Министерство науки и высшего образования

Российской Федерации

Пензенский Государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовой работе

По курсу «Программирование на языке Java»

На тему «Разработка многозадачного приложения на языке Java»

Выполнил студент группы 19ВВ2:

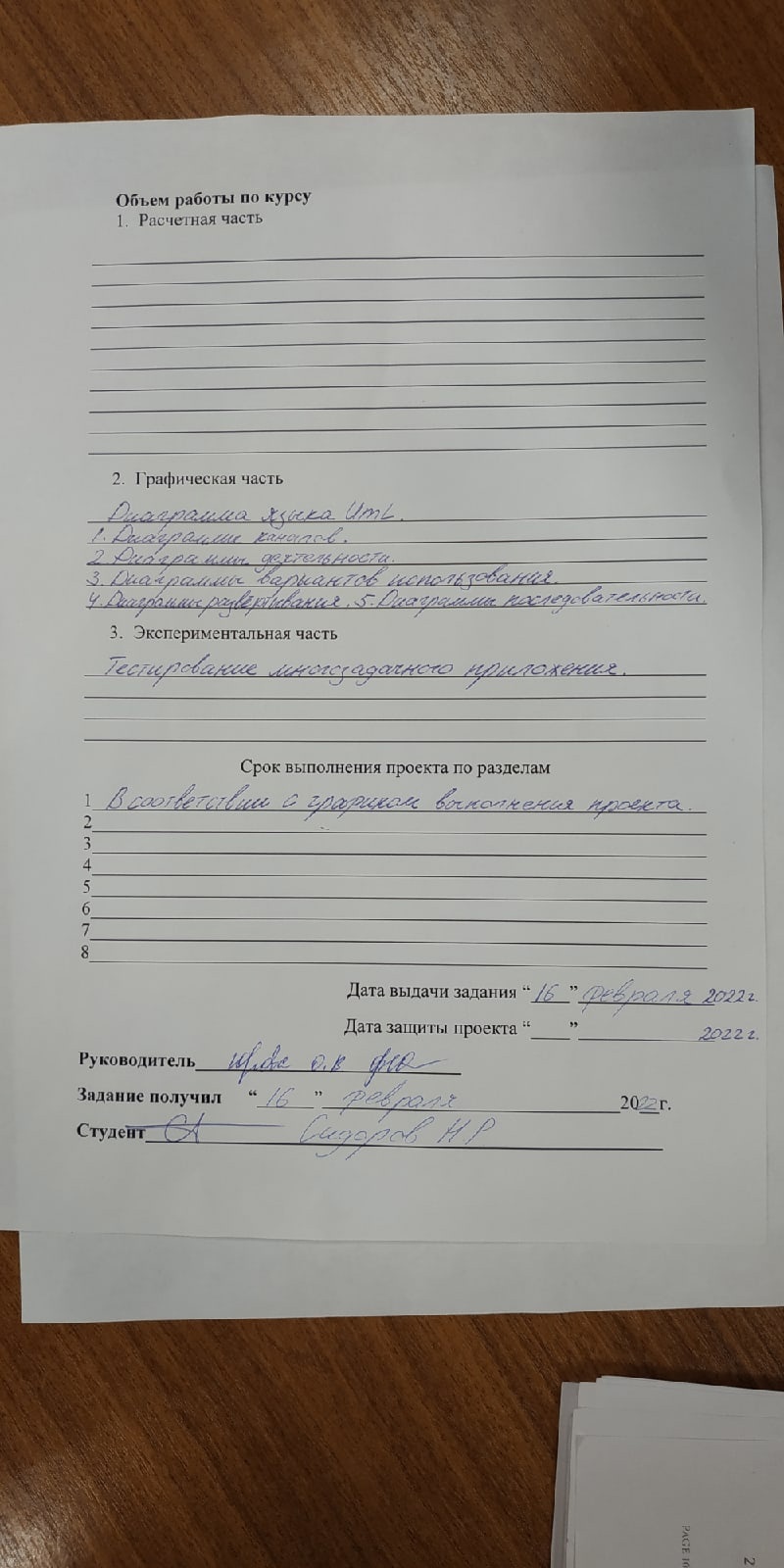
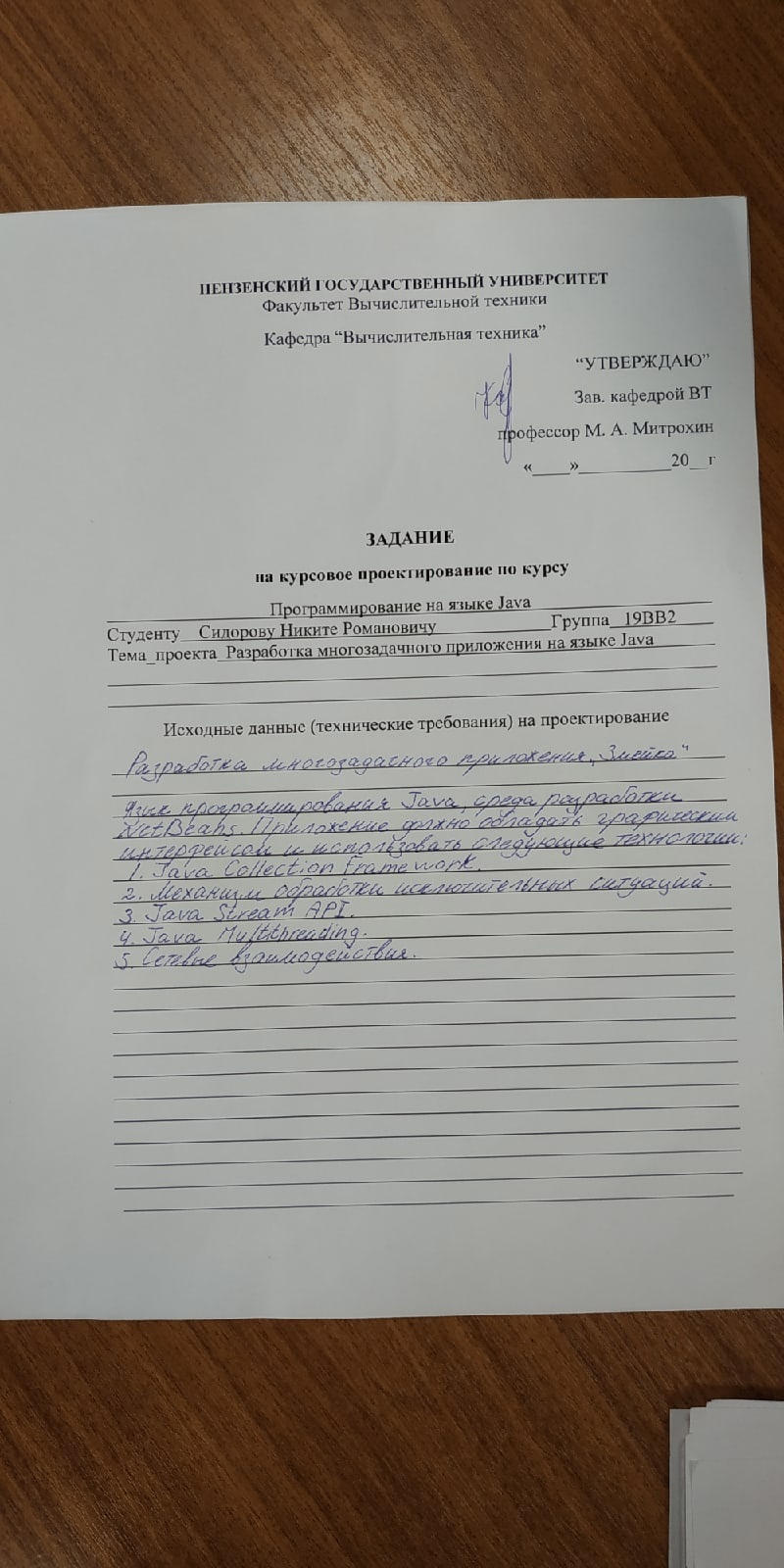
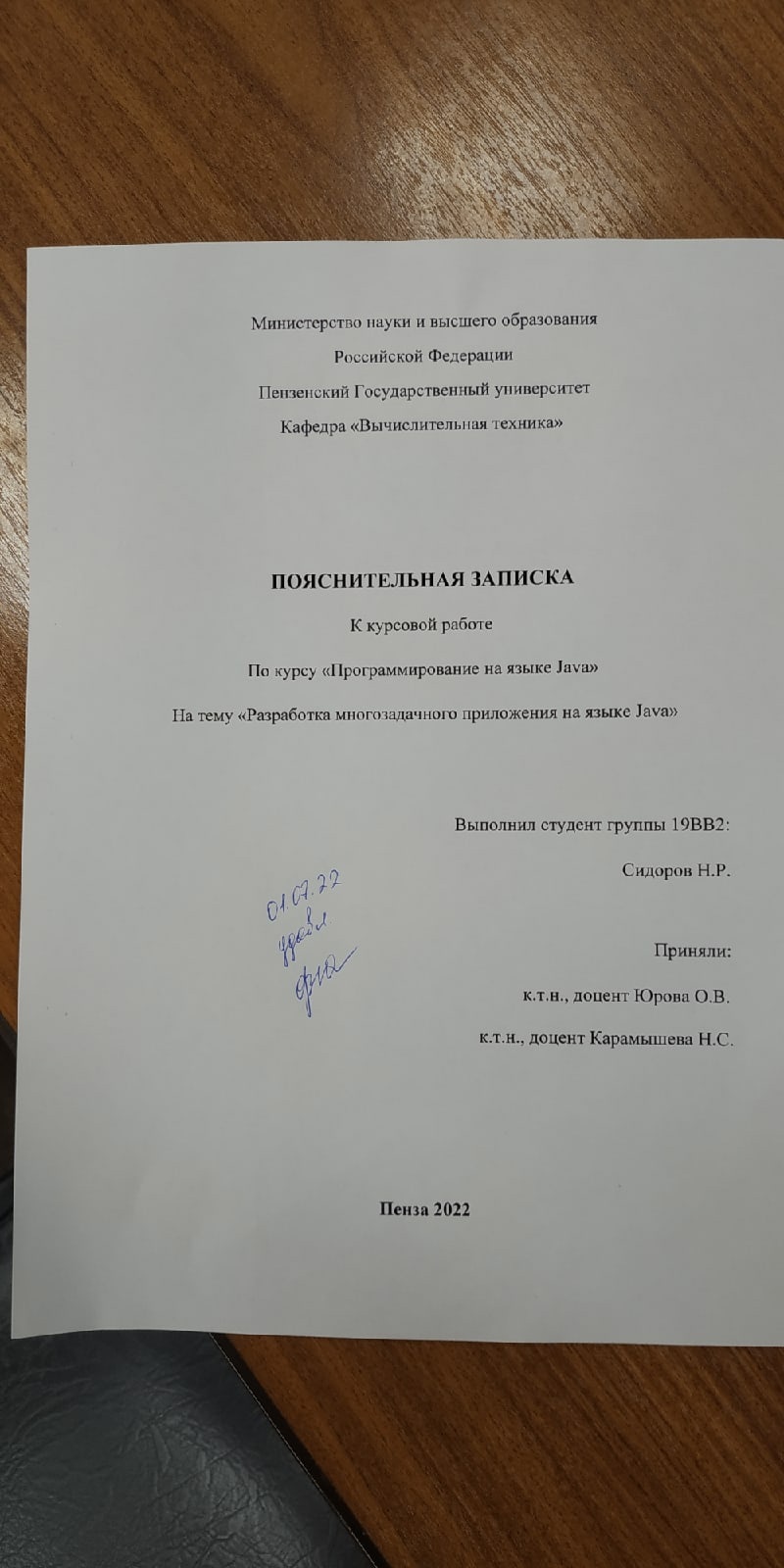
Сидоров Н.Р.

