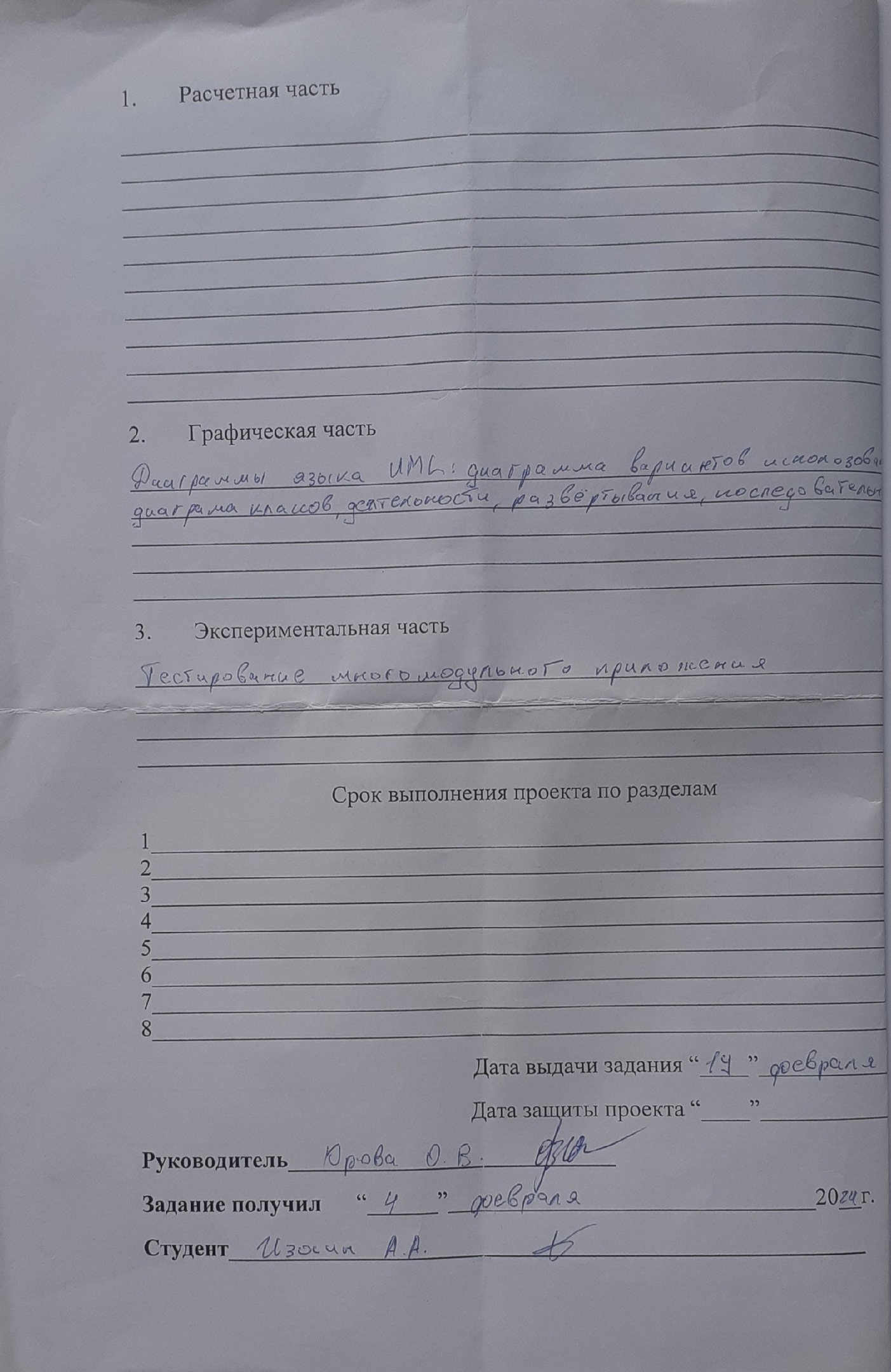
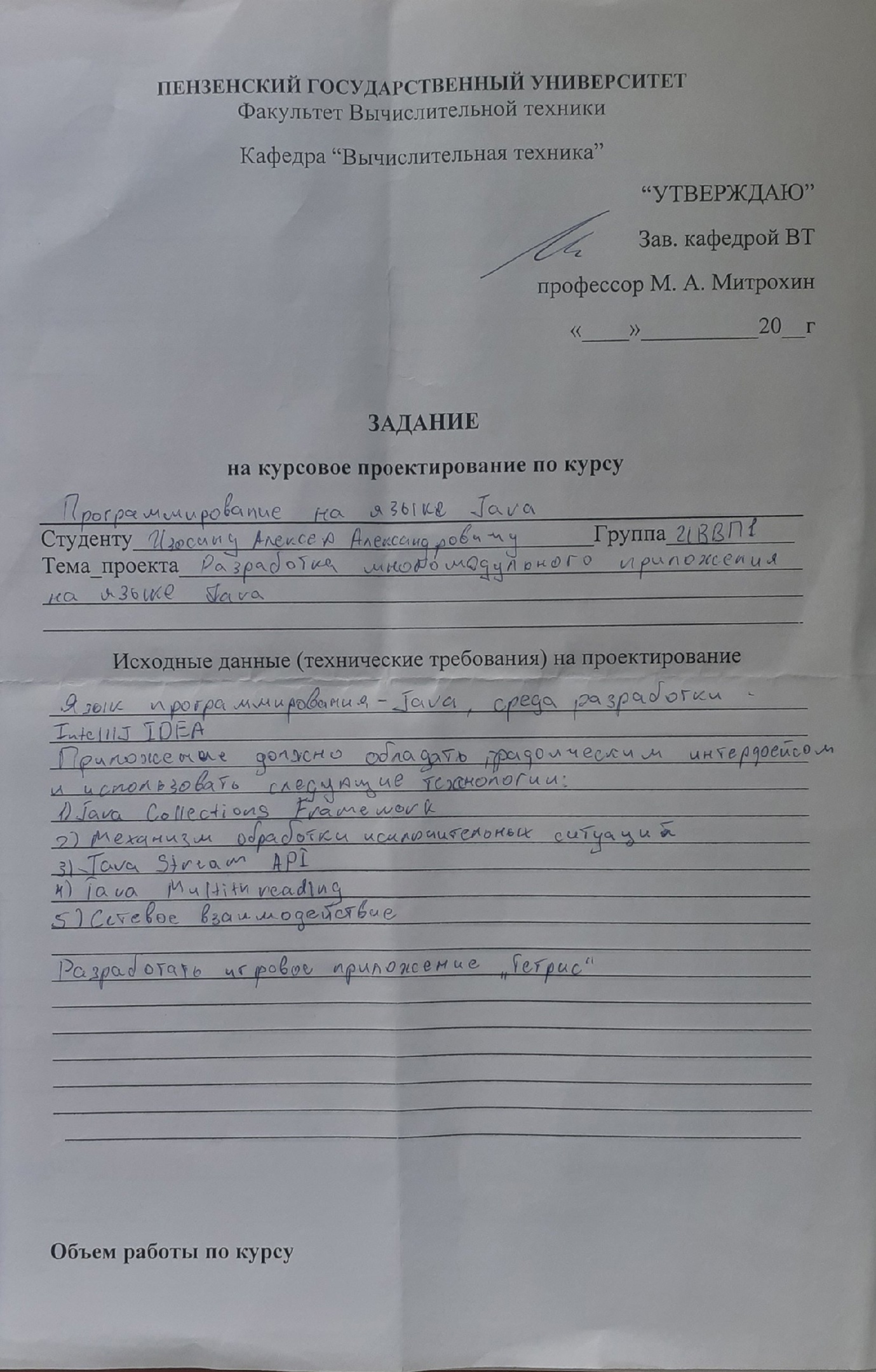
### 

