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 Тетрис»,
основанного на технологии RMI»

Выполнил:
обучающийся группы № 6401-020302D А.С. Титов

Проверил:
руководитель О. А. Гордеева

Дата защиты _____

Оценка _____

Самара 2025

ЗАДАНИЕ

Написать распределенное клиент-серверное приложение, используя технологию распределенной объектной обработки RMI.

Приложение реализует игру «Тетрис» со следующими правилами:

– в игре присутствует один игрок, который заполняет игровое поле фигурами;

– в игре есть базовый набор фигур, размер поля также можно задавать;

– серверное приложение предлагает игроку последовательно фигуры из набора, игрок должен их расположить на поле снизу-вверх;

– игрок может управлять фигурами до момента размещения их на поле (вращать, двигать);

– игра заканчивается, когда поле полностью заполнено фигурами снизу доверху, при этом баллы начисляются с учетом возможных пустых ячеек на поле.

РЕФЕРАТ

Пояснительная записка 43 с, 20 рисунков, 2 таблицы, 1 источников, 1 приложение.

ТЕТРИС, ИГРА, СОСТАВЛЕНИЕ ФИГУР, ГОЛОВОЛОМКА, ТЕТРАМИНО, ОБЪЕКТНАЯ РАСПРЕДЕЛЕННАЯ ОБРАБОТКА, JAVA, RMI, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ.

Во время выполнения курсовой работы было разработано распределенное клиент-серверное приложение с использованием технологии RMI, представляющее собой реализацию компьютерной игры «Тетрис». Серверное приложение обрабатывает запросы от клиентской части и возвращает сгенерированную случайно фигуру из списка базовых фигур тетрамино. Клиентское приложение отвечает за визуализацию игры, формирует запрос в виде, требуемом серверным приложением, и обрабатывает соответствующим образом полученный ответ.

Программа написана на языке Java в среде IntelliJ IDEA Community Edition 2023.2.2 и функционирует под управлением операционной системы Windows 10.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

«Тетрис» – это головоломка, построенная на использовании геометрических фигур, состоящих из четырёх квадратов. В 1984 году ее придумал инженер-компьютерщик Алексей Пажитнов, в том же году игра была представлена общественности. В 1985 году Алексей Пажитнов вместе с Дмитрием Павловским написали первоначальную версию игры на языке Паскаль для компьютера «Электроника-60», а чуть позже шестнадцатилетний школьник Вадим Герасимов переписал эту игру для IBM PC. Коммерческая версия игры была выпущена американской компанией Spectrum HoloByte в 1987 году. В последующие годы «Тетрис» во множестве различных версий был портирован на великое множество устройств, включая всевозможные компьютеры и игровые консоли, а также такие устройства, как графические калькуляторы, мобильные телефоны, медиаплееры, карманные персональные компьютеры. [1]

Во время выполнения курсовой работы необходимо разработать клиент-серверное приложение «Игра Тетрис». Разработка системы будет производиться с использованием технологии RMI (Java Remote Method Invocation), которая позволяет Java-приложению, запущенному на одной виртуальной машине, вызвать методы объекта, работающего на другой виртуальной машине JVM (Java Virtual Machine). [2].

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 [3].

1 Описание и анализ предметной области

Предметная область – это часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы. Другими словами, предметная область включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения конкретной задачи. [4]

1.1 Описание игры «Тетрис»

«Тетрис» представляет собой головоломку, построенную на использовании геометрических фигур: случайные игровые фигурки падают сверху в игровой стакан заданной ширины и высоты. [5]

Идея «Тетриса» родилась, когда Алексей Пажитнов, который увлекался головоломками ещё с детства, наткнулся на детскую настольную игру «Пентамино». Ее смысл заключается в том, чтобы сложить вместе несколько плоских фигурок, каждая из которых состоит из пяти одинаковых квадратов. Программист начал экспериментировать, решив создать её компьютерную версию, но остановился на более простом варианте, где элементы были сложены из четырёх квадратов, такие фигурки он назвал «тетрамино», от греческого «тетра» – «четыре». На рисунке 4 представлены всевозможные варианты фигурок «тетрамино». [6]

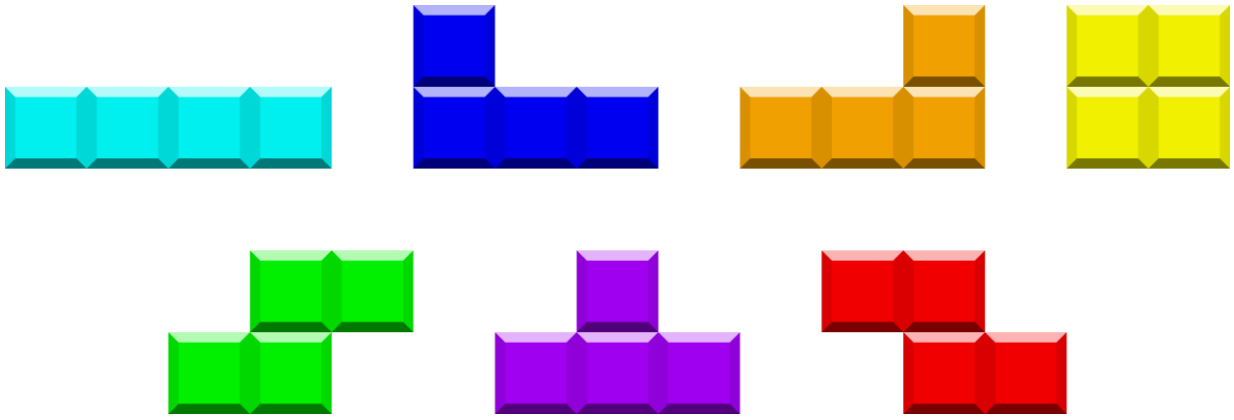

Рисунок 1 – Всевозможные варианты фигурок «тетрамино»

Первая классическая версия игры «Тетрис» была написана для советского микрокомпьютера «Электроника-60». Элементы игры состояли из пробелов, ограниченных с двух сторон скобками. Ни цвета, ни музыки не было. Стакан имел ширину 10 и высоту 20 клеток, все фигурки состояли из четырех квадратов, в полёте игрок мог поворачивать фигурку на 90°, двигать её по горизонтали налево и направо, а также ускорять падение фигурки и сбрасывать ее. Фигурка летела до тех пор, пока не натыкалась на другую фигурку или на дно стакана. Если при этом заполнился горизонтальный ряд из 10 клеток, он пропадал и всё, что выше него, опускалось на одну клетку. Темп игры постепенно ускорялся. Игра заканчивалась, когда новая фигурка не могла поместиться в стакан. [7]

На рисунке 2 представлена первая классическая версия игры «Тетрис».

Рисунок 2 – Классическая версия «Тетриса»

Начисление очков в разных версиях «Тетриса» довольно разнообразное. Очки могут начисляться за убранные линии, за сброшенные фигурки, за переход на новую скорость и т.д. При начислении очков за сброшенные фигурки могут учитываться высота, на которой остановилась фигурка, расстояние, которое пролетела фигурка после «сбрасывания». Хотя обычно приоритетом являются линии, а за фигурки начисляется относительно небольшое количество очков. [7]

1.2 Постановка задачи

Во время выполнения курсовой работы необходимо разработать распределенное клиент-серверное приложение для игры «Тетрис».

