Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проектированию

по курсу «Программирование на языке *Java*»

на тему «Разработка многомодульного приложения на языке *Java*»

Выполнил:

студент группы 22ВВП1

Перфилов А.В.

Принял:

Карамышева Н.В.

Пенза 2025

СОДЕРЖАНИЕ

[РЕФЕРАТ 5](#_Toc200300646)

[ВВЕДЕНИЕ 6](#_Toc200300647)

[1 ПОСТАНОВКА ЗАДАЧИ 7](#_Toc200300648)

[2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 8](#_Toc200300649)

[3 ОПИСАНИЕ ПРОГРАММЫ 9](#_Toc200300650)

[3.1 Описание выбора архитектуры приложения. 9](#_Toc200300651)

[3.2 Описание программы Клиента 9](#_Toc200300652)

[3.3 Описание программы Сервера 9](#_Toc200300653)

[3.4 Описание программы Шахмат 10](#_Toc200300654)

[4 ОТЛАДКА И ТЕСТИРОВАНИЕ 11](#_Toc200300655)

[4.1 Отладка с помощью средств, предоставляемых *NetBeans IDE* 24 11](#_Toc200300656)

[4.2 Тестирование пользовательского интерфейса 12](#_Toc200300657)

[ЗАКЛЮЧЕНИЕ 13](#_Toc200300658)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 14](#_Toc200300659)

[ПРИЛОЖЕНИE A – UML-диаграмма вариантов использования приложения 15](#_Toc200300660)

[ПРИЛОЖЕНИE B – UML-диаграмма деятельности 16](#_Toc200300661)

[ПРИЛОЖЕНИE C – UML-диаграмма классов и UML-диаграмма развертывания 17](#_Toc200300662)

[ПРИЛОЖЕНИE D UML-диаграмма развертывания 18](#_Toc200300663)

[ПРИЛОЖЕНИE E – Класс Board 19](#_Toc200300664)

[ПРИЛОЖЕНИE F – Программа сканера событий 25](#_Toc200300665)

[ПРИЛОЖЕНИE G – Программа главного окна 29](#_Toc200300666)

[ПРИЛОЖЕНИE H – Программа клиента 39](#_Toc200300667)

[ПРИЛОЖЕНИE I – Программа сервера 41](#_Toc200300668)

[ПРИЛОЖЕНИE J – Программа для отслеживания движений мышки 47](#_Toc200300669)

# РЕФЕРАТ

Пояснительная записка 49 с., 7 рис., 4 источника, 10 приложений

ФАЙЛОВАЯ СИСТЕМА, ОПЕРАЦИИ РАБОТЫ НАД ФАЙЛАМИ,   
ФАЙЛОВЫЙ МЕНЕДЖЕР, ДВУПАНЕЛЬНЫЙ ФАЙЛОВЫЙ МЕНЕДЖЕР

Целью проекта является создание программы, предоставляющей интерфейс для игры в шахматы.

В ходе работы был проведен анализ предметной области, установлены задачи проекта для достижения поставленной цели.

Результатом проведенной работы является создание функционирующего приложения «Шахматы», которое выполняет требуемый круг задач.

# ВВЕДЕНИЕ

**Приложения для игры в шахматы** — это специальные программы, позволяющие пользователям играть в шахматы против компьютера или других людей. С помощью таких приложений можно не только играть партии, но и анализировать ходы, обучаться стратегиям, решать шахматные задачи и отслеживать прогресс. Одной из первых популярных шахматных программ была Chessmaster, выпущенная для персональных компьютеров ещё в 1980-х годах. Она отличалась простым интерфейсом и возможностью играть с различными уровнями сложности.

Во многих современных шахматных приложениях реализовано разделение интерфейса на несколько ключевых областей: доска с фигурами, список сделанных ходов, таймер и панель с анализом партии. Приложение Lichess, к примеру, стало основоположником класса современных онлайн-шахматных платформ. К числу популярных шахматных приложений относятся также Chess.com, Play Magnus, Stockfish, и другие.

Современные шахматные приложения часто включают поддержку горячих клавиш, множество обучающих материалов и поддержку подключаемых модулей. Некоторые пользователи предпочитают использовать приложения с мощными движками, такими как Stockfish, за счёт их способности глубоко анализировать позиции и предлагать оптимальные ходы. Шахматные программы остаются востребованными как среди новичков, так и среди профессиональных игроков, предлагая мощные инструменты для обучения и развития навыков.

# 1 ПОСТАНОВКА ЗАДАЧИ

Целью работы является разработка приложения «Шахматы».

Разрабатываемая программа должна предоставлять следующие возможности:

* игры по локальной сети;
* игры на одном компьютере;

Приложение должно обладать графическим интерфейсом и использовать следующие технологии:

* Java Collections Framework ;
* механизм обработки исключительных ситуаций;
* Java Stream API;
* Java Multithreading;
* сетевое взаимодействие.

Диаграмма вариантов использования приложения приведена в приложении А.

# 2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

**Шахматное приложение** — это компьютерная программа, предоставляющая пользователю интерфейс для игры, анализа и обучения шахматам.

Игровой интерфейс чаще всего включает в себя шахматную доску и боковую панель с дополнительной информацией: список ходов, таймер, нотация, кнопки управления. Структура шахматных баз данных может отображаться в виде дерева, классифицированного по турнирным событиям, дебютам или игрокам.

Термин, который предшествовал понятию шахматного приложения, — это шахматная программа. Одной из ранних шахматных программ была **Mac Hack**, разработанная в MIT в 1960-х годах, а позже — известный **Chessmaster**, ориентированный на массового пользователя.

Современные шахматные приложения поддерживают множество функций управления партиями: возможность делать ходы с помощью мыши или сенсорного экрана, использование горячих клавиш, управление ходами с помощью кнопок «отменить» и «повторить», а также функцию перетаскивания фигур. Часто присутствует меню выбора уровня сложности, шахматного движка и других параметров игры.

Во время анализа партии приложение может показывать текущую позицию, график оценки позиции, лучшие ходы по мнению движка, глубину анализа, а также варианты, предложенные ИИ. Есть возможность приостановить анализ и продолжить его позже, не теряя прогресса.

Если пользователь делает неоднозначный или некорректный ход, приложение может предложить варианты: откатить ход, показать подсказку или включить учебный режим. Поддержка PGN-файлов позволяет сравнивать партии по позициям или ходам, но при большом объёме данных это может замедлить обработку.

# 3 ОПИСАНИЕ ПРОГРАММЫ

## 3.1 Описание выбора архитектуры приложения.

Для организации клиент-серверного взаимодействия используется соединение TCP. Основные функции с логикой шахмат реализованы в классе Board. Для графической реализации был выбран класс JFame. В *Netbeans IDE* есть возможность организации JFrame класса с помощью графического интерфейса.

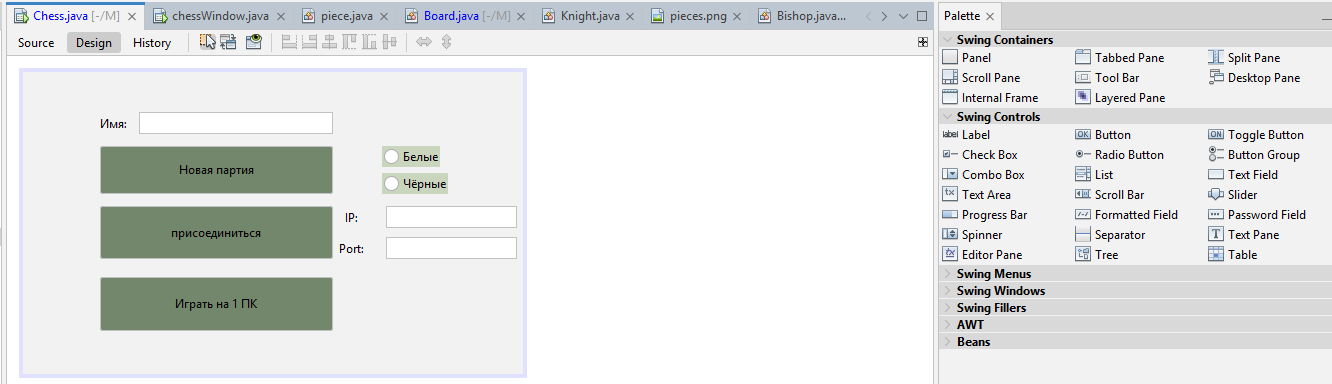


Рисунок 1 **–** Создание оконного интерфейса.

