Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Программирование на языке Java»

на тему «Разработка многомодульного приложения на языке Java»

Выполнил студент

группы 21ВВП1:

Хабибулин А.М.

Приняла:

Юрова О.В.

**Оглавление**

[Введение 3](#_Toc105022468)

[1. Постановка задачи 4](#_Toc105022469)

[2. Выбор решения 5](#_Toc105022470)

[3. Описание программы 6](#_Toc105022471)

[4. Экспериментальная часть 8](#_Toc105022472)

[5. Заключение 11](#_Toc105022473)

[Список используемой литературы. 12](#_Toc105022474)

[Приложение А. Исходный код серверной части. 13](#_Toc105022475)

[Приложение А.1 - Server.java 13](#_Toc105022476)

[Приложение Б. Исходный код клиентской части 17](#_Toc105022477)

[Приложение Б.1 – NewFrame.java 17](#_Toc105022478)

[Приложение Б.2 – Kurs.java 26](#_Toc105022479)

[Приложение В. UML-диаграммы приложения 28](#_Toc105022480)

[Приложение В.1 – UML-диаграмма вариантов использования 28](#_Toc105022481)

[Приложение В.2 – UML-диаграмма классов сервера 29](#_Toc105022482)

[Приложение В.3 – UML-диаграмма классов клиента 30](#_Toc105022483)

[Приложение В.4 – UML-диаграмма деятельности 31](#_Toc105022484)

[Приложение В.5 – UML-диаграмма развёртывания 32](#_Toc105022485)

[Приложение В.6 – UML-диаграмма последовательности 33](#_Toc105022486)

# Введение

Жизнь современного человека невозможно представить без ежедневного взаимодействия с компьютером. Сегодня люди с легкостью по всему миру могут обмениваться не только сообщениями, но и графикой и звуковой и другими видами информации.

Java – один из самых популярных языков программирования, с большим количеством достоинств:

1) Язык Java хорош для изучения объектно-ориентированного программирования.

2) Java имеет богатый API, и вы можете делать очень много в Java, например, работать с графикой, звуком и, что всего вероятнее, писать игры.

3) Java имеет сильную поддержку сообщества.

Целбю данной курсовой работы является создание игрового приложения “Крестики - нолики”.

Результатом работы должно стать игровое приложение, способное обеспечивать удобное взаимодействие пользователей. Полученный опыт и результаты исследования могут быть полезными для понимания процесса создания подобных приложений и их дальнейшего усовершенствования.

# Постановка задачи

Разработать клиент-серверное приложение, реализующее игру «Крестики-нолики»

Функции сервера:

1) Обработка данных от пользователей.

2) Отправка результата в соответствии с принятыми данными

3) Своевременное окончание игры.

Функции клиента:

1. Реализация интерфейса, и его изменение в соответствии с ответами сервера.
2. Отправка данных серверу

Приложение должно обладать графическим интерфейсом и использовать следующие технологии:

1. Java Collections Framework
2. Механизм обработки исключительных ситуаций
3. Java Stream API
4. Java Multithreading
5. Сетевое взаимодействие.

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

# 2. Выбор решения

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

Сокет — это 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.

Обычно приложение клиент-сервер, использующее сокеты, состоит из двух разных приложений - клиента, инициирующего соединение с целью (сервером), и сервера, ожидающего соединения от клиента.

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

Интерфейс пользователя реализован с помощью библиотеки Java Swing.

# 3. Описание программы

Программа, состоит из клиентской и серверной части. Сначала запускается сервер, который входит в состояние ожидания подключения клиентов.

На других устройствах запускается клиентская часть, которая по нажатию определенной кнопки, пытается подключиться к серверу используя сокет.

**3.1 Серверная часть**

Server.java – главный файл серверного процесса, котором находится интерфейс и точка входа в программу, функция Main.

run() – метод, через который происходит взаимодействие с клиентами. Сервер получает данные от одного клиента и передаёт их другому. По окончанию игры он также получает данные о завершении игры и закрываются нити взаимодействия.

Вначале сервер ожидает подключения двух клиентов через сокет, когда это происходит он отправляет сообщения «player1» и «player2», только после этого начинается игра. Он принимает сообщения от «1» до «9», и отправляет их же другому клиенту, когда приходит сообщение «exit», то после этого сервер закрывается.

