Содержание

[Введение 3](#_Toc533207633)

[1. Анализ объекта 4](#_Toc533207634)

[1.1 Описание предметной области. 4](#_Toc533207635)

[1.2 Построение концептуальной модели предметной области. 5](#_Toc533207636)

[2 Постановка задачи 6](#_Toc533207637)

[2.1 Определение требований к программной системе. 6](#_Toc533207638)

[2.2 Описание аналогов системы. 7](#_Toc533207639)

[2.3 Обзор и обоснование выбора средств реализации 8](#_Toc533207640)

[3 Проектирование 9](#_Toc533207641)

[3.1 Разработка архитектуры программного продукта 9](#_Toc533207642)

[3.2 Проектирование структур хранения данных. 10](#_Toc533207643)

[4 Реализация 11](#_Toc533207644)

[4.1 Разработка архитектуры программного продукта 11](#_Toc533207645)

[4.2 Разработка интерфейса программного продукта 13](#_Toc533207646)

[4.3 Разработка алгоритмов реализации вариантов использования. 18](#_Toc533207647)

[5 Тестирование 25](#_Toc533207648)

[Литература 29](#_Toc533207649)

[Приложение А 30](#_Toc533207650)

# Введение

Тема данной курсовой работы является актуальной для любителей старых добрых игр, которые не требуют обучения или много времени. Говоря конкретно, проект интересен всем любителям игры крестики-нолики. Данная программа значительно облегчит поиск соперника (которого приходилось искать самому в огромном мире), предотвращает любой мухлеж, дает возможность играть с любым человеком, подключенным к серверу. Программа также предоставляет возможность играть с компьютером, который играет согласно созданному алгоритму.

Данная курсовая работа выполнена на языке программирования высокого уровня Java с использованием IDE Eclipse.

# Анализ объекта

## Описание предметной области.

Крестики-нолики - логическая игра между двумя противниками на квадратном поле 3 на 3 клетки или большего размера (вплоть до «бесконечного поля»). Один из игроков играет «крестиками», второй -- «ноликами». Эта игра стала популярна задолго до появления компьютеров, только раньше в нее играли с помощью обычного листка бумаги и ручки. В традиционной китайской игре используются черные и белые камни.

Выигрыш фиксируется, если крестиками или ноликами полностью заполняется вертикаль, горизонталь или диагональ. Если свободные клетки поля закончились, но никто не одержал победы, то считается, что игра закончилась «ничьей».

## Построение концептуальной модели предметной области.

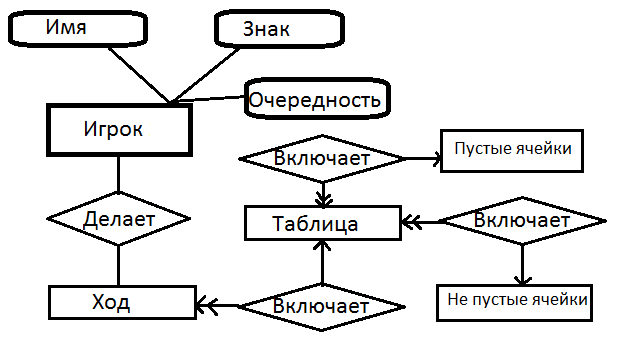


Рисунок 1.2 – модель предметной области

Концептуальная модель состоит из Игрока, имеющего имя, знак, которым ходит, и очередность хода. Когда игрок делает ход, знак заносится в пустую ячейку таблицы. Заполненные ячейки проверяются на выигрышную комбинацию, и при отсутствии таковой действия повторяются для другого пользователя. Если найдена выигрышная ситуация – игра завершается победой игрока с выигрышным знаком. Если выигрышной ситуации нет, как и пустых ячеек – игра заканчивается ничьей.