В разрабатываемой системе присутствует единственная роль – игрок. Основная функция игрока – это заполнение игрового поля фигурами.

Игрок может пройти регистрацию, для этого ему нужно ввести логин (от 4 до 20 символов), пароль (от 4 до 20 символов) и повтор пароля.

Перед началом игры игрок проходит авторизацию, введя свой логин и пароль, задает размеры игрового поля (ширина от 5 до 20 пикселей. а высота от 10 до 30 пикселей). Затем игрок нажимает кнопку «Начать игру», при нажатии на которую будут произведены проверки правильности заполнения полей и переход к окну игрового процесса в случае успешного прохождения проверок. При некорректном вводе пользователем система выдает соответствующее уведомление и предлагает повторить ввод заново.

В процессе игры серверное приложение предлагает игроку случайную фигуру из базового набора, состоящего из 5 фигур тетрамино. Игрок должен их расположить на поле снизу вверх: с помощью клавиш клавиатуры он может поворачивать фигурку на 90° по часовой стрелке (стрелка вверх ↑), двигать её по горизонтали налево и направо (стрелки влево ← и вправо →), а также сбрасывать ее (стрелка вниз ↓).

Игра заканчивается, когда новая фигурка не сможет поместиться в стакан. Баллы начисляются в процессе игры с учетом количества заполненных целиком линий. После завершения игры на экран выводится статистика.

Таким образом, приложение должно решать следующие задачи:

1) функции клиента:
   - регистрация и авторизация в системе;
   - просмотр справочной информации о разработчиках и о правилах игры;

– проверка введенных игроком размеров игрового поля;
– генерация заданного игрового поля;
– визуализация всех процессов игры;
– управление игровыми фигурами с помощью клавиш;
– подсчет заполненных линий игрового поля;
– вывод статистики после завершения игры;
– запрос серверу на генерацию случайной игровой фигуры;

2) функции сервера:
– идентификация пользователя;
– выдача справочной информации о разработчиках и о правилах игры;
– генерация случайной базовой фигуры из списка и передача ее клиенту.

Пользовательский интерфейс разрабатываемого приложения должен четко и полно отображать все игровые возможности для пользователя, делая взаимодействие интуитивным и удобным.

1.3 Описание комплекса программных средств

Для реализации системы был выбран язык программирование Java для серверной и клиентской части приложения, среда разработки IntelliJ IDEA и технология Remote Method Invocation (RMI) для организации удаленного взаимодействия различных частей разрабатываемого приложения. Также программное обеспечение будет функционировать под управлением операционной системы Windows 10 и выше.

1.3.1 Описание языка программирования

Java — это объектно-ориентированный, кроссплатформенный язык программирования, разработанный компанией Sun Microsystems (ныне Oracle). Он обладает высокой производительностью, автоматическим управлением памятью (сборка мусора) и строгой типизацией. Язык широко

используется в разработке веб-приложений, корпоративных систем, мобильных приложений (Android), а также серверных решений. Его популярность обусловлена стабильностью, масштабируемостью, обширной экосистемой библиотек и большим сообществом разработчиков. [8]

1.3.2 Описание среды разработки

IntelliJ IDEA — это мощная интегрированная среда разработки (IDE) для Java и других языков, разработанная компанией JetBrains. Она обеспечивает интеллектуальное автодополнение кода, удобную навигацию, встроенные инструменты отладки, рефакторинга и интеграцию с системами контроля версий. Благодаря поддержке плагинов и встроенных фреймворков IntelliJ IDEA подходит для разработки веб-приложений, мобильных решений и корпоративных систем. Её преимущества — высокая производительность, удобный интерфейс и продвинутые инструменты анализа кода, которые помогают писать чистый и эффективный код. [9]

1.3.3 Описание программного интерфейса

Remote Method Invocation (RMI) – это программная модель, в соответствии с которой Java-приложение может вызывать методы объекта другого Java-приложения, функционирующего на другой виртуальной машине, на другом узле компьютерной сети. Модель содержит набор объектов (классов) для организации удаленного (remote) взаимодействия. RMI-приложение, как правило, содержит две основные части – сервер и клиент. Серверная часть приложения создает удаленные объекты, публикует ссылки на них. Клиентская часть получает ссылку на удаленные объекты и вызывает их методы, обращаясь к ним как к «родным», находящимся в том же адресном пространстве на той же виртуальной машине Java. [10]

1.3.4 Описание операционной системы

Windows 10 и более новые версии — это операционные системы от Microsoft, ориентированные на производительность, безопасность и удобство работы. Windows 10 представила обновляемую модель с регулярными обновлениями, улучшенной многозадачностью, поддержкой гибридных рабочих процессов и оптимизированную производительность для современных процессоров. Windows 10 и более новые версии поддерживают широкий спектр устройств, имеют расширенные функции безопасности (Windows Defender, BitLocker) и обеспечивают высокую совместимость с программами и играми. [11]

2 Проектирование системы

2.1 Структурная схема системы

На рисунке 3 приведена структурная схема разрабатываемой системы.

В состав клиентской части входят следующие подсистемы:

– подсистема настройки игрового поля, которая отвечает за обработку введенных пользователем значений игрового поля и его генерацию при условии их корректности;

– подсистема игрового процесса, которая отвечает за управление игровыми фигурами игроком в процессе игры с помощью клавиш, за наполнение стакана;

– справочная подсистема, отвечающая за вывод информации о системе, правил игры и разработчиках;

– подсистема визуализации игры, которая отвечает за отображение игрового процесса;

– подсистема регистрации и авторизации, которая отвечает за регистрацию новых игроков и авторизацию старых;

– подсистема подсчета статистики, отвечающая за подсчет заполненных линий, а также за отображение статистики после окончания игры;

– подсистема взаимодействия с сервером, обеспечивающая взаимодействие клиентской части приложения с серверной.

В состав серверной части входят следующие подсистемы:

– подсистема взаимодействия с клиентом, обеспечивающая взаимодействие серверной части приложения с клиентской;

– подсистема аутентификации, обеспечивающая доступ к системе игрокам;

– подсистема работы генерации игровых фигур, которая отвечает за случайную генерацию игровой фигуры из списка базовых заданных в системе игровых фигур;

– база данных, которая хранит информацию о игроках и их статистике.