Приняли:

к.т.н., доцент Юрова О.В.

к.т.н., доцент Карамышева Н.С.

**Пенза 2022**



**Содержание**

[Введение 6](#_Toc107522260)

[Постановка задачи 9](#_Toc107522261)

[Выбор решения 10](#_Toc107522262)

[Описание программы 11](#_Toc107522263)

[Организация пользовательского интерфейса 13](#_Toc107522264)

[Экспериментальная часть 15](#_Toc107522265)

[Заключение 20](#_Toc107522266)

[Список используемой литературы. 21](#_Toc107522267)

[Приложение А. Код приложения 22](#_Toc107522268)

[Приложение В. UML-диаграммы приложения 42](#_Toc107522270)



# Введение

Java – язык программирования общего назначения. То есть язык, который применяется в разработке различных программных продуктов, без четкой специализации в конкретной сфере. Он во многом похож на Python, JavaScript и другие языки того же уровня, что и Java. Кроме того, Java заимствует массу синтаксических конструкций из C и C++.

Вся структура строится вокруг объектов, классов и прочих формальных сущностей, принятых в сообществе программистов за стандарт разработки ООП. Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины.

Язык Java имеет ряд достоинств:

Простота – первое техническое достоинство Java. У него чёткие синтаксические правила и понятная семантика. Рациональность и краткость полезны для обработки кода машинами с ограниченным объёмом ресурсов. Для встроенных устройств создана специальная платформа Java Micro Edition.

Объектно-ориентированный подход. За 3 десятилетия он доказал свою эффективность. Суть состоит в том, что в центре внимания находятся данные (объекты), интерфейсы и алгоритмы вторичны. Другими словами, мы отталкиваемся от результата при выборе инструментов, способов их применения.

Безопасность. Важнейший критерий, учитывая применение языка в сетевых / распределенных средах. Разработчики проделали большую работу по защите платформы Java. И она продолжается. Обойти или взломать механизмы защиты крайне сложно. Пример: использование классов, имеющих цифровую подпись. Полные права предоставляются только при полном доверии автору класса.

Производительность. Изначально она вызывала вопросы. Новые версии динамических компиляторов Java не уступают традиционным из других платформ. Мощный прирост скорости обработки даёт оптимизация тех фрагментов кода, которая исполняется чаще. При необходимости те или иные приёмы оптимизации включаются или отменяются JIT-компилятором.

Надёжность – одно из важнейших преимуществ. Программы на Java стабильно работают в любых условиях. Компилятор способен выявить ошибки ещё до выполнения кода, то есть на ранних стадиях. Контроль выполнения позволяет предотвратить сбои в памяти (например, из-за неточного указателя). Сами указатели можно применять не везде, а только там, где это необходимо (скажем, в работе со связными списками).

Независимость от аппаратной части и ОС. Важно лишь наличие исполняющей среды и JVM. А компьютерная архитектура в целом значения не имеет. Байт-код легко интерпретируется на любой машине. Подход доказал свою состоятельность во многом благодаря динамической компиляции. Кроссплатформенностью отличается также интерфейс, реализованный в системных библиотеках.

Динамичность и адаптируемость. Эта особенность позволяет Java не теряться в постоянно изменяющейся среде. При необходимости можно добавить в библиотеки новые объекты, методы. При этом трогать приложения, использующие данные библиотеки не нужно. Отследить информацию о структуре объектов, их поведении, о ходе выполнения программы очень легко.

Удобные и эффективные сетевые возможности. Приложения умеют находить нужные объекты в сети и открывать к ним доступ. Причём так же легко, как будто мы имеем дело с локальной файловой системой. Имеется обширная программная библиотека для передачи данных по самым распространённым протоколам: FTP, HTTP, TCP/IP. Работает механизм вызова удалённых методов.

Не надо забывать, что Java – это трио, состоящее из языка программирования, мощного универсального обработчика и внушительной библиотеки. Все эти наработки доступны программистам. Им не нужно разрабатывать с нуля многие необходимые процедуры (доступ к сети, базам данных и т.п.).

# Постановка задачи

Разработать многозадачное приложение «Змейка».

Функции сервера:

● Работа в отдельном потоке;

● Реализация многопоточности;

● Своевременное окончание сессии.

Функции клиента:

● Обработка данных пользователя;

● Возврат результата в соответствии с принятыми данными.

Приложение должно обладать графическим интерфейсом и использовать следующие технологии:

● Java Collections Framework;

● Механизм обработки исключительных ситуаций;

● Java Stream API;

● Java Multithreading;

● Сетевое взаимодействие.

ОС – Windows. Язык программирования: Java. Среда разработки NetBeans.

# Выбор решения

Частью курсовой работы является передача сообщений от клиента к серверу. Для реализации этого используются TCP-сокеты.

