

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский
университет имени академика С.П. Королева (Самарский университет)»

Институт _____ информатики и кибернетики _____

Кафедра _____ программных систем _____

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

_____ к курсовой работе по дисциплине «Объектная распределенная
разработка» по теме «Разработка клиент-серверного приложения «Игра
«Крестики-Нолики»» с использованием технологии RMI»

Обучающаяся _____ Н.А. Кирш

Руководитель _____ О.А. Гордеева

Самара 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский
университет имени академика С.П. Королева (Самарский университет)»

Институт _____ информатики и кибернетики _____

Кафедра _____ программных систем _____

ЗАДАНИЕ

на курсовую работу по дисциплине
«Объектная распределенная обработка»
обучающейся в группе № 6414-020302D

Н.А. Кирш

1 Тема работы: «Разработка клиент-серверного приложения «Игра
«Крестики-Нолики»» с использованием технологии RMI»

2 Исходные данные к работе: см. приложение к заданию

3 Перечень вопросов, подлежащих разработке:

3.1 Произвести анализ предметной области

3.2 Выполнить обзор используемых технологий

3.3 Разработать информационно-логический проект системы по
методологии UML

3.4 Разработать и реализовать программное и информационное
обеспечение, провести его тестирование и отладку

3.5 Оформить документацию

Задание приняла

к исполнению _____ Н.А. Кирш

РЕФЕРАТ

Пояснительная записка 28 с, 9 рисунков, 1 таблиц, 8 источников, 1 приложения.

ИГРА, ИГРА «КРЕСТИКИ-НОЛИКИ», ТЕХНОЛОГИЯ RMI, JAVA.

Объектом автоматизации является игра «Крестики-Нолики».

Во время курсовой работы был разработан алгоритм совершения хода игроком-компьютером и соответствующая программа, позволяющая выполнять генерацию игрового поля и отслеживать ход игры. Игра может быть перезапущена в любой момент пользователем.

Программа написана на языке Java в среде IntelliJ IDEA 2022 и функционирует под управлением операционной системы Windows 7 и выше.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 Описание и анализ предметной области	6
2 Постановка задачи.....	6
2.1 Характеристика объекта моделирования:.....	6
2.2 Общие требования к проектируемой системе:	7
3 Описание используемых технологий	8
3.2 Java Swing	9
4 Проектирование системы.....	9
4.1 Структурная схема системы	9
4.2 Разработка информационно-логического проекта системы	11
4.2.1 Диаграмма вариантов использования.....	11
4.2.2 Диаграмма деятельности.....	13
4.2.3 Диаграмма компонентов	14
4.2.4 Диаграмма последовательности	16
5 Описание интерфейса системы	17
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	20
ПРИЛОЖЕНИЕ А Листинг модулей программы	21

ВВЕДЕНИЕ

Целью курсовой работы является создание клиент-серверного приложения, закрепление навыков работы с технологиями для решения поставленной задачи распределённой обработки, определение и формализация требований к системе, накладываемых реализуемой технологией обработки.

Во время курсовой работы необходимо разработать автоматизированную систему распределенной обработки данных, реализующую игры «Крестики-нолики».

Разработка системы будет производиться по технологии быстрой разработки приложений RAD (Rapid Application Development), которая поддерживается методологией структурного проектирования и включает элементы объектно-ориентированного проектирования и анализа предметной области [1].

При проектировании системы будут использоваться методология ООАП (Object-Oriented Analysis/Design), в основу которой положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов, и язык моделирования UML (Unified Modeling Language), который является стандартным инструментом для разработки «чертежей» программного обеспечения [2].

1 Описание и анализ предметной области

Предметной областью часто называют ту часть реального мира, которая имеет непосредственное отношение к разработке той или иной программы. Предметная область состоит из тех частей, которые могут оказать влияние на создание конкретных объектов и их взаимосвязей между объектами.

Распределенная система – набор независимых узлов, не имеющих общей совместно используемой памяти и общего единого времени (таймера) и взаимодействующих через коммуникационную сеть посредством передачи сообщений. В курсовой работе для создания распределённой системы, реализующей игру, применено функциональное разделение на клиентскую и серверную часть. Таким образом, различные узлы системы будут выполнять разные задачи.

Крестики-нолики— логическая игра между двумя противниками на квадратном поле 3×3 клетки или большего размера (вплоть до «бесконечного поля»). В классической игре игроки по очереди ставят на свободные клетки поля $n \times n$ знаки (один всегда крестики, другой всегда нолики). Первый, выстроивший в ряд свои фигуры по вертикали, горизонтали или диагонали, выигрывает. Если игроки заполнили все ячейки и оказалось, что ни в одной вертикали, горизонтали или диагонали нет одинаковых знаков, партия считается закончившейся в ничью. Первый ход делает игрок, ставящий крестики [3].

2 Постановка задачи

2.1 Характеристика объекта моделирования:

- 1) объект моделирования: игра «Крестки-нолики;
- 2) виды моделируемой деятельности:
 - процесс генерирования игрового поля размера $N \times N$;
 - процесс ведения игры согласно правилам;

- процесс выбора хода противником;
 - процесс визуализации игрового процесса.
- 3) правила игры «Крестики-нолики»:
- игровое поле должно иметь размер не менее 3x3 и не более 5x5 клеток;
 - игроки ходят по очереди;
 - игра завершается победой одного из игроков, когда он выстроил в ряд свои фигуры по вертикали, горизонтали или диагонали. Если поле оказалось заполненным и оказалось, что ни в одной вертикали, горизонтали или диагонали нет одинаковых знаков, игра закончена в ничью.
- 4) входные данные системы:
- размеры поля $N \times N$;
 - координаты хода игрока.
- 5) выходные данные системы:
- двумерный массив клеток поля;
 - координаты хода компьютера.

