using System;

using System.Collections.Generic;

namespace Checkers

{

// Square class to represent each square on the board

public class Square

{

public string Color { get; set; }

public bool IsEmpty { get; set; }

public Piece TypePiece { get; set; }

public void PlacePiece(Piece piece)

{

TypePiece = piece;

IsEmpty = false;

}

public void RemovePiece()

{

TypePiece = null;

IsEmpty = true;

}

}

// Piece class to represent each piece in the game

public class Piece

{

public int PosX { get; set; }

public int PosY { get; set; }

public string Color { get; set; }

public bool IsKing { get; set; }

public bool IsAlive { get; set; }

public Piece(int x, int y, string color)

{

PosX = x;

PosY = y;

Color = color;

IsKing = false;

IsAlive = true;

}

// Movement logic (simplified)

public void Move(int newX, int newY)

{

PosX = newX;

PosY = newY;

}

}

// Player class to represent the players

public class Player

{

public string Color { get; set; }

public List<Piece> Pieces { get; set; }

public Player(string color)

{

Color = color;

Pieces = new List<Piece>();

}

public void MakeMove(Piece piece, int newX, int newY)

{

piece.Move(newX, newY);

}

}

// Board class to manage the game board

public class Board

{

public Square[,] BoardArr { get; set; }

public Board()

{

BoardArr = new Square[8, 8];

InitBoard();

}

// Initialize board with squares and pieces

public void InitBoard()

{

for (int x = 0; x < 8; x++)

{

for (int y = 0; y < 8; y++)

{

BoardArr[x, y] = new Square

{

Color = (x + y) % 2 == 0 ? "White" : "Black",

IsEmpty = true

};

}

}

}

public void ShowBoard()

{

// Displays a simplified board (for debugging)

for (int y = 0; y < 8; y++)

{

for (int x = 0; x < 8; x++)

{

var square = BoardArr[x, y];

if (square.IsEmpty)

Console.Write(" . ");

else

Console.Write(square.TypePiece.Color == "Black" ? " B " : " W ");

}

Console.WriteLine();

}

}

public bool IsMoveOk(Piece piece, int newX, int newY)

{

// Validate the movement (simplified)

return newX >= 0 && newX < 8 && newY >= 0 && newY < 8 && BoardArr[newX, newY].IsEmpty;

}

public void UpdateBoard(Piece piece, int newX, int newY)

{

BoardArr[piece.PosX, piece.PosY].RemovePiece();

piece.Move(newX, newY);

BoardArr[newX, newY].PlacePiece(piece);

}

}

// Game class to manage the overall game flow

public class Game

{

public Player PlayerOne { get; set; }

public Player PlayerTwo { get; set; }

public Board Board { get; set; }

public bool GameStatus { get; set; }

public Game()

{

PlayerOne = new Player("White");

PlayerTwo = new Player("Black");

Board = new Board();

GameStatus = true;

}

public void StartGame()

{

// Set up initial positions for the pieces

for (int y = 0; y < 3; y++)

{

for (int x = (y % 2 == 0) ? 0 : 1; x < 8; x += 2)

{

Piece blackPiece = new Piece(x, y, "Black");

PlayerTwo.Pieces.Add(blackPiece);

Board.BoardArr[x, y].PlacePiece(blackPiece);

}

}

for (int y = 5; y < 8; y++)

{

for (int x = (y % 2 == 0) ? 0 : 1; x < 8; x += 2)

{

Piece whitePiece = new Piece(x, y, "White");

PlayerOne.Pieces.Add(whitePiece);

Board.BoardArr[x, y].PlacePiece(whitePiece);

}

}

Board.ShowBoard();

}

public void StartNewGame()

{

// Reset game

Board.InitBoard();

PlayerOne.Pieces.Clear();

PlayerTwo.Pieces.Clear();

StartGame();

}

public bool IsGame()

{

// Checks if game is still running (simplified)

return PlayerOne.Pieces.Count > 0 && PlayerTwo.Pieces.Count > 0;

}

}

// Entry point for the Checkers game

class Program

{

static void Main(string[] args)

{

Game checkersGame = new Game();

checkersGame.StartGame();

// Add more game logic such as player turns and moves here

}

}

}