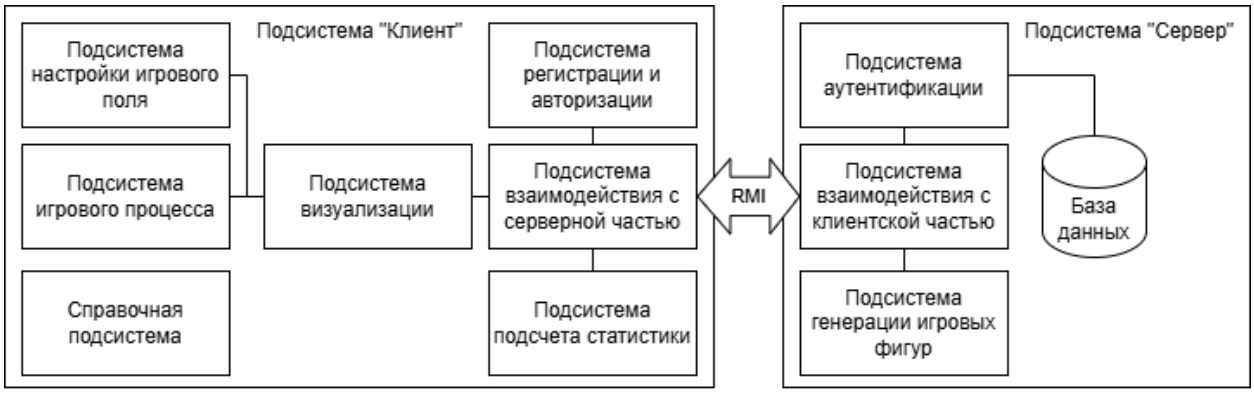

Рисунок 3 – Структурная схема системы

2.2 Разработка информационно-логического проекта системы

2.2.1 Диаграмма вариантов использования

На рисунке 4 приведена диаграмма вариантов использования.

Пользователь может просмотреть информацию о разработчике или сведения о системе (правила игры), а также пройти авторизацию, введя логин и пароль.

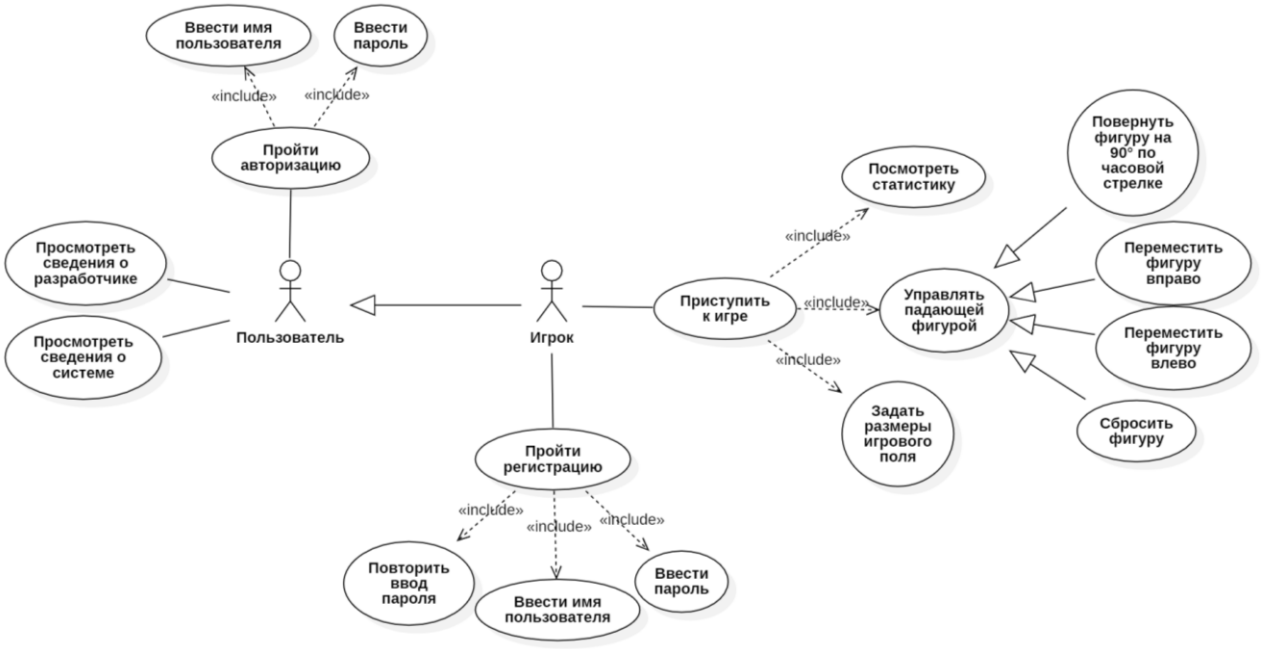

Рисунок 4 – Диаграмма вариантов использования

Игрок может пройти регистрацию, введя логин, пароль и повтор пароля, а также приступить к игре. В этом случае он может задать размеры

игрового поля, а также управлять фигурами, что включает в себя поворот фигуры на 90° по часовой стрелке, перемещение фигуры влево или вправо, а также сброс фигуры вниз. Игрок также может просмотреть статистику после завершения игры.

2.2.2 Диаграмма деятельности

На рисунке 5 приведена диаграмма деятельности системы.