2.2 Общие требования к проектируемой системе:

- 1) клиентское приложение:
- предоставление игроку возможности задать размер игрового поля;
 - предоставление возможности сделать ход игроку;
 - визуализация игрового процесса.
- 2) серверное приложение:
- генерация игрового поля согласно размерам, заданным пользователем;
 - совершение хода противником;
 - определение игрока, который имеет право сделать ход;

- определение итогов игры.
- 3) функции пользователя:
 - задать размеры игрового поля;
 - сделать ход.
- 3 Описание используемых технологий

3.1 Технология RMI

Технология Remote Method Invocation (RMI) впервые была внедрена еще в Java Development Kit(JDK) версии 1.1. Она определяет, как объекты себя ведут, в каком месте и из-за чего могут возникнуть исключения, каким образом происходит управление памятью, а также как параметры передаются в удаленные методы и как они оттуда возвращаются. Технология RMI, поддерживаемая платформой Java, предлагает способы создания объектов, методы которых могут быть вызваны из среды других виртуальных машин, – в том числе и работающих на других хост-компьютерах.

Архитектура технологии RMI основывается на одном важном правиле: определение поведения и реализация этого поведения должны быть физически разделены. Это значит, что RMI позволяет хранить отдельно описание поведения и код его реализации на нескольких отдельно работающих JVM. Изначально в качестве определения удаленного сервиса в RMI используют интерфейсы Java (interfaces). Ниже представлена диаграмма, которая наглядно демонстрирует эту концепцию разделения (см. рисунок 1).

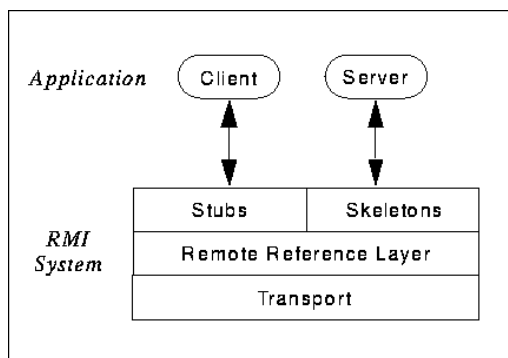


Рисунок 1 – Концепция разделения RMI

Интерфейсы не содержат никакого исполняемого кода. Когда клиентская программа делает вызов метода прокси-объекта, RMI посылает запрос удаленной JVM, которая пересылает этот запрос коду реализации [4].

3.2 Java Swing

Java Swing – это легкий инструментальный графического интерфейса пользователя (GUI), который включает в себя богатый набор виджетов. Он включает в себя пакет, позволяющий создавать компоненты графического интерфейса для приложений Java, и не зависит от платформы.

Компоненты Swing поддерживают специфические динамически подключаемые виды и поведения, благодаря которому возможна адаптация к графическому интерфейсу платформы (то есть к компоненту можно динамически подключить другой, специфический для операционной системы, в том числе и созданный программистом вид и поведение). Таким образом, приложения, использующие Swing, могут выглядеть как родные приложения для данной операционной системы. Основным минусом таких «легковесных» компонентов является относительно медленная работа. Положительная сторона — универсальность интерфейса созданных приложений на всех платформах [5].

4 Проектирование системы

4.1 Структурная схема системы

Система должна представлять собой совокупность элементов (объектов, субъектов), находящихся между собой в определенной зависимости и составляющих некоторое единство (целостность), направленное на достижение определенной цели. Система может являться элементом другой системы более высокого порядка (надсистема) и включать в себя системы более низкого порядка (подсистемы). То есть систему можно рассматривать как набор подсистем, организованных для достижения определенной цели и описанных с помощью набора моделей (возможно, с

различных точек зрения), а подсистему – как группу элементов, часть которых составляет спецификацию поведения, представленного другими ее составляющими.

Как следует из определения, отличительным (главным свойством) системы является ее целостность: комплекс объектов, рассматриваемых в качестве системы, должен обладать общими свойствами и поведением. Очевидно, необходимо рассматривать и связи системы с внешней средой. В самом общем случае понятие «система» характеризуется [6]:

- наличием множества элементов;
- наличием связей между ними;
- целостным характером данного устройства или процесса.

На рисунке 2 приведена структурная схема системы.

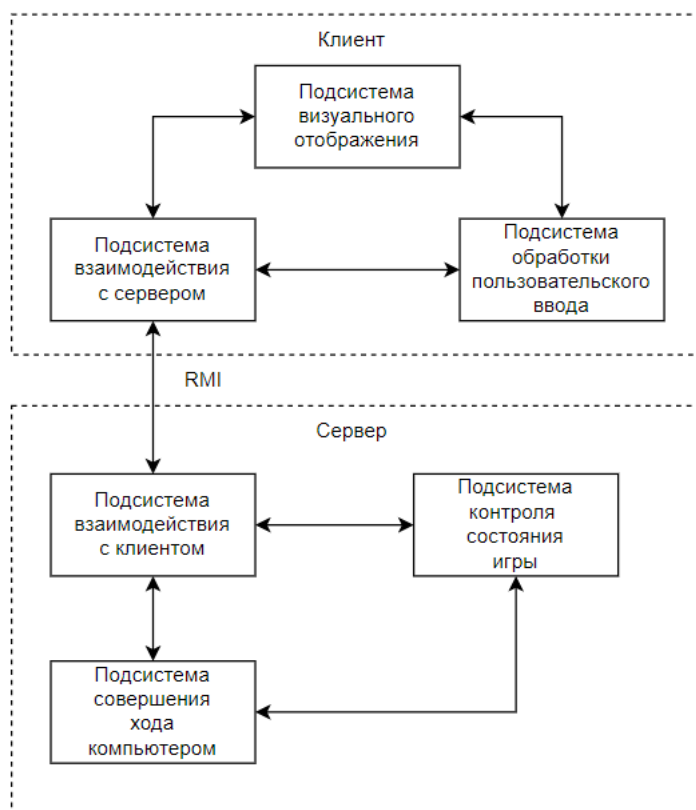


Рисунок 2 – Структурная схема системы

В состав разрабатываемой системы клиентской части входят следующие подсистемы:

- 1) подсистема взаимодействия с сервером, отвечающая за передачу на сервер информации от клиента;
- 2) подсистема визуального отображения, отвечающая за отображение текущего игрового поля;
- 3) подсистема обработки пользовательского ввода, позволяющая игроку ввести размеры игрового поля и сделать ход;

В состав серверной части системы входят следующие подсистемы:

- 1) подсистема взаимодействия с клиентом, отвечающая за получение и передачу игровых данных;
- 2) подсистема контроля состояния игры, контролирующая, какой игрок может сделать ход и хранящая текущее игровое поле;
- 3) подсистема совершения хода компьютером, симулирующая ход компьютера;

4.2 Разработка информационно-логического проекта системы

Для специфицирования (построения точных, недвусмысленных и полных моделей) системы и ее документирования используется унифицированный язык моделирования UML.

Унифицированный язык моделирования (Unified Modeling Language – UML) – это стандартный инструмент для разработки «чертежей» программного обеспечения. Его можно использовать для визуализации, спецификации, конструирования и документирования артефактов программных систем. UML подходит для моделирования любых систем – от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени [7].

4.2.1 Диаграмма вариантов использования

Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм.

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования со стороны пользователя приведена на рисунке 3.

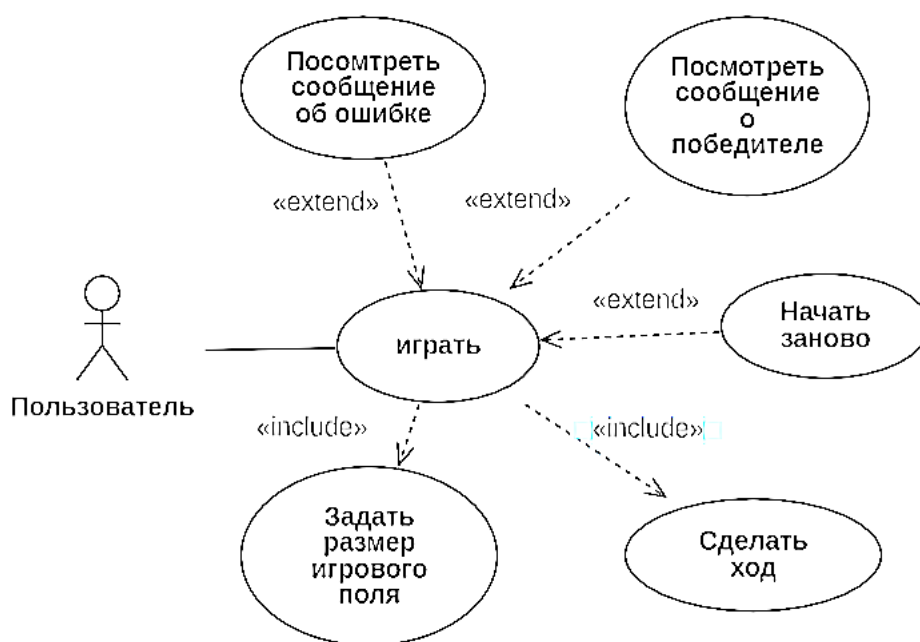


Рисунок 3 – Диаграмма вариантов использования системы

Для начала игры необходимо задать размер игрового поля. В процессе игры основной функцией пользователя является возможность сделать ход. Кроме того, находясь в процессе игры, пользователь может начать заново, нажав на соответствующую кнопку. По окончании игры пользователь может просмотреть сообщение о победителе, в случае ошибки ввода пользователь должен получать сообщение об ошибке.

4.2.2 Диаграмма деятельности

Диаграмма деятельности используется для моделирования динамических аспектов поведения системы. Она является частным случаем диаграмм состояний. Они позволяют реализовать особенности процедурного и синхронного управления, обусловленного завершением внутренних действий и активностей [7].

На рисунке 3 приведена диаграмма деятельности системы. Сначала система отображает главный экран и форму ввода размера игрового поля. В процессе игры система предоставляет пользователю возможность сделать ход. После каждого хода система производит проверку, определяющую, может ли игрок сходить снова, и проверку на окончание игры.

