**Checkers\_game**

Checkers\_game // Logic and models of the game /**Class Library**

Dependencies // Libraries and references required by the project (automatically)

Piece.cs // Class for managing individual pieces

Attributes:

- bool IsKing // Determines if the piece is a king

- bool IsPlayerOne // Determines if the piece belongs to Player 1

Methods:

+ PromoteToKing() // Promotes a piece to king

+ Move() // Handles logic for moving the piece

Board.cs // Clas for managing the game board

Attributes:

Piece[,] Grid // 2D array representing the 8x8 game board

Methods:

+ IsValidMove() : bool // Checks if a move is valid

+ MovePiece() : void // Moves a piece to a new position

+ HasPieces() : bool // Checks if a player still has pieces

+ PlayerHasMoves() : bool // Checks if a player has any valid moves

+ CanCapture() : bool // Checks if a piece can capture another

Game.cs // Class for managing the overall game

Attributes:

Board board // The board on which the game is played

- Player currentPlayer // Keeps track of the current player

Methods:

+ StartGame() : void // Starts the game and initializes the board

+ MakeMove() : bool // Executes a move and checks its validity

+ SwitchTurn() : void // Switches the turn between players

+ IsGameOver() : bool // Checks if the game is over

CheckersConsoleApp // **Console application** to run the game

Dependencies // Dependencies, including reference to Checkers game

Program.cs // Main program file and entry point of the application

Methods:

+ Main() : void // Entry point, starts the game in the console

+ PrintBoard() : void // Prints the current state of the board to the console

+ GetMoveFromUser() : (int, int, int, int) // Gets user input for moves

+ ParseMove() : (int, int, int, int) // Parses user input into coordinates

+ DisplayWinner() : void // Displays the winner at the end of the game

**About structure:**

1. **Checkers\_game**: **//***checkers logic*

This project contains all the core models and logic for the game. It is the "brain" of the entire game.

**Piece.cs**:

This file handles the logic for individual pieces (both regular pieces and kings).

It defines attributes that tell whether the piece belongs to Player 1 or Player 2, and whether it is a king.

**Board.cs**:

This file manages the game board, an 8x8 grid that holds the positions of all pieces.

The class contains methods to check if a move is valid, handle piece movement, and perform captures.

**Game.cs**:

This file manages the overall flow of the game. It includes the board, tracks which player's turn it is, and manages game rules such as checking for game over conditions.

It also switches turns between players and executes moves through interactions with the board.

1. **CheckersConsoleApp**:

This project is responsible for interacting with the user through the console.

**Program.cs**:

Contains the main code that launches the game in the console.

It prints the board, handles user input for moves, and shows whose turn it is.

It accepts moves in a format like "B2 C3" and converts them into coordinates used by the game logic.

It handles the game loop until the game ends.

**Ok\_how it might works:**

1. **Game Start**: The game begins when the Main() method in Program.cs is called. It creates an instance of the Game class, which initializes the board and starts the game.
2. **Player Moves**: Players input their moves through the console, which is processed by Program.cs and passed into the MakeMove() method of the Game class.
3. **Move Validation**: The Board and Piece classes work together to validate the move and update the state of the game, checking if a move is valid and whether any captures occur.
4. **Game End**: The game ends when one player either loses all their pieces or cannot make a valid move, at which point the game declares a winner.

----