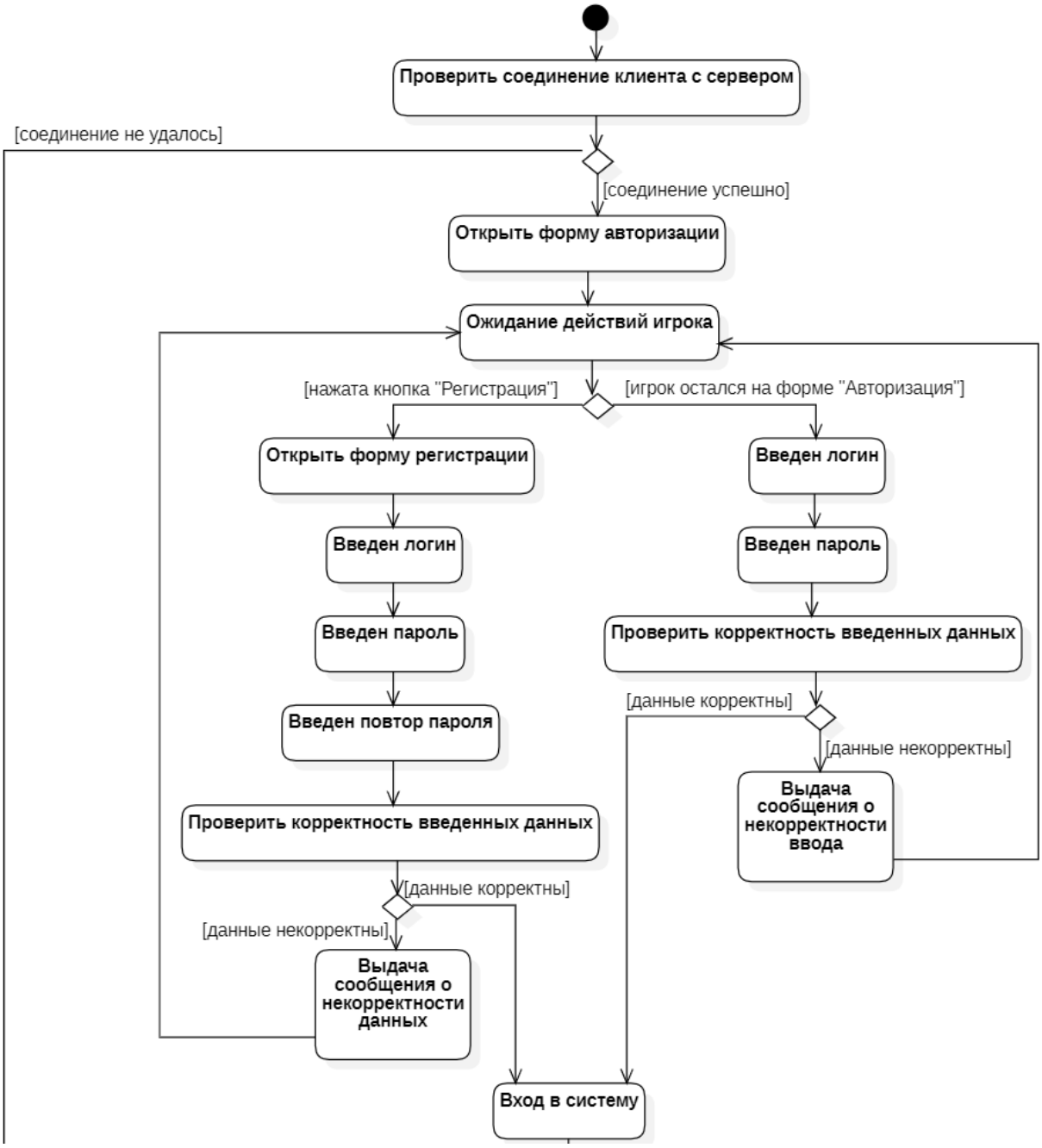

Рисунок 5 – Диаграмма деятельности системы (начало)

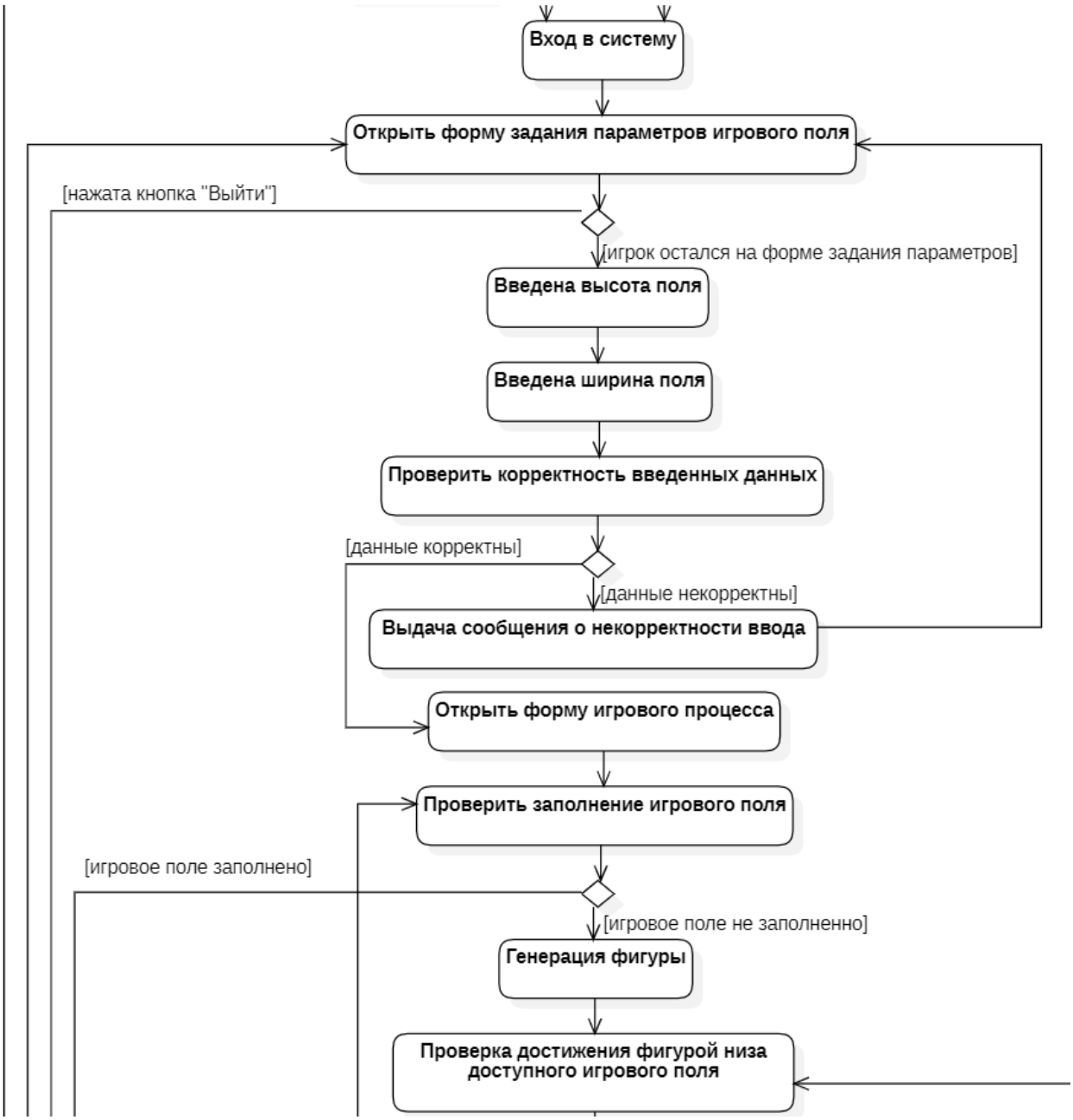

Рисунок 5 – Диаграмма деятельности системы (продолжение)