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc168390534)

[Введение 4](#_Toc168390535)

[1 Постановка задачи 6](#_Toc168390536)

[2 Теоретическая часть задания 7](#_Toc168390537)

[2.1 Среда разработки 7](#_Toc168390538)

[2.2 Игра «Тетрис» 7](#_Toc168390539)

[3 Описание алгоритма работы 9](#_Toc168390540)

[4 Описание программы 10](#_Toc168390541)

[4.1 Класс «Block» 10](#_Toc168390542)

[4.2 Класс «GameButtons» 10](#_Toc168390543)

[4.3 Класс «ParameterBuilder» 10](#_Toc168390544)

[4.4 Класс «Tetris1P» 10](#_Toc168390545)

[4.5 Класс «TetrisFocusListener» 11](#_Toc168390546)

[4.6 Класс «TetrisKeyListener» 11](#_Toc168390547)

[4.7 Класс «TetrisMouseListener» 11](#_Toc168390548)

[4.8 Класс «TetrisWindow» 12](#_Toc168390549)

[4.9 Класс «Tetromino» 12](#_Toc168390550)

[4.10 Класс «TetrisRunner» 12](#_Toc168390551)

[5 Руководство пользователя 13](#_Toc168390552)

[5.1 Запуск приложения 13](#_Toc168390553)

[5.2 Окно игры 14](#_Toc168390554)

[5.3 Окно при поражении 14](#_Toc168390555)

[5.4 Окно паузы 15](#_Toc168390556)

[ЗАКЛЮЧЕНИЕ 16](#_Toc168390557)

[СПИСОК ЛИТЕРАТУРЫ 17](#_Toc168390558)

[Приложение A. Листинг программы 18](#_Toc168390559)

[Приложение А.1. «Block.java» 18](#_Toc168390560)

[Приложение А.2. «GameBoard.java» 20](#_Toc168390561)

[Приложение А.3. «GameButton.java» 25](#_Toc168390562)

[Приложение А.4. «GameButtons.java» 27](#_Toc168390563)

[Приложение А.5. «GraphicsUtils.java» 28](#_Toc168390564)

[Приложение А.6. «ParameterBuilder.java» 29](#_Toc168390565)

[Приложение А.7. «Tetris1P.java» 30](#_Toc168390566)

[Приложение А.8. «TetrisFocusListener.java» 31](#_Toc168390567)

[Приложение А.9. «TetrisKeyListener.java» 32](#_Toc168390568)

[Приложение А.10. «TetrisMouseListener.java» 35](#_Toc168390569)

[Приложение А.11. «TetrisWindow.java» 36](#_Toc168390570)

[Приложение А.12. «Tetromino.java» 38](#_Toc168390571)

[Приложение А.13. «Tetrisrunner.java» 39](#_Toc168390572)

[Приложение Б. UML-диаграммы 41](#_Toc168390573)

[Приложение Б.1. UML-диаграмма вариантов использования 41](#_Toc168390574)

[Приложение Б.2. UML- диаграмма деятельности 42](#_Toc168390575)

[Приложение Б.3. UML-диаграмма классов 43](#_Toc168390576)

# Введение

Модульное программирование  — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам.

Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

Таким образом, поддержка принципов модульного программирования, инверсии управления и четкой архитектуры приложения поможет решить сразу несколько задач:

1. Обеспечить чёткое функциональное разделение кода. При возникновении ошибок можно быстро определить источник, а исправления не приведут к появлению новых сбоев.
2. Минимизировать количество связей. Это позволит упростить разработку, отдав на разработку нескольким разработчикам разные модули. Или можно самостоятельно разрабатывать каждый блок без оглядки на другие, что тоже экономит время и силы.
3. Создать иерархию с чёткой вертикалью наследования. Это повышает надёжность кода, так как тестирование провести проще, а результаты информативнее.

Соблюдение принципа модульности в больших проектах позволит сэкономить время и чётко организовать процесс разработки. Более того, у разработчика наконец появится возможность сосредоточиться на самом интересном – реализации оригинальных задумок в коде.

Данная курсовая работа предназначена для ознакомления со способами разработки многомодульных приложений. Она также позволяет более углублённо изучить язык программирования Java. Результатом данной курсовой станет многомодульное приложение всеми известной игры «Тетрис» с графическим интерфейсом и сетевым взаимодействием.

