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;

    public static void main(String[] args) {
        String coordsFrom;
        String coordsTo;
        Move nextMove = new Move(0,0); // skal nok ikke være en klasse ←
            variabel
        init();
        do { // main game loop
            printBoard();
            if (!isWhiteCPU && isWhite || !isBlackCPU && !isWhite) {
                boolean inputWithinRange = false;
                do { // loop for validating the players input
                    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();
```

```java
26                        if (isValidCoords(coordsFrom) && isValidCoords(↩
                              coordsTo)) { //Checks if input is a valid letter+↩
                              number
27                            nextMove = new Move(convertCoordinate(coordsFrom),↩
                                  convertCoordinate(coordsTo)); //Converts ↩
                                  coordinate to int position
28                            if (board.isLegal(nextMove))
29                                inputWithinRange = true;
30                        }
31                        if (!inputWithinRange)
32                            System.out.println(coordsFrom + " to " + coordsTo ↩
                                  + " is " +
33                                  "not a valid move, please try again.");
34                    } while (!inputWithinRange);
35                    board.move(nextMove);
36                } else if (!board.isGameOver()) {
37                    nextMove = new Minimax().nextMove(board, cpuDepth, isWhite↩
                          );
38                    System.out.println((isWhite ? whiteName : blackName) + " ↩
                          played " +
39                            convertPosition(nextMove.from()) + " to " + ↩
                                convertPosition(nextMove.to()));
40                    board.move(nextMove);
41                }
42                isWhite = !isWhite; // changes who's turn it is
43            } while (!board.isGameOver());
44            System.out.println("This is the final state of the board");
45            printBoard(); // prints the state of the board when game over
46            if (board.black().length > 0 && board.white().length <= 0)
47                System.out.println(blackName + " is the winner!");
48            else if (board.black().length <= 0 && board.white().length > 0)
49                System.out.println(whiteName + " is the winner!");
50            else
51                System.out.println("It's a draw!");
52        }
53
54        /**
55         * Initializes the program and runs the start menu.
56         */
57        private static void init() {
58            reader = new Scanner(System.in);
59            board = new Board();
60            whiteName = "White(CPU)";
61            blackName = "Black(CPU)";
62            isWhite = true;
63            int option;
64            System.out.println("*******************************************");
65            System.out.println("Greetings Master! And welcome to Alquerque.");
66            System.out.println("*******************************************");
67            do {
68                printOptions();
69                option = reader.nextInt();
70                switch (option) {
71                    case 0:
72                        System.out.println("You have chosen option " + option ↩
                              + ": Exit program");
73                        System.out.println("Thank you for playing, have a nice↩
                              day!");
74                        break;
75                    case 1: // Player vs Player
76                        System.out.println("You have chosen option " + option ↩
                              + ": Player vs Player");
```

Page 4

```java
77                          System.out.print("Please enter the name of player 1: "←
                                );
78                          reader.nextLine(); // clears terminal input
79                          whiteName = reader.nextLine().trim();
80                          System.out.print("Please enter the name of player 2: "←
                                );
81                          blackName = reader.nextLine().trim();
82                          break;
83                      case 2: // Player vs CPU
84                          System.out.println("You have chosen option " + option ←
                                + ": Player vs CPU");
85                          String color;
86                          reader.nextLine(); // clears input
87                          do {
88                              System.out.print("Please enter the color you want ←
                                    to play " +
89                                  "black or white (B/W): ");
90                              color = reader.nextLine();
91                              if (color.matches("[Bb]")){
92                                  System.out.println("\nYou have chosen to play ←
                                        black.\n" +
93                                          "The CPU will therefore play white");
94                                  System.out.print("Please enter the name of the←
                                         player: ");
95                                  blackName = reader.nextLine().trim();
96                                  System.out.println();
97                                  isWhiteCPU = true;
98                              } else if (color.matches("[Ww]")){
99                                  System.out.println("\nYou have chosen to play ←
                                        white.\n" +
100                                         "The CPU will therefore play black");
101                                 System.out.print("Please enter the name of the←
                                         player: ");
102                                 whiteName = reader.nextLine().trim();
103                                 isBlackCPU = true;
104                             } else {
105                                 System.out.println("'" + color + "'" + " is ←
                                        not a valid color " +
106                                         "option, please try again.\n");
107                             }
108                         } while (!color.matches("[B|b|W|w]"));
109                         System.out.print("How far ahead do you want the CPU to←
                                 analyze: ");
110                         cpuDepth = reader.nextInt();
111                         break;
112                     case 3: // CPU vs CPU
113                         System.out.println("You have chosen option " + option ←
                                + ": CPU vs CPU");
114                         System.out.print("How far ahead do you want the CPU's ←
                                to analyze: ");
115                         cpuDepth = reader.nextInt();
116                         isWhiteCPU = true;
117                         isBlackCPU = true;
118                         break;
119                     default:
120                         System.out.println("Invalid option, " + option + " is ←
                                not a valid option\n");
121                 }
122         } while (option > 3 && option < 0);
123     }
124
125     /**
```

