ОГБПОУ «ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Работу выполнил:

Студент 682 группы

Мочалов Н.

Работу проверил:

Павлов В.С.

Томск, 2020

**Chess.cs:**

using System;

using System.Linq;

namespace Шахматы

{

public enum PlayerColor { White, Black }

public enum PlayerState { Idle, Holding, AwaitPromote, GameOver }

public enum PromoteOptions { Queen = 0, Rook = 1, Bishop = 2, Knight = 3 }

public class Chess{

public bool Running { private set; get; }

private PlayerState playerState;

private PromoteOptions promoteOption; //Выбранный в настоящее время вариант продвижения

private PlayerColor currentPlayer;// текущий игрок

private int cursorX, cursorY;// Координаты для курсора на доске

private Doska board;//доска

private Doska.Yacheika holdedNode = null; // ячейка текущего элемента

private Doska.Yacheika moveTo = null;// Куда двигаться

public Chess(){

Running = true;

board = new Doska();

currentPlayer = PlayerColor.White;//текущий игрок белый цвет, первый

turnStart();}

public void Update(){

if (Console.KeyAvailable){

ConsoleKeyInfo keyInfo = Console.ReadKey(true);//считка и передача то что нажимаем

if (keyInfo.Key == ConsoleKey.LeftArrow && cursorX > 0 && playerState != PlayerState.AwaitPromote)//если нажата стрелочка влево, ожидает перемещения

cursorX--;//по Х-1

else if (keyInfo.Key == ConsoleKey.RightArrow && cursorX < 7 && playerState != PlayerState.AwaitPromote)//если вправо нажата, ожидает

cursorX++;//по Х+1

else if (keyInfo.Key == ConsoleKey.UpArrow){//если вверх стрелочка нажата

if (playerState != PlayerState.AwaitPromote && cursorY < 7)

cursorY++;//по У+1

else if ((int)promoteOption > 0) promoteOption--;}

else if (keyInfo.Key == ConsoleKey.DownArrow){//если вниз стрелочка нажата

if (playerState != PlayerState.AwaitPromote && cursorY > 0)

cursorY--;

else if ((int)promoteOption < 3)

promoteOption++;}

else if (keyInfo.Key == ConsoleKey.Enter)

interact();//если энтер то делать

else if (keyInfo.Key == ConsoleKey.D)

debugInteract();//отладка если Д

else if (keyInfo.Key == ConsoleKey.Escape)

cancel();}}//если эскейп то закр

public void Draw(ConsoleGraphics g){//Рисует игру

g.FillArea(new CChar(' ', ConsoleColor.Black, ConsoleColor.DarkGray), 10, 5, 8, 8);//заполнение

for (int i = 0; i < 8; i++){

for (int j = 0; j < 8; j++){

Doska.Yacheika yacheika = board.GetCell(i, j);

if (yacheika.Piece != null){

g.DrawTransparent(yacheika.Piece.Char, (yacheika.Piece.Color == PlayerColor.White) ? ConsoleColor.White : ConsoleColor.Black, 10 + i, 5 + (7 - j));

if (yacheika.Piece.LegalMoves.Count == 0)

{

g.SetBackground(ConsoleColor.DarkRed, 10 + i, 5 + (7 - j));

}}

if (yacheika.HitBy.Contains(debugPiece))

g.SetBackground(ConsoleColor.DarkMagenta, 10 + i, 5 + (7 - j));}}

if (holdedNode != null && playerState == PlayerState.Holding){

foreach (Doska.Yacheika move in holdedNode.Piece.LegalMoves){// Подсветка возможных ходов

g.SetBackground(ConsoleColor.DarkGreen, 10 + move.X, 5 + (7 - move.Y));}}

g.SetBackground(ConsoleColor.DarkYellow, 10 + cursorX, 5 + (7 - cursorY));// курсор- желтый

for (int i = 0; i < 8; i++){

for (int j = 0; j < 8; j++){

if ((i + j) % 2 == 1) g.LightenBackground(10 + i, 5 + j);}}

if (playerState == PlayerState.AwaitPromote){// Меню параметров продвижения

g.DrawTextTrasparent("Dama", promoteOption == PromoteOptions.Queen ? ConsoleColor.Yellow : ConsoleColor.White, 22, 7);

g.DrawTextTrasparent("LAdiya", promoteOption == PromoteOptions.Rook ? ConsoleColor.Yellow : ConsoleColor.White, 22, 9);

g.DrawTextTrasparent("Slon", promoteOption == PromoteOptions.Bishop ? ConsoleColor.Yellow : ConsoleColor.White, 22, 11);

g.DrawTextTrasparent("Kon", promoteOption == PromoteOptions.Knight ? ConsoleColor.Yellow : ConsoleColor.White, 22, 13);}

else{

g.ClearArea(22, 7, 6, 7);}}//очистка чтобы не светилось

private void interact(){// kогда нажимает энтер

switch (playerState){

case PlayerState.Idle:

holdedNode = board.GetCell(cursorX, cursorY);

if (holdedNode.Piece == null || holdedNode.Piece.Color != currentPlayer || holdedNode.Piece.LegalMoves.Count == 0){

holdedNode = null;

return;}

else playerState = PlayerState.Holding;

break;

case PlayerState.Holding:

playerState = PlayerState.Holding;

moveTo = board.GetCell(cursorX, cursorY);

if (!holdedNode.Piece.LegalMoves.Contains(moveTo)){

moveTo = null;

return;}

if (board.IsPromotable(holdedNode, moveTo))

showPromote();

else

turnOver();

break;

case PlayerState.AwaitPromote:

turnOver();

break;

case PlayerState.GameOver:

Running = false;

break;}}

private Piece debugPiece;

private void debugInteract() { debugPiece = board.GetCell(cursorX, cursorY).Piece; }

private void cancel(){ //если esc

playerState = PlayerState.Idle;

holdedNode = null;}

// Вызывается при каждом ходе

private void turnStart() { board.TurnStart(currentPlayer); }

private void showPromote(){

playerState = PlayerState.AwaitPromote;

promoteOption = PromoteOptions.Queen;}

private void turnOver(){// Вызывается, когда ход передается другому игроку

board.Move(holdedNode, moveTo, promoteOption);

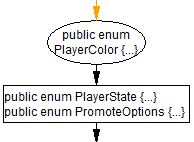
holdedNode = null;

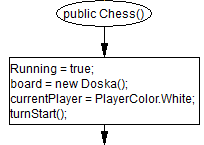
moveTo = null;

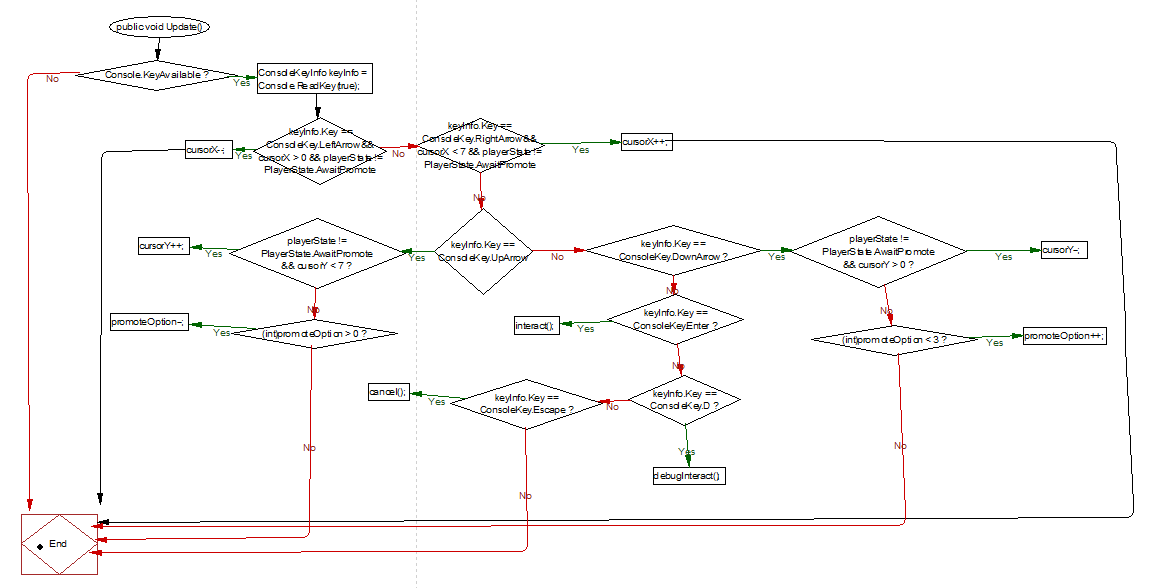
playerState = PlayerState.Idle;

currentPlayer = currentPlayer == PlayerColor.White ? PlayerColor.Black : PlayerColor.White;

turnStart();}}}







**Doska.cs:**

using Шахматы.Pieces;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

namespace Шахматы{

public class Doska{// Структура и алгоритмы шахматной доски

public class Yacheika{//Представляет одну шахматную ячейку

public Doska Parent { private set; get; }

public int X, Y;

public Piece Piece;

public List<Piece> HitBy;// Все, кто могут поразить эту клетку.

public Yacheika(Doska parent, int x, int y){

Parent = parent;

HitBy = new List<Piece>();

X = x;

Y = y;}

public IEnumerable<Yacheika> OpenLineOfSight(int dirX, int dirY, int desiredCount = 1){

for (int i = 1; i <= desiredCount; i++){

//Запрос элемента для ячейки, если пустая ячейка отсутствует, и мы должны остановиться

Yacheika yacheika = Parent.GetCell(X + dirX \* i, Y + dirY \* i);

if (yacheika == null) yield break;

yield return yacheika;

//Остановиться в любом случае, так как линия заблокирована

if (yacheika.Piece != null)

yield break;}}

public Yacheika Open(int x, int y){

//Запрос элемента для ячейки, если пустая ячейка отсутствует, и мы не должны возвращать

Yacheika yacheika = Parent.GetCell(X + x, Y + y);

return yacheika ?? null;}}

private Yacheika[,] cells; //информация о ячейках и связях между ними

public Yacheika Proxod { private set; get; }

public Yacheika Proxod1 { private set; get; } //Ячейка, в которой пешка будет захвачена

private List<Piece> pieces = new List<Piece>();

private Piece blackKing;

private Piece whiteKing;

private bool inCheck;

public Doska() { Reset(); }

public Yacheika GetCell(int x, int y){// Получить ячейку по абсолютным координатам

if (x < 0 || cells.GetLength(0) <= x || y < 0 || cells.GetLength(1) <= y) return null;

return cells[x, y];}

private void addPiece(Yacheika yacheika, Piece piece){// Добавляет фигуру в начале игры

yacheika.Piece = piece;

pieces.Add(piece);

piece.OnPlace(yacheika);}

public void Reset(){//Сбрoc

cells = new Yacheika[8, 8];

for (int i = 0; i < 8; i++){

for (int j = 0; j < 8; j++){

cells[i, j] = new Yacheika(this, i, j);}}

pieces.Clear();

Proxod = null;

