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 народных словесных игр, которые не требуют обучения или много времени. Говоря конкретно, проект интересен всем любителям игры слова, города и всех производных игр. Суть игры – переиграть игрока в слова, называя слово на последнюю букву слова соперника. Данная программа значительно облегчит поиск соперника (которого приходилось искать самому в огромном мире), 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.

Игра в слова – словесная игра между игроками. Количество игроков варьируется в зависимости от предпочтения самих игроков. Обычно играют от 2 до 7 человек. В данной программе решил сделать игру для двух человек. Сперва выбирают чей первый ход. Затем первый человек называет свое слово. Обычно договариваются над типом игры. Так, любо называют только существительные, только имена, только на одну букву и тд.

Выигрыш фиксируется, если соперник не смог сказать слово (к примеру на последнюю букву) за оговоренное время. Ничьей считается, если у соперников еще есть слова, но игра слишком затянулсь.

## Построение концептуальной модели предметной области.

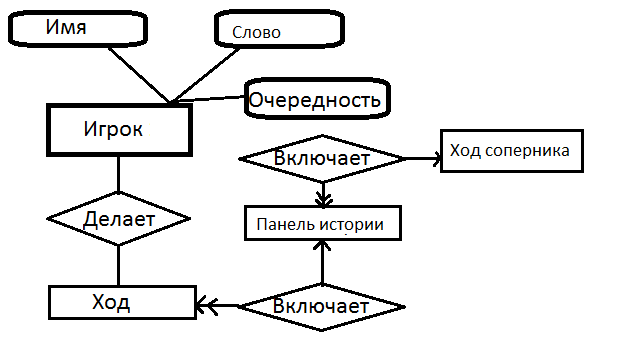


Рисунок 1.2 – модель предметной области

Концептуальная модель состоит из Игрока, имеющего имя, слово, которым ходит, и очередность хода. Когда игрок делает ход, слово заносится в панель истории. История проверяется на существования заданного слова в себе до этого. Если такое слово есть – игрок должен будет ввести слово, которое еще не называли. При успешном введении слова, действия повторяются для другого игрока, пока у одного из них не кончатся слова или пока оба не признают ничью.

# Постановка задачи

## Определение требований к программной системе.

1. Приложение, позволяющее пользователю играть в слова
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

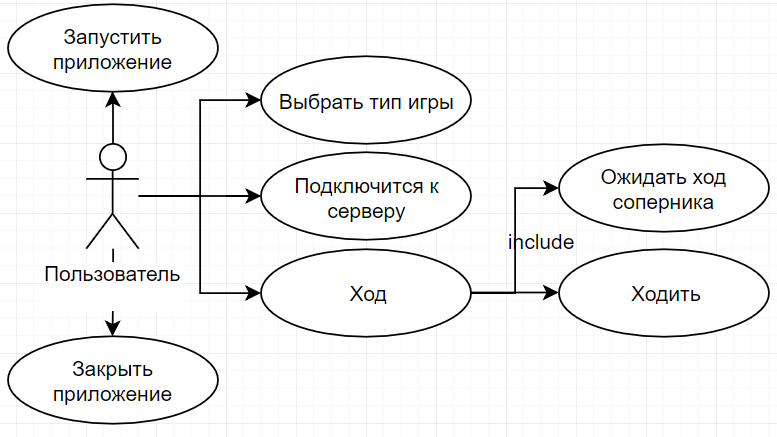


Рисунок 2.1 - диаграмма вариантов использования

## 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 «Слово за слово» и «Городки». Так же во многих играх меняются правила, например: называть слова на последнюю букву, на одну букву, на одну тему.

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 классической игрой в слова на последнюю букву.

### 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

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 панель с отображение ходов, небольшая история и кнопка ходить. По кнопке на сервер будет отправляться сообщение словом. Очередность хода будет смотреть по переменной iGo.

## Проектирование структур хранения данных.

Для хранения данных слов используется текстовый документс более чем 30 000 существительных русского языка. Все это дело переносится при старте игры в переменную words, к которой доступ получаем по индексу к определенному слову.

