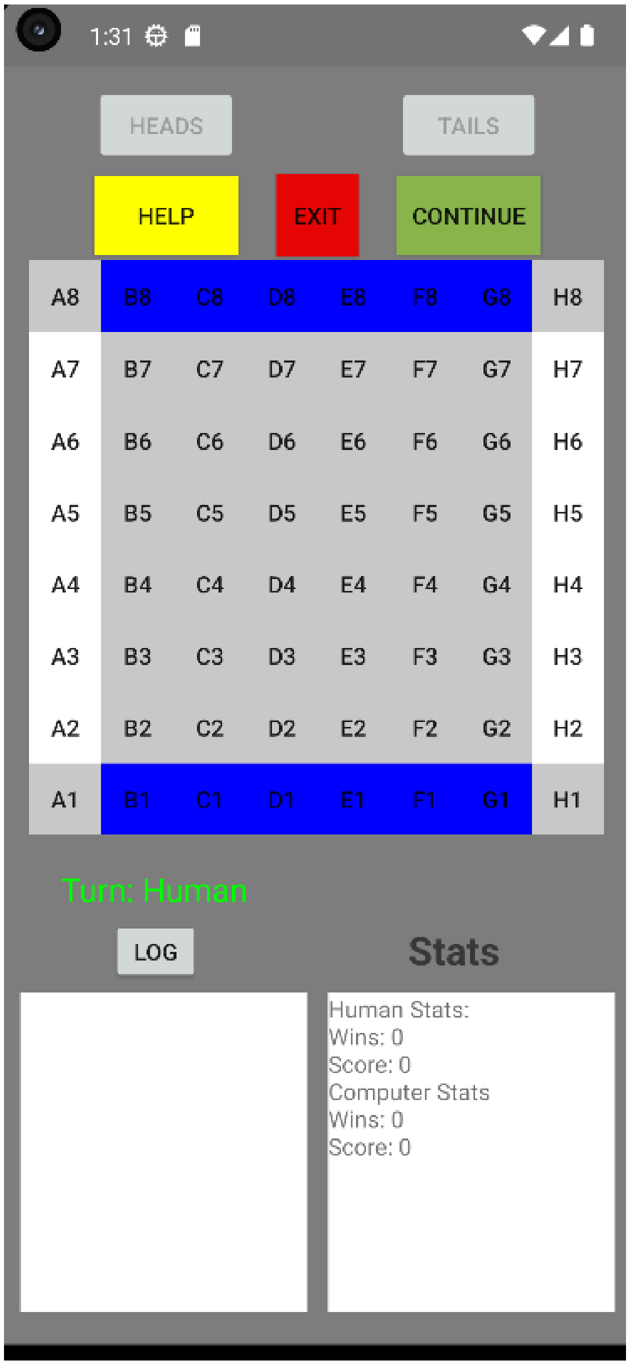
I made a listView to display all serialized files with an onClick listener event to choose and run the file. The listView displays all the files from Ra that are possible. Once the file is selected, loadGame is run, which gives information on the serialized file.

Time Spent: 2 hours

5.) Generative AI assistance:

None.

