Alquerque

Danny Nicolai Larsen, Mikkel Brix Nielsen, Steffen Bach

November 11, 2021

Contents

1 Introduction

Hello

2 Design

3 Implementation

4 Test

5 Conclusion

6 Appendix

6.1 Program Code

```java
import java.util.InputMismatchException;
import java.util.Scanner;
public class Alquerque {
    private static Scanner reader;
    private static Board board;
    public static final char EMPTY = ' ';
    private static String whiteName, blackName;
    private static int cpuDepth;
    private static boolean isWhiteCPU, isBlackCPU, isWhite;
    // ved ikke om de ånedenstende variabler skal ævre her, men det gjorde main mere clean.
    private static String coordsFrom;
    private static String coordsTo;
    private static Move nextMove; // skal nok ikke ævre en klasse variabel

    public static void main(String[] args) {
        init();
        do { // loop for making moves
            printBoard();
            if (!isWhiteCPU && isWhite || !isBlackCPU && !isWhite) {
                boolean inputWithinRange = false;
                do { // loop for validating player moves
                    System.out.print("It's " + (isWhite ? whiteName : blackName) + "'s turn" + ", please enter which " +
                        "piece you want to move: ");
                    coordsFrom = reader.nextLine().trim();
                    System.out.print("Please enter where you want to move the piece: ");
                    coordsTo = reader.nextLine().trim();
                    if (isValidCoords(coordsFrom) && isValidCoords(coordsTo)) { //Checks if input is a valid letter+number
                        nextMove = new Move(convertCoordinate(coordsFrom), convertCoordinate(coordsTo)); //Converts coordinate to int position
                        if (board.isLegal(nextMove))
                            inputWithinRange = true;
                    }
                    if (!inputWithinRange)
                        System.out.println(coordsFrom + " to " + coordsTo + " is " +
                                "not a valid move, please try again.");
                } while (!inputWithinRange);
                board.move(nextMove);
            } else if (!board.isGameOver()) {
                nextMove = new Minimax().nextMove(board, cpuDepth, isWhite);
                System.out.println((isWhite ? whiteName : blackName) + " played " +
                        convertPosition(nextMove.from()) + " to " + convertPosition(nextMove.to()));
                board.move(nextMove);
            }
            isWhite = !isWhite; // changes whos turn it is
        } while (!board.isGameOver());
        printBoard(); // prints the state of the board when game over
        if (board.black().length > 0 && board.white().length <= 0)
            System.out.println(blackName + " is the winner!");
        else if (board.black().length <= 0 && board.white().length > 0)
            System.out.println(whiteName + " is the winner!");
        else
            System.out.println("It's a draw!");
    }

    /**
     * Initializes the program and runs the start menu.
     */
    private static void init() {
        reader = new Scanner(System.in);
        board = new Board();
        whiteName = "White(CPU)";
        blackName = "Black(CPU)";
        isWhite = true;
        int option;
        System.out.println("*******************************************");
        System.out.println("Greetings Master! And welcome to Alquerque.");
        System.out.println("*******************************************");
        do {
            printOptions();
            System.out.print("Please enter the number corresponding " +
                    "to the option you want executed: ");
            option = intCheck(reader);
            switch (option) {
                case 0:
                    System.out.println("You have chosen option " + option + ": Exit program");
                    System.out.println("Thank you for playing, have a nice day!");
                    break;
                case 1: // Player vs Player
```

```java
                    System.out.println("You have chosen option " + option + ": Player vs Player");
                    System.out.print("Please enter the name of player 1: ");
                    reader.nextLine(); // clears terminal input
                    whiteName = reader.nextLine().trim();
                    System.out.print("Please enter the name of player 2: ");
                    blackName = reader.nextLine().trim();
                    break;
                case 2: // Player vs CPU
                    System.out.println("You have chosen option " + option + ": Player vs CPU");
                    String color;
                    reader.nextLine(); // clears input
                    do {
                        System.out.print("Please enter the color you want to play " +
                                "black or white (B/W): ");
                        color = reader.nextLine();
                        switch (Character.toUpperCase(color.charAt(0))) {
                            case 'B':
                                System.out.println("\nYou have chosen to play black.\n" +
                                        "The CPU will therefore play white");
                                System.out.print("Please enter the name of the player: ");
                                blackName = reader.nextLine().trim();
                                System.out.println();
                                isWhiteCPU = true;
                                break;
                            case 'W':
                                System.out.println("\nYou have chosen to play white.\n" +
                                        "The CPU will therefore play black");
                                System.out.print("Please enter the name of the player: ");
                                whiteName = reader.nextLine().trim();
                                isBlackCPU = true;
                                break;
                            default:
                                System.out.println("'" + color + "'" + " is not a valid color " +
                                        "option, please try again.\n");
                        }
                    } while (Character.toUpperCase(color.charAt(0)) != 'B' && Character.toUpperCase(color.charAt(0)) != 'W');