# Реализация

## Разработка архитектуры программного продукта

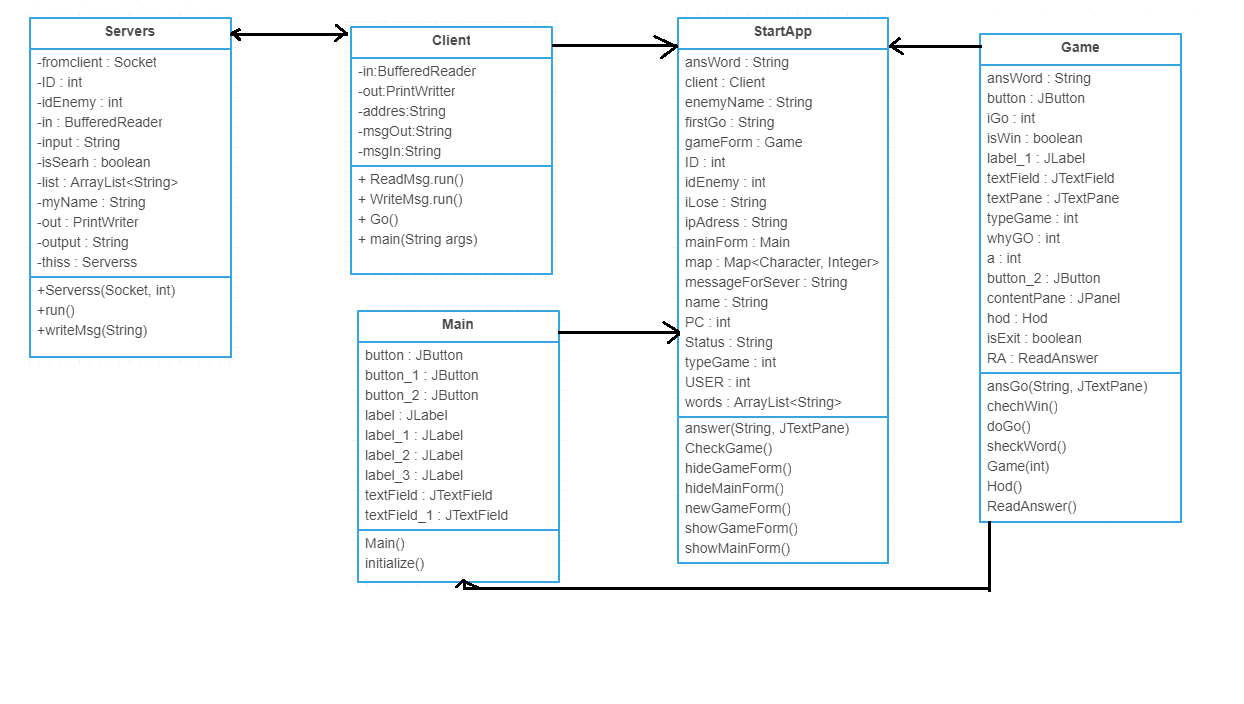


Рисунок 4.1 – Диаграмма классов

Главным классом является класс StartApp, у которого есть переменные имени, имени врага, хранения статуса программы. Методы ответа пк 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 пересылает в экземпляр класса Client. Последний в методе Go(), подключается к серверу 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

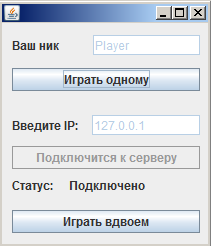
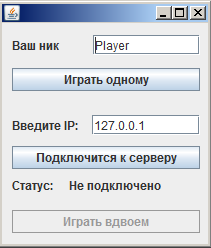


Рисунок 4.2.1 – Вид главного окна

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.

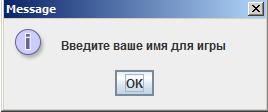


Рисунок 4.2.2 – Ошибка при отсутствии имени

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

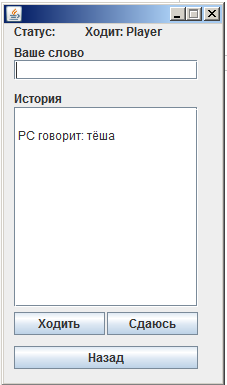


Рисунок 4.2.3 – Окно игры против ПК

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

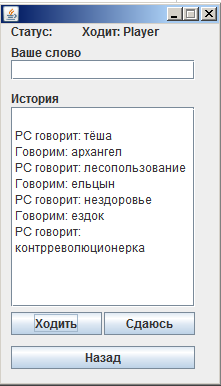


Рисунок 4.2.4 – Один из вариантов окончания игры.

Ходить можно нажатием на кнопку. После окончания игры в истории ходов можно видеть победителя. После окончания игры остается выйти в главное меню кнопкой «Назад».