6.) Screen Shots

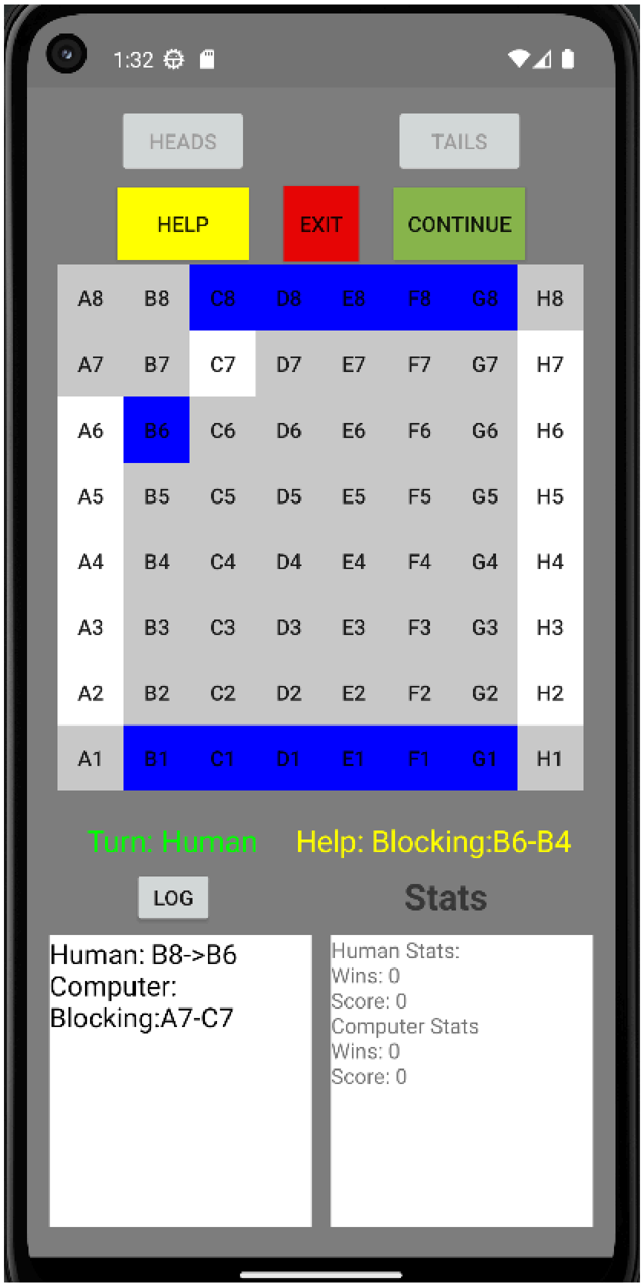


A8	B8	C8	D8	E8	F8	G8	H8
A7	B7	C7	D7	E7	F7	G7	H7
A6	B6	C6	D6	E6	F6	G6	H6
A5	B5	C5	D5	E5	F5	G5	H5
A4	B4	C4	D4	E4	F4	G4	H4
A3	B3	C3	D3	E3	F3	G3	H3
A2	B2	C2	D2	E2	F2	G2	H2
A1	B1	C1	D1	E1	F1	G1	H1

Turn: Human

Stats

Human Stats:
Wins: 0
Score: 0
Computer Stats
Wins: 0
Score: 0

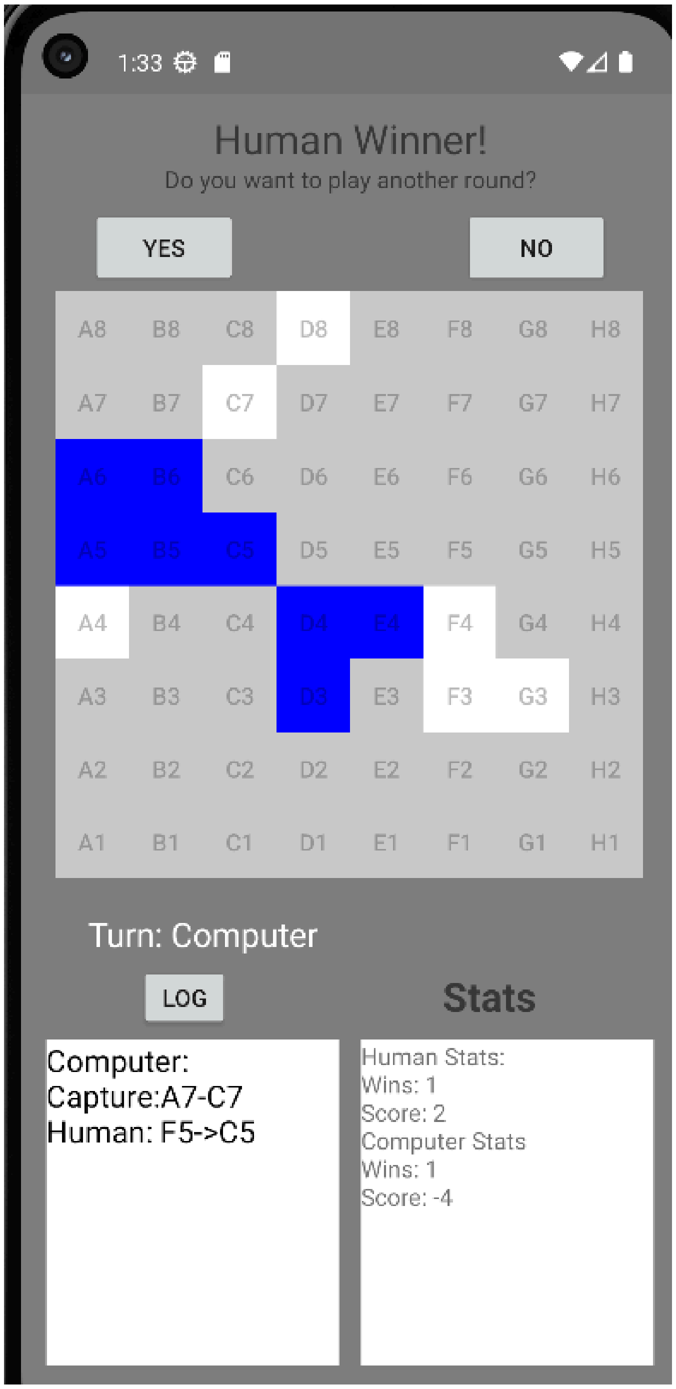


A8	B8	C8	D8	E8	F8	G8	H8
A7	B7	C7	D7	E7	F7	G7	H7
A6	B6	C6	D6	E6	F6	G6	H6
A5	B5	C5	D5	E5	F5	G5	H5
A4	B4	C4	D4	E4	F4	G4	H4
A3	B3	C3	D3	E3	F3	G3	H3
A2	B2	C2	D2	E2	F2	G2	H2
A1	B1	C1	D1	E1	F1	G1	H1

Turn: Human Help: Blocking:B6-B4

```
Human: B8->B6
Computer:
Blocking:A7-C7
```

```
Human Stats:
Wins: 0
Score: 0
Computer Stats
Wins: 0
Score: 0
```



Turn: Computer

```
Computer:
Capture:A7-C7
Human: F5->C5
```

```
Human Stats:
Wins: 1
Score: 2
Computer Stats
Wins: 1
Score: -4
```