**3.2 Клиентская часть.**

NewFrame.java – главный файл пользовательского процесса, содержащий функцию Main. Также реализует интерфейс.

Kurs.java – содержит логическое поле и функции проверки на выигрыш и старт игры.

jButton[1-9]ActionPerformed(java.awt.event.ActionEvent evt) – кнопки игрового поля, при нажатии на них отображаются либо крестики либо нолики. Также отправляются и принимаются данные сервера.

run() – запускает нить ожидания запроса от второго клиента.

send(String msg) – отправляет данные на сервер.

get() – получает данные с сервера.

jButton10ActionPerformed(java.awt.event.ActionEvent evt) – кнопка начала игры.

startGame() – инициализация начальных значений логического поля.

checkWin(String myToken, String otherToken) – проверяет победил или проиграл игрок, а также определяет ничью.

closeSocket() – закрывает сокеты.

Клиент отправляют на сервер сообщения от «1» до «9» и второй получает те же значения, в зависимости от этого как изменяется интерфейс. Как только игра завершилась, появляется сообщение о выигрыше игрока, а на сервер отправляется сообщение о выходе. После этого игра закрывается.

# 4. Экспериментальная часть

Для начала запускаем сервер.

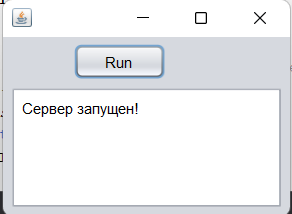


Рисунок – 1 Запуск сервера.

Далее запускаем клиента.

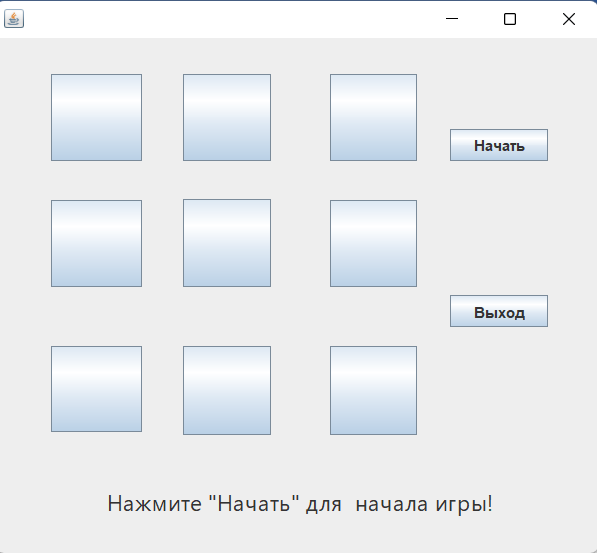


Рисунок – 2 Запуск Клиента.

После нажатия кнопки «Начать», происходит успешное подключение к серверу, значит второй клиент тоже смог подсоединиться. Далее нажимаем на свободную кнопку.