# Постановка задачи

Разработать многомодульное приложение-игру «Тетрис». Язык программирования - Java, среда разработки -Intelijer IDEA. Приложение должно обладать графическим интерфейсом и использовать следующие технологии:

1. Java Collection Framework;

2. Механики обработки исключительных ситуаций;

3. Java Stream API;

4. Java Multithreading;

5. Сетевое взаимодействие.

# Теоретическая часть задания

## **Среда разработки**

По сравнению с другими средами IDE, IntelliJ IDEA обеспечивает высококлассную комплексную поддержку новейших технологий Java и последних усовершенствований стандартов Java. Эта IDE поддерживает JDK 8, JDK 7, Java EE 7, включая соответствующие усовершенствования HTML5 и JavaFX 2.

Благодаря постоянно совершенствующемуся редактору Java, разнообразным возможностям и широкому спектру инструментов, шаблонов и образцов, IntelliJ IDEA устанавливает стандарт разработки с помощью новейших готовых технологий.

С помощью редакторов и функций перетаскивания в среде IDE можно быстро и эффективно разрабатывать графические интерфейсы для приложений Java SE, HTML5, Java EE, PHP, C/C++ и других языков.

IntelliJ IDEA автоматически обрабатывает правильные интервалы между строками и выравнивание для приложений, поддерживая редактирование внутри строки. Конструктор графического интерфейса настолько интуитивен и прост в использовании, что его часто применяют для создания живых прототипов интерфейсов во время демонстраций для клиентов.

## **Игра «Тетрис»**

Суть игры заключается в наборе как можно большего числа очков и собранных линий. С определённым временным периодом на игровом поле появляется одна из 7 фигур с отличной от других блоков формой. Задача игрока – как можно быстрее опустить фигуру вниз так, чтобы она смогла составить линию с остальными блоками. За каждую составленную линию игрок получает 50 очков, если игрок хочет набрать больше очков, он может опускать блоки быстрее самостоятельно, за каждое нажатие кнопки «Вниз» игрок получает 1 очко.

Игра заканчивается поражением тогда, когда игровое поле заполнилось до самой верхней линии и дальнейшее падение блоков невозможно.

# Описание алгоритма работы

* 1. **Описание клиент-серверной части**

Разработанная программа состоит из клиентской и серверной частей. Сперва необходимо запустить сервер с уникальным адресом для обслуживания TCP/IP, определяемый комбинацией IP-адреса хоста с номером порта обслуживания, который создает конечную точку для обслуживания. Далее сокет привязывается к локальной конечной точке. Чтобы сокет клиента мог идентифицировать потоковый сокет TCP, серверная программа присваивает имя своему сокету.

Затем сокет переходит в состояние прослушивания, в котором он будет ожидать входящие попытки соединения от клиента. В случае обнаружения входящего соединения, сервер проверяет в цикле switch-case входящее сообщение и в зависимости от него выполняет соответствующие действия. На этом установление соединения между клиентом и сервером заканчивается.

После установления соединения сервер ожидает запросы от клиента.

Как только клиент отправляет запрос, сервер их обрабатывает и присылает ответ на данный запрос.

* 1. **Описание работы приложения**

Приложение имеет несколько графических сцен, которым присвоены соответствующие кнопки. Для перехода из одного пункта меню в другой срабатывает событие, которое переключает сцену и отрисовывает её заново.

# Описание программы

### Класс «Block»

Класс Block представляет собой блок в игре Тетрис. Он содержит информацию о цвете блока и методы для его отрисовки. Этот класс обеспечивает основную функциональность для работы с блоками в игре Тетрис, включая их цветовое оформление и отрисовку на экране.

### Класс «GameButtons»

Класс GameButtons управляет коллекцией кнопок в игре Тетрис. Он обеспечивает добавление кнопок, проверку их состояния и обработку кликов. Этот класс организует и управляет кнопками, обеспечивая их корректную работу в зависимости от состояния игры и взаимодействия с пользователем.

### Класс «ParameterBuilder»

Класс ParameterBuilder используется для создания строки параметров из коллекции пар ключ-значение. Эта строка может использоваться, например, для передачи параметров в URL-запросе. Этот класс упрощает процесс формирования строки параметров для URL-запросов или других случаев, когда требуется последовательность пар ключ-значение в формате key=value&key2=value2.

### Класс «Tetris1P»

Класс Tetris1P представляет собой игровую доску для одиночной игры в Тетрис. Этот класс наследуется от GameBoard и добавляет функциональность для обработки очков, уровней, кнопок управления и отрисовки графики. Этот класс расширяет функциональность базового игрового поля, добавляя элементы управления и методы для обработки специфических аспектов одиночной игры, таких как уровень и счёт.

### Класс «TetrisFocusListener»

Класс TetrisFocusListener представляет собой слушателя фокуса окна для игры в Тетрис. Он реагирует на потерю фокуса окна, переводя игру в состояние паузы. Этот класс обеспечивает обработку событий фокуса окна для игры в Тетрис, гарантируя, что при потере фокуса игра будет автоматически приостановлена.

### Класс «TetrisKeyListener»

Класс TetrisKeyListener представляет собой слушателя клавиатуры для игры в Тетрис. Он обрабатывает нажатия клавиш и управляет игровым процессом в зависимости от текущего состояния игры. Этот класс обеспечивает взаимодействие с клавиатурой игрока и управление игровым процессом в зависимости от действий игрока и текущего состояния игры.

### Класс «TetrisMouseListener»

Класс TetrisMouseListener представляет собой слушателя мыши для игры в Тетрис. Он обрабатывает события мыши и взаимодействует с кнопками игрового интерфейса в зависимости от их состояния и положения указателя мыши. Этот класс обеспечивает взаимодействие с мышью игрока и обработку событий мыши в игровом интерфейсе, позволяя игроку взаимодействовать с кнопками и элементами игры с помощью мыши.

### Класс «TetrisWindow»

Класс TetrisWindow представляет окно игры Тетрис, отображаемое в пользовательском интерфейсе. Он управляет всеми элементами игры, включая игровое поле, состояние игры, кнопки меню и обработку событий для взаимодействия с игрой. Этот класс является центральным элементом пользовательского интерфейса игры Тетрис, координирующим отображение игровых элементов, управление состоянием игры и взаимодействие с пользователем через события клавиатуры и мыши.