# Постановка задачи

## Определение требований к программной системе.

1. Приложение, позволяющее пользователю играть в крестики нолики
2. Поддержка игры как с ПК, так и с другим пользователем
3. Поддержка имен пользователей
4. Дружелюбный интерфейс
5. Играть более 2-х человек одновременно
6. Поддержка Java версии 9 и более

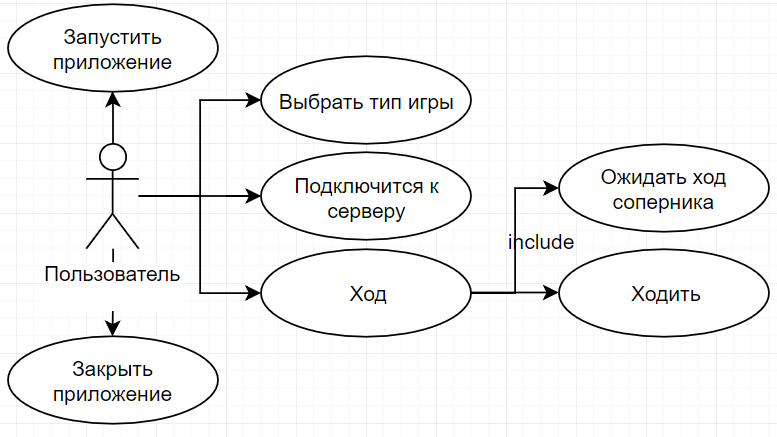


Рисунок 2.1 - диаграмма вариантов использования

## Описание аналогов системы.

В сети Интернет можно найти большое количество работ, которые реализуют данную игру. В настоящее время существует много аналогов этой игры, которые отошли от первоначальных стандартов. Примером таким программ являются «Крестики-нолики на бесконечном поле» и «Крестики-нолики 3D». Так же во многих играх «крестики» и «нолики» заменяются на другие символы, такие как, например, «камни».

Мой курсовой проект представляет собой приложение для ПК. Игра предназначена как для одного пользователя, соперником которого является компьютер, так и для двух пользователей. Она представлена на классическом поле 3х3.

### 2.3 Обзор и обоснование выбора средств реализации

Eclipse —интегрированная среда разработки ПО. Написана на языке Java в 2003 году. Также является кроссплатформенной. За счёт присоединяемых к этой среде дополнений — имеется возможность создавать программные продукты более чем на пяти языках программного кода.

Достоинства: Постоянное обновление версий среды разработки, поддержка многих языков (в том числе и русского), является бесплатной, поддержка многих языков программирования, среда имеет промышленный уровень, является гибкой — то есть легко настраивается как под любую платформу, так и под любого пользователя.

Для создания интерфейса было применено:

JFrame – класс, реализующий отображение окон приложения.

JLabel – элементы отображения надписей

JButton – элементы отображения кнопок

JTextField – элементы отображение полей ввода

Thread – класс, реализующий возможность многопоточной работы

JTable – элемент, отображающий ход

JTextPane – элемент, отображающий историю ходов пользователей

# Проектирование

## Разработка архитектуры программного продукта

Поскольку игра многопользовательская, необходимо иметь сервер, который синхронизирует игровой процесс между пользователями. Этим будет заниматься класс Server. Пользователей может быть множество, чтобы их не потерять, они все будут храниться в списке list и иметь свой id. Так как пользователи будут играть 1 на 1, у них должен храниться id врага и возможность общаться с врагом (методы принятия и отправки сообщений (in, out)). Сервер должен знать, в какой стадии игрок (играет, ищет игру), поэтому должна быть переменная состояния isSearh, переменная хранения имени name.

С серверной части клиента, должны быть методы получения имени, хода пользователя и принятия хода соперника – getName(), msgIn, msgOut.

Главное окно будет содержать поля ввода имени и хоста. Будут кнопки подключение к серверу (должен будет подключится), игра с пк и игра с пользователем (при успешном подключении).

После выбора типа игры, должно будет появится окно с самой игрой. На ней будет таблица с отображение ходов, небольшая история и кнопка ходить. По кнопке на сервер будет отправляться сообщение с координатами ячейки, в которую сделали ход. Очередность хода будет смотреть по переменной iGo.

## Проектирование структур хранения данных.

Для хранения данных бд не используется. Используется массив 3 на 3 типа integer для хранения данных при игре с пк и таблица 3 на 3 для хранения данных игры с другим пользователем