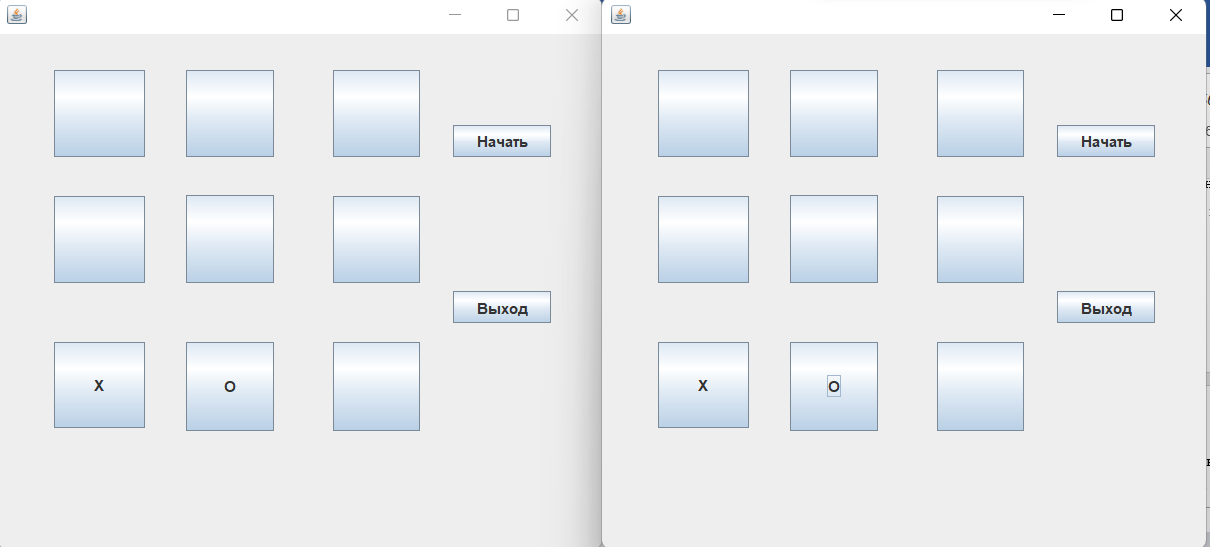


Рисунок – 3 Игра.

Играем до тех пор пока один из клиентов не соберёт три в ряд. После этого игра заканчивается.

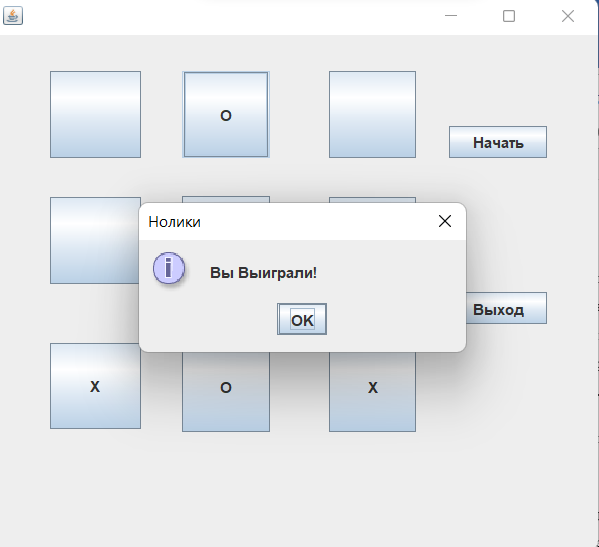


Рисунок – 4 Конец игры

# 5. 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 «Крестики-нолики».

Данная реализация не лишена недочетов в визуальной части и оптимизации кода, поэтому ее можно улучшить. Также есть возможность расширения функционала, например, добавление истории игр, рейтинга и т.д.

# 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. Исходный код серверной части.

# Приложение А.1 - Server.java

package com.server;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import static java.lang.Thread.sleep;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Server extends javax.swing.JFrame {

public Server() {

initComponents();

}

private static Socket clientSocket1; //сокет 1

private static Socket clientSocket2; //сокет 2

private static ServerSocket server; // серверсокет

private static BufferedReader in\_1; // поток чтения из сокета

private static BufferedWriter out\_1; // поток записи в сокет

private static BufferedReader in\_2; // поток чтения из сокета

private static BufferedWriter out\_2; // поток записи в сокет

private boolean stepOnePlayer = true;

private boolean stepTwoPlayer = false;

Thread thread = new Thread() {

@Override

public void run() {

try {

server = new ServerSocket(4004); // серверсокет прослушивает порт 4004

try {

sleep(1);

} catch (InterruptedException ex) {

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

}

jTextArea1.append("Сервер запущен!\n");

clientSocket1 = server.accept();//Ожидает подключения клиента 1 к серверу. Когда клиент подключается, создается новый сокет

jTextArea1.append("Клиент 1 подключен!\n");

clientSocket2 = server.accept();

jTextArea1.append("Клиент 2 подключен!\n");

in\_1 = new BufferedReader(new InputStreamReader(clientSocket1.getInputStream()));//Создает BufferedReader для чтения данных, отправленных клиентом 1 через сокет

out\_1 = new BufferedWriter(new OutputStreamWriter(clientSocket1.getOutputStream()));//Создает BufferedWriter для записи данных в сокет clientSocket1, чтобы отправить их клиенту 1

in\_2 = new BufferedReader(new InputStreamReader(clientSocket2.getInputStream()));

out\_2 = new BufferedWriter(new OutputStreamWriter(clientSocket2.getOutputStream()));

send\_1("player1");

send\_2("player2");

while (true){

//проверяет, должен ли сейчас ходить первый игрок

if (stepOnePlayer){

String word = in\_1.readLine(); // ждём пока клиент что-нибудь нам напишет

if ("exit".equals(word)){

jTextArea1.append("Клиенты отключились!\n");

clientSocket1.close();

in\_1.close();

out\_1.close();

clientSocket2.close();

in\_2.close();

out\_2.close();

break;