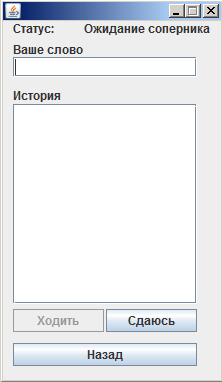


Рисунок 4.2.5 – Окно в режиме ожидания соперника

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

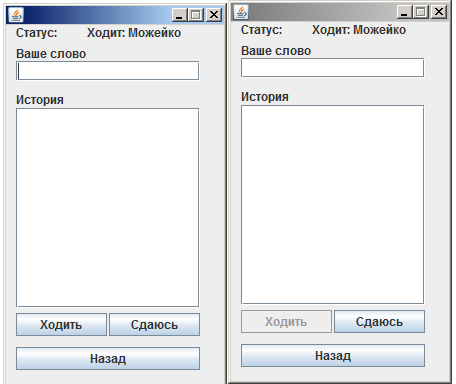


Рисунок 4.2.6 – Окна игры двух пользователей

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

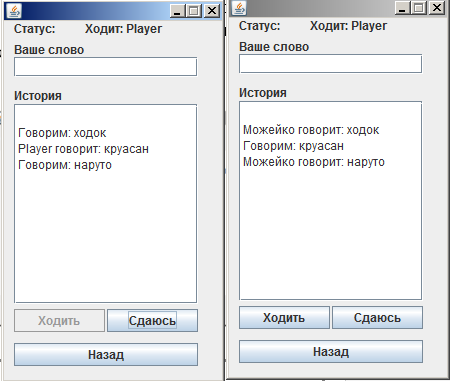


Рисунок 4.2.7 – Окна игры двух пользователей при игре против друг-друга

Когда пользователь пытается ходить словом, которое ранее называлось, ему выдается соответствующее уведомление

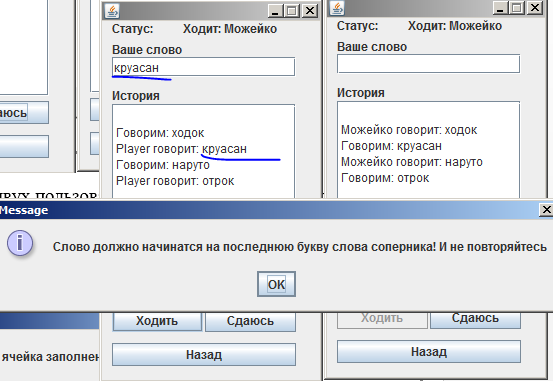


Рисунок 4.2.8 – Окно ошибки, при попытки заполнить заполненную ячейку

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

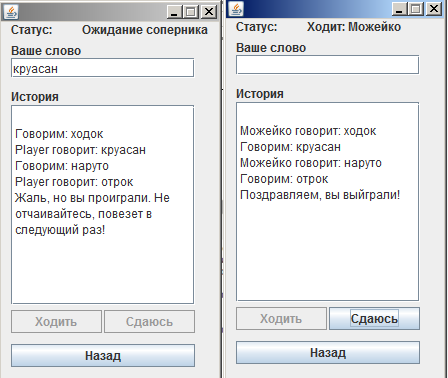


Рисунок 4.2.9 – Окна пользователей по окончанию игры.

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

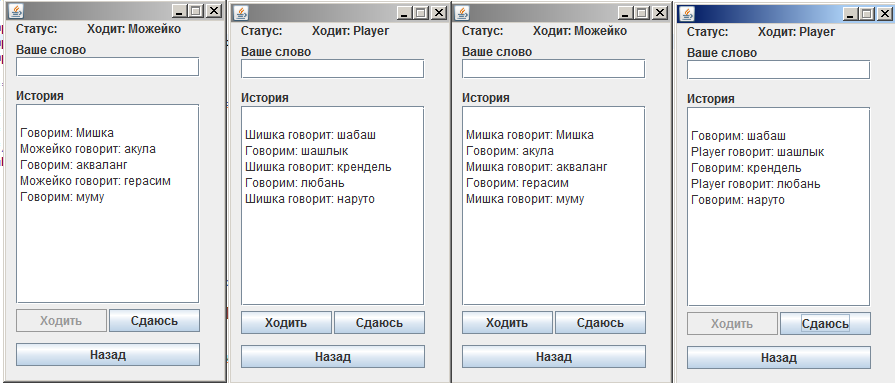


Рисунок 4.2.10 – Окна игры сразу четверых игроков

## 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 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
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 словом. Компьютер получает слово игрока, смотрит на какой символ он заканчиваетяс. Если наэтот символ нету слова, он берет предыдущий. Затем из списка слов он находит индекс начала и конца слов на эту букву. После чего рандомным образом выбирает слово из этого списка. До тех пор, пока он не найдет не использованное слово, он будет перебирать значения. Как только слово найдено, пк ставит ход и передает ход пользователю

**public** **static** String answer(String ansWord, JTextPane pane) {

String answer = "";

**if** (ansWord.charAt(ansWord.length() - 1) == 'ъ' ||ansWord.charAt(ansWord.length() - 1) == 'й' ||ansWord.charAt(ansWord.length() - 1) == 'ё'|| ansWord.charAt(ansWord.length() - 1) == 'ы'

|| ansWord.charAt(ansWord.length() - 1) == 'ь')

ansWord = ansWord.substring(0, ansWord.length() - 1);