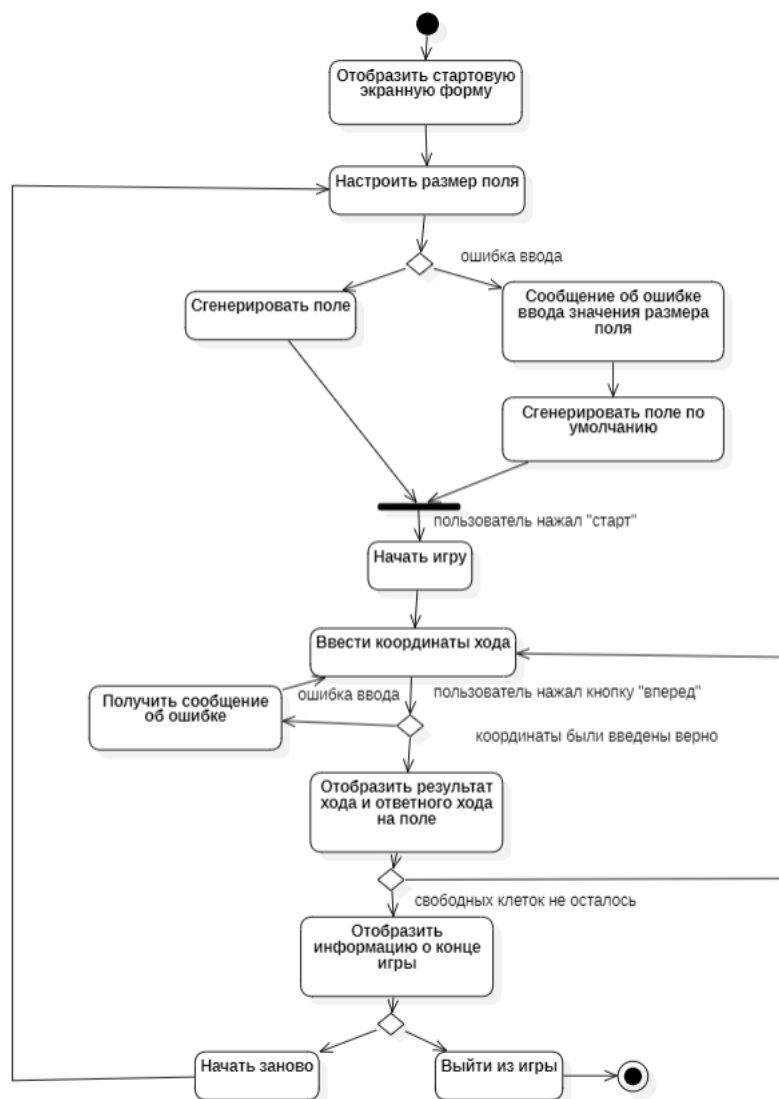


Рисунок 3 – Диаграмма деятельности системы

4.2.3 Диаграмма компонентов

Диаграмма компонентов – диаграмма физического уровня, которая служит для представления программных компонентов и зависимостей между ними [12].

Диаграмма компонентов разрабатывается для следующих целей:

- визуализация общей структуры исходного кода программной системы;
- спецификация исполнимого варианта программной системы;
- обеспечение многократного использования отдельных фрагментов программного кода;
- представление концептуальной и физической схем баз данных.

На рисунке 4 приведена диаграмма компонентов, их описание приведено в таблице 1.

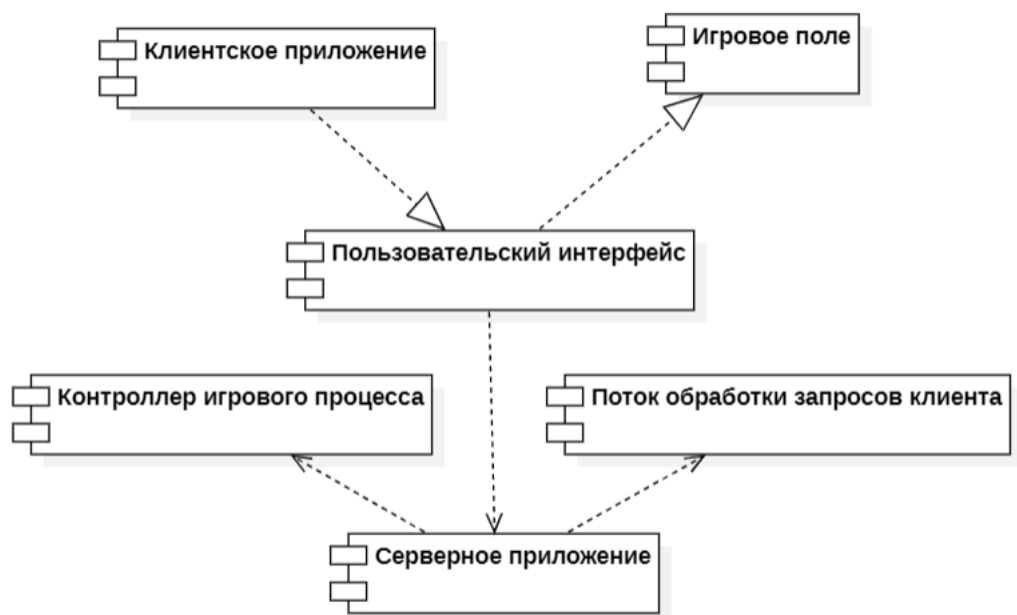


Рисунок 4 – Диаграмма компонентов системы

Таблица 1 – Описание компонентов системы

Название компонента	Назначение компонента	Подсистема
Клиентское приложение	Позволяет пользователю взаимодействовать с системой	Задания размеров поля, совершение хода пользователем
Пользовательский интерфейс		
Игровое поле	Отвечает за отображение текущего состояния игры	Отрисовка игрового поля
Серверное приложение	Позволяет работать с несколькими клиентскими приложениями одновременно	Взаимодействия с клиентом
Поток обработки запросов клиента	Отвечают за обработку запросов клиентов	
Контроллер игрового процесса	Отвечает за последовательность действий в игровом процессе и за хранение позиций на поле	Совершения хода компьютером, хранения клеток поля, определения

		активного игрока, определение итогов игры
--	--	--

4.2.4 Диаграмма последовательности

Диаграмма последовательности UML (sequence diagram) — такая диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления. Основные элементы диаграммы последовательности это: обозначения объектов (прямоугольники), вертикальные линии, отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами [20].