}

send\_2(word);

stepOnePlayer = false;

stepTwoPlayer = true;

}

//проверяет, должен ли сейчас ходить второй игрок

if (stepTwoPlayer){

String word = in\_2.readLine(); // ждём пока клиент что-нибудь нам напишет

if ("exit".equals(word)){

jTextArea1.append("Клиенты отключились!\n");

clientSocket1.close();

in\_1.close();

out\_1.close();

clientSocket2.close();

in\_2.close();

out\_2.close();

break;

}

send\_1(word);

stepOnePlayer = true;

stepTwoPlayer = false;

}

}

jTextArea1.append("Сервер отключился!\n");

server.close();

}

catch (IOException e)

{

System.err.println(e);

}

}

};

@SuppressWarnings("unchecked")

private void send\_1(String msg) {

try {

out\_1.write(msg + "\n");

out\_1.flush();

} catch (IOException ignored) {}

}

private void send\_2(String msg) {

try {

out\_2.write(msg + "\n");

out\_2.flush();

} catch (IOException ignored) {}

}

@SuppressWarnings("unchecked")

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

private void initComponents() {

jButton1 = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

jTextArea1 = new javax.swing.JTextArea();

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

jButton1.setText("Run");

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

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

jButton1ActionPerformed(evt);

}

});

jTextArea1.setColumns(20);

jTextArea1.setRows(5);

jScrollPane1.setViewportView(jTextArea1);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addGroup(layout.createSequentialGroup()

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

.addGroup(layout.createSequentialGroup()

.addGap(30, 30, 30)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addGap(103, 103, 103)

.addComponent(jButton1)))

.addContainerGap(30, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

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

.addGroup(layout.createSequentialGroup()

.addGap(30, 30, 30)

.addComponent(jButton1)

.addGap(18, 18, 18)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(30, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

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

thread.start();

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

//<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(Server.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

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

public void run() {

new Server().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTextArea jTextArea1;

// End of variables declaration

}

# Приложение Б. Исходный код клиентской части

## Приложение Б.1 – NewFrame.java

package client;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.net.Socket;

import java.util.logging.Level;

import java.util.logging.Logger;

public class NewJFrame extends javax.swing.JFrame {

public NewJFrame() {

initComponents();

}

Kurs logic = new Kurs(); //экземпляр класса Kurs, который,отвечает за логику игры.

String text = ""; //переменная, которая используется для хранения сообщений, получаемых через сетевое соединение.

String myToken; // токен текущего игрока

String otherToken; // токен соперника

boolean waiting = true;

private static Socket clientSocket; //переменная, которая представляет сокет для сетевого соединения с сервером.

private static BufferedReader in; // переменная, которая представляет поток чтения из сокета, используемый для чтения данных от сервера.

private static BufferedWriter out; // переменная, которая представляет поток записи в сокет, используемый для отправки данных на сервер.

Runnable thread = new Runnable() //экземпляр анонимного класса, реализующего интерфейс Runnable

{

//класс переопределяет метод run(), который будет выполняться в отдельном потоке.

@Override

public void run() {

try {

Thread.sleep(1000);

} catch (InterruptedException ex) {

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

}

get();

// метод,проверяет, нет ли победителя на игровом поле на основе текущего состояния игры.

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

};

private void closeSocket() {

try {

send("exit"); //метод отправляет сообщение "exit" на сервер через сокет

clientSocket.close();//закрывает сокет, прекращая все взаимодействия с сервером.

in.close();//метод закрывает поток чтения из сокета, который использовался для чтения данных от сервера

out.close();//метод закрывает поток записи в сокет, который использовался для отправки данных на сервер.