# Реализация

## Разработка архитектуры программного продукта

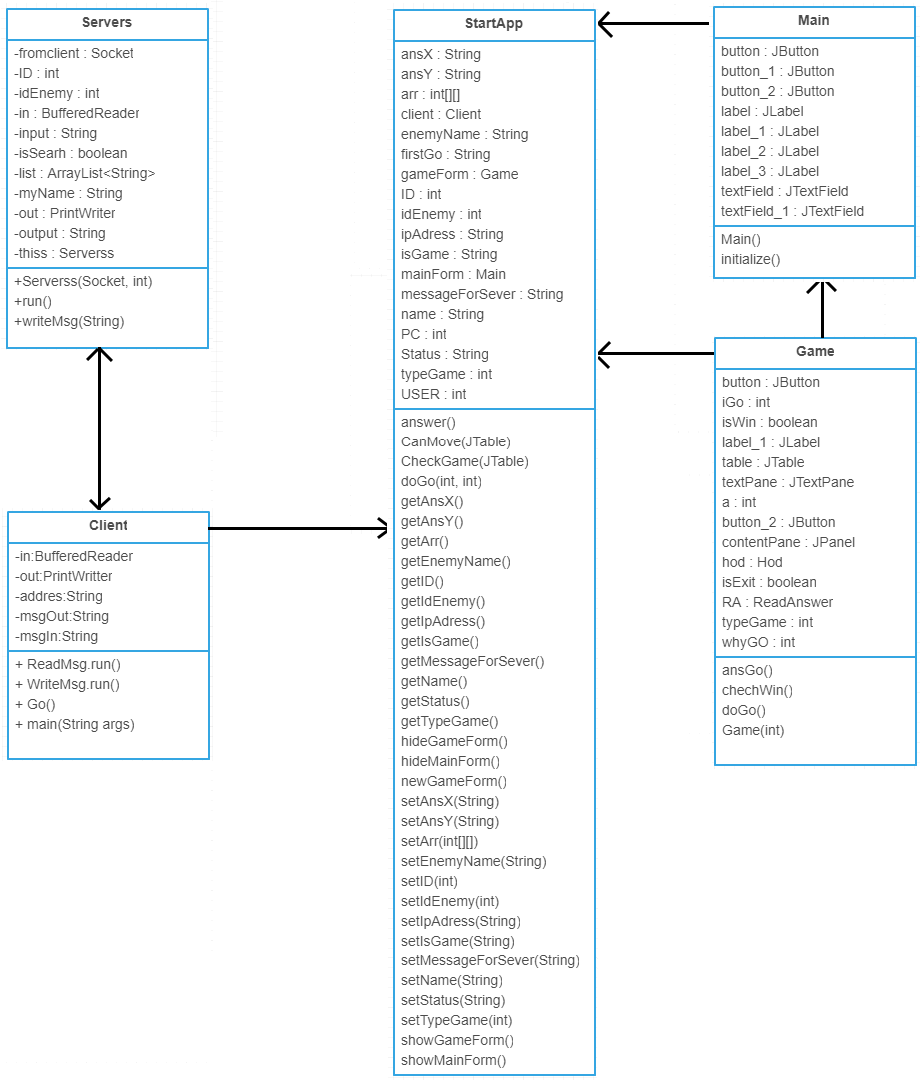


Рисунок 4.1 – Диаграмма классов

Главным классом является класс StartApp, у которого есть методы получения имени своего и врага – getName(), getEnemyName(), методы хранения статуса программы – setStatus(), проверки можно ли сделать ход – CanMove(), проверка на выигрыш – ChechGame() и др.

В нем также хранятся переменные типа Main и Game, отвечающие за вывод на экран главной формы и передачи информации между ними.

В форме Main по кнопке подключится к серверу в класс StartApp передается информации о запросе подключения, которую оный пересывает в экземпляр класса Client. Последний в методе Go(), подключается к сервету по введенному пользователем адресу, передает свое имя и запрос на поиск игры методом WriteMsg(). Ответ читает методом ReadMsg() и передает информацию пользователю.

## Разработка интерфейса программного продукта

Главное окно

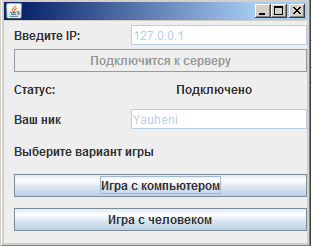
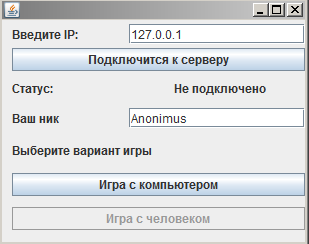


Рисунок 4.2.1 – Вид главного окна

При вводе имени и адреса сервера, пользователь получает доступ к игре с другими пользователями. Также обновляется статус приложения. Если не введено имя, будет показано соответствующее предупреждение.