## 3.2 Описание программы Клиента

Клиентская программа создает сокет на своем конце связи и пытается подключить этот сокет к серверу. Для взаимодействия на стороне Клиента используется класс *java.net.Socket*, представляющий собой сокет.

this.socket = new Socket(host, port);

После того, как соединения установлены, связь происходит с использованием потоков входных/выходных данных. Каждый сокет имеет и OutputStream (поток выходных данных), и InputStream (поток входных данных). OutputStream клиента подключен к InputStream сервера, а InputStream клиента подключен к OutputStream сервера.

Для создания функции просмотра файла необходимо организовать между Клиентом и Сервером пересылку байт. Поэтому Для чтения байтовых данных (не строк) применяется класс *DataInputStream*.

DataInputStream in = new DataInputStream(clientSocket.getInputStream());

DataOutputStream out = new DataOutputStream(clientSocket.getOutputStream());

## 3.3 Описание программы Сервера

На стороне Сервера используется класс *java.net.ServerSocket*, предоставляющий механизм серверной программы для прослушивания клиентов и установления соединений с ними. Сервер создает экземпляр объекта *ServerSocket*, определяющий, по какому номеру порта должна происходить связь. В бесконечном цикле осуществляется прослушивание наличия подключения нового клиента

this.serverSocket = new ServerSocket(port);

clientSocket = serverSocket.accept();

System.out.println("Player connected!");

out = new ObjectOutputStream(clientSocket.getOutputStream());

in = new ObjectInputStream(clientSocket.getInputStream());

## 3.4 Описание программы Шахмат

После создания окна и при каждом ходе доска со всеми фигурами отрисовывается заного (рис.2).

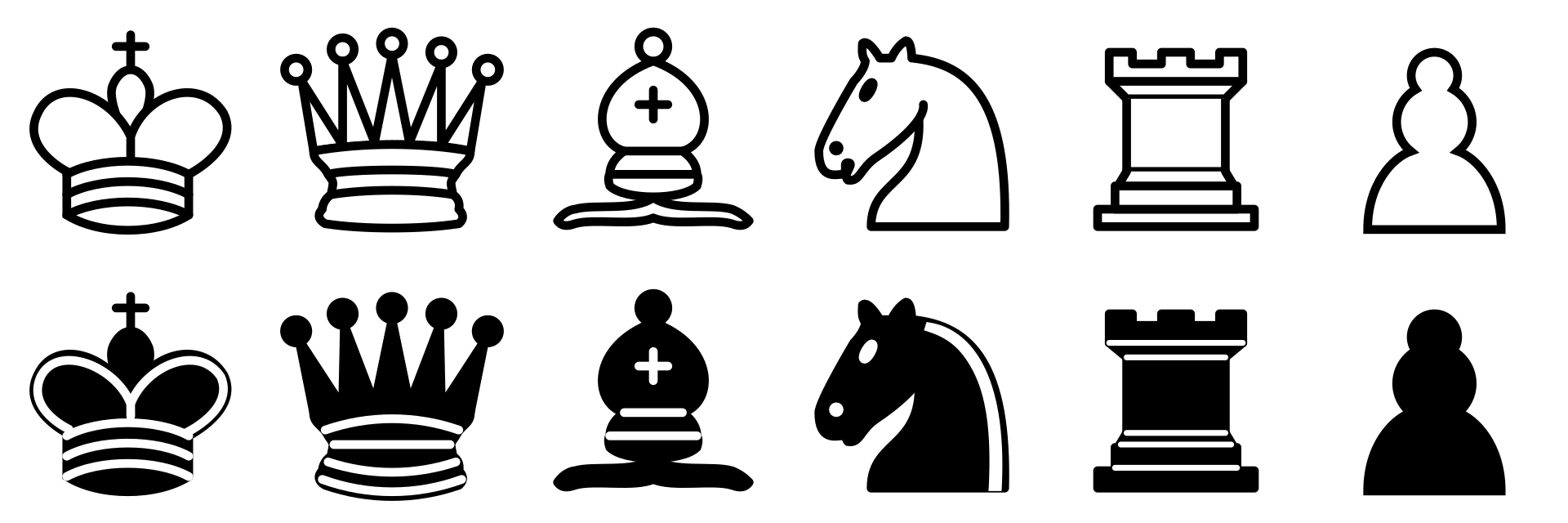


Рисунок 2 – фигуры

Для взаимодействия с фигурами используется библиотека MouseAdapter.

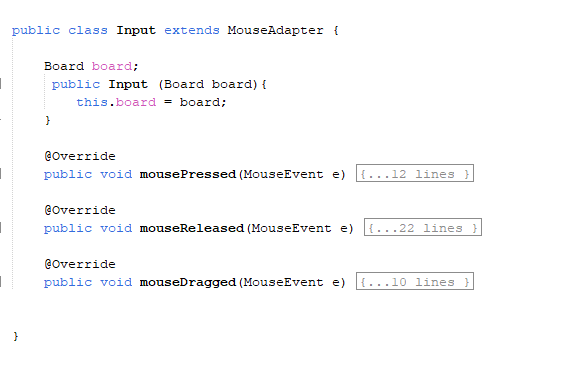


Рисунок 3 – функции для отслеживания мышки

# 4 ОТЛАДКА И ТЕСТИРОВАНИЕ

## 4.1 Отладка с помощью средств, предоставляемых *NetBeans IDE* 24

В качестве среды разработки была выбрана программа *NetBeans IDE* 24. Для отладки использовались несколько возможностей *NetBeans IDE*: точка останова, трассировка, анализ содержимого переменных. Тестирование проводилось во время разработки и также после завершения разработки.

Точка останова — это преднамеренная остановка или приостановка в программе, установленная в целях отладки. Точка останова — это средство получения знаний о программе во время ее выполнения. Во время прерывания есть возможность проверить тестовую среду (регистры общего назначения, память, журналы, файлы и т.д.), чтобы выяснить, функционирует ли программа должным образом.

Трассировка — процесс пошагового выполнения программы. В режиме трассировки программист видит последовательность выполнения команд и значения переменных на данном шаге выполнения программы, что позволяет легче обнаруживать ошибки. Трассировка может быть начата и окончена в любом месте программы, выполнение программы может останавливаться на каждой команде или на точках останова, трассировка может выполняться с заходом в процедуры и без заходов, а также осуществляться в обратном порядке.

Точка останова для проверки корректного получения ответа Сервера представлена на рисунке 13.