} catch (IOException ignored) {}

}

//метод отправки сообщения на сервер

private void send(String msg) {

try {

out.write(msg + "\n"); //Записываем сообщение в буфер вывода

out.flush(); // Принудительно отправляем данные на сервер

waiting = true; //Устанавливаем флаг ожидания.Oтправитель теперь должен ждать ответа от сервера

} catch (IOException ignored) {}

}

private void get() {

try {

text = in.readLine(); //считывает строку текста из потока ввода in

waiting = false;

switch(text){

case "1": jButton1.setText(otherToken); logic.ArrayField[0][0] = otherToken; break;

case "2": jButton2.setText(otherToken); logic.ArrayField[0][1] = otherToken; break;

case "3": jButton3.setText(otherToken); logic.ArrayField[0][2] = otherToken; break;

case "4": jButton4.setText(otherToken); logic.ArrayField[1][0] = otherToken; break;

case "5": jButton5.setText(otherToken); logic.ArrayField[1][1] = otherToken; break;

case "6": jButton6.setText(otherToken); logic.ArrayField[1][2] = otherToken; break;

case "7": jButton7.setText(otherToken); logic.ArrayField[2][0] = otherToken; break;

case "8": jButton8.setText(otherToken); logic.ArrayField[2][1] = otherToken; break;

case "9": jButton9.setText(otherToken); logic.ArrayField[2][2] = otherToken; break;

default: break;

}

} catch (IOException ignored) {}

}

@SuppressWarnings("unchecked")

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

private void initComponents() {

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jButton3 = new javax.swing.JButton();

jButton4 = new javax.swing.JButton();

jButton5 = new javax.swing.JButton();

jButton6 = new javax.swing.JButton();

jButton7 = new javax.swing.JButton();

jButton8 = new javax.swing.JButton();

jButton9 = new javax.swing.JButton();

jButton10 = new javax.swing.JButton();

jLabel1 = new javax.swing.JLabel();

jButton11 = new javax.swing.JButton();

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

jButton1.setMaximumSize(null);

jButton1.setMinimumSize(null);

jButton1.setPreferredSize(new java.awt.Dimension(75, 75));

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

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

jButton1ActionPerformed(evt);

}

});

jButton2.setPreferredSize(new java.awt.Dimension(75, 75));

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

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

jButton2ActionPerformed(evt);

}

});

jButton3.setPreferredSize(new java.awt.Dimension(75, 75));

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

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

jButton3ActionPerformed(evt);

}

});

jButton4.setMaximumSize(null);

jButton4.setMinimumSize(null);

jButton4.setPreferredSize(new java.awt.Dimension(75, 75));

jButton4.addActionListener(new java.awt.event.ActionListener() {

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

jButton4ActionPerformed(evt);

}

});

jButton5.setPreferredSize(new java.awt.Dimension(75, 75));

jButton5.addActionListener(new java.awt.event.ActionListener() {

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

jButton5ActionPerformed(evt);

}

});

jButton6.setPreferredSize(new java.awt.Dimension(75, 75));

jButton6.addActionListener(new java.awt.event.ActionListener() {

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

jButton6ActionPerformed(evt);

}

});

jButton7.setPreferredSize(new java.awt.Dimension(75, 75));

jButton7.addActionListener(new java.awt.event.ActionListener() {

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

jButton7ActionPerformed(evt);

}

});

jButton8.setPreferredSize(new java.awt.Dimension(75, 75));

jButton8.addActionListener(new java.awt.event.ActionListener() {

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

jButton8ActionPerformed(evt);

}

});

jButton9.setPreferredSize(new java.awt.Dimension(75, 75));

jButton9.addActionListener(new java.awt.event.ActionListener() {

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

jButton9ActionPerformed(evt);

}

});

jButton10.setText("Начать");

jButton10.addActionListener(new java.awt.event.ActionListener() {

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

jButton10ActionPerformed(evt);

}

});

jLabel1.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jLabel1.setText("Нажмите \"Начать\" для начала игры!");

jButton11.setText("Выход");

jButton11.addActionListener(new java.awt.event.ActionListener() {

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

jButton11ActionPerformed(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(21, 21, 21)

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

.addComponent(jButton1, javax.swing.GroupLayout.DEFAULT\_SIZE, 76, Short.MAX\_VALUE)

.addComponent(jButton4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton7, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(45, 45, 45)

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

.addComponent(jButton8, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jButton5, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(45, 45, 45)

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

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

.addComponent(jButton9, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(layout.createSequentialGroup()

.addComponent(jButton6, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 45, Short.MAX\_VALUE)

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

.addComponent(jButton10, javax.swing.GroupLayout.PREFERRED\_SIZE, 92, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton11, javax.swing.GroupLayout.PREFERRED\_SIZE, 92, javax.swing.GroupLayout.PREFERRED\_SIZE))))

