

1,2

2,3

0,5

1,4

2,3

3,4

2,5 (Bl 1)

4,3

3.2 (W 1)

5.4

4.5 (Bl 2)

6.3

4.1

3.2

6.3( bl3)

4.1

3.2

2.3

1.4 (bl4)

3.2

6.1

5.2

1.6

2.5

0.1

1.2

0.7

1.6

5.0

6.1

4.1 (b5)

6.3

1.2

2.3

2.5

1.4

3.0

4.1

3.2(b6) KING

5.0

2.1

3.2

3.6

4.5

6.1

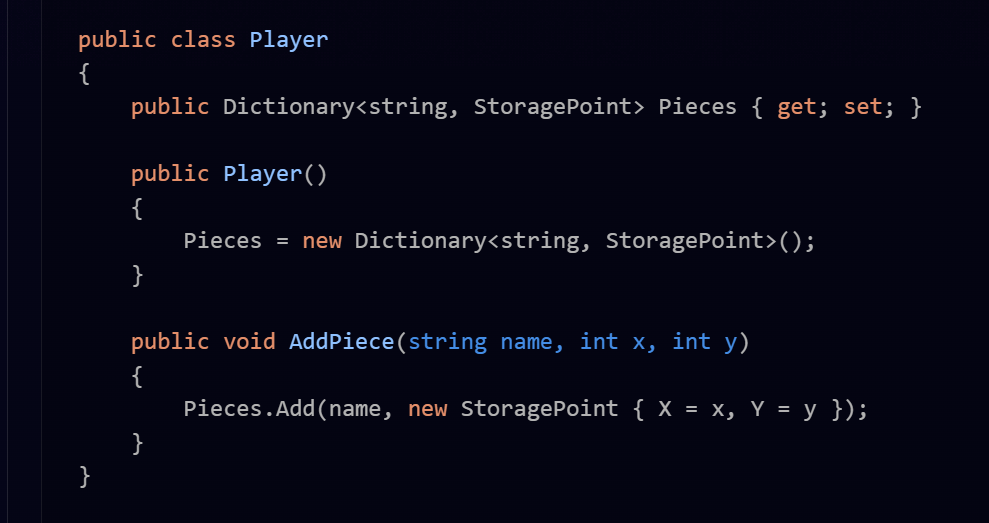
5.2

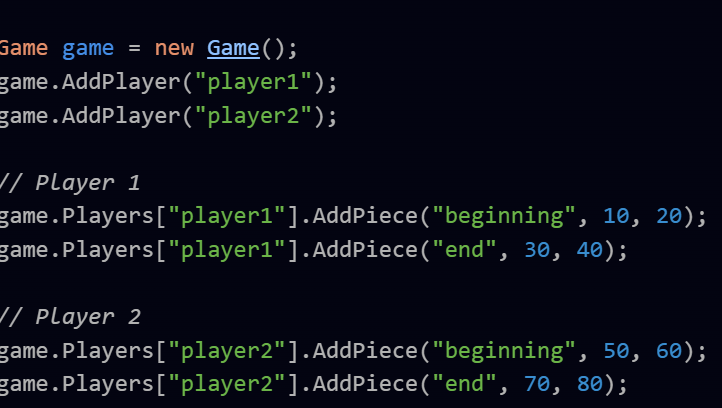
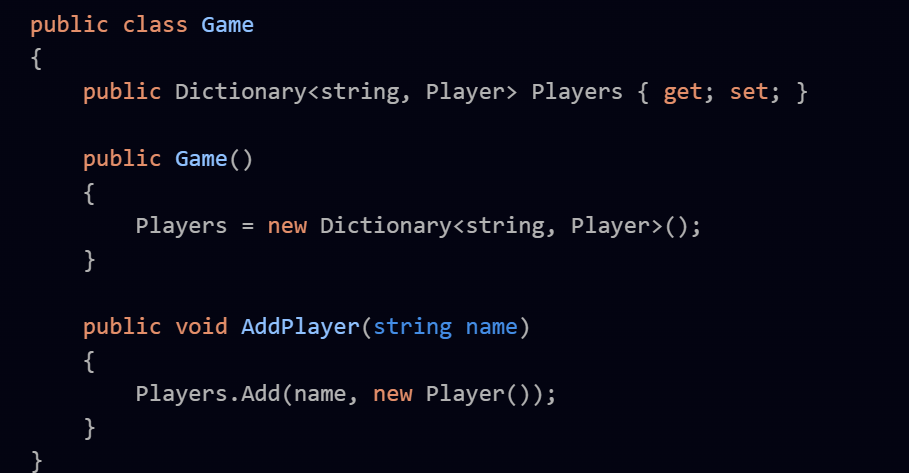
Dictionary nyimpen posisi,

Print pake idx (?)

Player menyimpan piece dimana saja apakai apa?