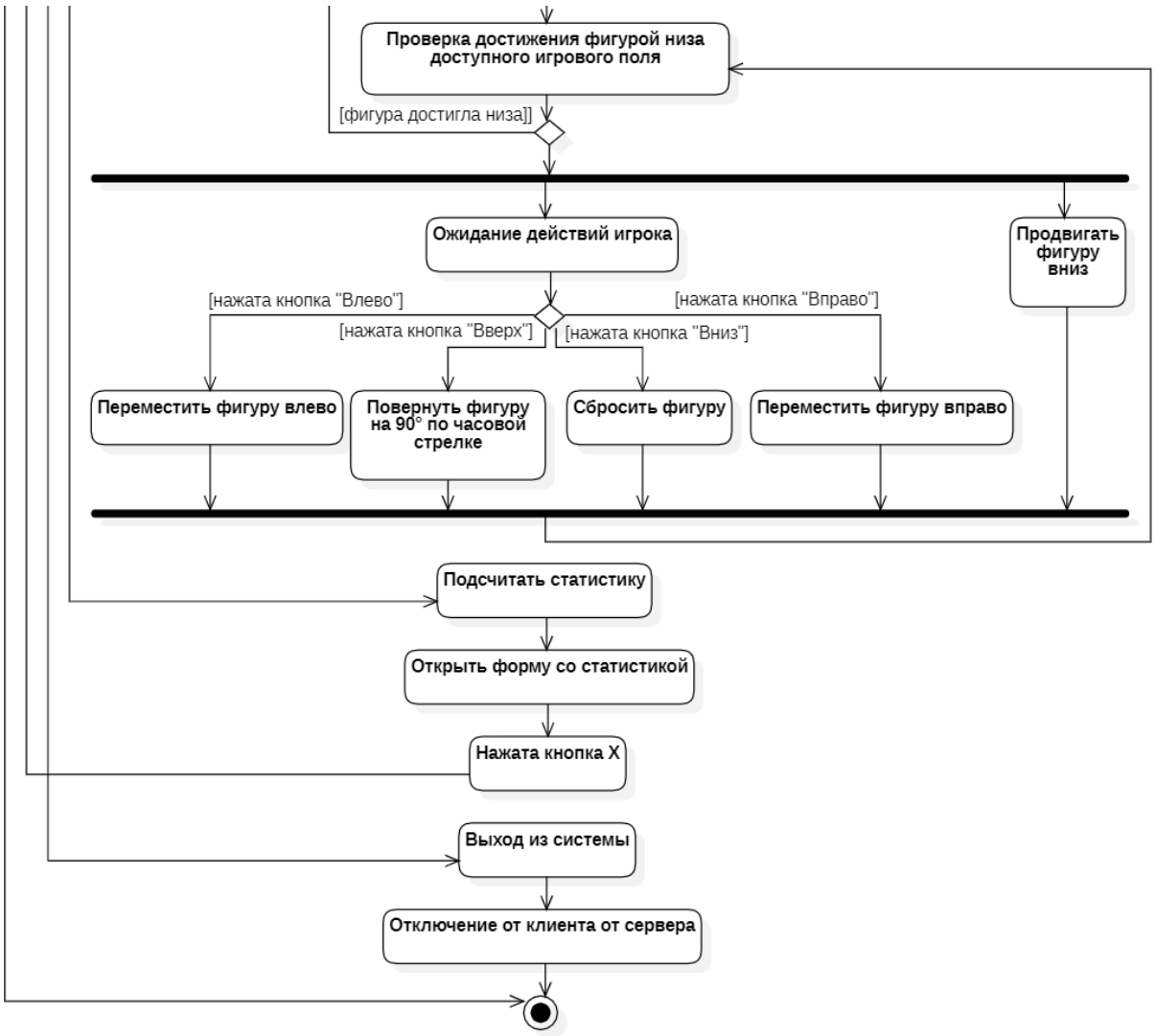

Рисунок 5 – Диаграмма деятельности системы (окончание)

Клиентская часть приложения соединяется серверной, и в случае успеха клиент открывает форму авторизации пользователя. В случае неудачи, система завершает свои процессы.

После открытия формы авторизации система ждет действия игрока. Он может ввести свой логин и пароль для авторизации, в таком случае сервер проверит корректность данных и выведет ошибку в случае некорректного ввода или пропустит в систему в случае успеха. Также игрок может зарегистрироваться, для этого ему нужно нажать кнопку «Зарегистрироваться», после чего откроется форма регистрации, куда ему необходимо ввести логин, пароль и повтор пароля. После этого сервер проверит корректность данных и выведет ошибку в случае некорректного ввода или пропустит в систему в случае успеха.

Далее открывается форма задания параметров игрового поля, тут игроку необходимо ввести высоту и ширину поля, после чего сервер опять проверит корректность введенных данных, в случае неудачи система выведет сообщение о некорректности ввода данных, а в случае успеха начнется откроется форма игрового процесса. Также игрок может нажать кнопку «Выйти», в таком случае система завершит свою работу.

При открытии формы игрового процесса начнется сама игра. Система проверят заполненность игрового поля, и в случае нет отправляет запрос на генерацию фигуры тетрамино на сервер. Пока эта фигура не достигла доступного низа игрового поля, клиент будет продвигать фигуру вниз, ожидания также действий игрока, который может нажать кнопку «Влево», тогда клиент переместит фигуру влево, кнопку «Вправо», тогда клиент переместит фигуру вправо, кнопку «Вниз», тогда клиент сбросит фигуру до низа доступного игрового поля, или кнопку «Вверх», тогда клиент повернет фигуру на 90° по часовой стрелке. Как только фигура достигла низа, система снова проверяет заполненность игрового поля, тем самым игровой процесс замыкается, пока игровое поле не заполнено.

Как только это происходит, клиент подсчитывает статистику за эту игру, и открывает с ней форму, которую можно закрыть, нажав кнопку «Х». В таком случае клиент снова открывает форму задания параметров игрового поля.

Как только происходит выход из системы, то клиент отключается от сервера и завершает свою работу.

2.2.3 Диаграмма последовательности

На рисунке 6 приведена диаграмма последовательности системы.

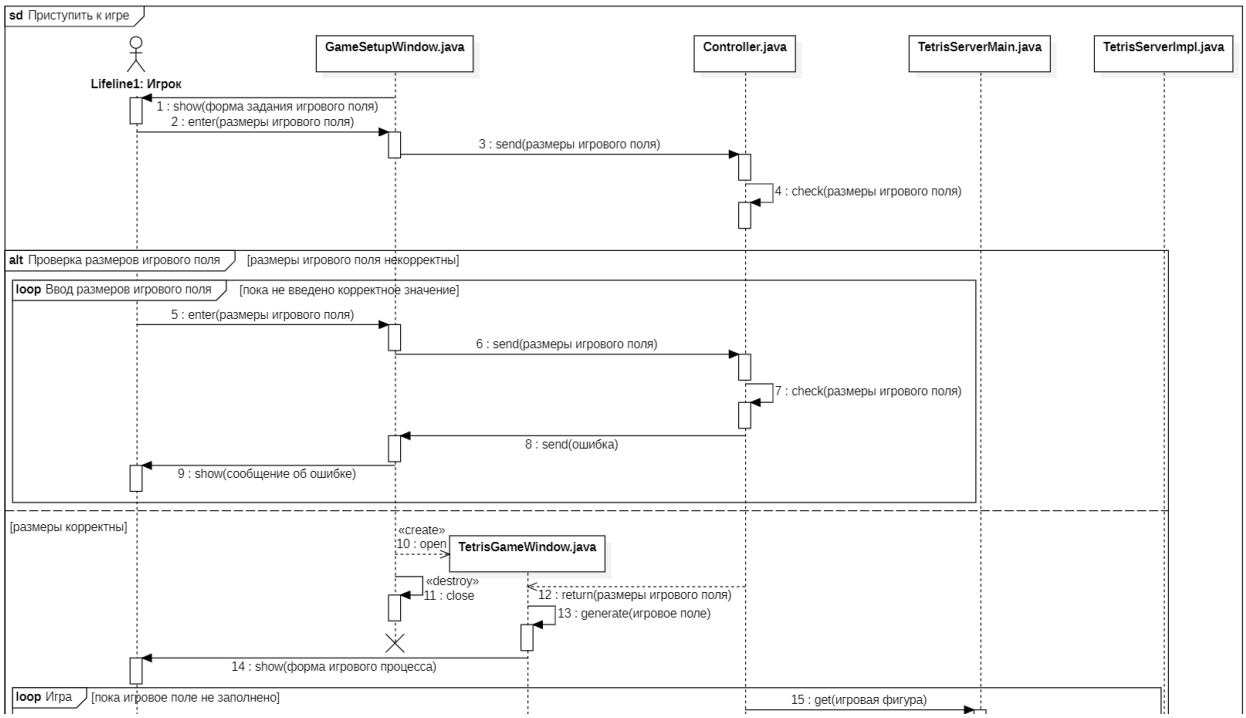

Рисунок 6 – Диаграмма последовательности (начало)