.addGap(46, 46, 46))

.addGroup(layout.createSequentialGroup()

.addGap(105, 105, 105)

.addComponent(jLabel1)

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

);

layout.setVerticalGroup(

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

.addGroup(layout.createSequentialGroup()

.addGap(41, 41, 41)

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

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

.addGroup(layout.createSequentialGroup()

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

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

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

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

.addGroup(layout.createSequentialGroup()

.addGap(45, 45, 45)

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

.addComponent(jButton4, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton5, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(45, 45, 45)

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

.addComponent(jButton7, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton8, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGroup(layout.createSequentialGroup()

.addGap(8, 8, 8)

.addComponent(jButton10)

.addGap(93, 93, 93)

.addComponent(jButton11))))

.addGroup(layout.createSequentialGroup()

.addGap(120, 120, 120)

.addComponent(jButton6, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(45, 45, 45)

.addComponent(jButton9, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 30, Short.MAX\_VALUE)

.addComponent(jLabel1)

.addGap(30, 30, 30))

);

pack();

}// </editor-fold>

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

if (waiting){}

else{

jButton1.setText(myToken);

logic.ArrayField[0][0] = myToken;

send("1");

Thread button\_1 = new Thread(thread);

button\_1.start();

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

}

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

if (waiting){}

else{

jButton2.setText(myToken);

logic.ArrayField[0][1] = myToken;

send("2");

Thread button\_2 = new Thread(thread);

button\_2.start();

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

}

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

if (waiting){}

else{

jButton3.setText(myToken);

logic.ArrayField[0][2] = myToken;

send("3");

Thread button\_3 = new Thread(thread);

button\_3.start();

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

if (waiting){}

else{

jButton4.setText(myToken);

logic.ArrayField[1][0] = myToken;

send("4");

Thread button\_4 = new Thread(thread);

button\_4.start();

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

if (waiting){}

else{

jButton5.setText(myToken);

logic.ArrayField[1][1] = myToken;

send("5");

Thread button\_5 = new Thread(thread);

button\_5.start();

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

if (waiting){}

else{

jButton6.setText(myToken);

logic.ArrayField[1][2] = myToken;

send("6");

Thread button\_6 = new Thread(thread);

button\_6.start();

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

if (waiting){}

else{

jButton7.setText(myToken);

logic.ArrayField[2][0] = myToken;

send("7");

Thread button\_7 = new Thread(thread);

button\_7.start();

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

}

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {

if (waiting){}

else{

jButton8.setText(myToken);

logic.ArrayField[2][1] = myToken;

send("8");

Thread button\_8 = new Thread(thread);

button\_8.start();

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

}

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

if (waiting){}

else{

jButton9.setText(myToken);

logic.ArrayField[2][2] = myToken;

send("9");

Thread button\_9 = new Thread(thread);

button\_9.start();

if(logic.CheckWin(myToken, otherToken)){

waiting = true;

closeSocket();

System.exit(0);

}

}

}

private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {

logic.StartGame();

try {

clientSocket = new Socket("localhost", 4004);

in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

out = new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream()));

get();

if("player1".equals(text)) {myToken = "X"; otherToken = "O"; waiting = false; jLabel1.setText(""); }

if("player2".equals(text)) {myToken = "O"; otherToken = "X"; waiting = true; jLabel1.setText(""); Thread button\_10 = new Thread(thread);

button\_10.start();}

} catch (IOException e) {

System.err.println(e);

}

}

private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {

closeSocket();

System.exit(0);

}

public static void main(String args[]) {

java.awt.EventQueue.invokeLater(() -> {

new NewJFrame().setVisible(true);

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton10;

private javax.swing.JButton jButton11;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JButton jButton4;

private javax.swing.JButton jButton5;

private javax.swing.JButton jButton6;

private javax.swing.JButton jButton7;

private javax.swing.JButton jButton8;

private javax.swing.JButton jButton9;

private javax.swing.JLabel jLabel1;