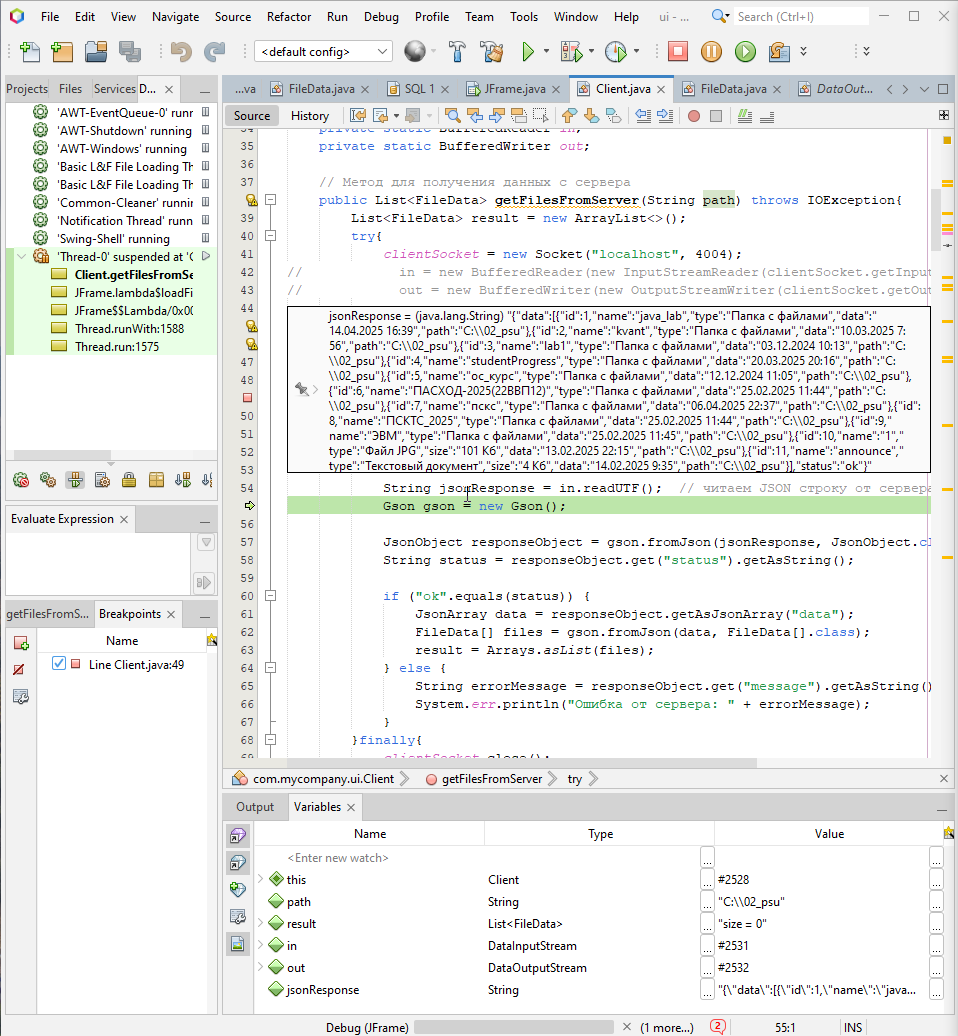


Рисунок 4 – Точка останова для проверки ответа от Сервера

## 4.2 Тестирование пользовательского интерфейса

На рисунке 5 представлена работа программы при нажатии на кнопку создания новой партии без ввода имени игрока.

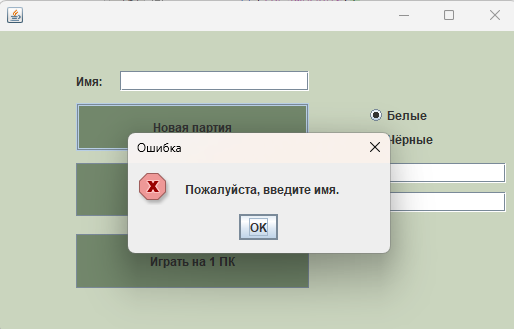


Рисунок 5 – Проверка корректного перемещения файла

На рисунке 6 представлена работа программы при нажатии на кнопку «Присоединиться» при условии пустого поля IP.

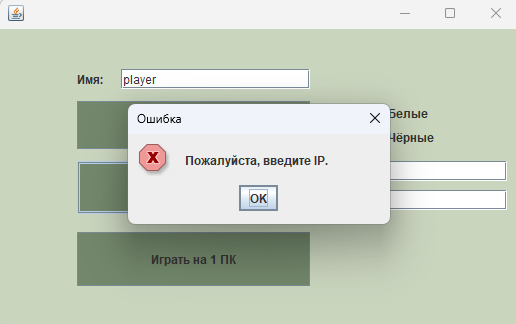


Рисунок 6 – Проверка корректного копирования файла

Полное описание поведения программы при тестировании выполнено в таблице 10.

# ЗАКЛЮЧЕНИЕ

Главным результатом проведенной работы является создание функционирующей программы, которая выполняет требуемый от неё круг задач. Она предоставляет возможность игры в шахматы по локальной сети. Интерфейс готовой программы представлен на рисунке 7.

В дальнейшем программу можно улучшить путём доработки пользовательского интерфейса в виде добавления различных видов иконок к файлам, а также привязки к основным операциям определенных сочетаний клавиш для выполнения функций.

В ходе выполнения курсового проекта были получены навыки разработки программ с использованием среды разработки приложений *NetBeans IDE*. Также были закреплены навыки и углублены знания программирования на языке *Java*.

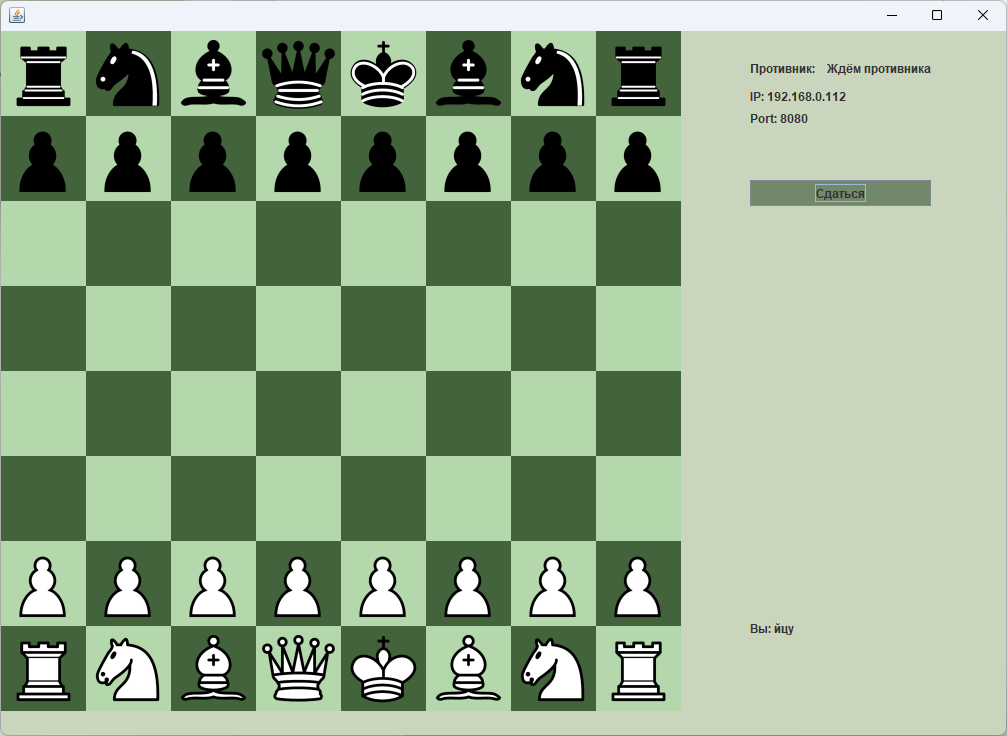
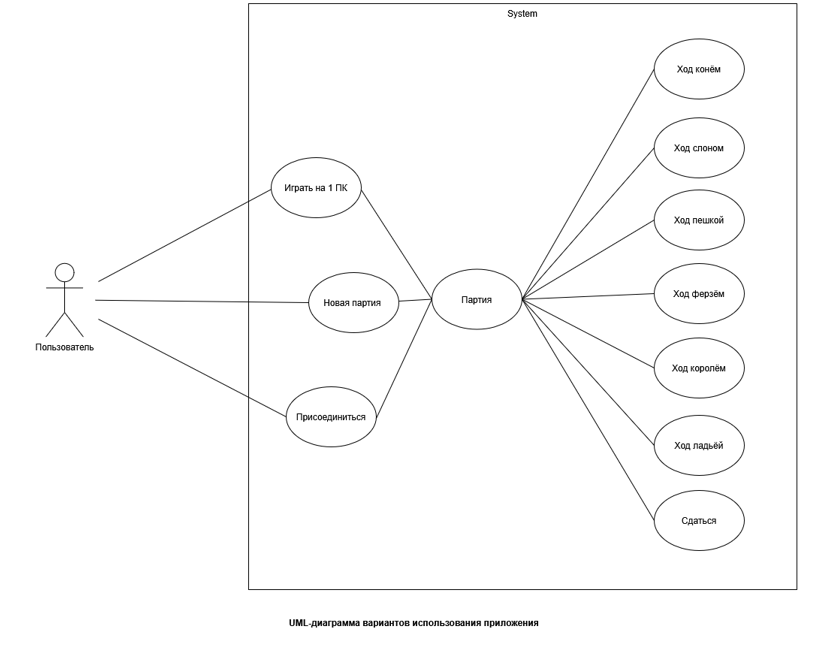


