Faculty of Engineering & Technology
Electrical & Computer Engineering Department

ARTIFICIAL INTELLIGENCE-ENCS3340
Project #1

Prepared by:
Yazeed Hamdan 1201133
Instructor: Dr. Ismail Khater

Section: 3
Date: 18/6/2023 Birzeit

Project description

1. Starting

When the user run the program, the following menu in the Console will observe, there is two modes in the Game, the first one is User VS. User, and the other mode is User VS. Ai (Minimax algorithm)

```
Console ×
Ai [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (م ٩:٥٩:٣٣ ,٢٠٢٣/٠٦/٢٠) [pid: 32560]
                 -------------------------
                     Magnetic Cave Game
                 -------------------------
1. Player vs. Player
        In this mode first player will play with Black, Player 2 with White
        ---------------------------------------------------------------
2. Player vs. AI (MiniMax)
        ---------------------------------------------------------------
3. Exit Game
        ---------------------------------------------------------------
Please Select the mode:
```

Initially, the cave (the board) is empty.

```
Please Select the mode:
1
Current board:

_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _

Player 1, enter your move (row and column):
```

If the user selects the second mode of the game there is two ways to start the game, whether to start playing with black and the algorithm with white, or to let the algorithm starts first.

```
Console ×
Ai [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (٢٠٢٣/٠٦/٢٠, ٩:٥٩:٣٣ م) [pid: 32560]
                    -------------------------
                        Magnetic Cave Game
                    -------------------------
1. Player vs. Player
        In this mode first player will play with Black, Player 2 with White
        ---------------------------------------------------------------------
2. Player vs. AI (MiniMax)
        ---------------------------------------------------------------------
3. Exit Game
        ---------------------------------------------------------------------
Please Select the mode:
2
Please choose your position:
1. First Player (Black)
2. Second Player (White)
```

2. Win cases

The following three figures show the three cases of win (horizontal, vertical, and diagonal), the screenshots token as results of the first mode in the game.

```
Player 1, enter your move (row and column):
7 5
Player 1 wins!
W W _ _ _ _ W W
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
B B B B B _ _ _
_ _ _ _ _ _ _ _
```

```
Player 2, enter your move (row and column):
5 8
Player 2 wins!
B B B _ _ _ _ W
B _ _ _ _ _ _ W
B _ _ _ _ _ _ W
_ _ _ _ _ _ _ W
W _ _ _ _ _ _ W
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
B _ _ _ _ _ _ _
```

```
Player 1, enter your move (row and column):
5 5
Player 1 wins!
B _ _ _ _ _ _ _
W B _ _ _ _ _ _
W W B _ _ _ _ _
W B W B _ _ _ _
B W B W B _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
B W _ _ _ _ _ _
```

The following figure shows the user when played against the algorithm and won the minimax here is with depth = 5 .

```
Player 1, enter your move (row and column):
 1 5
Player 1 wins!
B B B B B _ _ _
B _ _ _ _ _ _ _
W W W _ _ _ _ _
W W W _ _ _ _ _
W W W _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
B _ _ _ _ B B B
```

In this case the Ai minimax algorithm won after I changed the depth value to 10

```
AI's turn:
AI wins!
B W B B B B _ _
B B B B W _ _ _
W W W W W _ _ _
W W W W _ _ _ _
W W W B _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
B _ _ _ _ B B B
```

In this case Ai minimax won with Black (starts first) with depth = 7

```
_ _ _ _ _ _ W W

AI's turn:
AI wins!
B B B W B W W W
B W W W W _ _ _
B B B B _ _ _ _
B _ _ _ _ _ _ _
B _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ W W
```

3. Some methods in the code:

The following method is one of the most important methods in the code, I called it in all game modes methods, user vs. user, Ai vs. user, user vs. Ai, it gives the color to each player first one takes black the other with white, and to make the move in the board.

```java
private boolean makeMove(int row, int col) { //method is used to make a move by a player, placing their piece on the board.
    if (row < 1 || row >= 9 || col < 1 || col >= 9 || board[row][col] != 'P') {
        return false;
    }

    char symbol = currentPlayer == 2 ? 'W' : 'B';
    board[row][col] = symbol;

    // Update the cells next to the stone in the same row to be playable
    if (col - 1 >= 0 && board[row][col - 1] == '_') {
        board[row][col - 1] = 'P';
    }
    if (col + 1 < 8 && board[row][col + 1] == '_') {
        board[row][col + 1] = 'P';
    }

    return true;
}
```

Minimax algorithm for a two-player game. The method minimax recursively evaluates the game state and returns the best score for the current player at a given depth of the game tree

```java
private int minimax(int depth, int alpha, int beta, boolean isMaximizingPlayer) { // MiniMax algorithm
    if (depth == 0 || gameEnded) {
        return Evaluation();
    }

    List<int[]> possibleMoves = generateMoves();
    if (isMaximizingPlayer) {
        int bestScore = Integer.MIN_VALUE;
        for (int[] move : possibleMoves) {
            int row = move[0];
            int col = move[1];
            board[row][col] = 'W';
            int score = minimax(depth - 1, alpha, beta, false);
            board[row][col] = 'P';
            bestScore = Math.max(bestScore, score);
            alpha = Math.max(alpha, bestScore);
            if (beta <= alpha) {
                break; // Beta cutoff
            }
        }
        return bestScore;
    } else {
        int bestScore = Integer.MAX_VALUE;
        for (int[] move : possibleMoves) {
            int row = move[0];
            int col = move[1];
            board[row][col] = 'B';
            int score = minimax(depth - 1, alpha, beta, true);
            board[row][col] = 'P';
            bestScore = Math.min(bestScore, score);
            beta = Math.min(beta, bestScore);
            if (beta <= alpha) {
                break;
            }
        }
        return bestScore;
```