Proxod1 = null;

addPiece(cells[0, 0], new Ladiya(PlayerColor.White));

addPiece(cells[1, 0], new Kon(PlayerColor.White));

addPiece(cells[2, 0], new Slon (PlayerColor.White));

addPiece(cells[3, 0], new Dama (PlayerColor.White));

addPiece(cells[4, 0], (whiteKing = new Korol(PlayerColor.White)));

addPiece(cells[5, 0], new Slon(PlayerColor.White));

addPiece(cells[6, 0], new Kon(PlayerColor.White));

addPiece(cells[7, 0], new Ladiya(PlayerColor.White));

addPiece(cells[0, 1], new Peshka(PlayerColor.White));

addPiece(cells[1, 1], new Peshka(PlayerColor.White));

addPiece(cells[2, 1], new Peshka(PlayerColor.White));

addPiece(cells[3, 1], new Peshka(PlayerColor.White));

addPiece(cells[4, 1], new Peshka(PlayerColor.White));

addPiece(cells[5, 1], new Peshka(PlayerColor.White));

addPiece(cells[6, 1], new Peshka(PlayerColor.White));

addPiece(cells[7, 1], new Peshka(PlayerColor.White));

addPiece(cells[0, 6], new Peshka(PlayerColor.Black));

addPiece(cells[1, 6], new Peshka(PlayerColor.Black));

addPiece(cells[2, 6], new Peshka(PlayerColor.Black));

addPiece(cells[3, 6], new Peshka(PlayerColor.Black));

addPiece(cells[4, 6], new Peshka(PlayerColor.Black));

addPiece(cells[5, 6], new Peshka(PlayerColor.Black));

addPiece(cells[6, 6], new Peshka(PlayerColor.Black));

addPiece(cells[7, 6], new Peshka(PlayerColor.Black));

addPiece(cells[0, 7], new Ladiya(PlayerColor.Black));

addPiece(cells[1, 7], new Kon(PlayerColor.Black));

addPiece(cells[2, 7], new Slon(PlayerColor.Black));

addPiece(cells[3, 7], new Dama(PlayerColor.Black));

addPiece(cells[4, 7], (blackKing = new Korol(PlayerColor.Black)));

addPiece(cells[5, 7], new Slon(PlayerColor.Black));

addPiece(cells[6, 7], new Kon(PlayerColor.Black));

addPiece(cells[7, 7], new Ladiya(PlayerColor.Black));

foreach (Piece piece in pieces) { piece.Recalculate(); }}

public bool TurnStart(PlayerColor currentPlayer) {

inCheck = IsInCheck(currentPlayer, false);

bool anyLegalMove = false;

foreach (Yacheika yacheika in cells) {// Очистить списки попаданий в ячейки

yacheika.HitBy.Clear(); }

foreach (Piece piece in pieces){// Пересчитаем возможные ходы и списки попаданий для ячеек

piece.Recalculate();}

foreach (Piece piece in pieces){

piece.LegalMoves.Clear();

foreach (Yacheika move in piece.PossibleMoves){

if (piece.Color == currentPlayer && isMoveLegal(piece, move)){

piece.LegalMoves.Add(move);

anyLegalMove = true;}}}

return anyLegalMove;}

private bool isMoveLegal(Piece piece, Yacheika move){//Подтверждает, если ход является доступным для данной фигуры

Piece currentKing = piece.Color == PlayerColor.White ? whiteKing : blackKing;

if (piece is Korol){

foreach (Piece hitter in move.HitBy){

if (hitter.Parent != move && hitter.Color != piece.Color)

return false;}

// Подтвердить рокировку

if (Math.Abs(move.X - piece.Parent.X) == 2){

if (inCheck)

return false;

foreach (Piece hitter in GetCell(move.X > piece.Parent.X ? move.X - 1 : move.X + 1, move.Y).HitBy){

if (hitter.Color != piece.Color)

return false;}}}

else {// Некороль

if (inCheck){

foreach (Piece hitter in currentKing.Parent.HitBy){

if (hitter.Color == currentKing.Color) continue;

if (hitter.Parent == move) continue;

if (hitter.IsBlockedIfMove(piece.Parent, move, currentKing.Parent)) continue;

return false;}}

foreach (Piece hitter in piece.Parent.HitBy){

if (hitter.Color == currentKing.Color) continue;

if (hitter.Parent == move) continue;

if (!hitter.IsBlockedIfMove(piece.Parent, move, currentKing.Parent))

return false;}}

return true;}

public bool IsInCheck(PlayerColor player, bool useCache = true){// Проверяет, его король может быть поражен?

if (useCache)

return inCheck;

if (player == PlayerColor.White)

return whiteKing.Parent.HitBy.Any(hitter => hitter.Color != player);

else

return blackKing.Parent.HitBy.Any(hitter => hitter.Color != player);}

// Перемещаем фигуру из одной ячейки в другую, после того как эта функция будет вызвана

public void Move(Yacheika from, Yacheika to, PromoteOptions promoteOption){

if (to.Piece != null)

pieces.Remove(to.Piece);

to.Piece = from.Piece;

from.Piece = null;

if (to == Proxod && to.Piece is Peshka){

pieces.Remove(Proxod1.Piece);

Proxod1.Piece = null;}

if (to.Piece is Korol && to.X - from.X == 2) {// Рокировка вправо

Move(GetCell(7, to.Y), GetCell(to.X - 1, to.Y), promoteOption); }// Перемещаем ладью

if (to.Piece is Korol && to.X - from.X == -2){// Рокировка влево{

Move(GetCell(0, to.Y), GetCell(to.X + 1, to.Y), promoteOption);}// Перемещаем ладью

if (to.Piece is Peshka && to.Y == (to.Piece.Color == PlayerColor.White ? 7 : 0)){

Piece promoted = null;

switch (promoteOption){

case PromoteOptions.Queen:

promoted = new Dama(to.Piece);

break;

case PromoteOptions.Rook:

promoted = new Ladiya(to.Piece);

break;

case PromoteOptions.Bishop:

promoted = new Slon(to.Piece);

break;

case PromoteOptions.Knight:

promoted = new Kon(to.Piece);

break;}

//Обновляем список

pieces.Remove(to.Piece);

to.Piece = promoted;

promoted.OnPlace(to);

pieces.Add(promoted);}

// Пересчитаем возможные ходы

to.Piece.OnMove(to);

to.Piece.Recalculate();

Proxod = null;