Рисунок 7 – Результат работы

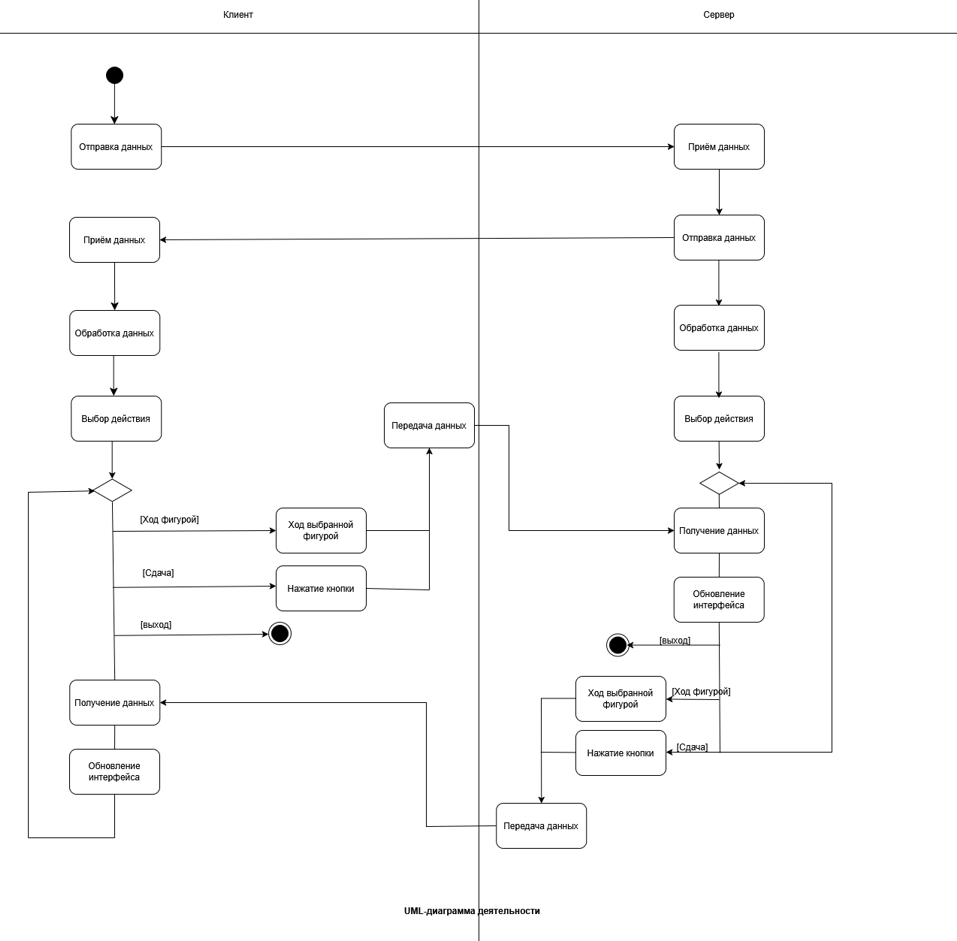
# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Зелязнов В. Основы программирования на Java. — М.: Диалект, 2020. — 340 с.
2. Иванов И. Ю. Учебник по Java для начинающих. — М.: Академия, 2022. — 290 с.
3. Сьерра К., Пайярс Б. Head First Java. — СПб.: Питер, 2008. — 720 с.
4. Шилдт Г. Java. Полное руководство. — М.: Вильямс, 2019. — 1152 с.

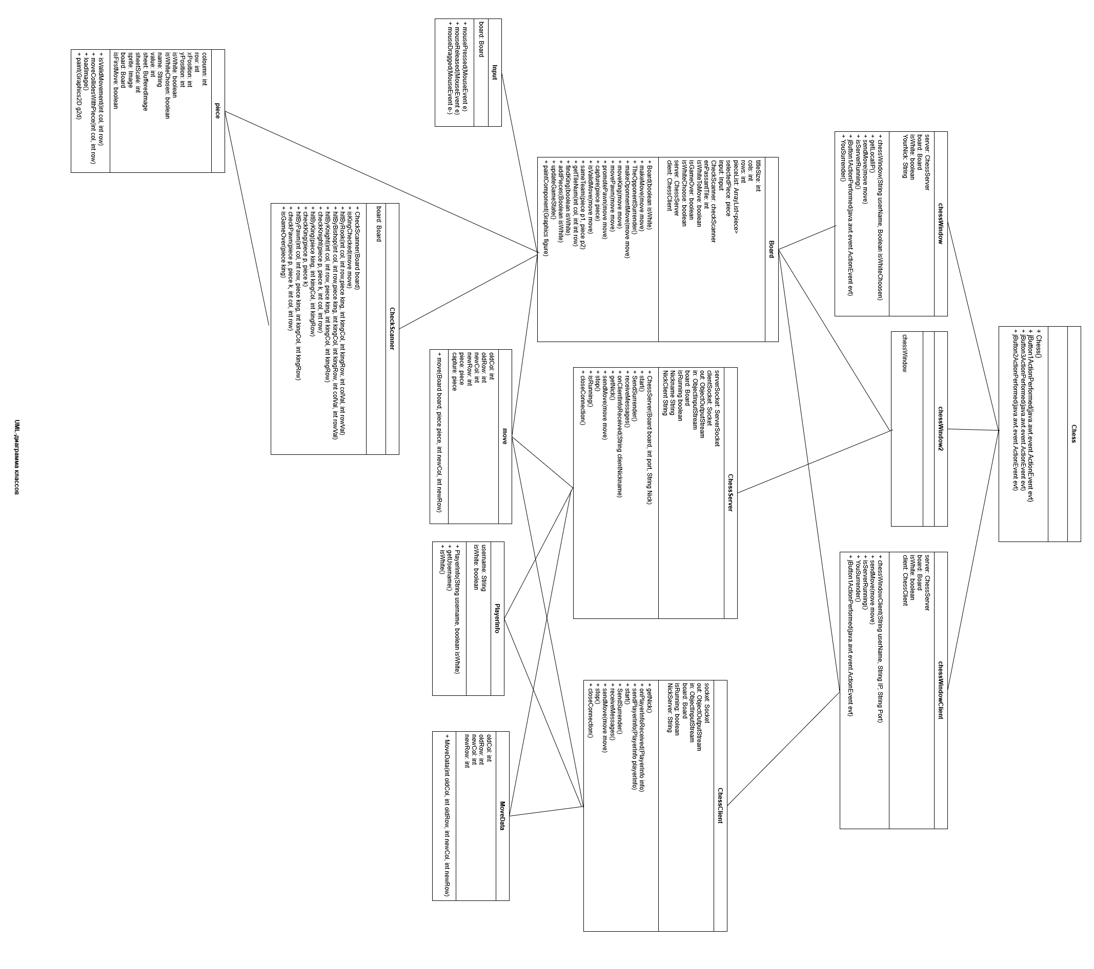
# ПРИЛОЖЕНИE A – UML-диаграмма вариантов использования приложения