Board.cs

using System.Drawing;

using System.Security.Cryptography.X509Certificates;

using GameCheckker.Interface;

namespace GameCheckker.Class;

public class Board : IBoard

{

    private const int \_size = 8;

    private IPiece[,] grid;

    // private IPlayer player;

    Dictionary<IPlayer, IPiece> pointPlayer = new Dictionary<IPlayer, IPiece>();

    private Dictionary<Coordinate, (PieceType, PieceState)> piecePositions;

    private Dictionary<IPlayer, List<Coordinate>> playerPiecePosition;

    private GameController gameController;

    public Board()

    {

        grid = new IPiece[\_size, \_size];

        playerPiecePosition = new Dictionary<IPlayer, List<Coordinate>>();

        InitBoard();

    }

    private void InitBoard()

    {

        for (int y = 0; y < \_size; y++)

        {

            for (int x = 0; x < \_size; x++)

            {

                if ((x + y) % 2 == 1)

                {

                    if (y < 3)

                    {

                        grid[x, y] = new Piece(PieceType.White);

                    }

                    else if (y > 4)

                    {

                        grid[x, y] = new Piece(PieceType.Black);

                    }

                    else

                    {

                        grid[x, y] = new Piece(PieceType.Empty);

                    }

                }

                else

                {

                    grid[x, y] = new Piece(PieceType.Empty);

                }

            }

        }

    }

    public void DisplayBoard()

    {

        System.Console.WriteLine("          ");

        for (int x = 0; x < \_size; x++)

        {

            Console.Write("  " + x);

        }

        Console.WriteLine();

        for (int y = 0; y < \_size; y++)

        {

            Console.Write(y + " ");

            for (int x = 0; x < \_size; x++)

            {

                IPiece piece = grid[x, y];

                // char symbol = piece.Type switch

                // {

                //     PieceType.White => 'W',

                //     PieceType.Black => 'B',

                //     \_ => '|'

                // };

                char symbol;

                if (piece.Type == PieceType.White)

                {

                    symbol = 'W';

                }

                else if (piece.Type == PieceType.Black)

                {

                    symbol = 'B';

                }else{

                    symbol='|';

                }

                Console.Write(symbol + "  ");

            }

            Console.WriteLine();

        }

    }

    public bool MovePiece(Coordinate start, Coordinate end, PieceType playerType)

    // public bool MovePiece(Coordinate start, Coordinate end, PieceType playerType)

    // public bool MovePiece(List<Coordinate> start, List<Coordinate> end, PieceType playerType, IPlayer username)

    // public bool MovePiece(List<Coordinate> start, List<Coordinate> end, PieceType playerType, String username)

    {

        IPiece piece = grid[start.X, start.Y];// get piece at the starting position

        // IPiece piece = grid[start[0].X, start[0].Y];// get piece at the starting position

        // if (piece == null || piece.Type == PieceType.Empty)

        // // if (piece == null || piece.Type != playerType)

        // {

        //     return false;

        // }

        int dx = end.X - start.X;

        int dy = end.Y - start.Y;

        //memeriksa apakah gerakan valid

        if (Math.Abs(dx) == 1 && Math.Abs(dy) == 1)

        {

            //Cek apakah ada bidak di posisi akhir

            IPiece endPiece = grid[end.X, end.Y];

            if (endPiece != null && endPiece.Type != PieceType.Empty)

            {

                return false; // tidak bisa bergerak ke posisi yang sudah ditempati

            }

            // Memindahkan ke posisi akhir

            grid[end.X, end.Y] = piece;

            grid[start.X, start.Y] = new Piece(PieceType.Empty);

            return true;

            // ---

            // if (grid[end[0].X, end[0].Y].Type == PieceType.Empty)

            // {

            //     grid[end[0].X, end[0].Y] = piece;

            //     grid[start[0].X, start[0].Y] = new Piece(PieceType.Empty);

            //     UpdatePiecePosition(username, start[0], end[0], playerType);

            //     if (playerType == PieceType.White && end[0].Y == 0)

            //     {

            //         piece.State = PieceState.King;

            //     }

            //     else if (playerType == PieceType.Black && end[0].Y == \_size - 1)

            //     {

            //         piece.State = PieceState.King;

            //     }

            //     return true;

            // }

            // else

            // {

            //     return false;

            // }

        }

        // 2

        if (Math.Abs(dx) == 2 && Math.Abs(dy) == 2)

        {

            int captureX = start.X + dx / 2;

            int captureY = start.Y + dy / 2;

            IPiece capturedPiece = grid[captureX, captureY];

            if (capturedPiece != null && capturedPiece.Type != PieceType.Empty && capturedPiece.Type != piece.Type)

            {

                // Menangkap bidak

                grid[captureX, captureY] = new Piece(PieceType.Empty);

                grid[end.X, end.Y] = piece;

                grid[start.X, start.Y] = new Piece(PieceType.Empty);

                // --

                // UpdatePiecePosition(username, start[0], end[0], playerType);

                // RemovePiecePosition(username, new Coordinate(captureX, captureY));

                // if (playerType == PieceType.White && end[0].Y == 0)

                // {

                //     piece.State = PieceState.King;

                // }

                // else if (playerType == PieceType.Black && end[0].Y == \_size - 1)

                // {

                //     piece.State = PieceState.King;

                // }

                return true;

            }

        }

        return false;