The getAIMove method is responsible for determining the best move for the AI player using the Minimax algorithm. Here's a breakdown of how the method work

```java
public int[] getAIMove(int depth) { // This method is responsible for determining the best move for the AI player using the Minimax algorithm
    List<int[]> possibleMoves = generateMoves();
    if (possibleMoves.isEmpty()) {
        gameEnded = true;
        return null;
    }

    int[] bestMove = possibleMoves.get(0);
    int bestScore = Integer.MIN_VALUE;
    int alpha = Integer.MIN_VALUE;
    int beta = Integer.MAX_VALUE;

    for (int[] move : possibleMoves) {
        int row = move[0];
        int col = move[1];
        board[row][col] = 'W';
        int score = minimax(depth - 1, alpha, beta, false);
        board[row][col] = 'P';

        if (score > bestScore) {
            bestScore = score;
            bestMove = move;
        }

        alpha = Math.max(alpha, bestScore);
        if (beta <= alpha) {
            break; // beta cut-off
        }
    }
    return bestMove;
}
```

The Evaluation () method is responsible for calculating the current "score" of the game board. It evaluates the state of the board and assigns a score based on the positioning of the pieces. For each direction, it examines the next three adjacent cells to determine if they contain the same piece.

If the cells form a winning pattern (four consecutive pieces of the same color), it increments the score by 1 for white or decrements it by 1 for black. the method returns the calculated score

```
private int Evaluation() { //This method calculates the current "score" of the game board, it is done by examining each cell on the board and c
    int score = 0;

    for (int row = 1; row < 9; row++) {
        for (int col = 1; col < 9; col++) {

            if (board[row][col] == 'W' || board[row][col] == 'B') {
                // check horizontal
                if (col <= 5) {
                    for (int i = 0; i < 4; i++) {
                        if (board[row][col + i] == board[row][col]) {
                            score = board[row][col] == 'W' ? score + 1 : score - 1;
                        } else {
                            break;
                        }
                    }
                }

                // check vertical
                if (row <= 5) {
                    for (int i = 0; i < 4; i++) {
                        if (board[row + i][col] == board[row][col]) {
                            score = board[row][col] == 'W' ? score + 1 : score - 1;
                        } else {
                            break;
                        }
                    }
                }

                // check diagonal
                if (row <= 5 && col <= 5) {
                    for (int i = 0; i < 4; i++) {
                        if (board[row + i][col + i] == board[row][col]) {
                            score = board[row][col] == 'W' ? score + 1 : score - 1;
                        } else {
                            break;
```

```
                    }

                    // check diagonal
                    if (row <= 5 && col <= 5) {
                        for (int i = 0; i < 4; i++) {
                            if (board[row + i][col + i] == board[row][col]) {
                                score = board[row][col] == 'W' ? score + 1 : score - 1;
                            } else {
                                break;
                            }
                        }
                    }

                    // check diagonal
                    if (row >= 4 && col <= 5) {
                        for (int i = 0; i < 4; i++) {
                            if (board[row - i][col + i] == board[row][col]) {
                                score = board[row][col] == 'W' ? score + 1 : score - 1;
                            } else {
                                break;
                            }
                        }
                    }
                }
            }
        }
```

4. Some invalid cases:

When the user enters wrong value for the game mode, Error message will display in the console and the list of the modes will print again

```
Ai [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (م ١١:٢٠:١٨ ,٢٠٢٣/٠٦/٢٠) [pid: 33828]
                ------------------------
                    Magnetic Cave Game
                ------------------------
1. Player vs. Player
        In this mode first player will play with Black, Player 2 with White
        ---------------------------------------------------------------
2. Player vs. AI (MiniMax)
        ---------------------------------------------------------------
3. Exit Game
        ---------------------------------------------------------------
Please Select the mode:
111
        Please choose one mode to start the game or 3 to Exit !!


                ------------------------
                    Magnetic Cave Game
                ------------------------
1. Player vs. Player
        In this mode first player will play with Black, Player 2 with White
        ---------------------------------------------------------------
2. Player vs. AI (MiniMax)
        ---------------------------------------------------------------
3. Exit Game
        ---------------------------------------------------------------
Please Select the mode:
```

When the user chooses to play in invalid cell (already there is a brick previously) , (Invalid move) message will observe in the console, and the user will return his turn again in a valid cell.

```
Ai [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\java
Current board:
B W _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ B
_ _ _ _ _ _ _ W
_ _ _ _ _ _ _ _
_ _ _ _ _ _ W B

Player 1, enter your move (row and column):

1 1
Invalid move. Try again.
B W _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ B
_ _ _ _ _ _ _ W
_ _ _ _ _ _ _ _
_ _ _ _ _ _ W B
```

If the user enters a character by mistake, Error message will observe and he will return his turn until enter a valid numbers of a valid cell

```
Current board:
B _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _

Player 2, enter your move (row and column):
1 q
Invalid input. Please enter valid row and column numbers.
Current board:
B _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _

Player 2, enter your move (row and column):
```