На рисунке 5 приведена диаграмма последовательности для варианта использования «Сделать ход».

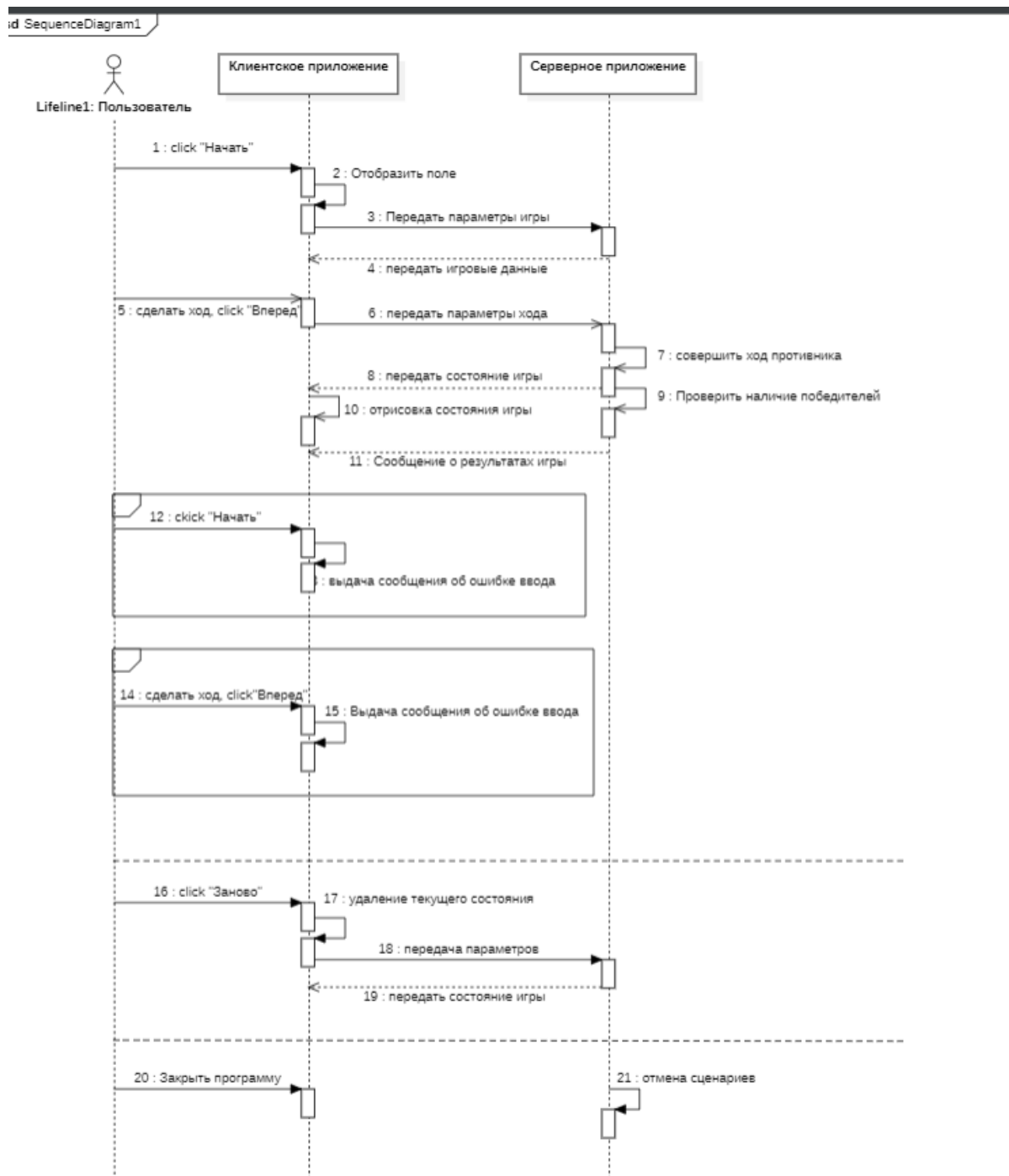


Рисунок 5 – Диаграмма последовательности для варианта использования «Сделать ход»

5 Описание интерфейса системы

При запуске клиентской части приложения пользователь видит начальный экран, представленный на рисунке 6. Здесь пользователь может задать размеры игрового поля, далее по нажатию на кнопку “Начать” генерируется

поле нужного размера. В любой момент игры пользователь может начать игру заново, нажав кнопку «Заново!»