# ПРИЛОЖЕНИE B – UML-диаграмма деятельности

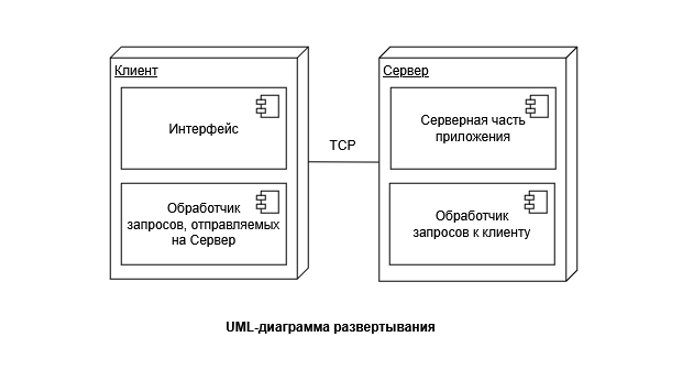


# ПРИЛОЖЕНИE C – UML-диаграмма классов и UML-диаграмма развертывания

UML

**UML-диаграмма классов**

# ПРИЛОЖЕНИE D UML-диаграмма развертывания



# ПРИЛОЖЕНИE E – Класс Board

package com.mycompany.chess;

import com.mycompany.chess.pieces.\*;

import java.awt.\*;

import java.io.IOException;

import java.util.ArrayList;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

public class Board extends JPanel{

public int titleSize = 85;

int cols = 8;

int rows = 8;

ArrayList<piece> pieceList = new ArrayList<>();

public piece selectedPiece;

Input input = new Input(this);

public CheckScanner checkScanner = new CheckScanner(this);

public int enPassantTile = -1;

public boolean isWhiteToMove = true;

public boolean isGameOver = false;

public boolean isWhiteChoose;

ChessServer server;

ChessClient client;

public Board(boolean isWhite)

{

isWhiteChoose = isWhite;

this.setPreferredSize(new Dimension(cols \* titleSize, rows \* titleSize));

this.addMouseListener(input);

this.addMouseMotionListener(input);

addPieces(isWhiteChoose);

}

public void setClient(ChessClient client){

this.client = client;

}

public void setServer(ChessServer server){

this.server = server;

}

public piece getPiece(int col, int row){

for(piece piece : pieceList){

if (piece.coloumn == col && piece.row == row){

return piece;

}

}

return null;

}

public void makeMove(move move) throws IOException{

if (move.piece.name.equals("Pawn")){

movePawn(move);

}else if (move.piece.name.equals("King")){

moveKing(move);

}

if (server != null) {

System.out.println("Sending a move via the server");

server.sendMove(move);

}

else if (client != null) {

System.out.println("Sending a move via the client");

client.sendMove(move);

}

else {

System.err.println("Error: Neither server nor client initialized!");

}

move.piece.coloumn = move.newCol;

move.piece.row = move.newRow;

move.piece.xPosition = move.newCol \* titleSize;

move.piece.yPosition = move.newRow \* titleSize;

move.piece.isFirstMove = false;

capture(move.capture);

isWhiteToMove = !isWhiteToMove;

updateGameState();

}

public void TheOpponentSurrender(){

String color = isWhiteChoose ? "Чёрные":"Белые";

JOptionPane.showMessageDialog(null, color + "выиграли", "Результаты партии", JOptionPane.INFORMATION\_MESSAGE);

}

public void makeOponnentMove(move move) throws IOException{

if (move.piece.name.equals("Pawn")){

movePawn(move);

}else if (move.piece.name.equals("King")){

moveKing(move);

}

move.piece.coloumn = move.newCol;

move.piece.row = move.newRow;

move.piece.xPosition = move.newCol \* titleSize;

move.piece.yPosition = move.newRow \* titleSize;

move.piece.isFirstMove = false;

capture(move.capture);

isWhiteToMove = !isWhiteToMove;

updateGameState();

}

private void moveKing(move move){

if (Math.abs(move.piece.coloumn - move.newCol) == 2){

piece rook;

if (move.piece.coloumn < move.newCol){

rook = getPiece(7, move.piece.row);

rook.coloumn = 5;

}else{

rook = getPiece(0,move.piece.row);

rook.coloumn = 3;

}

rook.xPosition = rook.coloumn \* titleSize;

}

}

public void movePawn(move move){

//на проходе

int colorIndex = move.piece.isWhite ? 1 : -1;

if(!isWhiteChoose){

colorIndex = move.piece.isWhite ? -1 : 1;

}

if (getTileNum(move.newCol, move.newRow) == enPassantTile){

move.capture = getPiece(move.newCol, move.newRow + colorIndex);

}

if (Math.abs(move.piece.row - move.newRow) == 2){

enPassantTile = getTileNum(move.newCol, move.newRow + colorIndex);

}else{

enPassantTile = -1;

}

//promotions

colorIndex = move.piece.isWhite ? 0 : 7;

if (move.newRow == colorIndex){

promotePawn(move);

}

}

public void promotePawn(move move){

pieceList.add(new Queen(this,move.newCol, move.newRow, move.piece.isWhite));

capture(move.piece);

}

public void capture(piece piece){

pieceList.remove(piece);

}

public boolean isValidMove(move move){

if(isGameOver){

return false;

}

if(move.piece.isWhite != isWhiteToMove){

return false;

}

if(sameTeam(move.piece, move.capture)){

return false;

}

if(!move.piece.isValidMovement(move.newCol, move.newRow)){

return false;

}

if(move.piece.moveCollidesWithPiece(move.newCol, move.newRow)){

return false;

}

if(checkScanner.isKingChecked(move)){

return false;

}

if(move.piece.isWhite != isWhiteChoose){

return false;

}

return true;

}

public boolean sameTeam(piece p1, piece p2){

if(p1 == null || p2 == null){

return false;

}

return p1.isWhite == p2.isWhite;

}

public int getTileNum(int col, int row){

return row \* rows + col;

}

piece findKing(boolean isWhite){

for (piece piece : pieceList){

if (isWhite == piece.isWhite && piece.name.equals("King")){

return piece;

}

}

return null;

}

public void addPieces(Boolean isWhite)

{

if(isWhite){

//knights

//black

pieceList.add(new Knight(this,1,0,false));

pieceList.add(new Knight(this,6,0,false));

//white

pieceList.add(new Knight(this,1,7,true));

pieceList.add(new Knight(this,6,7,true));

//pawn

//white

pieceList.add(new Pawn(this,0,6,true,isWhite));

pieceList.add(new Pawn(this,1,6,true,isWhite));

pieceList.add(new Pawn(this,2,6,true,isWhite));

pieceList.add(new Pawn(this,3,6,true,isWhite));

pieceList.add(new Pawn(this,4,6,true,isWhite));

pieceList.add(new Pawn(this,5,6,true,isWhite));

pieceList.add(new Pawn(this,6,6,true,isWhite));

pieceList.add(new Pawn(this,7,6,true,isWhite));

//black

pieceList.add(new Pawn(this,0,1,false,isWhite));

pieceList.add(new Pawn(this,1,1,false,isWhite));

pieceList.add(new Pawn(this,2,1,false,isWhite));

pieceList.add(new Pawn(this,3,1,false,isWhite));

pieceList.add(new Pawn(this,4,1,false,isWhite));

pieceList.add(new Pawn(this,5,1,false,isWhite));

pieceList.add(new Pawn(this,6,1,false,isWhite));

pieceList.add(new Pawn(this,7,1,false,isWhite));

//Rook

//white

pieceList.add(new Rook(this,7,7,true));

pieceList.add(new Rook(this,0,7,true));

//black

pieceList.add(new Rook(this,7,0,false));

pieceList.add(new Rook(this,0,0,false));