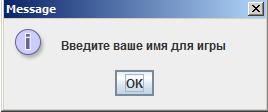


Рисунок 4.2.2 – Ошибка при отсутствии имени

Нажатие на кнопку игры с ПК, пользователь начинает игру непосредственно с ПК.

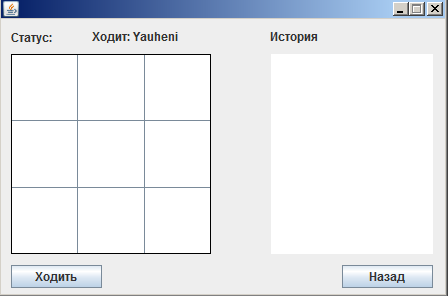


Рисунок 4.2.3 – Окно игры против ПК

В игре с ПК или другим пользователем в верху приложения показывается чей сейчас ход. Также справа предусмотрена история ходов, чтоб пользователи могли видеть кто куда уже походил.

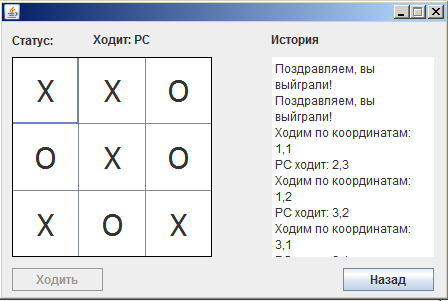


Рисунок 4.2.4 – Один из вариантов окончания игры.

Ходить можно как кликом мышки, так и нажатием на кнопку. После окончания игры в истории ходов можно видеть победителя. После окончания игры остается выйти в главное меню кнопкой «Назад».

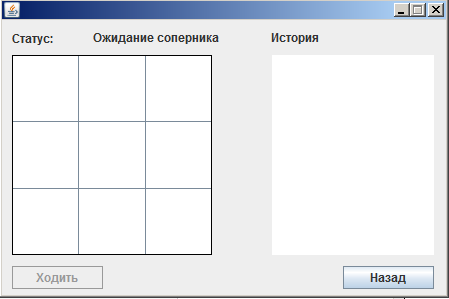


Рисунок 4.2.5 – Окно в режиме ожидания соперника

При нажатии кнопки игра с пользователем, вы переходите на окно игры с пользователем (инфетфейс один и то же) и одидаете соперника. Ходить при ожидании нельзя (это не чесно).

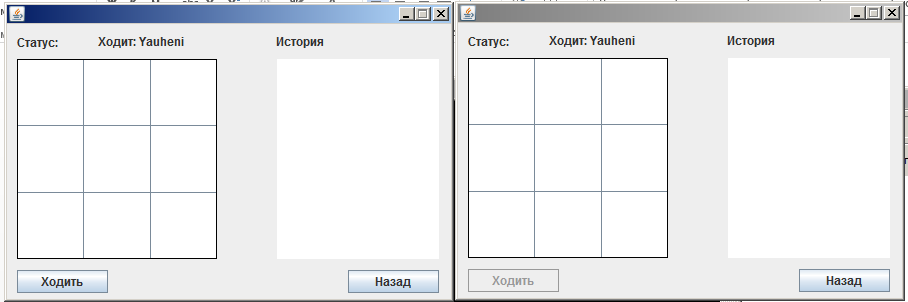


Рисунок 4.2.6 – Окна игры двух пользователей

Как только соперник подключился, сервер определяет кто будет ходить первым (рандом), и разрешае ему ходить. Опять же вверху экрана можем видеть чей сейчас ход. Также это видно по активной кнопке

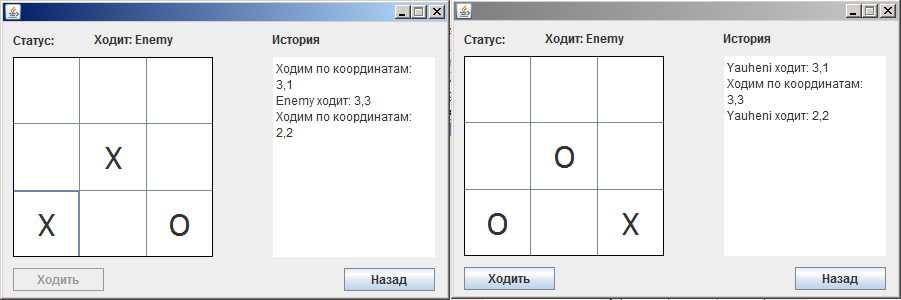


Рисунок 4.2.7 – Окна игры двух пользователей при игре против друг-друга

Когда пользователь пытается ходить в занятую ячейку, ему выдается соответствующее уведомление