Сокет — один конец двустороннего канала связи между двумя программами, работающими в сети. Соединив вместе два сокета, можно передавать данные между разными процессами (локальными или удаленными). Для сокетов жизнь дескриптора можно разделить на три фазы: открыть (создать) сокет, получить из сокета или отправить сокету и в конце концов закрыть сокет.

Клиент соединяется с сервером с помощью библиотеки *java.net.Socket*.

Интерфейс пользователя реализован с помощью библиотеки Java Swing.

# Описание программы

MainWindow.java - файл, реализующий главное окно многозадачного приложения, а также инициирующий метод GameField. MainWindow содержит серверную часть, с помощью которой в программе реализована многопоточность.

MainWindow() - метод, который обеспечивает работу приложения. Именно к нему обращается класс Menu при запуске.

Menu.java - файл, реализующий меню игры. Содержит функцию Main.

Menu составляющие:

● jButton1ActionPerformed – кнопка, создающая новое окно посредством метода MainWindow. Запуск игры;

● jButton2ActionPerformed – кнопка, закрывающая приложение.

Server.java – файл, содержащий класс Server(), реализующий работу сервера как «со стороны сервера», так и «со стороны клиента».

GameField.java – файл, содержащий метод GameField, реализующий техническую и графическую части игры «Змейка», а также осуществляющий работу с файлами в процессе расчета «рекорда» игрока. Последняя функция метода GameField, описанная ранее, реализована на «клиентской части» сервера. Вычисления происходят параллельно, со вторым, новым потоком «серверной части».

Метод GameField:

● initGame() – запускает генерацию элементов игры и отсчёт таймера;

● create…() – создаёт элементы игрового поля для взаимодействия;

● loadImages() – загрузка графических файлов;

● paintComponent(Graphics g) – использует графические файлы;

● move() – перемещение змейки;

● checkApple () – регистрайия взаимодействия змейки с яблоком;

● checkCollisions()– регистрайия взаимодействия змейки с препятствием;

● actionPerformed(ActionEvent e) – обновление игрового поля;

● class FieldKeyListener - реализация управления с клавиатуры;

● Audio.java – файл, содержащий класс Audio. Позволяет загружать, использовать музыку и звуковые эффекты. Повсеместно используется в приложении.

# Организация пользовательского интерфейса

В качестве среды разработки была выбрана программа ApacheNetBeanbs. Программа предоставляет все средства, необходимые при разработке графического приложения. А именно библиотеку JavaSwing.

GUI Builder в среде IDE разрешает основные проблемы, возникающие при создании графического интерфейса Java путем рационализации процесса создания графических интерфейсов, освобождая разработчиков от необходимости изучения особенностей диспетчеров компоновки Swing. Это выполняется путем расширения возможностей конструктора графического интерфейса пользователя IDE NetBeans для поддержки простой парадигмы "Произвольная структура" с простыми правилами компоновки, понятными и простыми в использовании. В процессе проектирования формы GUI Builder предоставляет визуальные средства поддержки, предлагая оптимальное расположение и выравнивание компонентов. GUI Builder способствует переносу пользовательских решений по разработке в функциональный пользовательский интерфейс, реализуемый при помощи диспетчера компоновки GroupLayout и других средств Swing. Благодаря динамической модели размещения компонентов поведение графического интерфейса в GUI Builder во время выполнения соответствует ожидаемому, что позволяет вносить корректировки без изменения установленных взаимосвязей между компонентами. При каждом изменении размеров форм, переключении локалей или применении нового общего стиля графический интерфейс автоматически изменяется в соответствии с новой настройкой вставок и смещений стиля.

В своем приложении я использовал 2 окна (Jframe). Окна являются основой пользовательского интерфейса любой операционной системы. Они визуально разделяют выполняемые в среде приложения. Окна, используемые в библиотеке Swing, представляют собой окна операционной системы.

Окно JFrame наследует свойства класса JWindow и представляет собой наиболее часто используемое в приложениях окно «общего назначения».

Заголовок (jLable1) используется для загрузки графического файла.

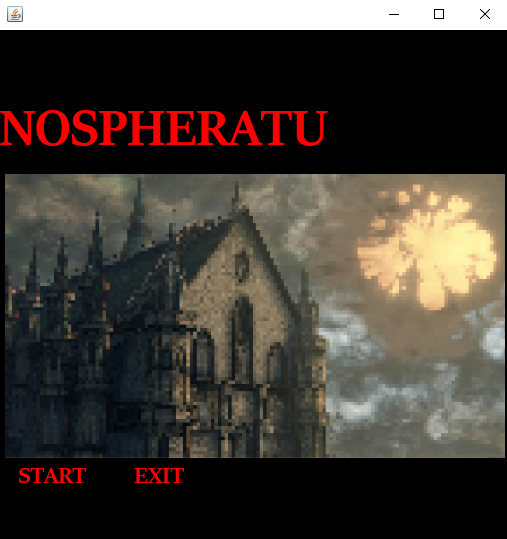
Сверху расположен Заголовок (jLable2).

Также были использованы кнопки (jButton) «START», «EXIT».

Библиотека Swing включает абсолютно все придуманные на сегодняшний день элементы управления. К ним относятся кнопки, флажки, переключатели, меню и его элементы, и многое другое. Все эти элементы в библиотеке связаны, поскольку они унаследованы от абстрактного класса AbstractButton, определяющего поведение любого компонента, претендующего на звание элемента управления. Кнопки JButton кроме собственного внешнего вида не включают практически ничего уникального. Поэтому всё, что верно для кнопок, будет верно и для остальных элементов управления. Основное время работы с кнопками связано не столько с их созданием и настройкой, сколько с размещением в контейнере и написанием обработчиков событий. Внешний вид кнопок JButton можно легко изменить, не меняя менеджера внешнего вида и поведения. С интерфейсом кнопок можно делать практически все — сопоставлять каждому действию пользователя своё изображение, убирать рамку, закрашивать в любой цвет, перемещать содержимое по разным углам.