Proxod1 = null;

if (to.Piece is Peshka && Math.Abs(to.Y - from.Y) == 2){

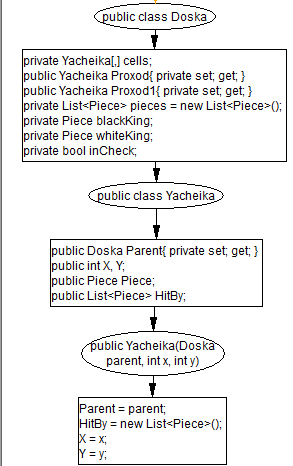
Proxod = GetCell(to.X, (from.Y > to.Y) ? from.Y - 1 : from.Y + 1);

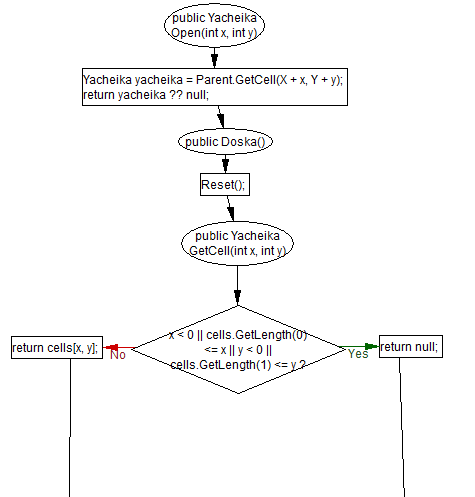
Proxod1 = to;}}

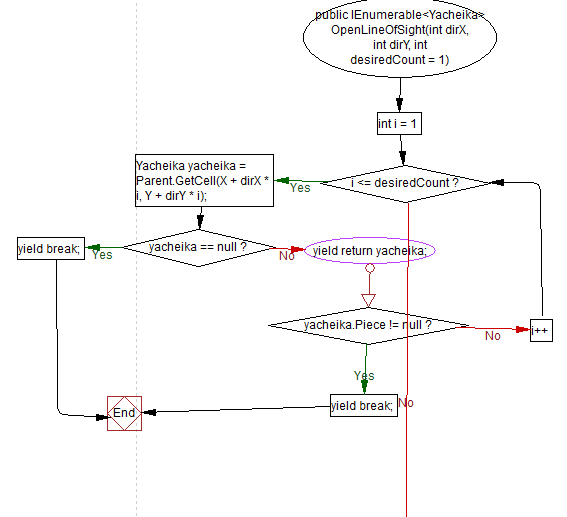
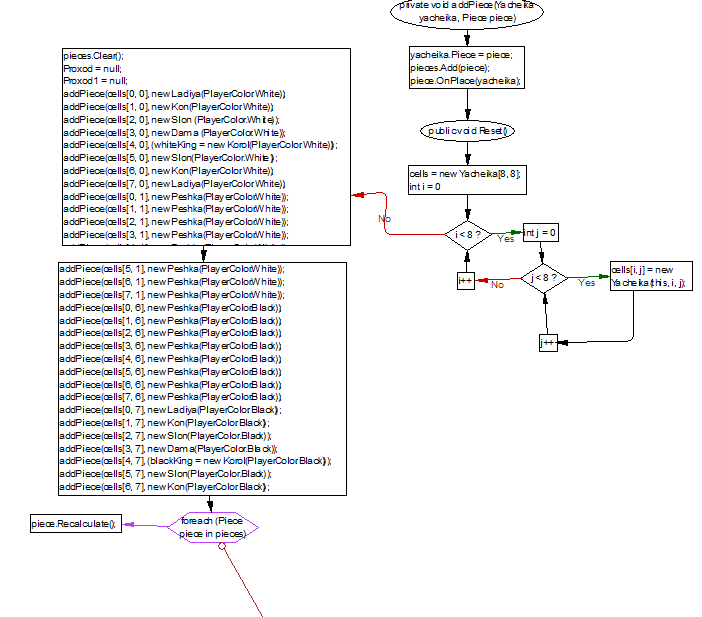
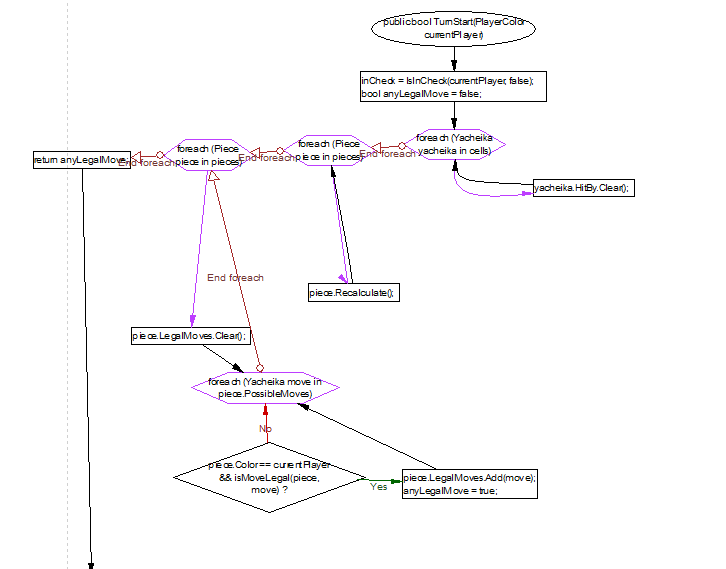
// Продвигается ли фигура ?