### Класс «Tetromino»

Класс Tetromino представляет тетромино, который является одним из игровых элементов в игре Тетрис. Он представляет собой фигуру, состоящую из четырех квадратных блоков. Этот класс предоставляет функциональность для создания, управления и отображения тетромино в игре Тетрис. Каждое тетромино имеет свой набор ориентаций, тип и текущие координаты на игровом поле.

### Класс «TetrisRunner»

Класс TetrisRunner является точкой входа в игру Тетрис. В целом, класс TetrisRunner инициализирует игровое окно, настраивает его параметры

# Руководство пользователя

## **Запуск приложения**

При запуске клиента первым окном является меню (рис.5.1). В нем присутствует 3 кнопки:

1. Play 1P – запускает непосредственно саму игру «Тетрис»;
2. Play 2p – отвечает за мультиплеерный режим;
3. How to play – отвечает за вывод подсказки по игре.

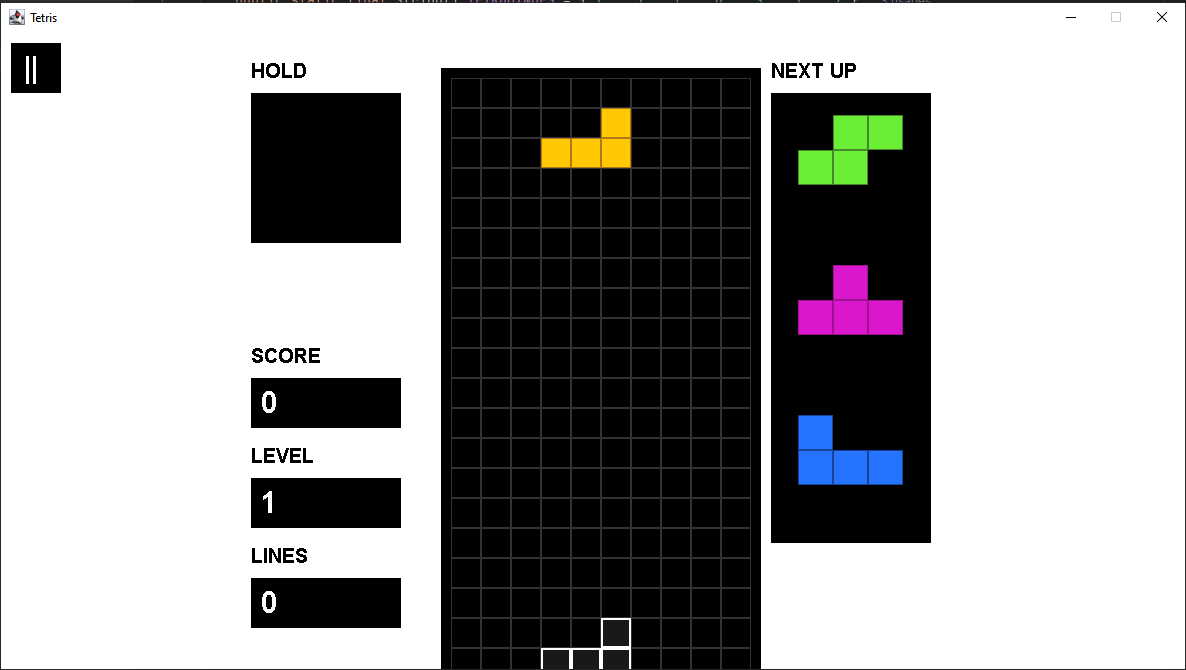
****

**Рисунок 5.1 – Графическое представление меню.**

## **Окно игры**

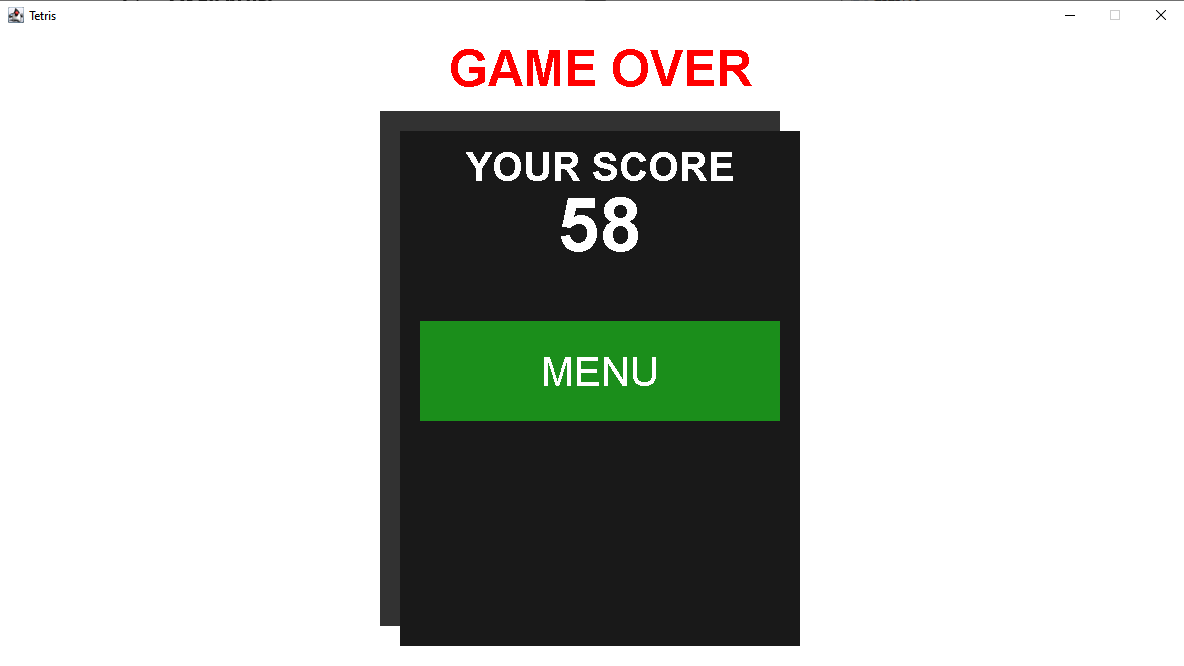
Данное окно является игрой «Тетрис» (рис. 5.2). Что бы передвигать фигуры необходимо нажать одну из следующих кнопок:

1. Стрелка вправо - сдвиг фигуры вправо;
2. Стрелка вниз – сдвиг фигуры вниз;
3. Стрелка вверх – поворот фигуры;
4. Стрелка влево – сдвиг фигуры влево.