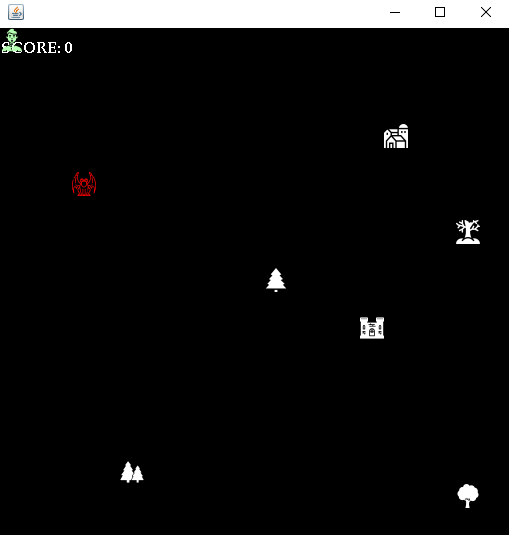
# Экспериментальная часть

Запускаем приложение:



**Рисунок 1 – Запуск приложения**

Нажимаем кнопку «START», «Змейка»:



**Рисунок 2 – Запуск игры «Змейка»**

В случае нажатия кнопки «EXIT», произойдет закрытие приложения.

Игровой процесс:



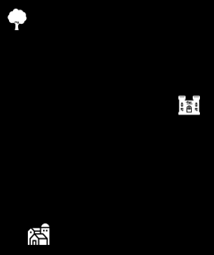
**Рисунок 3 – Игровой процесс**

В левом верхнем углу окна находится счетчик набранных игроком очков (По 10 за каждое «яблоко»):



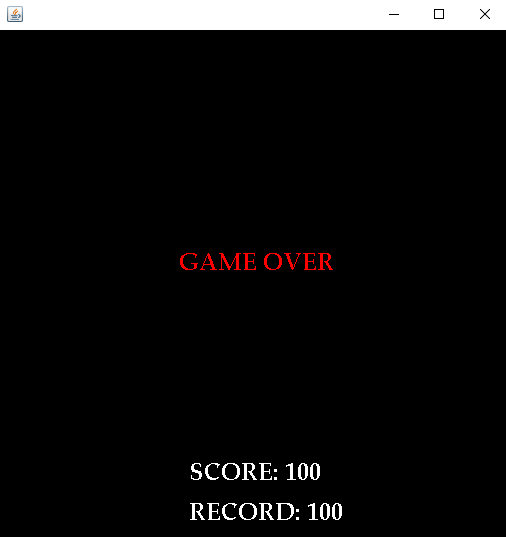
**Рисунок 4 – Счетчик очков**

Также, помимо границ экрана, на игровом поле опасность игроку представляют случайно генерируемые препятствия:



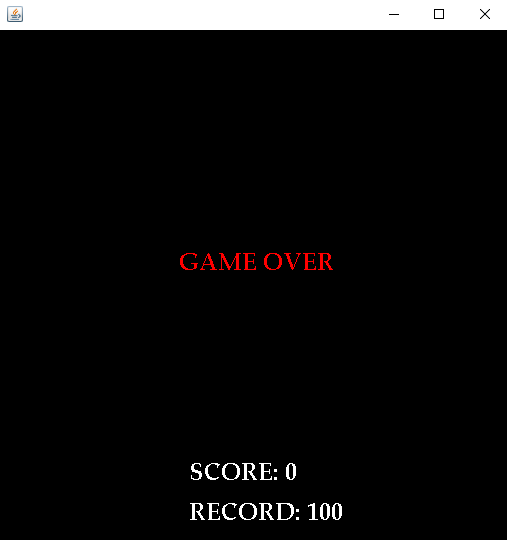
**Рисунок 5 – Препятствия**

В случае проигрыша, игрок видит сообщение «GAME OVER», Кол-во набранных очков и свой рекорд (Рис. 6-8):

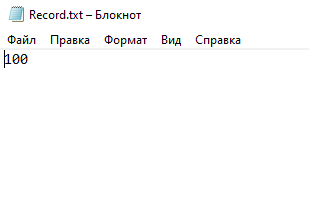


**Рисунок 6 – Проигрыш**

Сохранение рекорда происходит посредством записи в файл и чтения:



**Рисунок 7 – Проигрыш**



**Рисунок 8 – Проигрыш**

Ожидание в меню, игровой процесс, нажатие кнопок и взаимодействия змейки сопровождаются соответствующими звуками и музыкой.

# Заключение

При выполнении данной курсовой работы были получены навыки разработки программ на языке высокого уровня Java. Были освоены навыки разработки графического интерфейса с помощью библиотеки Swing. Изучены возможности среды разработки NetBeans. Освоены навыки разработки клиент-серверного приложения.

В рамках данной работы было написано приложение «Змейка».

Данная реализация не лишена недочетов визуальной части, присутствуют некритические баги.

# Список используемой литературы.

1. Берд, Барри Java для чайников / Барри Берд. - М.: Диалектика / Вильямс, 2013. - 521 c.

2. Гурвиц Г. Разработка реального приложения в среде клиент-сервер –

ДВГУПС, 2005, 206 с.

3. Дубаков А.А. Сетевое программирование: учебное пособие / А.А. Дубаков – СП: НИУ ИТМО, 2013. – 248 с.

4. Мейер Б. Объектно-ориентированное программирование и программная инженерия / Мейер Б.— Электрон. текстовые данные.— М.: Интернет Университет Информационных Технологий (ИНТУИТ), 2016.— 285с.