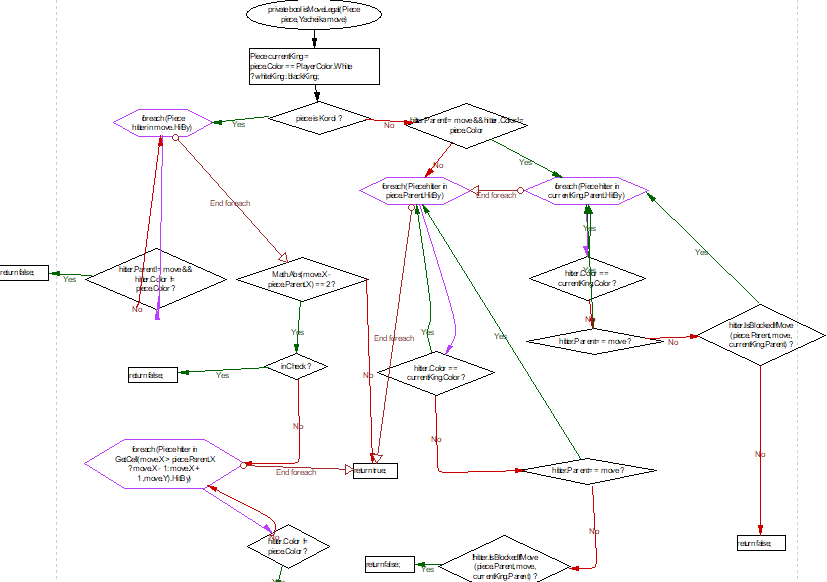
public bool IsPromotable(Yacheika from, Yacheika to){

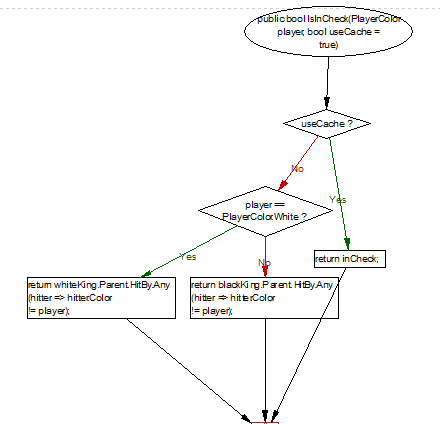
return from.Piece is Peshka && to.Y == (from.Piece.Color == PlayerColor.White ? 7 : 0);}}}

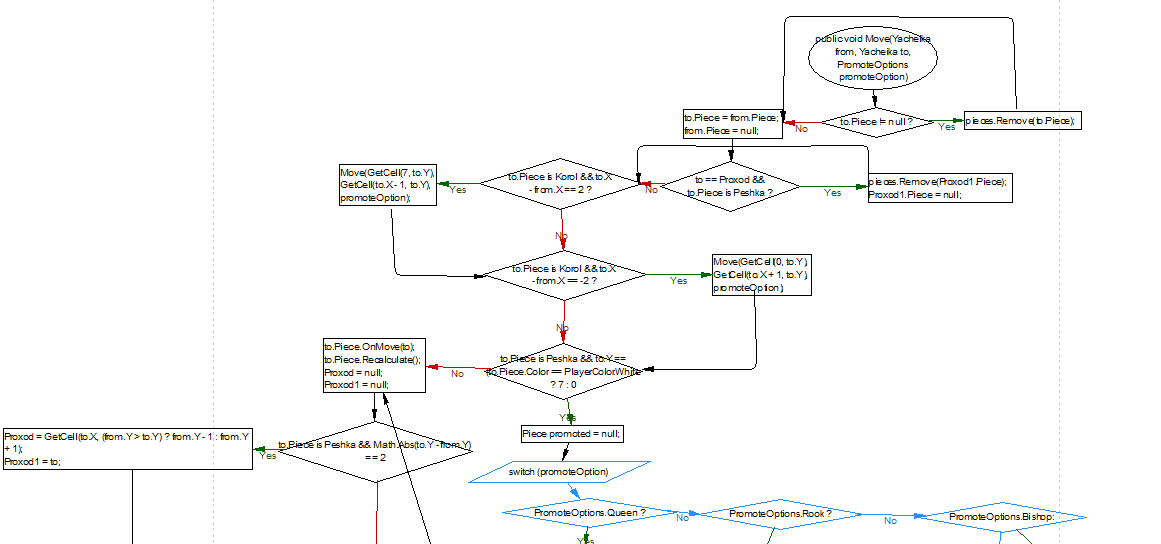


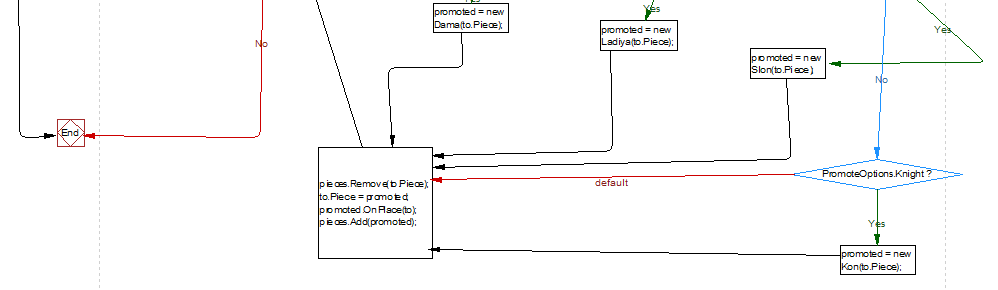


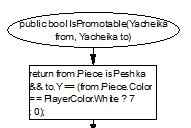
  











**ConsoleGraphics.cs:**

using System;

namespace Шахматы

{

public struct CChar : IEquatable<CChar>

{

public ConsoleColor Foreground;

public ConsoleColor Background;

public char C;

public CChar(char c = ' ', ConsoleColor foreground = ConsoleColor.White, ConsoleColor background = ConsoleColor.Black)

{

Foreground = foreground;

Background = background;

C = c;

}

public bool Equals(CChar other) { return other.Foreground == Foreground && other.Background == Background && other.C == C; }

public static bool operator ==(CChar lhs, CChar rhs) { return lhs.Equals(rhs); }

public static bool operator !=(CChar lhs, CChar rhs) { return !lhs.Equals(rhs); }

}

public class ConsoleGraphics

{

private CChar[,] backBuffer;

private CChar[,] frontBuffer;

public ConsoleGraphics()