```
126         * Prints the option menu to the terminal.
127         */
128        private static void printOptions() {
129            System.out.println("Now, what do you wish to do?");
130            System.out.println("***************************");
131            System.out.println("Option 0: Exit program");
132            System.out.println("Option 1: Player vs Player");
133            System.out.println("Option 2: Player vs CPU");
134            System.out.println("Option 3: CPU vs CPU");
135            System.out.println("***************************");
136            System.out.println();
137            System.out.print("Please enter the number corresponding " +
138                    "to the option you want executed: ");
139        }
140
141        /**
142         * Creates a representation of the game board with the pieces ←
               correctly placed
143         * in the form of a two dimensional array.
144         * Precondition: Relies on method black() and white() to return valid ←
               positions numbered from 1-25
145         * @return a two dimensional array 5 x 5 with the game pieces placed ←
               correctly
146         */
147        private static char[][] boardWithPieces() {
148            char[][] boardArr = new char[6][5]; //A-E & (no 0) 1-5
149            for (int j = 1; j < boardArr.length; j++)
150                for (int i = 0; i < boardArr[j].length; i++)
151                    boardArr[j][i] = EMPTY; // Fills board with empty spaces
152            for (int i = 0; i < board.black().length; i++)
153                boardArr[((board.black()[i] - 1) / 5) + 1][((board.black()[i] ←
                    - 1) % 5)] = 'B'; // Places black pieces
154            for (int i = 0; i < board.white().length; i++)
155                boardArr[((board.white()[i] - 1) / 5) + 1][((board.white()[i] ←
                    - 1) % 5)] = 'W'; // Places white pieces
156            return boardArr;
157        }
158
159        /**
160         * prints a representation of the board to the terminal
161         */
162        private static void printBoard() {
163            System.out.println(); // new line
164            int i = 0, j = 1;
165            System.out.println("   A   B   C   D   E"); //upper-coordinate-←
                line (A-E)
166            char[][] boardWithPieces = boardWithPieces();
167            while (j < 6) {
168                System.out.print(j + " "); //left-hand coordinate (1-5)
169                while (i < 5) {
170                    System.out.print("[" + boardWithPieces[j][i] + "]");
171                    if (i < 4)
172                        System.out.print("-");
173                    i++;
174                }
175                System.out.print(" " + (j)); //right-hand coordinate (1-5)
176                System.out.println("");
177                i = 0;
178                if (j % 2 == 1 && j < 5)
179                    System.out.println("   | \\ | / | \\ | / |");
180                else if (j % 2 == 0)
181                    System.out.println("   | / | \\ | / | \\ |");
```

```java
182            j++;
183        }
184        System.out.println("   A   B   C   D   E"); //bottom-coordinate-↵
                line (A-E)
185        System.out.println(""); // new line
186    }
187
188    /**
189     * Test wether an enterede coordinate is a valid coordinat
190     * @param coords, a coordinate to be tested
191     * @return true if the coordinat enterede is a valid coordinat else ↵
                returns false
192     */
193    private static boolean isValidCoords(String coords){
194        return (coords.matches("[A-Ea-e][1-5]")); // Regex for matching
195    }
196
197    /**
198     * Converts an input coordinate to the corresponding position on the ↵
                board, determined by numbers 1-25
199     * @param coord move coordinate input from user
200     * @return position on board, represented by an integer (1-25)
201     */
202    private static int convertCoordinate(String coord){
203        int position = 0;
204        switch(Character.toUpperCase(coord.charAt(0))){
205            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)
206                position = (1+(5*((Integer.parseInt(coord.substring(1))-1)↵
                        )));
207                break;
208            case 'B':
209                position = (2+(5*((Integer.parseInt(coord.substring(1))-1)↵
                        )));
210                break;
211            case 'C':
212                position = (3+(5*((Integer.parseInt(coord.substring(1))-1)↵
                        )));
213                break;
214            case 'D':
215                position = (4+(5*((Integer.parseInt(coord.substring(1))-1)↵
                        )));
216                break;
217            case 'E':
218                position = (5+(5*((Integer.parseInt(coord.substring(1))-1)↵
                        )));
219                break;
220            default:
221                return 0;
222        }
223        return position;
224    }
225    /**
226     * Converts an input position, represented by a number 1-25 to the ↵
                corresponding coordinates in form [A-E][1-5]
227     * @param position position represented by an int
228     * @return coord position represented by coordinates [A-E][1-5]
229     */
230    private static String convertPosition(int position){
231        String coord = "";
232        switch ((position - 1) % 5){
```

```
233              case 0:
234                  coord = "A";
235                  break;
236              case 1:
237                  coord = "B";
238                  break;
239              case 2:
240                  coord = "C";
241                  break;
242              case 3:
243                  coord = "D";
244                  break;
245              case 4:
246                  coord = "E";
247                  break;
248          }
249          coord = coord + ((position / 5) + 1);
250          return coord;
251      }
252  } //close of class, m.i.s.
```