**Рисунок 5.2 – Окно игры.**

## **Окно при поражении**

После того, как игрок проиграет в игре, отображается диалоговое окно, которое указывает пользователю о завершении игры и отображает набранное им количество очков.



**Рисунок 5.3 – Окно с окончанием игры.**

### Окно паузы

Так же, при желании, игрок может поставить паузу по время игры и продолжить игру позже.



**Рисунок 5.4 – Окно паузы.**

# ЗАКЛЮЧЕНИЕ

При выполнении данного курсового проекта были были получены навыки разработки многомодульного приложения с пользовательским интерфейсом и работы в среде NetBeans. Получены навыки разработки клиент-серверных приложений.

В результате выполнения данной курсовой работы было разработано модульное приложение игры «Тетрис», позволяющее проверить реакцию пользователя, а также отслеживать лучшие показатели за 5 последних игр.

# СПИСОК ЛИТЕРАТУРЫ

1. Берд, Барри Java для чайников / Барри Берд. – М.: Диалектика / Вильямс, 2013. – **521** с.
2. Кэти Сьерра, Берт Бэйтс Head First Java, Изучаем Java. – ЭКСМО, 2012. – 708 с.
3. Савитч, Уолтер Язык Java. Курс программирования / Уолтер Савитч. – М.: Вильямс, **2015.** – 928 с.
4. Хабибуллин, Ильдар Самоучитель Java / Ильдар Хабибуллин. – М.: БХВ-Петербург, **2014**. – 768 с.

# Приложение A. Листинг программы

# Приложение А.1. «Block.java»

import java.awt.\*;  
  
public class Block {  
 public static final Color  
 EMPTY = Color.black,  
 CYAN = Color.cyan,  
 BLUE = new Color(37,115,255),  
 ORANGE = Color.orange,  
 YELLOW = new Color(255, 255, 0),  
 GREEN = new Color(106, 238, 54),  
 PURPLE = new Color(218,23,204),  
 GRAY = Color.gray,  
 RED = Color.red;  
  
 private Color color;  
  
 private Color getAccent(Color c) {  
 if (color.equals(EMPTY)) {  
 return new Color(50, 50, 50);  
 } else if (color.equals(CYAN)) {  
 return new Color(56,186,241);  
 } else if (color.equals(BLUE)) {  
 return new Color(22,68,149);  
 } else if (color.equals(ORANGE)) {  
 return new Color(174,122,24);  
 } else if (color.equals(YELLOW)) {  
 return new Color(173, 173, 0);  
 } else if (color.equals(GREEN)) {  
 return new Color(69, 134, 45);  
 } else if (color.equals(PURPLE)) {  
 return new Color(158,17,148);  
 } else if (color.equals(RED)) {  
 return new Color(167,0,0);  
 }  
 return Color.black;  
 }  
  
 public Block(Color color) {  
 this.color = color;  
 }  
  
 public Color getColor() {  
 return color;  
 }  
  
 public void draw(Graphics g, int x, int y, int width) {  
 g.setColor(getAccent(color));  
 g.fillRect(x, y, width, width);  
 g.setColor(color);  
 g.fillRect(x + 1, y + 1, width - 2, width - 2);  
 }  
  
 public void drawOutline(Graphics g, int x, int y, int width) {  
 g.setColor(Color.white);  
 g.fillRect(x, y, width, width);  
 g.setColor(new Color(25, 25, 25));  
 g.fillRect(x + 2, y + 2, width - 4, width - 4);  
 }  
}

# Приложение А.2. «GameBoard.java»

import java.awt.\*;  
  
public abstract class GameBoard {  
 TetrisWindow window;  
 String gameState;  
  
 // Constants  
 public static int  
 BLOCK\_WIDTH = 30,  
 TETRION\_WIDTH = 10,  
 TETRION\_HEIGHT = 20;  
  
 // State variables consistent with both 1P and 2P  
 public boolean haveHeldThisTurn = false;  
 public Block[][] Tetrion = new Block[TETRION\_HEIGHT][TETRION\_WIDTH];  
 public boolean movingLeft = false;  
 public boolean movingRight = false;  
 public boolean movingDown = false;  
 public int movingDelay = 0;  
 public boolean oneMoveDone = false;  
 public double tetrominoDelay = 1;  
 public Tetromino[] queue = new Tetromino[3];  
 public Tetromino currentHand;  
 public Tetromino hold;  
 public int linesBroken = 0;  
  
 public GameBoard(TetrisWindow window) {  
 this.window = window;  
  
 for (int i = 0; i < queue.length; i ++) {  
 queue[i] = new Tetromino(Tetromino.getRandomTetromino(), window);  
 }  
 currentHand = new Tetromino(Tetromino.getRandomTetromino(), window);  
 }  
 public void tick() {  
 currentHand.tick();  
  