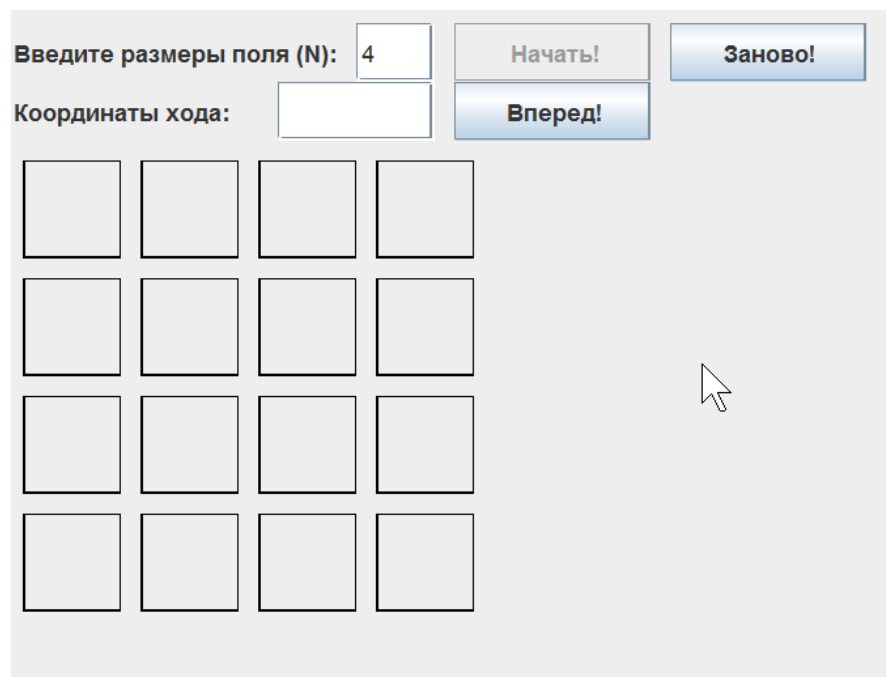


Рисунок 6 – Стартовая экранная форма

После начала игры, пользователь вводит координаты хода в соответствующую область, по нажатию кнопки «вперед» (Рисунок 7), система отображает ход пользователя и компьютер совершает ответный ход.

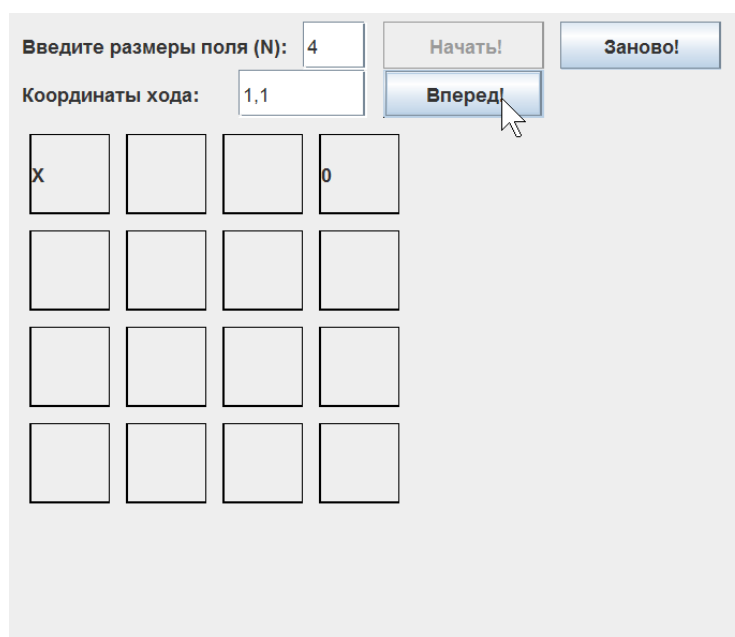
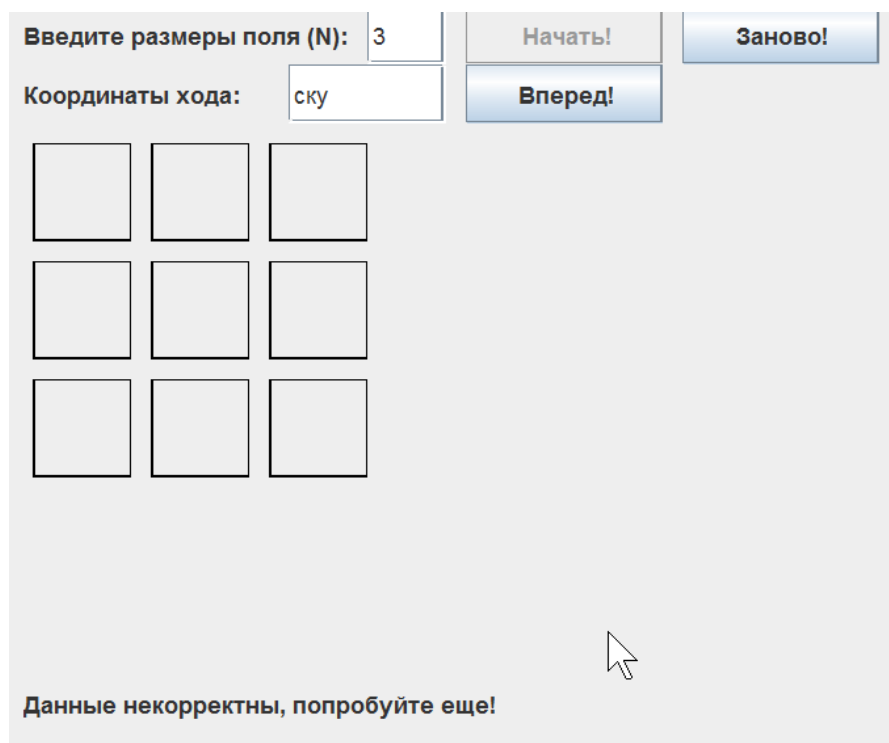