    }

    // private void UpdatePiecePosition(IPlayer playerName, Coordinate start, Coordinate end, PieceType playerType)

    // {

    //     if (!playerPiecePosition.ContainsKey(playerName))

    //     {

    //         playerPiecePosition[playerName] = new List<Coordinate>();

    //     }

    //     playerPiecePosition[playerName].Remove(start);

    //     playerPiecePosition[playerName].Add(end);

    //     piecePositions.Remove(start);

    //     piecePositions[end] = (playerType, grid[end.X, end.Y].State);

    // }

    // private void RemovePiecePosition(IPlayer playerName, Coordinate position)

    // {

    //     if (playerPiecePosition.ContainsKey(playerName))

    //     {

    //         playerPiecePosition[playerName].Remove(position);

    //     }

    //     piecePositions.Remove(position);

    // }

}

GC

using System.Security.Cryptography.X509Certificates;

using GameCheckker.Class;

using GameCheckker.Interface;

namespace GameCheckker;

public class GameController

{

    private IBoard \_board;

    private IPlayer \_player1;

    private IPlayer \_player2;

    private IPlayer \_currentPlayer;

    // Action<int, int> Addition2 = AddNumbers2;

    //     Addition2(2, 3);

    private Dictionary<string, List<(List<int> start, List<int> end)>> playerMove;

    private Dictionary<IPlayer, List<(List<Coordinate> start, List<Coordinate> end)>> playerMoves;

    public GameController(string username1, string username2)

    {

        \_board = new Board();

        \_player1 = new Player(username1, PieceType.White);

        \_player2 = new Player(username2, PieceType.Black);

        \_currentPlayer = \_player1;

        //player, piecenya dimana aja dictionary.

        // playerMoves = new Dictionary<IPlayer, List<(List<Coordinate> start, List<Coordinate> end)>>();

    }

    public void StartGame()

    {

        while (true)

        {

            // Console.Clear();

            \_board.DisplayBoard();

            while (!\_currentPlayer.MakeMove(\_board))

            {

                System.Console.WriteLine("Invalid, try again");

            }

            SwitchPlayer();

            // Func<int, int, int> addFunc = (x, y) => x + y;

            // Action<IPlayer>? SwitchPlayer;

            // Func<IPlayer, IPlayer> DecidePlayer = SwitchPlayer;

            // DecidePlayer(\_currentPlayer);

        }

    }

    // public void SwitchPlayer(IPlayer \_currentPlayer)

    private void SwitchPlayer()

    {

        if (\_currentPlayer == \_player1)

        {

            \_currentPlayer = \_player2;

            // System.Console.WriteLine(\_currentPlayer);

        }

        else

        {

            \_currentPlayer = \_player1;

            // System.Console.WriteLine(\_currentPlayer);

        }

        // System.Console.WriteLine(\_currentPlayer);

        // return \_currentPlayer;

    }

    // public void RecordMove(IPlayer playerName, List<Coordinate> start, List<Coordinate> end)

    // {

    //     if (!playerMoves.ContainsKey(playerName))

    //     {

    //         playerMoves[playerName] = new List<(List<Coordinate> start, List<Coordinate> end)>();

    //     }

    //     playerMoves[playerName].Add((start, end));

    // }

    // public void DisplayPlayerMoves()

    // {

    //     foreach (var playerMove in playerMoves)

    //     {

    //         Console.WriteLine($"Moves by {playerMove.Key}:");

    //         foreach (var move in playerMove.Value)

    //         {

    //             Console.WriteLine($"From: ({move.start[0].X}, {move.start[0].Y}) To: ({move.end[0].X}, {move.end[0].Y})");

    //         }

    //     }

    // }

}

Lain

// public bool MovePiece(int startX, int startY, int endX, int endY)

// {

// IPiece piece = grid[startX, startY];

// if (piece == null || piece.Type == PieceType.Empty)

// {

// return false;

// }

// if (piece.Move(startX, startY, endX, endY))

// {

// grid[endX, endY] = piece;

// grid[startX, startY] = new Piece(PieceType.Empty);

// return true;

// }

// return false;

// }

//2 loncat

if (Math.Abs(dx) == 2 && Math.Abs(dy) == 2)

{

int captureX = startX + dx / 2;

int captureY = startY + dy / 2;

IPiece capturedPiece = grid[captureX, captureY];

if (capturedPiece != null && capturedPiece.Type != PieceType.Empty && capturedPiece.Type != piece.Type)

{

// Menangkap bidak

grid[captureX, captureY] = new Piece(PieceType.Empty);

grid[endX, endY] = piece;

grid[startX, startY] = new Piece(PieceType.Empty);

return true;

}

}

88

 // Validate move direction

        // if (playerType == PieceType.White && dy >= 0)

        // {

        //     return false; // Red pieces must move down

        // }

        // if (playerType == PieceType.Black && dy <= 0)

        // {

        //     return false; // Black pieces must move up

        // }

// \_currentPlayer = \_currentPlayer == \_player1 ? \_player2 : \_player1;

Apakah hasil pengurangan atau dy dan dx ini hasilnya selalu 1 dan 2?