 if (!oneMoveDone) {  
 if (movingLeft) {  
 currentHand.moveLeft();  
 movingDelay = 200;  
 oneMoveDone = true;  
 }  
 if (movingRight) {  
 currentHand.moveRight();  
 movingDelay = 200;  
 oneMoveDone = true;  
 }  
 if (movingDown) {  
 currentHand.moveDown();  
 movingDelay = 50;  
 oneMoveDone = true;  
 }  
 } else {  
 if (movingDelay <= 0) {  
 if (movingLeft) {  
 currentHand.moveLeft();  
 movingDelay = 50;  
 oneMoveDone = true;  
 }  
 if (movingRight) {  
 currentHand.moveRight();  
 movingDelay = 50;  
 oneMoveDone = true;  
 }  
 if (movingDown) {  
 currentHand.moveDown();  
 movingDelay = 50;  
 oneMoveDone = true;  
 }  
 }  
 else {  
 movingDelay -= TetrisRunner.FPS\_DELAY;  
 }  
 }  
 }  
 public void hold() {  
 if (haveHeldThisTurn) {  
 window.musicUtils.playSoundEffect("noHold.wav");  
 return;  
 }  
 window.musicUtils.playSoundEffect("hold.wav");  
 haveHeldThisTurn = true;  
 if (hold == null) {  
 hold = currentHand;  
 currentHand = queue[0];  
 queue[0] = queue[1];  
 queue[1] = queue[2];  
 queue[2] = new Tetromino(Tetromino.getRandomTetromino(), window);  
 } else {  
 Tetromino oldHold = hold;  
 hold = currentHand;  
 currentHand = new Tetromino(oldHold.type, window);  
 }  
 }  
 public void nextCycle() {  
 currentHand = queue[0];  
 queue[0] = queue[1];  
 queue[1] = queue[2];  
 do {  
 queue[2] = new Tetromino(Tetromino.getRandomTetromino(), window);  
 } while (queue[2].equals(queue[1]));  
  
 while (!currentHand.isValidAtCoordinates()) {  
 currentHand.moveUp();  
 }  
  
 breakLines();  
 }  
  
 public int breakLines() {  
 int lines = 0;  
 for (int row = Tetrion.length - 1; row >= 0; row --) {  
 boolean rowFull = true;  
 boolean isGray = true;  
 for (int col = 0; col < Tetrion[0].length; col ++) {  
 if (Tetrion[row][col] == null) rowFull = false;  
 else if (!Tetrion[row][col].getColor().equals(Block.GRAY)) isGray = false;  
 }  
 if (rowFull && !isGray) {  
 breakLine(row++);  
 lines ++;  
 }  
 }  
 if (lines > 0) window.musicUtils.playSoundEffect("breakLines.wav");  
 linesBroken += lines;  
 return lines;  
 }  
  
 // Private method to break a specific line at a row, used by breakLines()  
 private void breakLine(int row) {  
 for (int i = row; i > 0; i --) {  
 Tetrion[i] = Tetrion[i - 1];  
 }  
 Tetrion[0] = new Block[Tetrion[0].length];  
 }  
  
 public abstract void paintGame(Graphics g);  
 public abstract void lose();  
}

# Приложение А.3. «GameButton.java»

public class GameButton {  
 private int x, y, width, height;  
 private String gameState;  
 private boolean isBeingHighlighted = false;  
 private TetrisWindow window;  
  
 public GameButton(TetrisWindow window, int x, int y, int width, int height, String gameState) {  
 this.window = window;  
 this.x = x;  
 this.y = y;  
 this.width = width;  
 this.height = height;  
 this.gameState = gameState;  
  
 window.buttons.addButton(this);  
 }  
  
 public void highlightIfMouseInRange(int mouseX, int mouseY) {  
 if (mouseX >= x && mouseX <= x + width && mouseY >= y && mouseY <= y + height) {  
 if (!isBeingHighlighted) window.musicUtils.playSoundEffect("move.wav");  
 isBeingHighlighted = true;  
 }  
 else isBeingHighlighted = false;  
 }  
  
 public boolean isHighlighted() {  
 return isBeingHighlighted;  
 }  
  
 public void removeHighlight() {  
 isBeingHighlighted = false;  
 }  
  
 public void click() {  
 window.musicUtils.playSoundEffect("addToBoard.wav");  
 }  
  
 public String getGameState() {  
 return gameState;  
 }  
}

# Приложение А.4. «GameButtons.java»

import java.util.ArrayList;  
  
public class GameButtons {  
 public ArrayList<GameButton> buttons = new ArrayList<GameButton>();  
 private TetrisWindow window;  
  
 public GameButtons(TetrisWindow window) {  
 this.window = window;  
 }  
  
 public void addButton(GameButton b) {  
 if (!buttons.contains(b)) buttons.add(b);  
 }  
  
 public void checkAllButtons(int mouseX, int mouseY) {  
 for (GameButton b : buttons) {  
 if (b.getGameState().equals(window.gameState)) b.highlightIfMouseInRange(mouseX, mouseY);  
 }  
 }  
  
 public void clickAllButtons() {  
 for (GameButton b : buttons) {  
 if (b.getGameState().equals(window.gameState) && b.isHighlighted()) {  
 b.click();  
 b.removeHighlight();  
 }  
 }  
 }  
}

# Приложение А.5. «GraphicsUtils.java»

import java.awt.\*;  
  
public class GraphicsUtils {  
 public static void centerString(Graphics g, String s, int XPos, int YPos){  
 int stringLen = (int) g.getFontMetrics().getStringBounds(s, g).getWidth();  
 int start = -stringLen / 2;  
 g.drawString(s, start + XPos, YPos);  
 }  
  
 public static void multilineString(Graphics g, String text, int x, int y) {  
 for (String line : text.split("\n"))  
 g.drawString(line, x, y += g.getFontMetrics().getHeight());  
 }  
}

# Приложение А.6. «ParameterBuilder.java»

import java.util.HashMap;  
import java.util.Map;  
  
public class ParameterBuilder {  
 public static String build(HashMap<String, String> data) {  
 String output = "";  
 boolean firstComplete = false;  
  
 for (Map.Entry<String, String> entry : data.entrySet()) {  
 String key = entry.getKey();  
 String value = entry.getValue();  
  
 if (!firstComplete) {  
 output += key + "=" + value;  
 firstComplete = true;  
 } else output += "&" + key + "=" + value;  
 }  
  
 return output;  
 }  
}

# Приложение А.7. «Tetris1P.java»

import java.util.HashMap;  
import java.util.Map;  
  
public class ParameterBuilder {  
 public static String build(HashMap<String, String> data) {  
 String output = "";  
 boolean firstComplete = false;  
  
 for (Map.Entry<String, String> entry : data.entrySet()) {  
 String key = entry.getKey();  
 String value = entry.getValue();  
  
 if (!firstComplete) {  
 output += key + "=" + value;  
 firstComplete = true;  
 } else output += "&" + key + "=" + value;  
 }  
  
 return output;  
 }  
}

# Приложение А.8. «TetrisFocusListener.java»

import java.awt.event.FocusEvent;  
import java.awt.event.FocusListener;  
  
public class TetrisFocusListener implements FocusListener {  
 TetrisWindow window;  
  
 public TetrisFocusListener(TetrisWindow window) {  
 this.window = window;  
 }  
  
 @Override  
 public void focusLost(FocusEvent e) {  
 if (window.gameState.equals("1p")) window.gameState = "paused";  
 }  
  