//Bishop

//white

pieceList.add(new Bishop(this,2,7,true));

pieceList.add(new Bishop(this,5,7,true));

//black

pieceList.add(new Bishop(this,2,0,false));

pieceList.add(new Bishop(this,5,0,false));

//king

//white

pieceList.add(new King(this,4,7,true));

//black

pieceList.add(new King(this,4,0,false));

//queen

//white

pieceList.add(new Queen(this,3,7,true));

//black

pieceList.add(new Queen(this,3,0,false));

}

else{

// Knights

//black

pieceList.add(new Knight(this, 1, 7, false));

pieceList.add(new Knight(this, 6, 7, false));

//white

pieceList.add(new Knight(this, 1, 0, true));

pieceList.add(new Knight(this, 6, 0, true));

// Pawns

// black

pieceList.add(new Pawn(this, 0, 6, false,isWhite));

pieceList.add(new Pawn(this, 1, 6, false,isWhite));

pieceList.add(new Pawn(this, 2, 6, false,isWhite));

pieceList.add(new Pawn(this, 3, 6, false,isWhite));

pieceList.add(new Pawn(this, 4, 6, false,isWhite));

pieceList.add(new Pawn(this, 5, 6, false,isWhite));

pieceList.add(new Pawn(this, 6, 6, false,isWhite));

pieceList.add(new Pawn(this, 7, 6, false,isWhite));

// white

pieceList.add(new Pawn(this, 0, 1, true,isWhite));

pieceList.add(new Pawn(this, 1, 1, true,isWhite));

pieceList.add(new Pawn(this, 2, 1, true,isWhite));

pieceList.add(new Pawn(this, 3, 1, true,isWhite));

pieceList.add(new Pawn(this, 4, 1, true,isWhite));

pieceList.add(new Pawn(this, 5, 1, true,isWhite));

pieceList.add(new Pawn(this, 6, 1, true,isWhite));

pieceList.add(new Pawn(this, 7, 1, true,isWhite));

// Rooks

// black

pieceList.add(new Rook(this, 0, 7, false));

pieceList.add(new Rook(this, 7, 7, false));

// white

pieceList.add(new Rook(this, 0, 0, true));

pieceList.add(new Rook(this, 7, 0, true));

// Bishops

// black

pieceList.add(new Bishop(this, 2, 7, false));

pieceList.add(new Bishop(this, 5, 7, false));

// white

pieceList.add(new Bishop(this, 2, 0, true));

pieceList.add(new Bishop(this, 5, 0, true));

// Kings

// black

pieceList.add(new King(this, 4, 7, false));

// white

pieceList.add(new King(this, 4, 0, true));

// Queens (ферзи)

// black

pieceList.add(new Queen(this, 3, 7, false));

// white

pieceList.add(new Queen(this, 3, 0, true));

}

}

private void updateGameState(){

piece king = findKing(isWhiteToMove);

if(checkScanner.isGameOver(king)){

if(checkScanner.isKingChecked(new move(this, king, king.coloumn, king.row))){

if(isWhiteToMove){

JOptionPane.showMessageDialog(null, "Чёрные выиграли", "Результаты партии", JOptionPane.INFORMATION\_MESSAGE);

}else{

JOptionPane.showMessageDialog(null, "Белые выиграли", "Результаты партии", JOptionPane.INFORMATION\_MESSAGE);

}

}

}

}

protected void paintComponent(Graphics figure)

{

Graphics2D g2d = (Graphics2D) figure;

//рисуем доску

for (int row = 0; row < rows; row++)

for (int coloumn = 0; coloumn < cols; coloumn++)

{

g2d.setColor((coloumn + row) % 2 == 0 ?

Color.decode("#b4d8ab") : Color.decode("#43633c"));

g2d.fillRect(coloumn \* titleSize, row \* titleSize,titleSize,titleSize);

}

//рисуем ходы

if (selectedPiece != null)

for (int row = 0; row < rows; row++)

for (int coloumn = 0; coloumn < cols; coloumn++){

if (isValidMove(new move(this,selectedPiece,coloumn,row))){

g2d.setColor(new Color(68,180,57,190));

g2d.fillRect(coloumn \* titleSize,row \* titleSize,titleSize,titleSize);

}

}

//рисуем фигуры

for (piece Piece : pieceList)

{

Piece.paint(g2d);

}

}

}

# ПРИЛОЖЕНИE F – Программа сканера событий

package com.mycompany.chess;

import com.mycompany.chess.pieces.piece;

public class CheckScanner {

Board board;

public CheckScanner(Board board){

this.board = board;

}

public boolean isKingChecked(move move){

piece king = board.findKing(move.piece.isWhite);

assert king != null;

int kingCol = king.coloumn;

int kingRow = king.row;

if(board.selectedPiece != null && board.selectedPiece.name.equals("King")){

kingCol = move.newCol;

kingRow = move.newRow;

}

return hitByRook(move.newCol, move.newRow, king, kingCol, kingRow, 0, 1)|| //up

hitByRook(move.newCol, move.newRow, king, kingCol, kingRow, 1, 0)|| //right

hitByRook(move.newCol, move.newRow, king, kingCol, kingRow, 0, -1)|| //down

hitByRook(move.newCol, move.newRow, king, kingCol, kingRow, -1, 0)|| //left

hitByBishop(move.newCol, move.newRow, king, kingCol, kingRow, -1, -1)|| //up left

hitByBishop(move.newCol, move.newRow, king, kingCol, kingRow, 1, -1)|| //up right

hitByBishop(move.newCol, move.newRow, king, kingCol, kingRow, 1, 1)|| //down right

hitByBishop(move.newCol, move.newRow, king, kingCol, kingRow, -1, 1)|| //down left

hitByKnight(move.newCol, move.newRow, king, kingCol, kingRow)||

hitByPawn(move.newCol, move.newRow, king, kingCol, kingRow)||

hitByKing(king, kingCol, kingRow);

}

private boolean hitByRook(int col, int row,piece king, int kingCol, int kingRow, int colVal, int rowVal){

for(int i = 1; i < 8; i++){

if(kingCol + (i \* colVal) == col && kingRow + (i \* rowVal) == row){

break;

}

piece piece = board.getPiece(kingCol + (i \* colVal), kingRow + (i \* rowVal));

if (piece != null && piece != board.selectedPiece){

if(!board.sameTeam(piece, king) && (piece.name.equals("Rook") || piece.name.equals("Queen"))){

return true;

}

break;

}

}

return false;

}

private boolean hitByBishop(int col, int row,piece king, int kingCol, int kingRow, int colVal, int rowVal){

for(int i = 1; i < 8; i++){

if(kingCol - (i \* colVal) == col && kingRow - (i \* rowVal) == row){

break;

}

piece piece = board.getPiece(kingCol - (i \* colVal), kingRow - (i \* rowVal));

if (piece != null && piece != board.selectedPiece){

if(!board.sameTeam(piece, king) && (piece.name.equals("Bishop") || piece.name.equals("Queen"))){

return true;

}

break;

}

}

return false;

}

private boolean hitByKnight(int col, int row, piece king, int kingCol, int kingRow){

return checkKnight(board.getPiece(kingCol - 1, kingRow - 2), king, col,row) ||

checkKnight(board.getPiece(kingCol + 1, kingRow - 2), king, col,row) ||

checkKnight(board.getPiece(kingCol + 2, kingRow - 1), king, col,row) ||

checkKnight(board.getPiece(kingCol + 2, kingRow + 1), king, col,row) ||

checkKnight(board.getPiece(kingCol + 1, kingRow + 2), king, col,row) ||

checkKnight(board.getPiece(kingCol - 1, kingRow + 2), king, col,row) ||

checkKnight(board.getPiece(kingCol - 2, kingRow + 1), king, col,row) ||

checkKnight(board.getPiece(kingCol - 2, kingRow - 1), king, col,row);

}

private boolean checkKnight(piece p, piece k, int col, int row){

return p != null && !board.sameTeam(p,k)&& p.name.equals("Knight") && !(p.coloumn == col && p.row == row);