# Приложение А. Код приложения

# Приложение А.1 - MainWindow.java

package SnakeGameCourse;

import java.io.IOException;

import java.net.ServerSocket;

import javax.swing.\*;

/\*\*

\* Created by infuntis on 15/01/17.

\*/

public class MainWindow extends JFrame {

public MainWindow(){

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setSize(525,550);

setLocation(600,300);

add(new GameField());

setVisible(true);

}

public static void main(String[] args) throws IOException {

MainWindow mw = new MainWindow();

//Server

try(ServerSocket server = new ServerSocket(8000);){

System.out.println("Server started!");

while(true)

try(Server record = new Server(server)){

new Thread(() -> {

System.out.println("multithreading");

}).start();

}catch (NullPointerException e){

e.printStackTrace();

}

}catch (IOException e){

throw new RuntimeException(e);

}

}

}

**Приложение А.2 – GameField.java**

package SnakeGameCourse;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Random;

import java.util.Scanner;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.stream.Collectors;

import java.util.stream.Stream;

public class GameField extends JPanel implements ActionListener{

private final int SIZE = 480;

private final int DOT\_SIZE = 24;

private final int ALL\_DOTS = 480;

private Image dot;

private Image apple;

private Image tree1;

private Image tree2;

private Image tree3;

private Image tree4;

private Image castle;

private Image castle1;

private int speed = 150;

private int forR;

private int Record;

private int appleX;

private int appleY;

private int tree1X;

private int tree1Y;

private int tree2X;

private int tree2Y;

private int tree3X;

private int tree3Y;

private int tree4X;

private int tree4Y;

private int castle1X;

private int castle1Y;

private int castleX;

private int castleY;

private int apple11;

private int tree11;

private int tree22;

private int tree33;

private int tree44;

private int castle11;

private int castle12;

private int score;

private int[] x = new int[ALL\_DOTS];

private int[] y = new int[ALL\_DOTS];

private int dots;

private Timer timer;

private boolean left = false;

private boolean right = true;

private boolean up = false;

private boolean down = false;

private boolean inGame = true;

public static Audio khit;

public static Audio hit;

public static Audio ost;

public GameField(){

setBackground(Color.black);

loadImages();

initGame();

addKeyListener(new FieldKeyListener());

setFocusable(true);

}

public void initGame(){

khit = new Audio("res/khit.wav", 0.75);

hit = new Audio("res/hit.wav", 0.75);

ost = new Audio("res/ost.wav", 0.7);

dots = 1;

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

x[i] = 48 - i\*DOT\_SIZE;

y[i] = 48;

}

if(inGame){

ost.play();

ost.setVolume();

}

timer = new Timer(speed,this);

timer.start();

createApple();

createTree1();

createTree2();

createTree3();

createTree4();

createCastle();

createCastle1();

}

public void createApple(){

if (apple11 != (tree11 | tree22 | tree33 | tree44 | castle11 | castle12)){

appleX = new Random().nextInt(20)\*DOT\_SIZE;

appleY = new Random().nextInt(20)\*DOT\_SIZE;

apple11 = appleX + appleY;

}

}

public void createTree1(){

tree1X = new Random().nextInt(20)\*DOT\_SIZE;

tree1Y = new Random().nextInt(20)\*DOT\_SIZE;

tree11 = tree1X + tree1Y;

}

public void createTree2(){

tree2X = new Random().nextInt(20)\*DOT\_SIZE;

tree2Y = new Random().nextInt(20)\*DOT\_SIZE;

tree22 = tree2X + tree2Y;

}

public void createTree3(){

tree3X = new Random().nextInt(20)\*DOT\_SIZE;

tree3Y = new Random().nextInt(20)\*DOT\_SIZE;

tree33 = tree3X + tree3Y;

}

public void createTree4(){

tree4X = new Random().nextInt(20)\*DOT\_SIZE;

tree4Y = new Random().nextInt(20)\*DOT\_SIZE;

tree44 = tree4X + tree4Y;

}

public void createCastle(){

castleX = new Random().nextInt(20)\*DOT\_SIZE;

castleY = new Random().nextInt(20)\*DOT\_SIZE;

castle11 = castleX + castleY;

}

public void createCastle1(){

castle1X = new Random().nextInt(20)\*DOT\_SIZE;

castle1Y = new Random().nextInt(20)\*DOT\_SIZE;

castle12 = castle1X + castle1Y;

}

public void loadImages(){

ImageIcon iia = new ImageIcon("apple.png");

apple = iia.getImage();

ImageIcon iid = new ImageIcon("dot.png");

dot = iid.getImage();

ImageIcon iif = new ImageIcon("tree1.png");

tree1 = iif.getImage();

ImageIcon iig = new ImageIcon("tree2.png");

tree2 = iig.getImage();

ImageIcon iih = new ImageIcon("tree3.png");

tree3 = iih.getImage();

ImageIcon iij = new ImageIcon("tree4.png");

tree4 = iij.getImage();

ImageIcon iic = new ImageIcon("castle.png");

castle = iic.getImage();

ImageIcon iiu = new ImageIcon("castle1.png");

castle1 = iiu.getImage();

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

if(inGame){

String str1 = "SCORE: " + score;

g.setColor(Color.white);

g.setFont(new Font("Palatino Linotype", Font.BOLD, 16));

g.drawString(str1,1,25);

g.drawImage(apple,appleX,appleY,this);

g.drawImage(tree1,tree1X,tree1Y,this);

g.drawImage(tree2,tree2X,tree2Y,this);

g.drawImage(tree3,tree3X,tree3Y,this);

g.drawImage(tree4,tree4X,tree4Y,this);

g.drawImage(castle,castleX,castleY,this);

g.drawImage(castle1,castle1X,castle1Y,this);