 @Override  
 public void focusGained(FocusEvent e) {  
  
 }  
}

# Приложение А.9. «TetrisKeyListener.java»

import java.awt.event.KeyEvent;  
import java.awt.event.KeyListener;  
  
public class TetrisKeyListener implements KeyListener {  
 private TetrisWindow window;  
  
 public TetrisKeyListener(TetrisWindow window) {  
 this.window = window;  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 if ((window.gameState.equals("1p") || window.gameState.equals("2p")) && window.gameBoard.gameState.equals("game")) {  
 if (e.getKeyCode() == KeyEvent.VK\_UP) window.gameBoard.currentHand.rotate();  
 else if (e.getKeyCode() == KeyEvent.VK\_DOWN) window.gameBoard.movingDown = true;  
 else if (e.getKeyCode() == KeyEvent.VK\_LEFT) window.gameBoard.movingLeft = true;  
 else if (e.getKeyCode() == KeyEvent.VK\_RIGHT) window.gameBoard.movingRight = true;  
 else if (e.getKeyCode() == KeyEvent.VK\_SPACE) {  
 window.gameBoard.movingLeft = false;  
 window.gameBoard.movingRight = false;  
 window.gameBoard.currentHand.jumpScore();  
 window.gameBoard.currentHand.addToBoard();  
 } else if (e.getKeyCode() == KeyEvent.VK\_C) {  
 window.gameBoard.movingLeft = false;  
 window.gameBoard.movingRight = false;  
 window.gameBoard.hold();  
 }  
 }  
  
 if (window.gameState.equals("1p") && window.gameBoard.gameState.equals("game")) {  
 if (e.getKeyCode() == KeyEvent.VK\_ESCAPE) {  
 window.gameState = "paused";  
 }  
 }  
 else if ((window.gameState.equals("1p") || window.gameState.equals("2p")) && window.gameBoard.gameState.equals("game\_over")) {  
 if (e.getKeyCode() == KeyEvent.VK\_ENTER) {  
 window.gameState = "menu";  
 }  
 }  
 else if (window.gameState.equals("paused")) {  
 if (e.getKeyCode() == KeyEvent.VK\_ESCAPE || e.getKeyCode() == KeyEvent.VK\_ENTER) {  
 window.gameState = "1p";  
 }  
 }  
 }  
  
 @Override  
 public void keyTyped(KeyEvent e) {  
  
 }  
  
 @Override  
 public void keyReleased(KeyEvent e) {  
 if (window.gameState.equals("1p") || window.gameState.equals("2p")) {  
 if (e.getKeyCode() == KeyEvent.VK\_LEFT) {  
 window.gameBoard.movingLeft = false;  
 window.gameBoard.movingDelay = 50;  
 if (!window.gameBoard.oneMoveDone) window.gameBoard.currentHand.moveLeft();  
 window.gameBoard.oneMoveDone = false;  
 }  
 else if (e.getKeyCode() == KeyEvent.VK\_RIGHT) {  
 window.gameBoard.movingRight = false;  
 window.gameBoard.movingDelay = 50;  
 if (!window.gameBoard.oneMoveDone) window.gameBoard.currentHand.moveRight();  
 window.gameBoard.oneMoveDone = false;  
 }  
 else if (e.getKeyCode() == KeyEvent.VK\_DOWN) {  
 window.gameBoard.movingDown = false;  
 window.gameBoard.movingDelay = 50;  
 if (!window.gameBoard.oneMoveDone) window.gameBoard.currentHand.moveDown();  
 window.gameBoard.oneMoveDone = false;  
 }  
 }  
 }  
}

# Приложение А.10. «TetrisMouseListener.java»

import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
import java.awt.event.MouseMotionListener;  
  
public class TetrisMouseListener implements MouseListener, MouseMotionListener {  
 private TetrisWindow window;  
  
 public TetrisMouseListener(TetrisWindow window) {  
 this.window = window;  
 }  
  
 @Override  
 public void mouseClicked(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mousePressed(MouseEvent e) {  
 window.buttons.clickAllButtons();  
 }  
  
 @Override  
 public void mouseReleased(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseEntered(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseExited(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseDragged(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseMoved(MouseEvent e) {  
 window.buttons.checkAllButtons(e.getX(), e.getY());  
 }  
}

# Приложение А.11. «TetrisWindow.java»

import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
import java.awt.event.MouseMotionListener;  
  
public class TetrisMouseListener implements MouseListener, MouseMotionListener {  
 private TetrisWindow window;  
  
 public TetrisMouseListener(TetrisWindow window) {  
 this.window = window;  
 }  
 @Override  
 public void mouseClicked(MouseEvent e) {  
  
 }  
 @Override  
 public void mousePressed(MouseEvent e) {  
 window.buttons.clickAllButtons();  
 }  
 @Override  
 public void mouseReleased(MouseEvent e) {  
  
 }  
 @Override  
 public void mouseEntered(MouseEvent e) {  
  
 }  
 @Override  
 public void mouseExited(MouseEvent e) {  
  
 }  
 @Override  
 public void mouseDragged(MouseEvent e) {  
  
 }  
 @Override  
 public void mouseMoved(MouseEvent e) {  
 window.buttons.checkAllButtons(e.getX(), e.getY());  
 }  
}