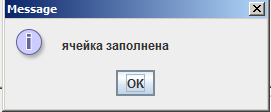


Рисунок 4.2.8 – Окно ошибки, при попытки заполнить заполненную ячейку

По окончанию игры, каждому игроку будет показано свое уведомление, в зависимости от конца игры

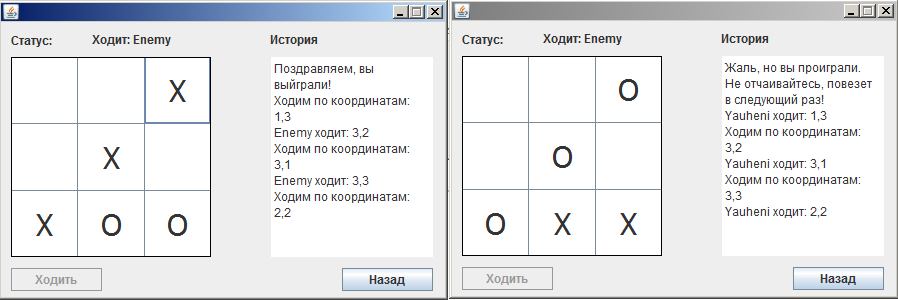


Рисунок 4.2.9 – Окна пользователей по окончанию игры.

Одновременно играть может любое количество игроков как один на один, так и один против ПК.

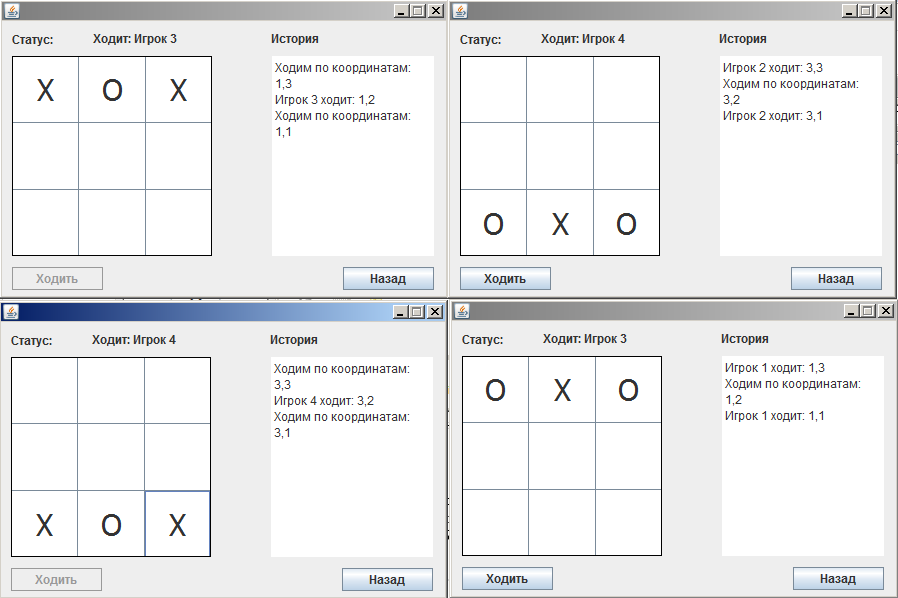


Рисунок 4.2.10 – Окна игры сразу четверых игроков

## Разработка алгоритмов реализации вариантов использования.

Общий алгоритм работы приложения:

1. Открывается главное меню, переход к 2
2. Вводит свое имя, переход к 3
3. Вводит адрес сервера, переход к 4
4. Если нажата кнопка игра с пк, переход к пункту 7, иначе к 5
5. Если нажата кнопка игра с человеком, переход к пункту 11, иначе к
6. Если нажата кнопка выход, переход к 15
7. Открывается окно игры с пк, переход к 8
8. Ход пользователя, проверка на конец игры. Если игра окончена – переход к 10, иначе к 9
9. Ход пк, проверка на конец игры. Если игра окончена – переход к 10, иначе к 8
10. Вывод результатов игры, выход в главное меню, переход к 1
11. Ожидание подключения соперника, переход к 12
12. Ход пользователя, проверка на конец игры. Если игра окончена – переход к 14, иначе к 13
13. Ход соперника, проверка на конец игры. Если игра окончена – переход к 14, иначе к 13
14. Вывод результатов игры, выход в главное меню, переход к 1
15. Завершение работы приложения
16. Для игры с пк

Сперва проверяем, введено ли имя, если это так, открываем форму с игрой и создаем массив 3 на 3 для хранения данных ходов. Главную форму скрываем

button\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

if (!textField.getText().isEmpty()) {

StartApp.setName(textField.getText());

StartApp.setTypeGame(StartApp.PC);

StartApp.setArr(new int[3][3]);

StartApp.newGameForm();

StartApp.showGameForm();

StartApp.hideMainForm();} else

JOptionPane.showMessageDialog(null, "Введите ваше имя для игры");}});

1. Для игры с соперником используется тот же алгоритм, только не создается массив
2. При подключении к серверу

Устанавливаем имя пользователя в переменные, закрываем возможность редактирования адреса сервера и имени, подключаемся к самому серверу и открываем возможность играть с другими пользователями. При возникновении ошибки, откатываем изменения и выводим в статус ошибку.

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

// работать лишь при условии что имя введено

if (!textField.getText().isEmpty()) {

// Сохранении имени и запуск подключения

StartApp.setName(textField.getText());

textField.setEnabled(false);

textField\_1.setEnabled(false);

button.setEnabled(false);

button\_2.setEnabled(true);

try {

StartApp.client.main(null);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Ошибка подключения к серверу");

StartApp.setStatus("Ошибка");

e.printStackTrace(); }} else

JOptionPane.showMessageDialog(null, "Введите ваше имя для подключения");}});

1. Ход пк

Ход ПК реализован рандомным значением. Компьютер получает массив, в котором отмечено занята ли ячейка. До тех пор, пока он не найдет свободную ячейку, н будет перебирать значения. Как только ячейка найдена, пк ставит ход и передает ход пользователю

**public** **static** **int**[] answer() {

**int** x, y; **do** {

x = (**int**) (Math.*random*() \* 3);

y = (**int**) (Math.*random*() \* 3);

} **while** (*arr*[x][y] != 0); *arr*[x][y] = 5;

**int**[] ans = **new** **int**[2]; ans[0] = x; ans[1] = y;

**return** ans; }

1. Проверка выигрыша реализована следующим алгоритмом:

Просматриваются сперва строки и столбцы на одинаковое значение. Если не найдено таких, просматривается главная и побочная диагональ. Если найдено такое значение – переменная отправляется в вызывающий метод. Если нет – возвращается пробел

**public** **static** String CheckGame(JTable table) {

String winner = " ";

// Проверка по горизонтали

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

**if** (table.getModel().getValueAt(i, 0) == table.getModel().getValueAt(i, 1)

&& table.getModel().getValueAt(i, 1) == table.getModel().getValueAt(i, 2)

&& table.getModel().getValueAt(i, 0) != "") {

winner = "" + table.getModel().getValueAt(i, 0);

**break**; } }

// Проверка по вертикали если победитель пока не найден

**if** (winner == " ") {

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

**if** (table.getModel().getValueAt(0, i) == table.getModel().getValueAt(1, i)

&& table.getModel().getValueAt(1, i) == table.getModel().getValueAt(2, i)

&& table.getModel().getValueAt(0, i) != "") {

winner = "" + table.getModel().getValueAt(0, i);

**break**;}}}

// Проверка главной диагонали если победитель пока не найден

**if** (winner == " ") {

**if** (table.getModel().getValueAt(0, 0) == table.getModel().getValueAt(1, 1)

&& table.getModel().getValueAt(1, 1) == table.getModel().getValueAt(2, 2)

&& table.getModel().getValueAt(1, 1) != "") {

winner = "" + table.getModel().getValueAt(1, 1);} }

// Проверка побочной диагонали если победитель пока не найден

**if** (winner == " ") {

**if** (table.getModel().getValueAt(0, 2) == table.getModel().getValueAt(1, 1)

&& table.getModel().getValueAt(1, 1) == table.getModel().getValueAt(2, 0)