}

private boolean hitByKing(piece king, int kingCol, int kingRow){

return checkKing(board.getPiece(kingCol - 1, kingRow - 1), king) ||

checkKing(board.getPiece(kingCol + 1, kingRow - 1), king) ||

checkKing(board.getPiece(kingCol, kingRow - 1), king) ||

checkKing(board.getPiece(kingCol - 1, kingRow), king) ||

checkKing(board.getPiece(kingCol + 1, kingRow), king) ||

checkKing(board.getPiece(kingCol - 1, kingRow + 1), king) ||

checkKing(board.getPiece(kingCol + 1, kingRow + 1), king) ||

checkKing(board.getPiece(kingCol, kingRow + 1), king);

}

private boolean checkKing(piece p, piece k){

return p != null && !board.sameTeam(p,k) && p.name.equals("King");

}

private boolean hitByPawn(int col, int row, piece king, int kingCol, int kingRow){

int colorVal = king.isWhite ? -1 : 1;

return checkPawn(board.getPiece(kingCol + 1, kingRow + colorVal), king, col, row)||

checkPawn(board.getPiece(kingCol - 1, kingRow + colorVal), king, col, row);

}

private boolean checkPawn(piece p, piece k, int col, int row){

return p!= null && !board.sameTeam(p,k) && p.name.equals("Pawn") && !(p.coloumn == col && p.row == row);

}

public boolean isGameOver(piece king){

for(piece piece : board.pieceList){

if(board.sameTeam(piece, king)){

board.selectedPiece = piece == king ? king : null;

for(int row = 0; row < board.rows; row++){

for(int col = 0; col < board.cols; col++){

move move = new move(board, piece, col, row);

if(board.isValidMove(move)){

return false;

}

}

}

}

}

return true;

}

}

# ПРИЛОЖЕНИE G – Программа главного окна

package com.mycompany.chess;

import java.awt.Color;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.ButtonGroup;

public class Chess extends javax.swing.JFrame {

public Chess() {

this.getContentPane().setBackground(Color.decode("#cad5be"));

initComponents();

ButtonGroup buttonGroup = new ButtonGroup();

buttonGroup.add(WhiteButton);

buttonGroup.add(BlackButton);

WhiteButton.setSelected(true);

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jLabel1 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

jTextField2 = new javax.swing.JTextField();

jTextField3 = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

BlackButton = new javax.swing.JRadioButton();

WhiteButton = new javax.swing.JRadioButton();

jButton3 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setBackground(new java.awt.Color(127, 151, 101));

setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jButton1.setBackground(new java.awt.Color(114, 135, 107));

jButton1.setText("Новая партия");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jButton2.setBackground(new java.awt.Color(114, 135, 107));

jButton2.setText("присоединиться");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

jLabel1.setText("Имя:");

jTextField1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jTextField1ActionPerformed(evt);

}

});

jTextField2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jTextField2ActionPerformed(evt);

}

});

jLabel2.setText("IP:");

jLabel3.setText("Port:");

BlackButton.setBackground(new java.awt.Color(202, 213, 190));

BlackButton.setText("Чёрные");

BlackButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

BlackButtonActionPerformed(evt);

}

});

WhiteButton.setBackground(new java.awt.Color(202, 213, 190));

WhiteButton.setText("Белые");

WhiteButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

WhiteButtonActionPerformed(evt);

}

});

jButton3.setBackground(new java.awt.Color(114, 135, 107));

jButton3.setText("Играть на 1 ПК");

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(77, 77, 77)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jButton3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton2, javax.swing.GroupLayout.DEFAULT\_SIZE, 233, Short.MAX\_VALUE)

.addComponent(jButton1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jTextField1)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(12, 12, 12)

.addComponent(jLabel2)

.addGap(12, 12, 12))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel3)

.addGap(18, 18, 18)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(BlackButton)

.addComponent(WhiteButton))

.addGap(0, 69, Short.MAX\_VALUE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addGap(0, 0, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jTextField2, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, 131, Short.MAX\_VALUE)

.addComponent(jTextField3, javax.swing.GroupLayout.Alignment.TRAILING))))

.addContainerGap())

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(40, 40, 40)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(layout.createSequentialGroup()

.addComponent(WhiteButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(BlackButton))

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 48, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(12, 12, 12)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel2))

.addGap(9, 9, 9)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel3)))

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 53, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addComponent(jButton3, javax.swing.GroupLayout.PREFERRED\_SIZE, 54, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(43, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

private ButtonGroup buttonGroup;

boolean isWhiteButtonSelected;

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

String playerName = jTextField1.getText().trim();

if (playerName.isEmpty()) {

javax.swing.JOptionPane.showMessageDialog(this, "Пожалуйста, введите имя.", "Ошибка", javax.swing.JOptionPane.ERROR\_MESSAGE);

} else {

isWhiteButtonSelected = WhiteButton.isSelected();

chessWindow gameWindow = null;

try {

gameWindow = new chessWindow(jTextField1.getText(), isWhiteButtonSelected);

} catch (IOException ex) {

Logger.getLogger(Chess.class.getName()).log(Level.SEVERE, null, ex);

}

gameWindow.setVisible(true);

}

}

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void BlackButtonActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void WhiteButtonActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

chessWindow2 gameWindow = new chessWindow2();

gameWindow.setVisible(true);

this.dispose();

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

chessWindowClient gameWindow = null;

if (jTextField1.getText().trim().isEmpty()) {

javax.swing.JOptionPane.showMessageDialog(this, "Пожалуйста, введите имя.", "Ошибка", javax.swing.JOptionPane.ERROR\_MESSAGE);

}

else if(jTextField2.getText().trim().isEmpty()){

javax.swing.JOptionPane.showMessageDialog(this, "Пожалуйста, введите IP.", "Ошибка", javax.swing.JOptionPane.ERROR\_MESSAGE);

}

else if(jTextField2.getText().trim().isEmpty()){

javax.swing.JOptionPane.showMessageDialog(this, "Пожалуйста, введите Port.", "Ошибка", javax.swing.JOptionPane.ERROR\_MESSAGE);

}else{

try {

gameWindow = new chessWindowClient(jTextField1.getText(),jTextField2.getText(),jTextField3.getText());

} catch (IOException ex) {

Logger.getLogger(Chess.class.getName()).log(Level.SEVERE, null, ex);

}

gameWindow.setVisible(true);

}

}

public static void main(String args[]) {

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Chess().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JRadioButton BlackButton;

private javax.swing.JRadioButton WhiteButton;

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

private javax.swing.JTextField jTextField3;

// End of variables declaration

}

# ПРИЛОЖЕНИE H – Программа клиента

package com.mycompany.chess;

import com.mycompany.chess.pieces.piece;

import java.io.\*;

import java.net.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

import javax.swing.SwingUtilities;

public class ChessClient {

private Socket socket;

private ObjectOutputStream out;

private ObjectInputStream in;

private final Board board;

private volatile boolean isRunning;

public String NickServer;

public ChessClient(Board board, String host, int port) throws IOException {

this.board = board;

this.socket = new Socket(host, port);

this.out = new ObjectOutputStream(socket.getOutputStream());

this.in = new ObjectInputStream(socket.getInputStream());

this.isRunning = true;

}

public String getNick(){

return NickServer;

}

public void onPlayerInfoReceived(PlayerInfo info) {

// Пустая реализация по умолчанию

}

public void sendPlayerInfo(PlayerInfo playerInfo) throws IOException {

out.writeObject(playerInfo);

out.flush();

System.out.println("PlayerInfo sent to server:" + playerInfo.getUsername());

}

public void start() {

new Thread(() -> {

try {

Object received = in.readObject();

if (!(received instanceof PlayerInfo)) {

throw new IOException("Получен некорректный формат данных");

}

PlayerInfo playerInfo = (PlayerInfo) received;