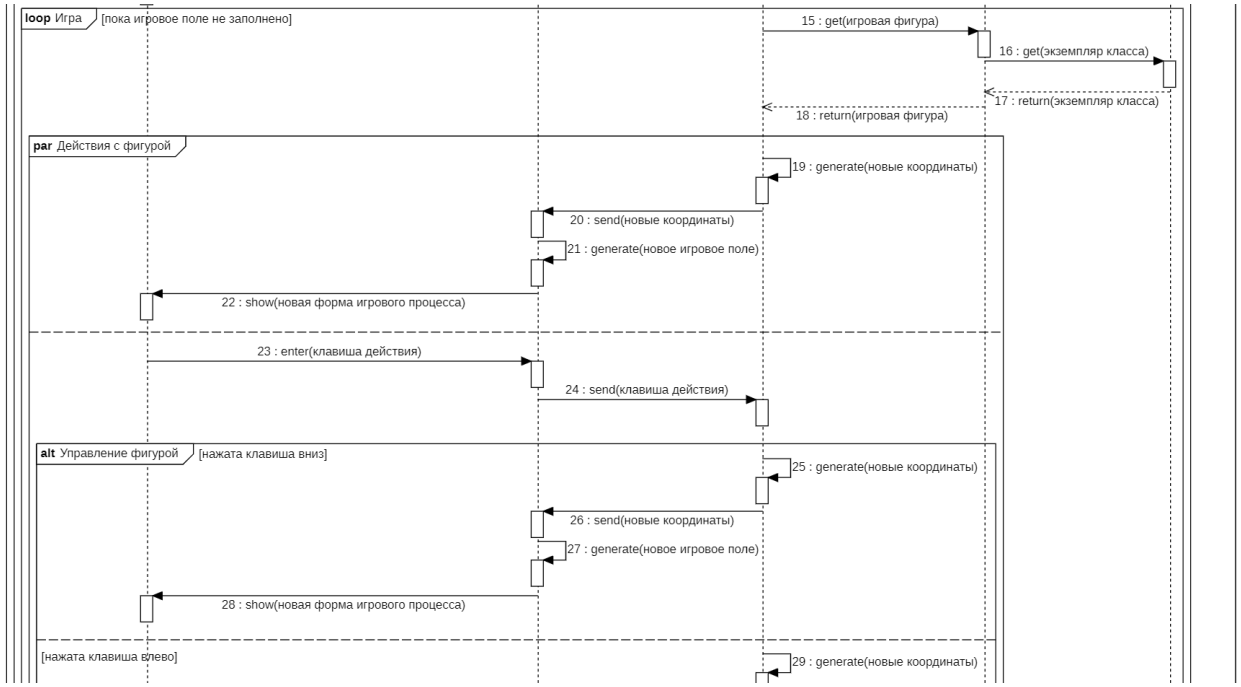

Рисунок 6 – Диаграмма последовательности (продолжение)

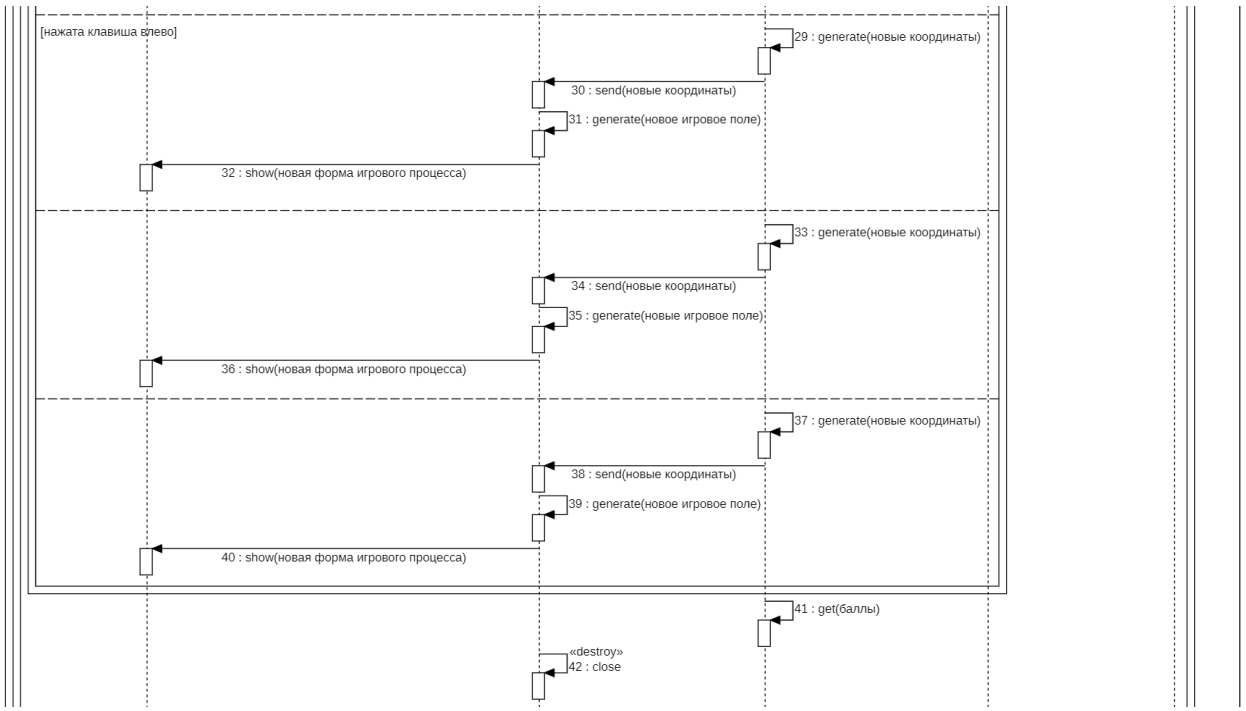

Рисунок 6 – Диаграмма последовательности (продолжение)

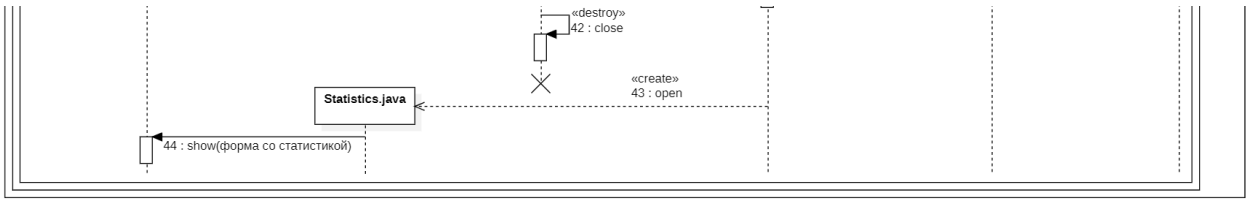

Рисунок 6 – Диаграмма последовательности (окончание)

3 Реализация системы

3.1 Диаграмма компонентов

На рисунке 8 представлена диаграмма компонентов. В нее входят основные компоненты, представленные в таблице 1.