Рисунок 7 – Экранная форма процесса игры

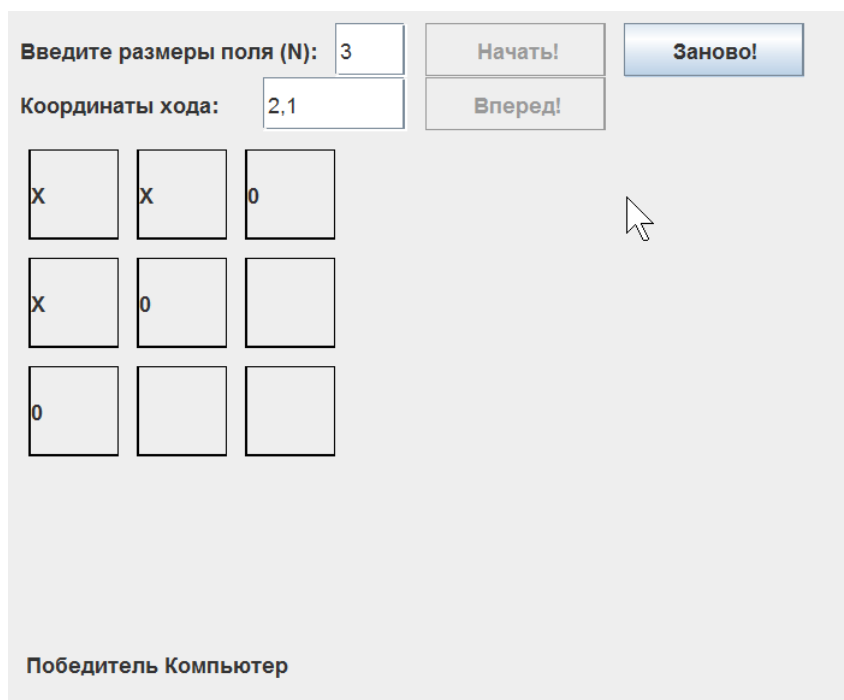
Если какие-либо данные введены некорректно, система выдает сообщение об ошибке, представленное на рисунке 8.



The screenshot shows a game interface with a 3x3 grid. At the top, there is a label "Введите размеры поля (N):" followed by a text input containing the number "3". To the right of this input are two buttons: "Начать!" (disabled) and "Заново!" (active). Below the input is a label "Координаты хода:" followed by a text input containing "ску". To the right of this input is a button "Вперед!". The 3x3 grid is empty. At the bottom of the interface, a message reads "Данные некорректны, попробуйте еще!". A mouse cursor is visible near the bottom right of the grid.

Рисунок 8 – Сообщение об ошибке ввода

По окончании игры система выводит сообщение о победителе, представленное на рисунке 9.



The screenshot shows the same game interface as in Figure 8, but with a different state. The "Введите размеры поля (N):" input still contains "3". The "Начать!" button is now active, and the "Заново!" button is disabled. The "Координаты хода:" input now contains "2,1", and the "Вперед!" button is disabled. The 3x3 grid is partially filled with 'X' and 'O' characters. The bottom message now reads "Победитель Компьютер". A mouse cursor is visible near the bottom right of the grid.

Рисунок 9 – Информация о победителе

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Принципы разработки учебных программ: Методические указания к лабораторному практикуму по дисциплине «Программная инженерия»/ Л. С. Зеленко. – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2014. - 64 с.
- 2 Чарльз Уэзерел. Этюды программирования Изд. 2-е. М.: ДМК Пресс, 2006. 150 с
- 3 Игра «Крестики-нолики» [Электронный ресурс]. URL: <https://ru.wiktionary.org/wiki/%D0%BB%D0%B0%D0%B1%D0%B8%D1%80%D0%B8%D0%BD%D1%82> (дата обращения: 01.04.2023).
- 4 Технология RMI [Электронный ресурс]. URL: <https://studfile.net/preview/4676627/page:26/> (дата обращения 20.12.2022).
- 5 Объектный подход к проектированию [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/11876/1156/lecture/18262> (дата обращения: 15.04.2023).
- 6 Диаграмма состояний (UML) [Электронный ресурс]. URL: [https://ru.wikipedia.org/wiki/Диаграмма_состояний_\(UML\)](https://ru.wikipedia.org/wiki/Диаграмма_состояний_(UML)) (дата обращения: 15.04.2023).
- 7 Построение диаграммы последовательности (UML) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Диаграмма_последовательности (дата обращения: 15.04.2023).
- 8 Java: Особенности языка [Электронный ресурс]. URL: <https://ru.hexlet.io/blog/posts/yazyk-programmirovaniya-java-osobennosti-populyarnost-situatsiya-na-rynke-truda> (дата обращения: 16.04.2023).

ПРИЛОЖЕНИЕ А

Листинг модулей программы

```
public class Game implements IGame {
    private static int SIZE = 0;
    public static final String FILLER = " ";
    private static String[][] board;
    private static Player[] players =
        { new Player("Computer", "X"), new Player("Компьютер", "O") };
    private static Player currPlayer = null;

    public Game() {}

    public String getGame(String player, int grid)
    {
        if (grid<=3)
            this.SIZE = grid < 3 ? 3 : grid;
        else this.SIZE=grid > 5 ? 5 : grid;
        this.board = new String[SIZE][SIZE];
        this.players[0] = new Player(player, "X");
        currPlayer = players[0];
        System.out.println("fff");
        for (int i = 0; i < this.SIZE; i++) {
            for (int j = 0; j < this.SIZE; j++) {
                this.board[i][j] = FILLER;
            }
        }
        return "Ваш ход!";
    }

    public Integer[] move(int xPosition, int yPosition) { //
        Integer[] moves={-1, -1};
        if (!isValidMove(xPosition, yPosition))
            return moves;
    }
}
```

```

else {
    board[xPosition][yPosition] = currPlayer.getSymbol();
    changePlayer();
    if (!isGameOver() && getName().equals("Компьютер")) {
        moves = currPlayer.playAIMove(this);
        board[moves[0]][moves[1]] = currPlayer.getSymbol();
        System.out.println("[ход" + moves[0] + moves[1]);
        changePlayer();
    }
}

return moves;
}

public String getName() {
    return currPlayer.getName();
}

public boolean isValidMove(int xPosition, int yPosition) {
    if (xPosition >= SIZE || yPosition >= SIZE || xPosition < 0 || yPosition < 0)
        return false;
    if (!board[xPosition][yPosition].equals(FILLER))
        return false;
    return true;
}

public void changePlayer() {
    currPlayer = currPlayer.equals(players[1]) ? players[0] : players[1];
}

public boolean isGameOver() {
    return getWinner() != null || isNoMovesLeft();
}

public boolean isNoMovesLeft() {
    for (int i = 0; i < SIZE; i++) {