&& table.getModel().getValueAt(1, 1) != "") {

winner = "" + table.getModel().getValueAt(1, 1);} }

// Возвращаем победителя или пробел, если такового пока нет

**return** winner;}

1. Ход пользователя

**public** **static** **void** doGo() {

// проверяем ход клиента или нет, выбрана ли ячейка

**if** (*label\_1*.getText().equals("Ходит: " + StartApp.*getName*())&&*table*.getModel().getValueAt(*table*.getSelectedRow(), *table*.getSelectedColumn()) == ""&&!*textPane*.getText().contains("!")) {

// если да, ставим значение в пустую ячейку

*table*.getModel().setValueAt("X", *table*.getSelectedRow(), *table*.getSelectedColumn());

//добавляем запись в массив

StartApp.*doGo*(*table*.getSelectedRow(), *table*.getSelectedColumn());

// выводим в консоль для отладки

System.***out***.println("Ходим по координатам: " + *table*.getSelectedRow() + "," + *table*.getSelectedColumn()+ "\n" + *textPane*.getText());

//выводим ход в историю ходов

*textPane*.setText("Ходим по координатам: " + (*table*.getSelectedRow() + 1) + ","+ (*table*.getSelectedColumn() + 1) + "\n" + *textPane*.getText());

// говорим что походили

*iGo* = 0;

//если играем с пк, говорим ему ходить

**if** (StartApp.*getTypeGame*() == StartApp.***PC***) { *ansGo*();

//если играем с пользователем, отправляем ему свой ход

} **else** **if** (StartApp.*getTypeGame*() == StartApp.***USER***) {

StartApp.*setMessageForSever*("play," + *table*.getSelectedRow() + "," + *table*.getSelectedColumn()); } // проверяем выиграл ли кто

*chechWin*(); }

}

1. Выбор и ожидание соперника

// читаем чат

input = in.readLine();

System.out.println("В чате пишут: " + input);

// делим на команды

String[] ans = input.split(",");

// принимаем: поиск, имя, кто ходит

if (ans[0].contains("find")) {

// если ищем мы

boolean findYes = false;

myName = ans[1];

// прогоняем весь список игроков

for (Serverss serv : Server.serverList) {

// если ищет кто-то еще, устанавливаем ему врагами нас, делаем его своим врагом. Отправляем запрос на старт игры

if (serv.isSearh) {

this.idEnemy = Server.serverList.indexOf(serv);

serv.idEnemy = Server.serverList.indexOf(this);

serv.isSearh = false;

boolean firstHod = false;

int iGo = Math.random() > 0.5 ? 1 : 0;

if (iGo == 1)

firstHod = true;

else

firstHod = false;

out.println("findYes," + firstHod + "," + serv.myName);

serv.writeMsg("findYes," + !firstHod + "," + myName);

findYes = true;

System.out.println("Нашли врагов и направили их друг на друга");

break;}}

// если никто больше не ищет, говорим всем что ищем мы и ожидаем соперника

if (!findYes) {

isSearh = true;

out.println("Ожидаем");}

// если нажата кнопка назад, прощается с соперником

} else if (ans[0].contains("out")) {

idEnemy = -1;

// для отладки выводим ход соперника в консоль

} else if (ans[0].contains("play")) {

System.out.println("Пришла ответка от " + ID);

if(idEnemy != -1)

Server.serverList.get(idEnemy).writeMsg(input);}}

# Тестирование

Проведем тестирование программы, пройдя все варианты взаимодействия. Запустим само приложение. Запуск происходит без проблем. Удалим имя и адрес сервера, выберем играть с пк

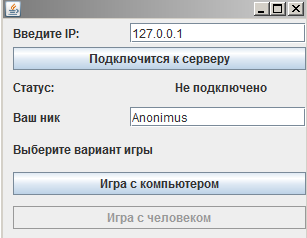


Рисунок 5.1 – Запуск главного окна

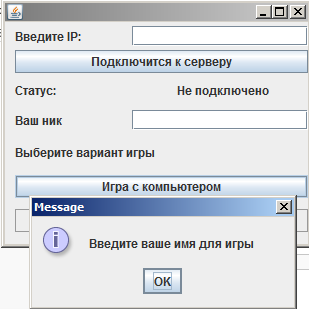


Рисунок 5.2 – Попытка запуска игры без введенного имени

Как и ожидалось, нам выдало уведомление,что неободимо перед началом игры ввести имя. Поддерживается как английкий, так и русский язык, спец символы и цифры. Игра с пк проходит без проблем.