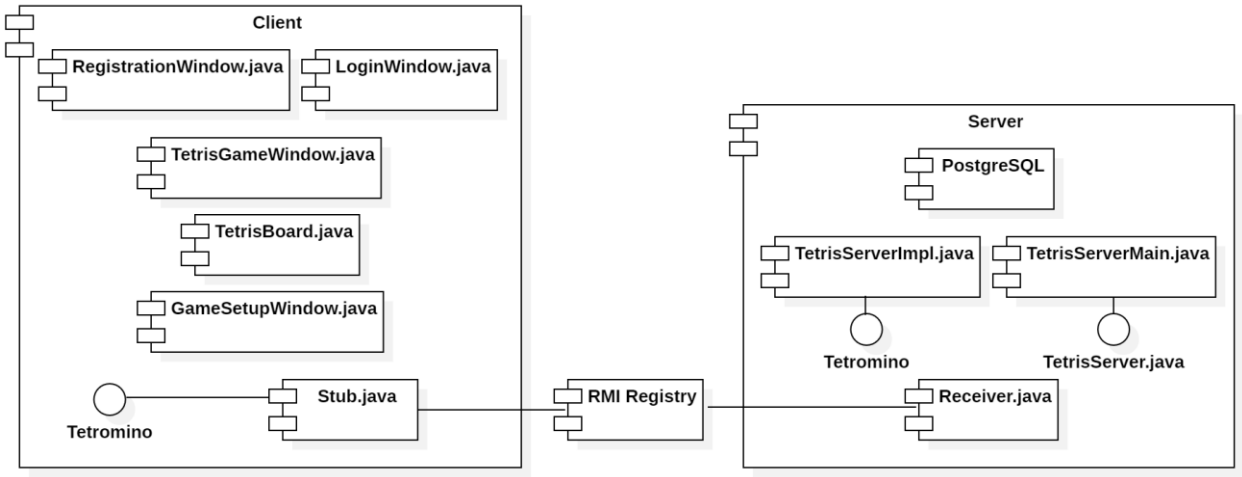

Рисунок 8 – Диаграмма компонентов

Таблица 1 – Описание компонентов системы

Компонент	Назначение компонента
RegistrationWindow.java	Обеспечивает регистрацию новых пользователей в систему.
LoginWindow.java	Обеспечивает авторизацию уже зарегистрированных пользователей.
TetrisGameWindow.java	Отвечает за визуализацию игры.
TetrisBoard.java	Отвечает сохранение игры в рамках игрового поля (позицию фигур тетрамино, размеров игрового поля и статистику).
GameSetupWindow.java	Отвечает за настойку игрового поля игры.
Stub.java	Отвечает за связь клиентской части приложения с серверной с помощью интерфейса RMI Registry.
RMI Registry	Принимает запросы с клиента или сервера и обслуживает их.
Receiver.java	Отвечает за связь серверной части приложения с клиентской с помощью интерфейса RMI Registry.
PostgreSQL	Обеспечивает сохранение данных о пользователях (таких как логинов, паролей и статистики).

Продолжение таблицы 1.

TetrisServerImpl.java	Отвечает за взаимодействие с базой данных, а также за генерацию фигур тетрамино
TetrisServerMain.java	Отвечает за инициализацию серверных классов, а также доступ к ним.

3.2 Физическая модель базы данных

База данных для разработанной системы представляет из себя одну сущность. В таблице 2 приведено описание этой сущности.

Таблица 2 – Описание единственно сущности базы данных.

Имя поля	Имя атрибута	Тип	Размер (байт)
ID	Идентификатор	int	4
Login	Имя пользователя	varchar(20)	30
Password_Tetris	Пароль	varchar(20)	30
Best_Score	Рекорд	int	4

3.3 Разработка и описание интерфейса пользователя

На рисунке 9 представлено окно авторизации в систему. Здесь пользователю необходимо ввести логин и пароль от своей учетной записи в системе. Далее ему необходимо нажать кнопку «Войти», и в случае корректного ввода пользователь перейдет на окно задания настроек игрового поля.

На рисунке 10 показано окно, всплывающее при некорректном вводе пользователем логина или пароля.

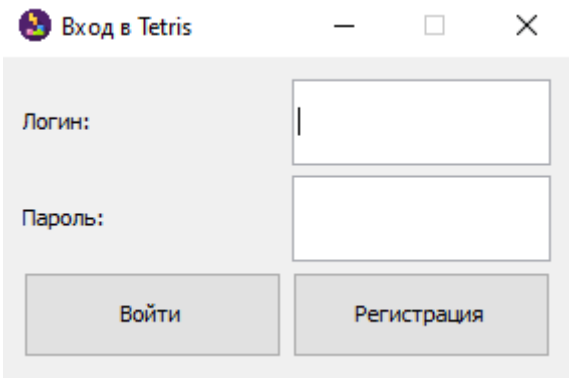

Рисунок 9 – Окно авторизации

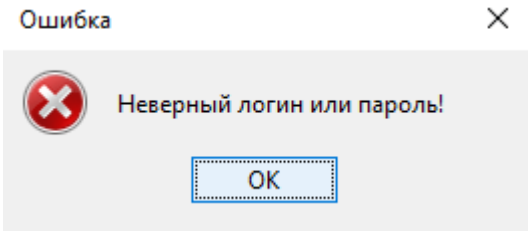

Рисунок 10 – Ввод неверного логина или пароля

На рисунке 11 показано сообщение об успешной авторизации игрока в системе.

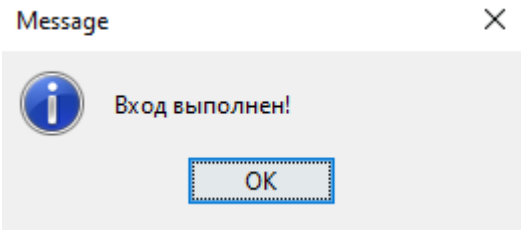

Рисунок 11 – Успешная авторизация игрока в системе

На окне авторизации пользователь также может нажать на кнопку «Зарегистрироваться», тогда система откроет окно регистрации нового игрока в систему.

На рисунке 12 показана форма регистрации нового пользователя в систему. Здесь необходимо ввести логин, пароль, повтор пароля и нажать кнопку «Зарегистрироваться». При нажатии на кнопку «Назад» или при успешной регистрации в системе, отроется окно авторизации.

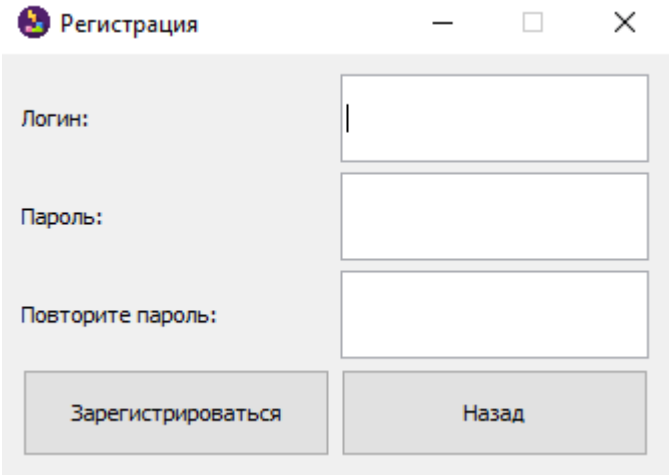

Рисунок 12 – Окно регистрации

На рисунке 13 продемонстрировано окно с ошибкой при регистрации, когда пользователь пытается зарегистрировать учетную запись на уже существующий логин.