{

backBuffer = new CChar[Console.BufferWidth, Console.BufferHeight];

frontBuffer = new CChar[Console.BufferWidth, Console.BufferHeight];

}

public void Clear()//Очищаем буфер

{

for (int i = 0; i < backBuffer.GetLength(0); i++)

{

for (int j = 0; j < backBuffer.GetLength(1); j++)

{

backBuffer[i, j] = new CChar();

}

}

}

public void Draw(CChar cchar, int x, int y)//Рисует цветной символ в буфере

{

backBuffer[x, y] = cchar;

}

public void DrawTransparent(char c, ConsoleColor foreground, int x, int y)//Рисует цветной символ в буфере, он не меняет цвет фона

{

backBuffer[x, y].C = c;

backBuffer[x, y].Foreground = foreground;

}

public void DrawArea(CChar[,] cchars, int x, int y)//Рисует область цветных символов. Длина массивов используется как ширина и высота области

{

for (int i = 0; i < cchars.GetLength(0); i++)

{

for (int j = 0; j < cchars.GetLength(1); j++)

{

backBuffer[x + i, y + j] = cchars[i, j];

}

}

}

public void DrawText(string text, ConsoleColor foreground, ConsoleColor background, int x, int y)// Рисует текст на экране.

{

CChar[,] area = new CChar[text.Length, 1];

for (int i = 0; i < text.Length; i++)

{

area[i, 0] = new CChar(text[i], foreground, background);

}

DrawArea(area, x, y);

}

public void DrawTextTrasparent(string text, ConsoleColor foreground, int x, int y)// Рисует текст на экране с прозрачным фоном

{

CChar[,] area = new CChar[text.Length, 1];

for (int i = 0; i < text.Length; i++)

{

area[i, 0] = new CChar(text[i], foreground, backBuffer[x + i, y].Background);

}

DrawArea(area, x, y);

}

public void FillArea(CChar cchar, int x, int y, int width, int height)// Заполняет область буфера одним цветным символом

{

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

{

backBuffer[x + i, y + j] = cchar;

}

}

}

public void ClearArea(int x, int y, int width, int height)// Очищает область на экране

{

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

{

backBuffer[x + i, y + j] = new CChar();

}

}

}

public void DarkenBackground(int x, int y)//Если фоновый цвет уже темный или нет темной версии, он остается неизменным.

{

switch (backBuffer[x, y].Background)

{

case ConsoleColor.Blue:

backBuffer[x, y].Background = ConsoleColor.DarkBlue;

break;

case ConsoleColor.Green:

backBuffer[x, y].Background = ConsoleColor.DarkGreen;

break;

case ConsoleColor.Yellow:

backBuffer[x, y].Background = ConsoleColor.DarkYellow;

break;

case ConsoleColor.Magenta:

backBuffer[x, y].Background = ConsoleColor.DarkMagenta;

break;

case ConsoleColor.Gray:

backBuffer[x, y].Background = ConsoleColor.DarkGray;

break;

case ConsoleColor.Cyan:

backBuffer[x, y].Background = ConsoleColor.DarkCyan;

break;

case ConsoleColor.Red:

backBuffer[x, y].Background = ConsoleColor.DarkRed;

break;

}

}

public void DarkenForeground(int x, int y)

{

switch (backBuffer[x, y].Foreground)

{

case ConsoleColor.Blue:

backBuffer[x, y].Foreground = ConsoleColor.DarkBlue;

break;

case ConsoleColor.Green:

backBuffer[x, y].Foreground = ConsoleColor.DarkGreen;

break;

case ConsoleColor.Yellow:

backBuffer[x, y].Foreground = ConsoleColor.DarkYellow;

break;

case ConsoleColor.Magenta:

backBuffer[x, y].Foreground = ConsoleColor.DarkMagenta;

break;

case ConsoleColor.Gray:

backBuffer[x, y].Foreground = ConsoleColor.DarkGray;

break;

case ConsoleColor.Cyan:

backBuffer[x, y].Foreground = ConsoleColor.DarkCyan;

break;

case ConsoleColor.Red:

backBuffer[x, y].Foreground = ConsoleColor.DarkRed;

break;

}

}

// Осветляет цвет фона цветного символа в буфере.Если фоновый цвет уже светлый или нет светлой версии, он остается неизменным

public void LightenBackground(int x, int y)

{

switch (backBuffer[x, y].Background)

{

case ConsoleColor.DarkBlue:

backBuffer[x, y].Background = ConsoleColor.Blue;

break;

case ConsoleColor.DarkGreen:

backBuffer[x, y].Background = ConsoleColor.Green;

break;

case ConsoleColor.DarkYellow:

backBuffer[x, y].Background = ConsoleColor.Yellow;

break;

case ConsoleColor.DarkMagenta:

backBuffer[x, y].Background = ConsoleColor.Magenta;

break;

case ConsoleColor.DarkGray:

backBuffer[x, y].Background = ConsoleColor.Gray;

break;

case ConsoleColor.DarkCyan:

backBuffer[x, y].Background = ConsoleColor.Cyan;

break;

case ConsoleColor.DarkRed:

backBuffer[x, y].Background = ConsoleColor.Red;

break;

}

}

public void LightenForeground(int x, int y)

{

switch (backBuffer[x, y].Foreground)

{

case ConsoleColor.DarkBlue:

backBuffer[x, y].Foreground = ConsoleColor.Blue;

break;

case ConsoleColor.DarkGreen:

backBuffer[x, y].Foreground = ConsoleColor.Green;

break;

case ConsoleColor.DarkYellow:

backBuffer[x, y].Foreground = ConsoleColor.Yellow;

break;

case ConsoleColor.DarkMagenta:

backBuffer[x, y].Foreground = ConsoleColor.Magenta;

break;

case ConsoleColor.DarkGray:

backBuffer[x, y].Foreground = ConsoleColor.Gray;

break;

case ConsoleColor.DarkCyan:

backBuffer[x, y].Foreground = ConsoleColor.Cyan;

break;

case ConsoleColor.DarkRed:

backBuffer[x, y].Foreground = ConsoleColor.Red;

break;