                    System.out.print("How far ahead do you want the CPU to analyze: ");
                    cpuDepth = intCheck(reader);
                    break;
                case 3: // CPU vs CPU
                    System.out.println("You have chosen option " + option + ": CPU vs CPU");
                    System.out.print("How far ahead do you want the CPU's to analyze: ");
                    cpuDepth = intCheck(reader);
                    isWhiteCPU = true;
                    isBlackCPU = true;
                    break;
                default:
                    System.out.println("Invalid option, " + option + " is not a valid option\n");
            }
        } while (option > 3 && option < 0);
        reader.nextLine(); // clears input before proceeding
    }

    /**
     * Prints the option menu to the terminal.
     */
    private static void printOptions() {
        System.out.println("Now, what do you wish to do?");
        System.out.println("****************************");
        System.out.println("Option 0: Exit program");
        System.out.println("Option 1: Player vs Player");
        System.out.println("Option 2: Player vs CPU");
        System.out.println("Option 3: CPU vs CPU");
        System.out.println("****************************");
        System.out.println();
    }

    /**
     * Creates a representation of the game board with the pieces correctly placed
     * in the form of a two dimensional array.
     * Precondition: Relies on method black() and white() to return valid positions numbered from 1-25
     * @return a two dimensional array 5 x 5 with the game pieces placed correctly
     */
    private static char[][] boardWithPieces() {
        char[][] boardArr = new char[6][5]; //A-E & (no 0) 1-5
        for (int j = 1; j < boardArr.length; j++)
            for (int i = 0; i < boardArr[j].length; i++)
                boardArr[j][i] = EMPTY; // Fills board with empty spaces
        for (int i = 0; i < board.black().length; i++)
            boardArr[((board.black()[i] - 1) / 5) + 1][((board.black()[i] - 1) % 5)] = 'B'; // Places black pieces
        for (int i = 0; i < board.white().length; i++)
            boardArr[((board.white()[i] - 1) / 5) + 1][((board.white()[i] - 1) % 5)] = 'W'; // Places white pieces
        return boardArr;
    }

    /**
     * prints a representation of the board to the terminal
     */
    private static void printBoard() {
        System.out.println(); // new line
        int i = 0, j = 1;
        System.out.println("   A   B   C   D   E"); //upper-coordinate-line (A-E)
        char[][] boardWithPieces = boardWithPieces();
        while (j < 6) {
            System.out.print(j + " "); //left-hand coordinate (1-5)
            while (i < 5) {
                System.out.print("[" + boardWithPieces[j][i] + "]");
                if (i < 4)
                    System.out.print("-");
                i++;
            }
            System.out.print(" " + (j)); //right-hand coordinate (1-5)
            System.out.println("");
            i = 0;
            if (j % 2 == 1 && j < 5)
                System.out.println("   | \\ | / | \\ | / |");
            else if (j % 2 == 0)
                System.out.println("   | / | \\ | / | \\ |");
            j++;
        }
        System.out.println("   A   B   C   D   E"); //bottom-coordinate-line (A-E)
        System.out.println(""); // new line
    }

    /**
     * Test wether an enterede coordinate is a valid coordinat
     * @param coords, a coordinate to be tested
     * @return true if the coordinat enterede is a valid coordinat else returns false
     */
    private static boolean isValidCoords(String coords){
        return (coords.matches("[A-Ea-e][1-5]")); // Regex for matching
    }

    /**
     * Converts an input coordinate to the corresponding position on the board, determined by numbers 1-25
     * @param coord move coordinate input from user
     * @return position on board, represented by an integer (1-25)
     */
    private static int convertCoordinate(String coord){
        int position = 0;
        switch(Character.toUpperCase(coord.charAt(0))){
            case 'A':  //value of each column is added to the row-determined multiplum of 5 (e.g. D is 4'th, so positional value is +4)
                position = (1+(5*((Integer.parseInt(coord.substring(1))-1))));
                break;
            case 'B':
                position = (2+(5*((Integer.parseInt(coord.substring(1))-1))));
                break;
            case 'C':
                position = (3+(5*((Integer.parseInt(coord.substring(1))-1))));
                break;
            case 'D':
                position = (4+(5*((Integer.parseInt(coord.substring(1))-1))));
                break;
            case 'E':
                position = (5+(5*((Integer.parseInt(coord.substring(1))-1))));
                break;
            default:
                return 0;
        }
        return position;
    }
    /**
```

```java
     * Converts an input position, represented by a number 1-25 to the corresponding coordinates in form [A-E][1-5]
     * @param position position represented by an int
     * @return coord position represented by coordinates [A-E][1-5]
     */
    private static String convertPosition(int position){
        String coord = "";
        switch ((position - 1) % 5){
            case 0:
                coord = "A";
                break;
            case 1:
                coord = "B";
                break;
            case 2:
                coord = "C";
                break;
            case 3:
                coord = "D";
                break;
            case 4:
                coord = "E";
                break;
        }
        coord = coord + ((position / 5) + 1);
        return coord;
    }
    /**
     * Catches exceptions when input doesn't match an integer
     */
    public static int intCheck(Scanner keyboard){
        try{
            return keyboard.nextInt(); // gets input from the user and checks if it throws input mismatch error
        } catch (InputMismatchException e) {
            System.out.print("Please input a number: ");
            keyboard.next(); // clears cache
            return intCheck(keyboard); // if error it prints that it is an error, and returns a recursive call of it self
        }
    }
} //close of class, m.i.s.
```