**CSD311: Artificial Intelligence**

**Assignment 2: Checkers**

* Aditya Srivastava, 191011034 (40%)
* Arinjay Srivastava, 1910110087 (30%)
* Prabhat Kumar Singh Gaur, 1910110272 (30%)

**Classes and Functions Used:**

1. **Checkers.java:**

* **public Checkers();**

Constructor to initialize Logic.java to invoke the GUI for checkers.

* **public createMenu(Logic board); return type: void**

Function to create the Menu of the algorithms to be used in Checkers, namely “Random” and “Minimax” and display it in the GUI.

* **public newGame(); return type: void**

Function to reset the game.

* **public static dialogWindow(String infoMessage, String titleBar); return type: void**

Function to show the result of the game i.e., Win, Lose or Draw.

1. **Pieces.java:**

* **public Pieces(PieceType pieceType);**

Constructor to initialize Piece type i.e., Black or White.

* **public remove(Graphics graphics); return type: void**

Function to remove a Piece if it is eliminated.

* **public make(Graphics graphics, int j, int i); return type: void**

Function to make/represent a Piece if it becomes a King.

* **public static getSize(); return type: int**

Function to return the size of each block of the board.

* **public static within(int a, int b, int j, int i); return type: Boolean**

Function to check if the piece is within the board.

1. **PieceLocation.java:**

**public class PieceLocation:**

Class to get a Piece from Piece.java and initialize its location.

1. **PieceType.java:**

**public enum PieceType:**

Class to initialize whether either of the players’ piece is a regular or a king.

1. **Turn.java:**

* **public Turn (int x, int y, int X, int Y);**

Constructor to initialize the initial and final locations of a piece.

* **public getStartX(); return type: int**

Function to return the Initial X-coordinate of the Piece.

* **public getStartY(); return type: int**

Function to return the Initial Y-coordinate of the Piece.

* **public getEndX(); return type: int**

Function to return the Final X-coordinate of the Piece.

* **public getEndY(); return type: int**

Function to return the Final Y-coordinate of the Piece.

1. **AlreadyOccupiedException.java:**

**public AlreadyOccupiedException(String message):**

Constructor to return an error message if the user tries to use a block that is already occupied by another piece.

1. **Logic.java:**

* **Logic (int algoNum);**

Constructor to initialize the board and display appropriate messages after each move is played.

* **public chooseAlgo(int algo, char[][] game, int depth, String player); return type: int**

Function to switch between the two algorithms.

* **public moveMaker(); return type: void**

Function to determine which player should play when.

* **public taken (char [][] game, int x1, int y1, int x2, int y2); return type: void**

Function to implement capturing moves by the opponent.

* **public checkKing(char [][] game); return type: void**

Function to check if a piece is eligible to become a king or not.

* **public checkPos(int x1, int y1, int x2, int y2, char[][] game, boolean errorMessage); return type: boolean**

Function to check if a move is valid or not.

* **public getCorrectPos(String team, char[][] game); return type: List<Turn>**

Function to create a list of moves for either of the teams.

* **public captureTurnPerformed(char[][] game, int x1, int y1, int x2, int y2); return type: boolean**

Function to check if the move just performed was a capturing move or not.

* **public captureTurnAt(char[][] game, int col, int row); return type: boolean**

Function to find whether a capture move exists at the given co-ordinates or not.

* **public checkCaptureMove(char[][] game, String player); return type: boolean**

Function to check if a capture move exists anywhere on the board for a given team or not.

* **public updateGame(int x1, int y1, int x2, int y2); return type: void**

Function to update the underlying board representation**.**

* **public alterGUI(); return type: void**

Function to update the GUI of the board.

* **public randomAlgo(char[][] game, String player); return type: int**

Function which contains the logic for “Random” algorithm.

* **public minimaxAlgo(char[][] game,int depth, String player); return type: int**

Function which contains the logic for “Minimax” algorithm.

* **public AMove(char[][] game, int x1, int y1, int x2, int y2); return type: char[][]**

Function to move a piece within the board in random algorithm.

* **public getPoints(char[][] game, String winningPlayer); return type: int**

Function to return the real-time score of either of the players.

* **public blackWon(); return type: boolean**

Function to check if Black (Human) won or not.

* **public whiteWon(); return type: Boolean**

Function to check if White (AI) won or not.

* **public changePlayer(String player); return type: String**

Function used in Minimax algorithm to switch between MIN and MAX players.

* **public resetGame(); return type: void**

Function to reset the game.

* **public putPieces(Pieces piece, int u, int v); return type: void**

Function to put the pieces back to their initial positions once the game resets.

* **protected paintComponent(Graphics graphics); return type: void**

Function to paint the pieces Black and White respectively.

* **private paintGame(Graphics graphics) ; return type: void**

Function to paint the board blocks as required by the game.

* **public getPreferredSize(); return type: Dimension**

Function to pass on the size of blocks of the board.

* **public getGame(); return type: char[][]**

Function to prevent the use of white blocks while playing the game i.e., it provides the suitable format of the board for checkers.