// End of variables declaration

}

## Приложение Б.2 – Kurs.java

package client;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

public class Kurs extends JFrame {

String player;//переменная, которая используется для хранения идентификатора текущего игрока. В зависимости от токена, который используется

public String [][] ArrayField = new String [3][3];//двумерный массив строк, который представляет собой игровое поле для игры

public boolean CheckWin(String myToken, String otherToken){

if("X".equals(myToken)) player = "Крестики";

if("O".equals(myToken)) player = "Нолики";

if ((ArrayField[0][0].equals(myToken) && ArrayField[0][1].equals(myToken) && ArrayField[0][2].equals(myToken)) ||

(ArrayField[1][0].equals(myToken) && ArrayField[1][1].equals(myToken) && ArrayField[1][2].equals(myToken)) ||

(ArrayField[2][0].equals(myToken) && ArrayField[2][1].equals(myToken) && ArrayField[2][2].equals(myToken)) ||

(ArrayField[0][0].equals(myToken) && ArrayField[1][0].equals(myToken) && ArrayField[2][0].equals(myToken)) ||

(ArrayField[0][1].equals(myToken) && ArrayField[1][1].equals(myToken) && ArrayField[2][1].equals(myToken)) ||

(ArrayField[0][2].equals(myToken) && ArrayField[1][2].equals(myToken) && ArrayField[2][2].equals(myToken)) ||

(ArrayField[0][0].equals(myToken) && ArrayField[1][1].equals(myToken) && ArrayField[2][2].equals(myToken)) ||

(ArrayField[0][2].equals(myToken) && ArrayField[1][1].equals(myToken) && ArrayField[2][0].equals(myToken))){

JOptionPane.showMessageDialog(null, "Вы Выиграли!", player, JOptionPane.INFORMATION\_MESSAGE);

return true;

}

if ((ArrayField[0][0].equals(otherToken) && ArrayField[0][1].equals(otherToken) && ArrayField[0][2].equals(otherToken)) ||

(ArrayField[1][0].equals(otherToken) && ArrayField[1][1].equals(otherToken) && ArrayField[1][2].equals(otherToken)) ||

(ArrayField[2][0].equals(otherToken) && ArrayField[2][1].equals(otherToken) && ArrayField[2][2].equals(otherToken)) ||

(ArrayField[0][0].equals(otherToken) && ArrayField[1][0].equals(otherToken) && ArrayField[2][0].equals(otherToken)) ||

(ArrayField[0][1].equals(otherToken) && ArrayField[1][1].equals(otherToken) && ArrayField[2][1].equals(otherToken)) ||

(ArrayField[0][2].equals(otherToken) && ArrayField[1][2].equals(otherToken) && ArrayField[2][2].equals(otherToken)) ||

(ArrayField[0][0].equals(otherToken) && ArrayField[1][1].equals(otherToken) && ArrayField[2][2].equals(otherToken)) ||

(ArrayField[0][2].equals(otherToken) && ArrayField[1][1].equals(otherToken) && ArrayField[2][0].equals(otherToken))){

JOptionPane.showMessageDialog(null, "Вы Проиграли!", player, JOptionPane.INFORMATION\_MESSAGE);

return true;

}

if(isFull()){

JOptionPane.showMessageDialog(null, "Ничья!", player, JOptionPane.INFORMATION\_MESSAGE);

return true;

}

else return false;

}

//метод инициализирует игровое поле пустыми строками (""), очищая его для новой игры.

public void StartGame(){

for (int i = 0; i < 3; i++){

for (int j = 0; j < 3; j++){

ArrayField[i][j] = "";

}

}

}

//метод проверяет, заполнено ли игровое поле.

private boolean isFull(){

int count = 0;

for(int i = 0; i < 3; i++){

for (int j = 0; j < 3; j++){

if(("X".equals(ArrayField[i][j])) || ("O".equals(ArrayField[i][j]))) count++;

}

}

if (count == 9) return true;//Возвращает true, если все ячейки заполнены, иначе false

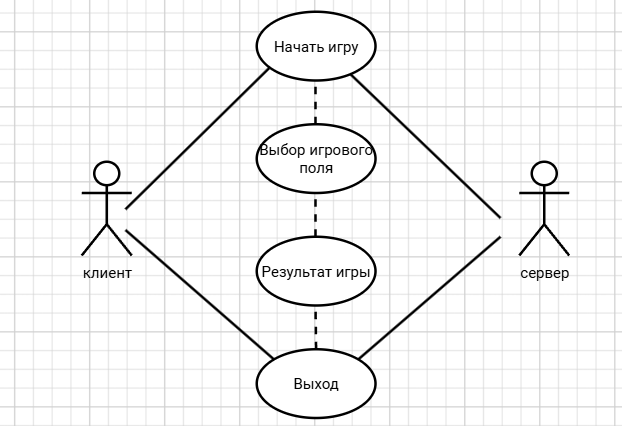
else return false;