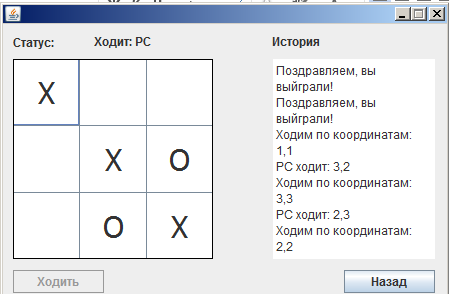


Рисунок 5.3 – Игра с ПК

Попробуем подключится, не вводя адрес сервера

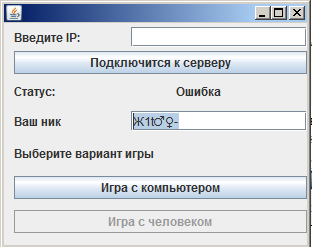


Рисунок 5.4 – Вход в игру при ошибочном адресе

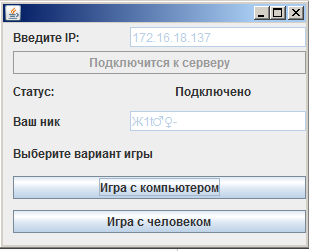


Рисунок 5.5 – Успешное подключение

Приложение попыталось подключится к несуществующему серверу, у него не получилось, и он уведомляет пользоватею об ошибке. Так будет до тех пор, пока не будет введен правильный и рабочий адрес сервера.

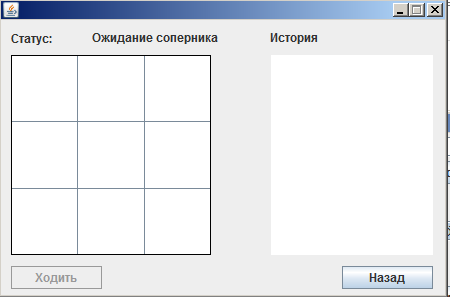


Рисунок 5.6 – Окно ожидание соперника

При нажати играть с человеком, происходит ожидание соперника. В этот момент можно выйти и начать игру с пк, ожидание прерверся без ошибок, и можно легко начать поиск соперника еще раз.

Процесс игры происходит без ошибок

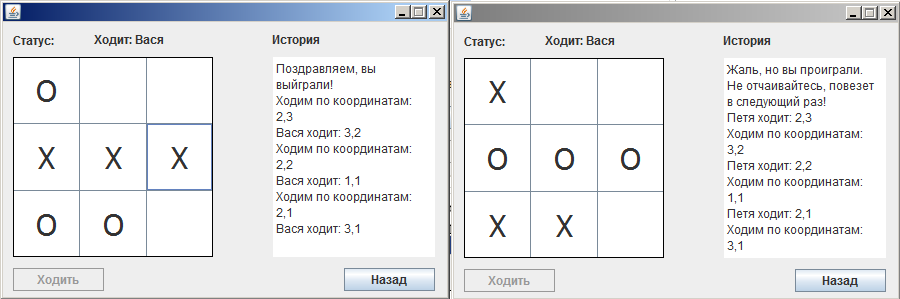


Рисунок 5.7 – успешное окончание игры

## Литература

1. Хорстманн, К.С. Java 2. Библиотека профессионала, том1. Основы. 8-е издание, : Пер. с англ. / К. С. Хорстманн, Г. Корнелл - М.: ООО «Вильямс», 2012. - 816 с.
2. Дирк, Л. Самоучитель Java 7: Пер. с нем. / Л. Дирк, П. Мюллер. – СПб: БХВ-Петербург, 2013. – 464 с.: илл.
3. Шилдт, Г. Java. Полное руководство. 8-е издание, : Пер. с англ. / Г. Шилдт. – М.:ООО «Вильямс», 2012. – 1104 с.
4. Блинов, И.Н. Java. Промышленное программирование./ И.Н. Блинов, В.С. Романчик –Минск: «Четыре четверти», 2013. – 896 с.

## Приложение А

Исходный код, откомпилированная программа и записка находятся на диске.