}

}

public void SetBackground(ConsoleColor color, int x, int y)//Устанавливает цвет фона буфера в(x, y)

{

backBuffer[x, y].Background = color;

}

public ConsoleColor GetBackground(int x, int y)//Получает цвет фона буфера в(x, y)

{

return backBuffer[x, y].Background;

}

public void SetForeground(ConsoleColor color, int x, int y)

{

backBuffer[x, y].Foreground = color;

}

public ConsoleColor GetForeground(int x, int y)

{

return backBuffer[x, y].Foreground;

}

/// Перезаписывает фронт и перерисовывает символ, если он отличается от бэка

public void SwapBuffers()

{

for (int i = 0; i < backBuffer.GetLength(0); i++)

{

for (int j = 0; j < backBuffer.GetLength(1); j++)

{

if (frontBuffer[i, j] != backBuffer[i, j])

{

Console.SetCursorPosition(i, j);

Console.ForegroundColor = backBuffer[i, j].Foreground;

Console.BackgroundColor = backBuffer[i, j].Background;

Console.Write(backBuffer[i, j].C);

frontBuffer[i, j] = backBuffer[i, j];

}

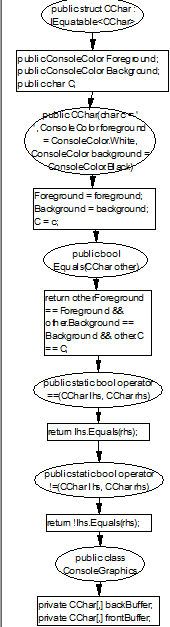
}

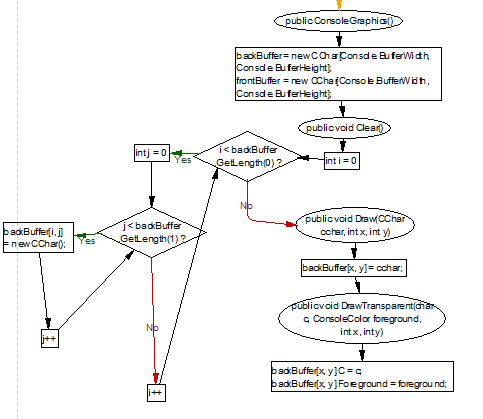
}

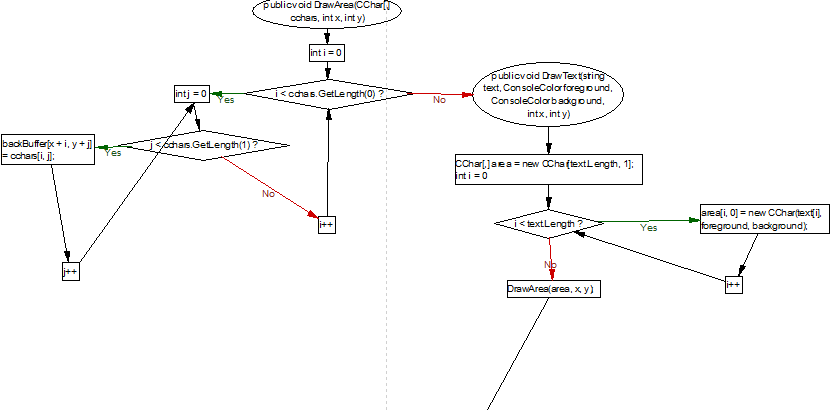
}

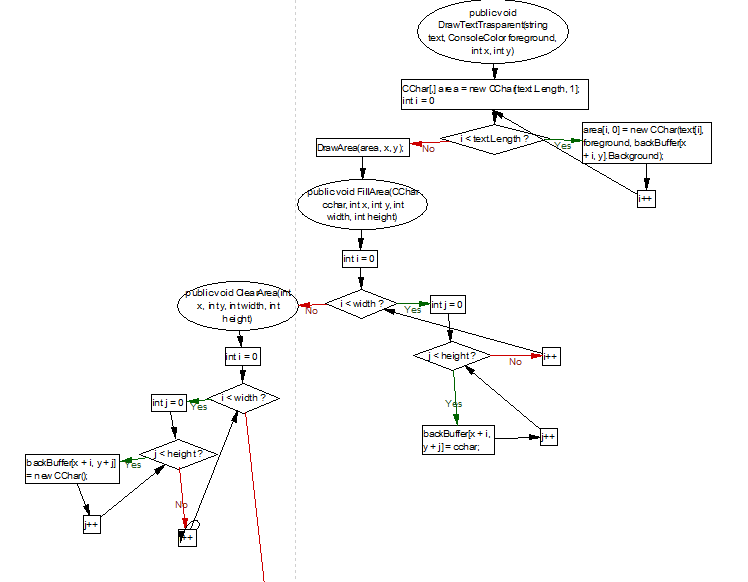
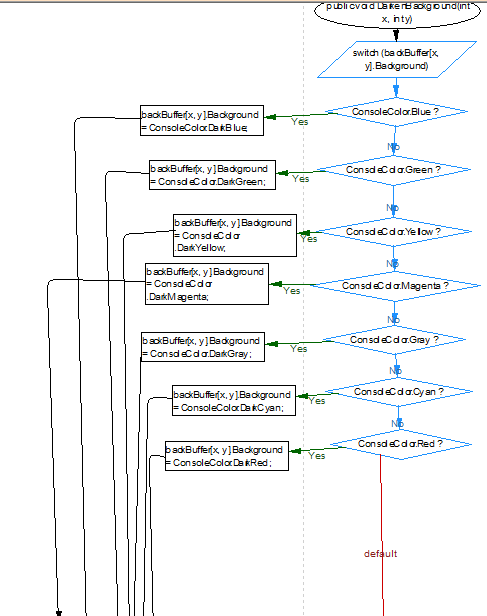
}