# Приложение А.12. «Tetromino.java»

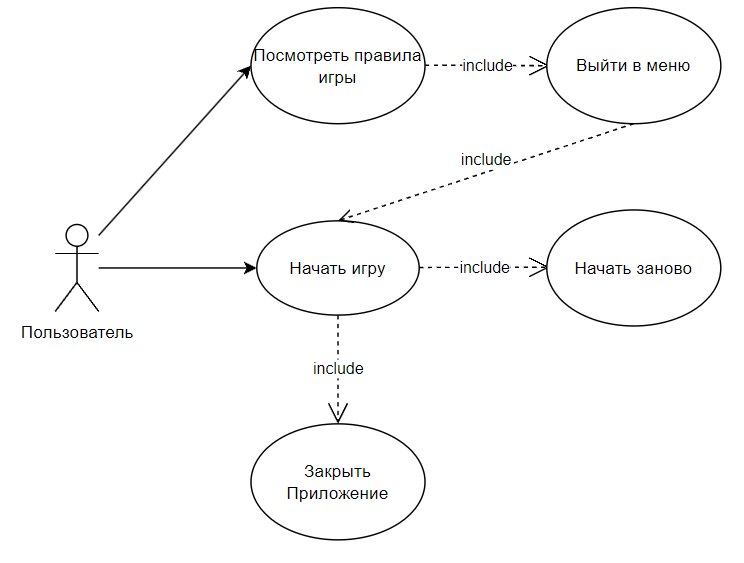
import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
import java.awt.event.MouseMotionListener;  
  
public class TetrisMouseListener implements MouseListener, MouseMotionListener {  
 private TetrisWindow window;  
  
 public TetrisMouseListener(TetrisWindow window) {  
 this.window = window;  
 }  
 @Override  
 public void mouseClicked(MouseEvent e) {  
  
 }  
 @Override  
 public void mousePressed(MouseEvent e) {  
 window.buttons.clickAllButtons();  
 }  
 @Override  
 public void mouseReleased(MouseEvent e) {  
  
 }  
 @Override  
 public void mouseEntered(MouseEvent e) {  
  
 }  
 @Override  
 public void mouseExited(MouseEvent e) {  
  
 }  
 @Override  
 public void mouseDragged(MouseEvent e) {  
  
 }  
 @Override  
 public void mouseMoved(MouseEvent e) {  
 window.buttons.checkAllButtons(e.getX(), e.getY());  
 }  
}

# Приложение А.13. «Tetrisrunner.java»

import javax.swing.\*;  
  
public class TetrisRunner {  
 public static final int WIDTH = 1200, HEIGHT = 675, FPS\_DELAY = 10;  
 public static final String[] TETROMINOES = {"I", "J", "L", "O", "S", "T", "Z"};  
  
 public static void main(String[] args) {  
 JFrame f = new JFrame("Tetris");  
 f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
  
 TetrisWindow w = new TetrisWindow(f);  
 f.add(w);  
 f.setSize(WIDTH, HEIGHT);  
 f.setResizable(false);  
 f.setLocationRelativeTo(null);  
 f.setVisible(true);  
  
 Runtime.getRuntime().addShutdownHook(new Thread() {  
 public void run() {  
 if (w.gameBoard instanceof Tetris2P) ((Tetris2P) w.gameBoard).disconnect();  
 }  
 });  
 }  
}

# Приложение Б. UML-диаграммы

## **Приложение Б.1. UML-диаграмма вариантов использования**



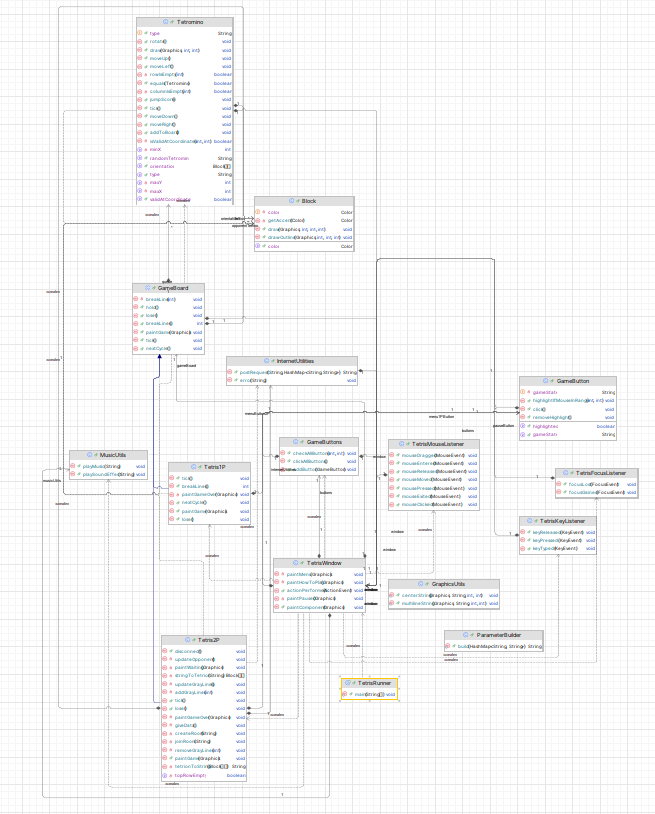
**Рисунок Б.1 – UML-диаграмма вариантов использования.**

## **Приложение Б.2. UML- диаграмма деятельности**



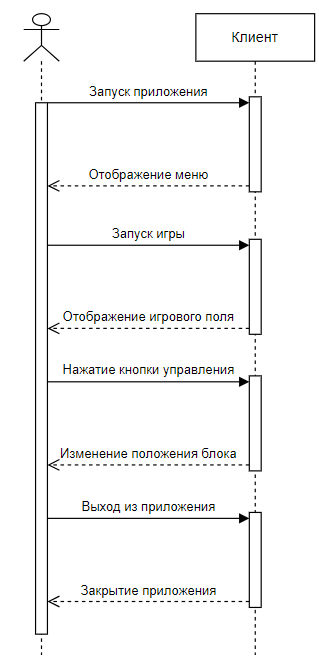
**Рисунок Б.2 – UML-диаграмма деятельности.**

## **Приложение Б.3. UML-диаграмма классов**



**Рисунок Б.3 – UML-диаграмма классов.**

## **Приложение Б.4. UML-диагама последовательности**



**Рисунок Б.4 – UML-диаграмма последовательности.**