SwingUtilities.invokeLater(() -> {

this.NickServer = playerInfo.getUsername();

onPlayerInfoReceived(playerInfo);

System.out.println("playerInfo was received");

});

new Thread(this::receiveMessages).start();

} catch (Exception e) {

if (isRunning) {

SwingUtilities.invokeLater(() ->

JOptionPane.showMessageDialog(null,

"Ошибка получения данных: " + e.getMessage(),

"Ошибка",

JOptionPane.ERROR\_MESSAGE));

}

}

}).start();

}

public void SendSurrender() throws IOException{

out.writeObject("Surrender");

out.flush();

}

private void receiveMessages() {

try {

while (isRunning) {

Object obj = in.readObject();

System.out.println("Получено сообщение: " + obj);

if (obj instanceof MoveData) {

MoveData data = (MoveData) obj;

piece piece = board.getPiece(data.oldCol, data.oldRow);

move move = new move(board, piece, data.newCol, data.newRow);

SwingUtilities.invokeLater(() -> {

synchronized (board) {

try {

board.makeOponnentMove(move);

} catch (IOException ex) {

Logger.getLogger(ChessClient.class.getName()).log(Level.SEVERE, null, ex);

}

board.repaint();

}

});

} else if (obj instanceof String) {

String message = (String) obj;

if ("Surrender".equals(message)) {

SwingUtilities.invokeLater(() -> {

board.TheOpponentSurrender();

});

} else {

System.out.println("Неизвестная строка: " + message);

}

}

else {

System.out.println("Неизвестный тип объекта: " + obj.getClass());

}

}

} catch (EOFException e) {

System.out.println("The server has gone down");

} catch (Exception e) {

if (isRunning) {

System.err.println("Connection error:" + e.getMessage());

}

} finally {

closeConnection();

}

}

public synchronized void sendMove(move move) throws IOException {

if (out == null) {

throw new IOException("Connection lost or not established!");

}

System.out.println("Sending a move:" + move);

MoveData data = new MoveData(move.oldCol,move.oldRow,move.newCol,move.newRow);

out.writeObject(data);

out.flush();

System.out.println("The move has been sent.");

}

public void stop() {

isRunning = false;

closeConnection();

}

private void closeConnection() {

try {

if (in != null) in.close();

if (out != null) out.close();

if (socket != null) socket.close();

} catch (IOException e) {

System.err.println("Ошибка закрытия соединения: " + e.getMessage());

}

}

}

# ПРИЛОЖЕНИE I – Программа сервера

package com.mycompany.chess;

import com.mycompany.chess.Board;

import com.mycompany.chess.move;

import com.mycompany.chess.pieces.piece;

import java.io.\*;

import java.net.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.SwingUtilities;

public class ChessServer {

private ServerSocket serverSocket;

private Socket clientSocket;

private ObjectOutputStream out;

private ObjectInputStream in;

private final Board board;

private volatile boolean isRunning;

public String Nickname;

public String NickClient;

public ChessServer(Board board, int port, String Nick) throws IOException {

this.board = board;

this.serverSocket = new ServerSocket(port);

this.isRunning = true;

this.Nickname = Nick;

}

public void start() {

System.out.println("Server started. Waiting for player...");

new Thread(() -> {

try {

clientSocket = serverSocket.accept();

System.out.println("Player connected!");

out = new ObjectOutputStream(clientSocket.getOutputStream());

in = new ObjectInputStream(clientSocket.getInputStream());

PlayerInfo playerInfo = new PlayerInfo(Nickname,!board.isWhiteChoose );

out.writeObject(playerInfo);

out.flush();

Object received = in.readObject();

if (received instanceof PlayerInfo) {

PlayerInfo clientInfo = (PlayerInfo) received;

System.out.println("Received PlayerInfo from client:" + clientInfo.getUsername());

this.NickClient = clientInfo.getUsername();

onClientInfoReceived(clientInfo.getUsername());

}

System.out.println("playerInfo was sended");

new Thread(this::receiveMessages).start();

} catch (IOException e) {

if (isRunning) {

System.err.println("Connection error:" + e.getMessage());

}

} catch (ClassNotFoundException ex) {

Logger.getLogger(ChessServer.class.getName()).log(Level.SEVERE, null, ex);

}

}).start();

}

public void SendSurrender() throws IOException{

out.writeObject("Surrender");

out.flush();

}

private void receiveMessages() {

try {

while (isRunning) {

Object obj = in.readObject();

System.out.println("Message received: " + obj);

if (obj instanceof MoveData) {

MoveData data = (MoveData) obj;

piece piece = board.getPiece(data.oldCol, data.oldRow);

move move = new move(board, piece, data.newCol, data.newRow);

SwingUtilities.invokeLater(() -> {

synchronized (board) {

try {

board.makeOponnentMove(move);

} catch (IOException ex) {

Logger.getLogger(ChessClient.class.getName()).log(Level.SEVERE, null, ex);

}

board.repaint();

}

});

} else if (obj instanceof String) {

String message = (String) obj;

if ("Surrender".equals(message)) {

SwingUtilities.invokeLater(() -> {

board.TheOpponentSurrender();

});

} else {

System.out.println("Unknown string: " + message);

}

} else {

System.out.println("Unknown object type: " + obj.getClass());

}

}

} catch (EOFException e) {

System.out.println("The server has gone down");

} catch (Exception e) {

if (isRunning) {

System.err.println("Connection error:" + e.getMessage());

}

} finally {

closeConnection();

}

}

public void onClientInfoReceived(String clientNickname) {

}

public String getNick(){

return NickClient;

}

public synchronized void sendMove(move move) throws IOException {

if (out == null || clientSocket.isClosed()) {

throw new IOException("Connection lost or not established!");

}

System.out.println("Sending a move:" + move);

MoveData data = new MoveData(move.oldCol,move.oldRow,move.newCol,move.newRow);

out.writeObject(data);

out.flush();

System.out.println("The move has been sent.");

}

public void stop() {

isRunning = false;

closeConnection();

try {

if (serverSocket != null) {

serverSocket.close();

}

} catch (IOException e) {

System.err.println("Server shutdown error:" + e.getMessage());

}

}

public boolean isRunning() {

return isRunning;

}

private void closeConnection() {

try {

if (in != null) in.close();

if (out != null) out.close();

if (clientSocket != null) clientSocket.close();

} catch (IOException e) {

System.err.println("Error closing connection:" + e.getMessage());

}

}

}

# ПРИЛОЖЕНИE J – Программа для отслеживания движений мышки

package com.mycompany.chess;

import com.mycompany.chess.pieces.piece;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.io.IOException;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Input extends MouseAdapter {

Board board;

public Input (Board board){

this.board = board;

}

@Override

public void mousePressed(MouseEvent e) {

int col = e.getX() / board.titleSize;

int row = e.getY() / board.titleSize;

piece pieceXY = board.getPiece(col, row);

if(pieceXY != null){

board.selectedPiece = pieceXY;

}

}

@Override

public void mouseReleased(MouseEvent e) {

int col = e.getX() / board.titleSize;

int row = e.getY() / board.titleSize;

if(board.selectedPiece != null){

move move = new move(board, board.selectedPiece, col, row);

if (board.isValidMove(move)){

try {

board.makeMove(move);

} catch (IOException ex) {

Logger.getLogger(Input.class.getName()).log(Level.SEVERE, null, ex);

}

}else{

board.selectedPiece.xPosition = board.selectedPiece.coloumn \* board.titleSize;

board.selectedPiece.yPosition = board.selectedPiece.row \* board.titleSize;

}

}

board.selectedPiece = null;

board.repaint();

}

@Override

public void mouseDragged(MouseEvent e) {

if (board.selectedPiece != null){

board.selectedPiece.xPosition = e.getX() - board.titleSize / 2;

board.selectedPiece.yPosition = e.getY() - board.titleSize / 2;

board.repaint();

}

}

}