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

g.drawImage(dot,x[i],y[i],this);

}

}

if(inGame == false){

/\*//Client

try (Server record = new Server("127.0.0.1", 8000)){

System.out.println("thread");

if (forR < score){

forR = score;

}

try {

FileInputStream fis = new FileInputStream("Record.txt");

Scanner sc = new Scanner(fis);

while(sc.hasNext()) {

Record = sc.nextInt();

}

}catch (FileNotFoundException ex) {

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

}

if (Record < forR){

try (FileWriter fileWriter = new FileWriter("Record.txt")) {

fileWriter.write(String.valueOf(forR));

fileWriter.flush();

}catch (IOException ex) {

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

}

}

} catch (IOException e){

e.printStackTrace();

}\*/

if (forR < score){

forR = score;

}

try {

FileInputStream fis = new FileInputStream("Record.txt");

Scanner sc = new Scanner(fis);

while(sc.hasNext()) {

Record = sc.nextInt();

}

}catch (FileNotFoundException ex) {

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

}

if (Record < forR){

try (FileWriter fileWriter = new FileWriter("Record.txt")) {

fileWriter.write(String.valueOf(forR));

fileWriter.flush();

}catch (IOException ex) {

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

}

}

String str = Stream.of("GAME"," ","OVER").collect(Collectors.joining());

g.setColor(Color.red);

g.setFont(new Font("Palatino Linotype", Font.BOLD, 24));

g.drawString(str,180,SIZE/2);

String str1 = "SCORE: " + score;

g.setColor(Color.white);

g.setFont(new Font("Palatino Linotype", Font.BOLD, 24));

g.drawString(str1,190,450);

String str2 = "RECORD: " + Record;

g.setColor(Color.white);

g.setFont(new Font("Palatino Linotype", Font.BOLD, 24));

g.drawString(str2,190,490);

}

}

public void move(){

for (int i = dots; i > 0; i--) {

x[i] = x[i-1];

y[i] = y[i-1];

}

if(left){

x[0] -= DOT\_SIZE;

}

if(right){

x[0] += DOT\_SIZE;

} if(up){

y[0] -= DOT\_SIZE;

} if(down){

y[0] += DOT\_SIZE;

}

}

public void checkApple(){

if(x[0] == appleX && y[0] == appleY){

dots++;

score += 10;

khit.sound();

khit.setVolume();

createApple();

}

}

public void checkCollisions(){

for (int i = dots; i >0 ; i--) {

if(i>4 && x[0] == x[i] && y[0] == y[i]){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

}

if(x[0]>SIZE){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

if(x[0]<0){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

if(y[0]>SIZE){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

if(y[0]<0){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

if(x[0] == tree1X && y[0] == tree1Y){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

if(x[0] == tree4X && y[0] == tree4Y){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

if(x[0] == tree2X && y[0] == tree2Y){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

if(x[0] == tree3X && y[0] == tree3Y){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

if(x[0] == castleX && y[0] == castleY){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

if(x[0] == castle1X && y[0] == castle1Y){

hit.sound();

hit.setVolume();

ost.stop();

inGame = false;

}

}

@Override

public void actionPerformed(ActionEvent e) {

if(inGame){

checkApple();

checkCollisions();

move();

}

repaint();

}

class FieldKeyListener extends KeyAdapter{

@Override

public void keyPressed(KeyEvent e) {

super.keyPressed(e);

int key = e.getKeyCode();

if(key == KeyEvent.VK\_LEFT && !right){

left = true;

up = false;

down = false;

right = false;

}

if(key == KeyEvent.VK\_RIGHT && !left){

right = true;

up = false;

down = false;

left = false;

}

if(key == KeyEvent.VK\_UP && !down){

right = false;

up = true;

left = false;

down = false;

}

if(key == KeyEvent.VK\_DOWN && !up){

right = false;

down = true;

left = false;

up = false;

}

}

}

}

**Приложение А.3 – Audio.java**

package SnakeGameCourse;

import java.io.File;

import java.io.IOException;

import javax.sound.sampled.\*;

public class Audio {

private String track;

private Clip clip = null;

private FloatControl volumeC = null;

private double wt;

private boolean pl\_audio;

public Audio(String track, double wt){

this.track = track;

this.wt = wt;

this.pl\_audio = false;

}

public void sound(){

File f = new File(this.track);

AudioInputStream tr = null;

try {

tr = AudioSystem.getAudioInputStream(f);

} catch (UnsupportedAudioFileException e){

e.printStackTrace();

} catch (IOException e){

e.printStackTrace();

}

try{

clip = AudioSystem.getClip();

clip.open(tr);

volumeC = (FloatControl) clip.getControl(FloatControl.Type.MASTER\_GAIN);

clip.setFramePosition(0);

clip.start();

} catch (LineUnavailableException e){

e.printStackTrace();

} catch (IOException e){

e.printStackTrace();

}

}

public void play(){

File f = new File(this.track);

AudioInputStream tr = null;

try {

tr = AudioSystem.getAudioInputStream(f);

} catch (UnsupportedAudioFileException e){

e.printStackTrace();

} catch (IOException e){

e.printStackTrace();

}

try{

clip = AudioSystem.getClip();

clip.open(tr);

volumeC = (FloatControl) clip.getControl(FloatControl.Type.MASTER\_GAIN);

if(!this.pl\_audio){

clip.setFramePosition(0);

clip.start();

this.pl\_audio = true;

}

} catch (LineUnavailableException e){

e.printStackTrace();

} catch (IOException e){

e.printStackTrace();