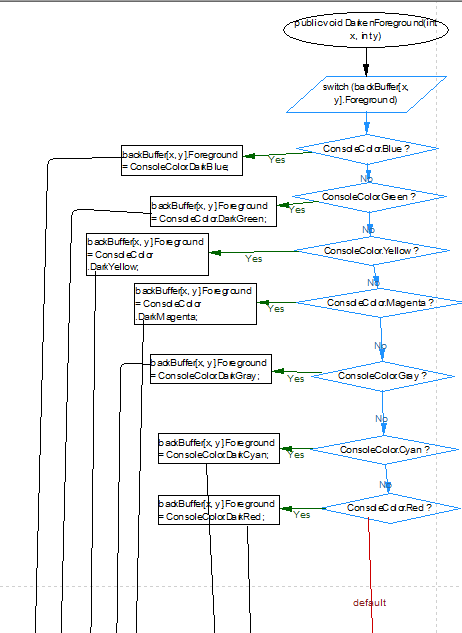
}

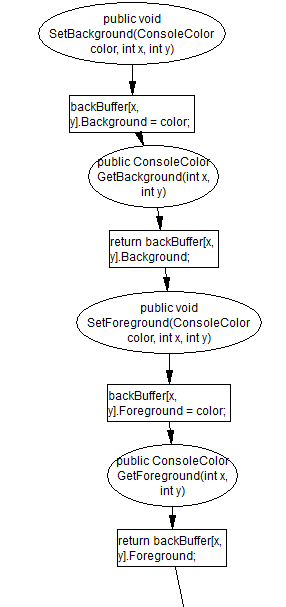
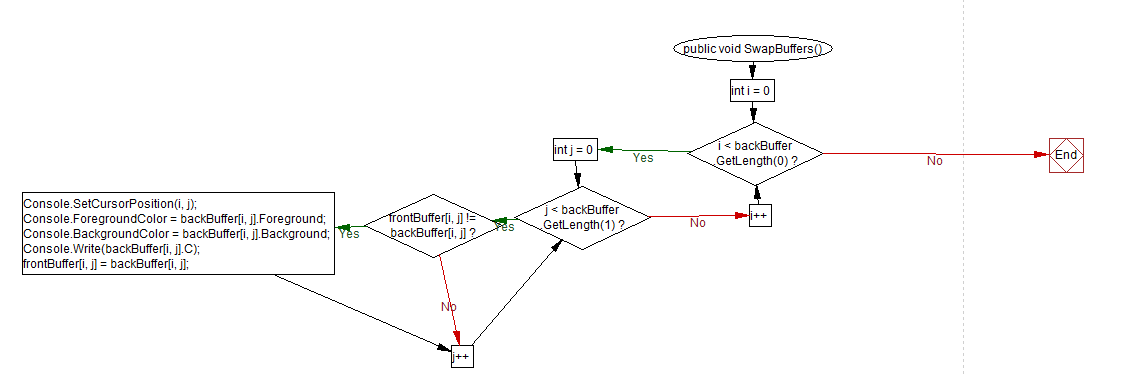








**Direction.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Шахматы

{

public class Napravlenie//Все возможные ходы и проверка линии хода

{

public Piece Piece { private set; get; }/// Часть на которую может двигаться

public int X { private set; get; }

public int Y { private set; get; }

private List<Doska.Yacheika> possibleMoves;//То, куда можно сходить, шаги, которые можно сделать в этом направлении

public IEnumerable<Doska.Yacheika> GetPossibleMoves(bool enemyHittable = true)

{

if (possibleMoves.Count == 0)

yield break; //указывает, что последовательность больше не имеет элементов.

for (int i = 0; i < possibleMoves.Count - 1; i++)

{

yield return possibleMoves[i];//определяет возвращаемый элемент

}

if (possibleMoves.Last().Piece == null)

yield return possibleMoves.Last();//определяет возвращаемый элемент

else if (enemyHittable && possibleMoves.Last().Piece.Color != Piece.Color)

yield return possibleMoves.Last();//определяет возвращаемый элемент

}

public int GetPossibleMoveCount(bool enemyHittable = true)/// Количество возможных ходов

{

if (possibleMoves.Count == 0)

return 0;

if (possibleMoves.Last().Piece == null)

return possibleMoves.Count;

else if (!enemyHittable || possibleMoves.Last().Piece.Color == Piece.Color)

return possibleMoves.Count - 1;

else

return possibleMoves.Count;

}

public int DesiredCount { private set; get; }

private bool updateHitGraph;

public Napravlenie(Piece piece, int x, int y, int desiredCount = 8, bool updateHitGraph = true)

{

Piece = piece;

X = x;

Y = y;

DesiredCount = desiredCount;

this.updateHitGraph = updateHitGraph;

possibleMoves = new List<Doska.Yacheika>();

possibleMoves.AddRange(piece.Parent.OpenLineOfSight(x, y, desiredCount));

foreach (Doska.Yacheika move in possibleMoves)

{

if (updateHitGraph)

move.HitBy.Add(Piece);

}

}

public bool IsBlockedIfMove(Doska.Yacheika from, Doska.Yacheika to, Doska.Yacheika blocked)

{

if (possibleMoves.Contains(blocked) && !possibleMoves.Contains(to))

{

return false;

}

else if (possibleMoves.Contains(from))

{

int toIndex = possibleMoves.IndexOf(to);

if (0 <= toIndex && toIndex < possibleMoves.Count - 1)

return true;

else

{

// Если мы двинулись дальше

foreach (Doska.Yacheika move in from.OpenLineOfSight(X, Y, DesiredCount - possibleMoves.Count))

{

if (move == to) //перешел на новый путь

return true;

if (move == blocked) //заблокирован

return false;

}

}

}

return true;

}

}

}

