Coursework Report

Kevin Falconer
401726410@napier.ac.uk
Edinburgh Napier University - Algorithms and Data Structure's (SET09117)

1 Introduction

The coursework assignment for the Algorithm's and Data Structure's was to design and implement a working checkers game using a combination of teaching from the module, and self-learning. The coursework assignment could be implemented in the programming language of the student's choice. The game must represent a board, two players, and the positions of each checkers piece. The core feature of the game would allow to human player's to complete a game of checker's, using text commands input using a console. Alongside the implementation of the core checkers game, other feature's such as the amendment of moves, recording play, and the inclusion of artificial intelligence which would allow human to computer play, and computer to computer play, alongside the human to human aspect of the game.

2 Design

The checker game is separated into four key classes. Move, Game, Board, and Main.

2.1 Board

The board class is used to contain the checkers board. The checker board in this checkers game is represented through the use of an integer 2d array data structure. The array is initialized manually, with each piece being represented using a number. The integer one represents a standard red checker piece, two, which represents a red king piece, three, which represents a standard black piece, and four, which represents a red king piece. The integer zero is also used to represent the empty spaces on the checker board. Within the class each of these values are assigned to variable's which can be accessed and used by other classes such as Move, to move pieces on the board, and to change standard pieces to kings. The decision to implement the array in this method rather than using for loop's for was extremely handful when it came to testing the game. This meant that scenarios could be created in which aspects of movement such as capturing an opponent, restricting movement for a certain piece, and the implementation of jumping could be tested without having to spend time playing the game, and waiting for those scenarios to unfold. This was especially helpful in the implantation of movement for the King Piece's, and allowing pieces the ability to jump across the board.

Alongside the array, the class also includes a method which is used to print the checker board, alongside each piece, within the console.

2.2 Move

The move class is key in regards to the movement of piece's on the checker board. This class is used to apply the rules for movement such as not allowing standard pieces to move only horizontally or vertically. The class includes two algorithms. The first algorithm is movePiece. This algorithm is used to move the pieces from the board to the location which the user has input, and then assign the location the user has moved from to a 0. The algorithm includes code which if the row the user wants to move to is either 0, for the black player, or 7 for the red player, then that piece on the board will become a king piece. The other algorithm in the class is a Boolean algorithm called canPieceMove. This algorithm dictates this rules of the game, alongside the rule's in which how piece's can move around the checker board. The algorithm is split into four section's, one for each checker piece. There's a universal rule that any move input by a user which is less than zero and greater than eight will return an error, and false value as the move would out of bounds. Each checker piece also have rule's which are in each section which include piece's not being allowed to move vertically or horizontally, ensuring that a user select's one of it's own piece's and ensuring that the destination of the checker which the user input does not land on one of it's own piece's or an opponent's, as a checker must jump over a piece in order to capture the piece. For the movement aspect of each piece is programmed using nested if statements based on the value of the checker piece which is being moved. Standard Red and Black piece's can only move horizontally down left and right, while the King Pieces can be moved in any direction. Each move which a user try to input's which is illegal is a standard if statement which will return false, alongside a message which is written to the console. When a piece is captured, 1 point is added to the variable's for red and black piece's and this ensures that once a player reaches twelve points that the game will end.

2.3 Game

The game class is the key component in getting the game up and running. There are two algorithms within this class. The first is a user input algorithm which takes the user input for origin and destination of each checker piece. The other algorithm within is the class is called gameLoop. This algorithm is the key aspect of the checkers game which actually allows the game to run, and allow users to play the game. The algorithm contains a while loop, which while the game is running, checks that the move which is input by the user legal, and then moves the piece. The algorithm also includes an if statement which means that if a player reaches twelve piece's captured that the game will end.

2.4 Main

The main class, although small is used to allow the game to be played using the Main Java method. Within the method it includes methods from other classes which always the game to be player such as the game loop, which prints the board

3 Enhancements

The key enhancements which I would implement for the checker's game would be the inclusion of an artificial intelligence aspect of the game to represent an opponent. The implementation of this feature would mean that the game could function as a single player game as well as a multiplayer game. The implantation of the artificial intelligence would include difficulty level's, which would attract both new players, looking to learn the rules of checkers and improve, and experienced players who are looking to test their skills. Other enhancements would include recording play, which would record the moves of each player, alongside the result of the match, which could be loaded by the player, the ability to undo, and redo moves if a player inputs a mistake, or finds a better move, and a visual GUI interface meaning that the player's would not need to input each command manually using a console.

4 Critical Evaluation

As a whole, the checkers game works well, and as it should in representing a standard human-human game. All of the rules of the game in regards to movement have been implemented, and I am pleased I was able. I was also pleased I was able to include the ability for a user to jump, removing two of the opponent's pieces from the board, as this functionality was not in place even a few hours prior to the deadline, so overall from a critical perspective of the feature's implemented in the game, I am pleased with the outcome, however I hoped that it would also include some of the enhancements I have cited in the previous section. One aspect of the game which I was not able to get functioning prior to the deadline of the assignment was a try-catch command for user input. The try-catch command would stop user's inputting any data other than one integer for each query. If a player inputs a non integer input then an out of bounds array exception will be thrown, so I would liked to have fixed that aspect of the coursework assignment.

5 Personal Evaluation

From a personal perspective, I feel that I performed extremely poorly during the duration of the coursework assignment. Programming is an aspect of Computing which I do not enjoy in comparison to other aspect's such as Web Development, and using Linux. I spent far too much time not getting the help I needed on aspects of the task I was unsure about such as movement, and inevitably, it may have resulted in the coursework assignment not being completed had I not been granted an extension. Personally, I really see this assignment as a wakeup call in regards to Programming. Although I do, I should at least be competent in implementing a simple checkers game such as this assignment, and from now on I will be working on improving my programming skills.

References