}

}

public void stop(){

clip.stop();

clip.close();

this.pl\_audio = false;

}

public void setVolume(){

if (wt < 0) wt = 0;

if (wt > 1) wt = 1;

float min = volumeC.getMinimum();

float max = volumeC.getMaximum();

volumeC.setValue((max-min)\*(float)wt+min);

}

}

**Приложение А.4 – Server.java**

package SnakeGameCourse;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.Closeable;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.net.ServerSocket;

import java.net.Socket;

public class Server implements Closeable {

private final Socket socket;

private final BufferedReader reader;

private final BufferedWriter writer;

public Server(String ip, int port){

try{

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

this.reader = createReader();

this.writer = createWriter();

}catch (IOException e){

throw new RuntimeException(e);

}

}

public Server(ServerSocket server){

try{

this.socket = server.accept();

this.reader = createReader();

this.writer = createWriter();

}catch (IOException e){

throw new RuntimeException(e);

}

}

public void write (String message){

try{

writer.write(message);

writer.newLine();

writer.flush();

}catch (IOException e){

throw new RuntimeException(e);

}

}

public String readline (){

try{

return reader.readLine();

} catch (IOException e){

throw new RuntimeException(e);

}

}

private BufferedReader createReader() {

try{

return new BufferedReader(new InputStreamReader(socket.getInputStream()));

}catch (IOException e){

throw new RuntimeException(e);

}

}

private BufferedWriter createWriter() {

try{

return new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));

}catch (IOException e){

throw new RuntimeException(e);

}

}

@Override

public void close() throws IOException {

writer.close();

reader.close();

socket.close();

}

String readLine() {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

void writeLine(String response) {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

}

**Приложение А.5 – Menu.java**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template

\*/

package SnakeGameCourse;

/\*\*

\*

\* @author Админ

\*/

public class Menu extends javax.swing.JFrame {

public static Audio menu;

public static Audio click;

/\*\*

\* Creates new form Menu

\*/

public Menu() {

setLocation(600,300);

initComponents();

menu = new Audio("res/menu.wav", 0.75);

click = new Audio("res/click.wav", 0.75);

menu.play();

menu.setVolume();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

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

setMaximumSize(new java.awt.Dimension(525, 550));

setMinimumSize(new java.awt.Dimension(525, 550));

setPreferredSize(new java.awt.Dimension(525, 550));

jPanel1.setBackground(new java.awt.Color(0, 0, 0));

jLabel1.setIcon(new javax.swing.ImageIcon("C:\\Users\\Админ\\Desktop\\проекты джава\\SnakeGameCourse\\back.png")); // NOI18N

jLabel1.setText("jLabel1");

jLabel2.setFont(new java.awt.Font("Sitka Banner", 1, 55)); // NOI18N

jLabel2.setForeground(new java.awt.Color(255, 0, 0));

jLabel2.setText("NOSPHERATU");

jButton1.setBackground(new java.awt.Color(0, 0, 0));

jButton1.setFont(new java.awt.Font("Sitka Banner", 1, 24)); // NOI18N

jButton1.setForeground(new java.awt.Color(255, 0, 0));

jButton1.setText("START");

jButton1.setBorder(null);

jButton1.setBorderPainted(false);

jButton1.setContentAreaFilled(false);

jButton1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));

jButton1.setFocusPainted(false);

jButton1.setFocusable(false);

jButton1.setMaximumSize(new java.awt.Dimension(50, 31));

jButton1.setMinimumSize(new java.awt.Dimension(50, 31));

jButton1.setPreferredSize(new java.awt.Dimension(50, 31));

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

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

jButton1ActionPerformed(evt);

}

});

jButton2.setBackground(new java.awt.Color(0, 0, 0));

jButton2.setFont(new java.awt.Font("Sitka Banner", 1, 24)); // NOI18N

jButton2.setForeground(new java.awt.Color(255, 0, 0));

jButton2.setText("EXIT");

jButton2.setBorder(null);

jButton2.setBorderPainted(false);

jButton2.setContentAreaFilled(false);

jButton2.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));

jButton2.setFocusPainted(false);

jButton2.setFocusable(false);

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

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

jButton2ActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel2, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, 459, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 95, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 95, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(79, 79, 79)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 59, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 284, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap(83, Short.MAX\_VALUE))

);

jLabel2.getAccessibleContext().setAccessibleName("NOSPHERATU");

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

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

new MainWindow();

menu.stop();

click.play();

click.setVolume();// TODO add your handling code here:

}

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

click.play();

click.setVolume();

System.exit(0); // TODO add your handling code here:

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Menu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Menu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Menu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Menu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

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

public void run() {

new Menu().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JPanel jPanel1;

// End of variables declaration

private void initGame() {

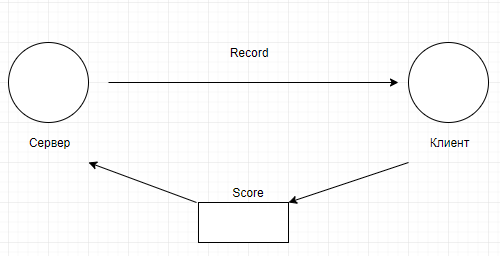
throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

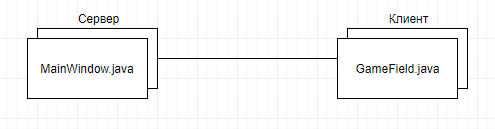
}

# Приложение В. UML-диаграммы приложения

# Приложение В.1 – UML-диаграмма вариантов использования



**Приложение В.2 – UML-диаграмма развёртывания**



**Приложение В.3 – UML-диаграмма последовательности**