}

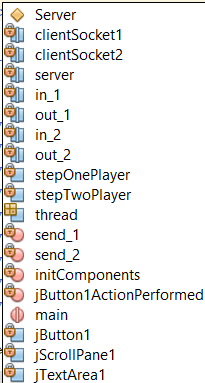
}

# Приложение В. UML-диаграммы приложения

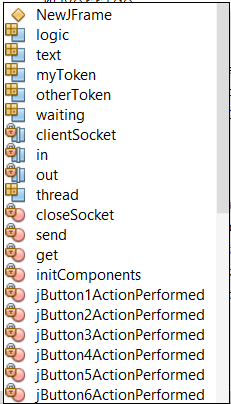
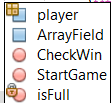
# Приложение В.1 – UML-диаграмма вариантов использования



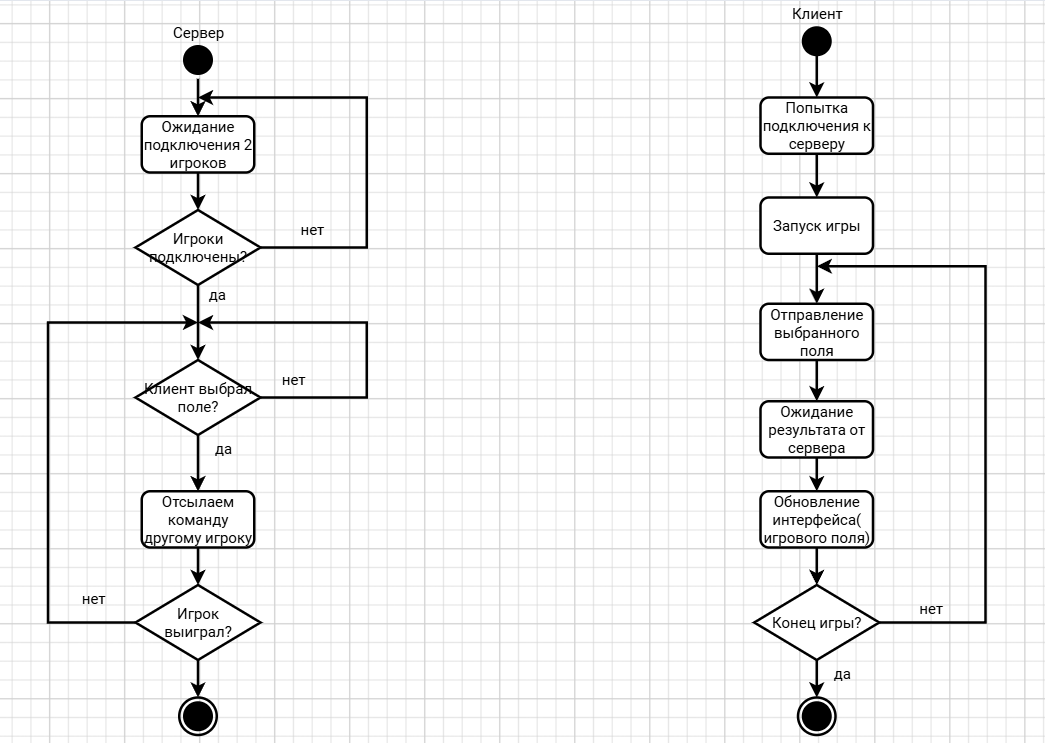
## Приложение В.2 – UML-диаграмма классов сервера



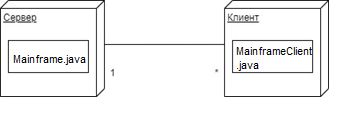
## Приложение В.3 – UML-диаграмма классов клиента



# Приложение В.4 – UML-диаграмма деятельности

A

# Приложение В.5 – UML-диаграмма развёртывания



NewFrame.java

Kurs.java

Server.java

# Приложение В.6 – UML-диаграмма последовательности