**char** chr = ansWord.charAt(ansWord.length() - 1);

**int** wordsEnd = *map*.get(chr);

**char** ch = (**char**) (chr-1);

**int** wordsStart;

**if** (chr == 'а')

wordsStart = 0;

**else**

wordsStart = *map*.get(ch);

System.***out***.println("слово врага - " + ansWord + ", буква на которую говорим - " + chr

+ ", конец слов на эту букву - " + wordsEnd + ", начало слов на эту букву - " + ch+","+wordsStart);

**boolean** find = **true**;

**while** (find) {

**int** index = (**int**) (Math.*random*() \* (wordsEnd - wordsStart));

answer = *words*.get(index + wordsStart);

find = pane.getText().contains(answer);

}

Game.*ansWord* = answer;

**return** answer;

}Проверка выигрыша реализована следующим алгоритмом:

**public** **static** **void** chechWin() {

String check = StartApp.*CheckGame*();

**if** (check.equals("Win") || check.equals("Lose")) {

**if** (check.equals("Win"))

*textPane*.setText(*textPane*.getText() + "\n" + "Поздравляем, вы выйграли!");

**if** (check.equals("Lose"))

*textPane*.setText(

*textPane*.getText() + "\n" + "Жаль, но вы проиграли. Не отчаивайтесь, повезет в следующий раз!");

*isWin* = **true**;

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

*button*.setEnabled(**false**);

// StartApp.setMessageForSever("out");

}

}Ход пользователя

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

// проверка можем ли ходить

**if** (*label\_1*.getText().equals("Ходит: " + StartApp.*name*) && !*textPane*.getText().contains("!")) {

**if** (!*sheckWord*()) {

JOptionPane.*showMessageDialog*(**null**,

"Слово должно начинатся на последнюю букву слова соперника! И не повторяйтесь");

**return**;

}

// выводим ход в историю ходов

*textPane*.setText(*textPane*.getText() + "\n" + "Говорим: " + *textField*.getText());

// говорим что походили

*iGo* = 0;

// если играем с пк, говорим ему ходить

**if** (StartApp.*typeGame* == StartApp.***PC***) {

*ansGo*(*textField*.getText(), *textPane*);

// если играем с пользователем, отправляем ему свой ход

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

StartApp.*messageForSever* = "play," + *textField*.getText();

}

*textField*.setText("");

// проверяем выиграл ли кто

*chechWin*();

}

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

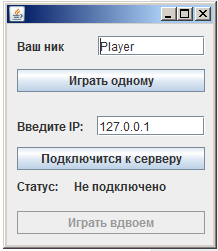


Рисунок 5.1 – Запуск главного окна

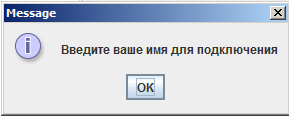


Рисунок 5.2 – Попытка запуска игры без введенного имени

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.

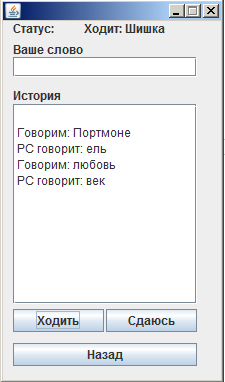


Рисунок 5.3 – Игра с ПК

Попробуем подключится, не вводя адрес сервера

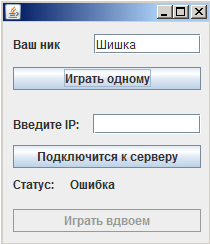


Рисунок 5.4 – Вход в игру при ошибочном адресе

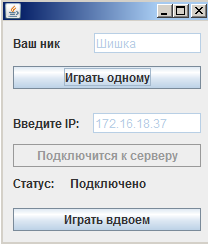


Рисунок 5.5 – Успешное подключение

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.

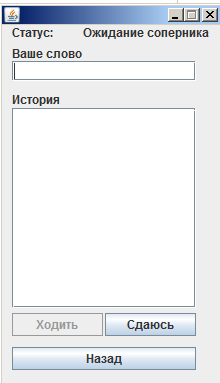


Рисунок 5.6 – Окно ожидание соперника

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

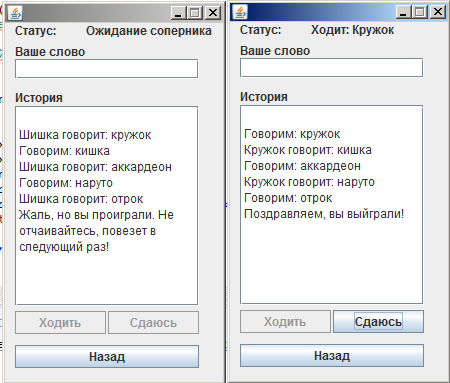


Рисунок 5.7 – успешное окончание игры

## 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.