```

```

        for (int j = 0; j < SIZE; j++) {
            if (board[i][j].equals(FILLER)) {
                return false;
            }
        }
    }
    return true;
}

public String getWinner() {
    String rowSymbol = rowCrossed();
    String columnSymbol = columnCrossed();
    String diagonalSymbol = diagonalCrossed();
    for (Player player : players) {
        if (player.getSymbol().equals(rowSymbol)) return getWinnerName(player);
        if (player.getSymbol().equals(columnSymbol)) return getWinnerName(player);
        if (player.getSymbol().equals(diagonalSymbol)) return getWinnerName(player);
    }
    return null;
}

public String getWinnerName(Player player){
    return player.getName();
}

public String rowCrossed() {
    for (int i = 0; i < SIZE; i++) {
        String check = board[i][0];
        for (int j = 1; j < SIZE; j++) {
            if (!check.equals(board[i][j])) {
                check = FILLER;
                break;
            }
        }
        if (!check.equals(FILLER)) {
            return check;
        }
    }
}

```

```

    }
    return FILLER;
}

public String columnCrossed() {
    for (int i = 0; i < SIZE; i++) {
        String check = board[0][i];
        for (int j = 1; j < SIZE; j++) {
            if (!check.equals(board[j][i])) {
                check = FILLER;
                break;
            }
        }
        if (!check.equals(FILLER)) {
            return check;
        }
    }
    return FILLER;
}

public String diagonalCrossed() {
    String check = board[0][0];
    for (int i = 1; i < SIZE; i++) {
        if (!check.equals(board[i][i])) {
            check = FILLER;
            break;
        }
    }
    if (!check.equals(FILLER)) {
        return check;
    }
    check = board[0][2];
    for (int i = 1; i < SIZE; i++) {
        if (!check.equals(board[i][SIZE - 1 - i])) {
            check = FILLER;

```



```

        break;
    }
}
if (!check.equals(FILLER)) {
    return check;
}
return FILLER;
}
public String[][] getBoard() {
    return board;
}
}

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class Player {
    public Player(String name, String symbol) {
        this.name = name;
        this.symbol = symbol;
    }

    public String getName() {
        return name;
    }

    public String getSymbol() {
        return symbol;
    }

    @Override
    public boolean equals(Object o) {

```

```

    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Player player = (Player) o;
    return Objects.equals(name, player.name) &&
        Objects.equals(symbol, player.symbol);
}

```

@Override

```

public int hashCode() {
    return Objects.hash(name, symbol);
}

```

```

private String name;

```

```

private String symbol;

```

```

public Integer[] playAIMove(Game game) {

    String[][] board = game.getBoard();
    Move bestMove = new Move(-1, -1, Integer.MIN_VALUE);
    int gridSize = board.length;
    for (int i = 0; i < gridSize; i++) {
        for (int j = 0; j < gridSize; j++) {
            if (board[i][j].equals(Game.FILLER)) {
                board[i][j] = getSymbol();
                Move move = new Move(i, j, calculateMoveCost(getSymbol(),
board));
                if (move.getScore() > bestMove.getScore()) {
                    bestMove = move;
                }
            }
        }
    }
}

```

```

        board[i][j] = Game.FILLER;
    }
}

Integer[] best={bestMove.getRow(), bestMove.getColumn()};
//game.move(bestMove.getRow(), bestMove.getColumn());
return best;
}

private int calculateMoveCost(final String symbol, String[][] board) {
    int size = board.length;
    List<Scorer> scorerList = new ArrayList<>();
    for (int i = 0; i < size; i++) {
        Scorer rowScorer = new Scorer();
        Scorer colScorer = new Scorer();
        for (int j = 0; j < size; j++) {
            scoreBasedOnSymbol(symbol, board[i][j], rowScorer);
            scoreBasedOnSymbol(symbol, board[j][i], colScorer);
        }
        scorerList.add(rowScorer);
        scorerList.add(colScorer);
    }

    Scorer diagonal1Scorer = new Scorer();
    Scorer diagonal2Scorer = new Scorer();
    for (int i = 0; i < size; i++) {
        scoreBasedOnSymbol(symbol, board[i][i], diagonal1Scorer);
        scoreBasedOnSymbol(symbol, board[i][size - i - 1], diagonal2Scorer);
    }
    scorerList.add(diagonal1Scorer);

```

```

    scorerList.add(diagonal2Scorer);

    int score = 0;
    for (Scorer scorer : scorerList) {
        score += scorer.getScore(size);
    }

    return score;
}

private void scoreBasedOnSymbol(String symbol, String symbolToCompare,
Scorer scorer) {
    if (symbol.equals(symbolToCompare))
        scorer.addWin();
    else if (Game.FILLER.equals(symbolToCompare))
        scorer.addTie();
    else
        scorer.addLoss();
}
}

```