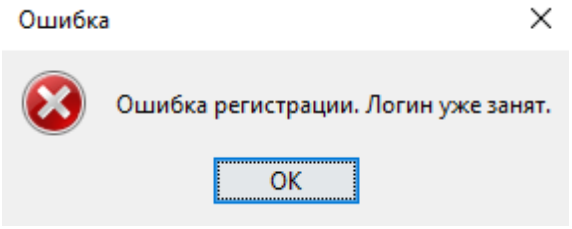

Рисунок 13 – Попытка использовать уже занятый логин

На рисунке 14 продемонстрировано окно с ошибкой при регистрации, когда пользователь пытается ввести меньше, чем 4 символа или больше, чем 20 для логина или пароля.

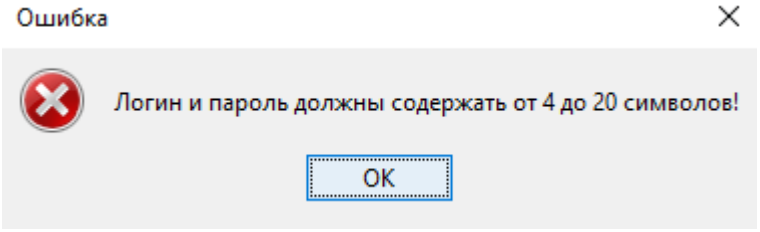

Рисунок 14 – Попытка ввести недопустимую длину логина или пароля

На рисунке 15 показано сообщение об успехе регистрации нового пользователя в системе.

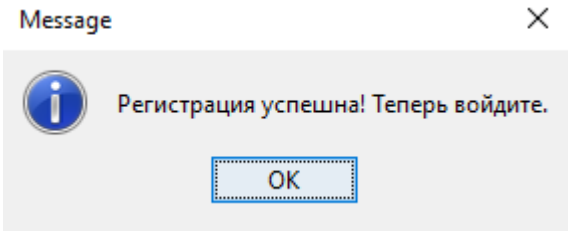

Рисунок 15 – Регистрация нового пользователя

На рисунке 16 представлено окно настройки игрового поля. Здесь игроку необходимо выбрать ширину и высоту игрового поля среди доступных вариантов. Как только он это сделал, далее ему необходимо нажать на кнопку «Начать игру», тогда система откроет окно с игровым

процессом. В добавок ко всему этому, игрок может нажать на кнопку «Правила», тогда откроется окно с правилами игры.

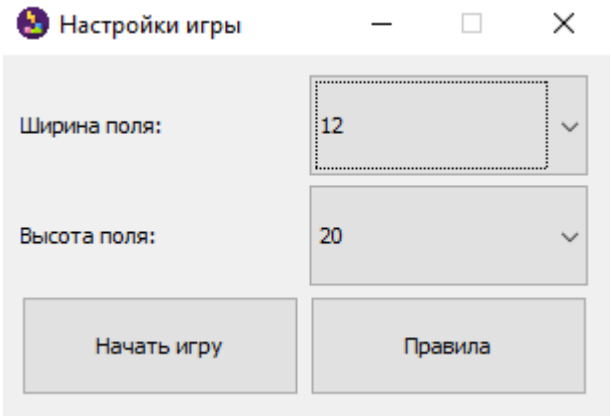

Рисунок 16 – Окно задания параметров игрового поля

На рисунке 17 представлено окно с правилами игры. Здесь также расположена информация о разработчике.

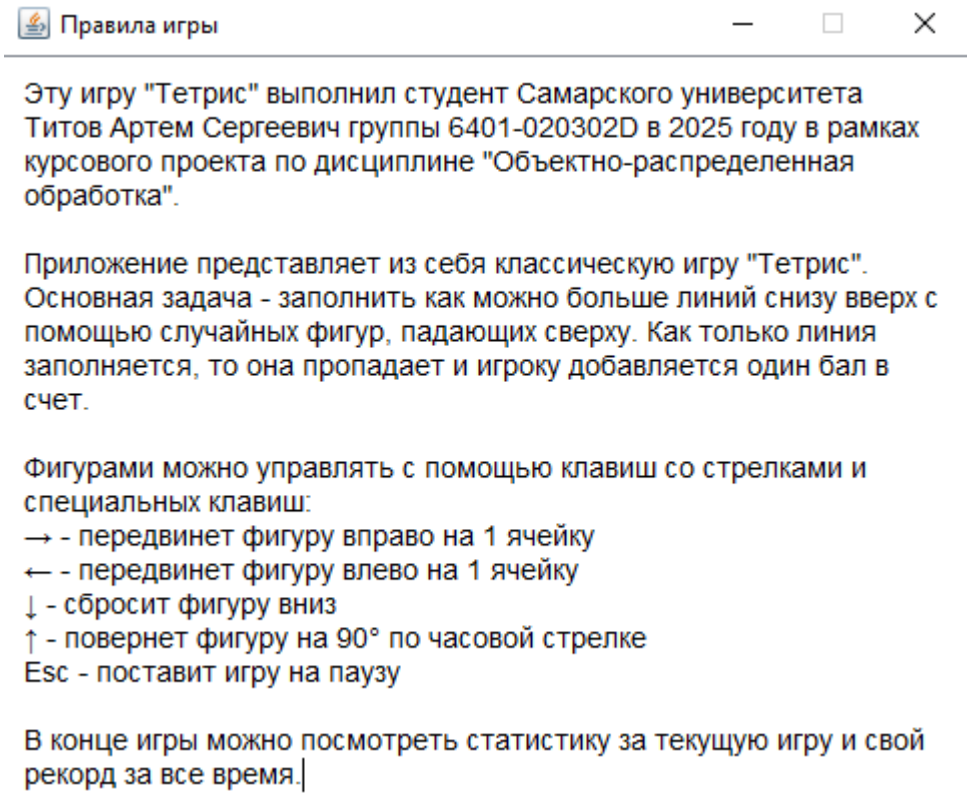

Рисунок 17 – Окно с правила и информацией о разработчике

На рисунке 18 продемонстрированно окно с игровым процессом. Здесь игрок может управлять фигурами с помощью клавиш со стрелками на клавиатуре, чтобы набирать баллы.

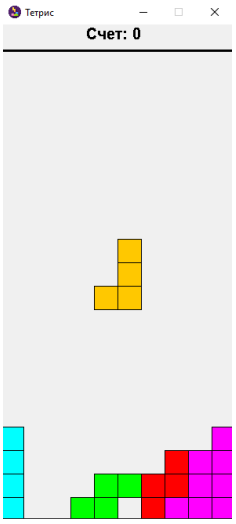

Рисунок 18 – Окно игрового процесса

На рисунке 19 показано окно с результатами игры, которое отображается, когда игрок завершает свою игру. Тут он может нажать на кнопку «ОК», тогда система отобразить окно с настройкой игрового поля или на кнопку «Таблица лидеров», которая отроет окно с таблицей лидеров.

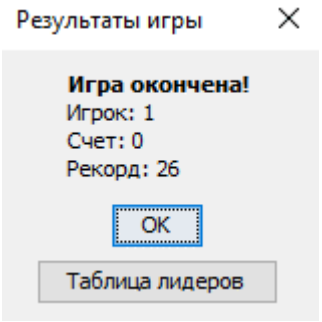

Рисунок 19 – Окно завершения игрового процесса

На рисунке 20 показано окно с таблицей лидеров.

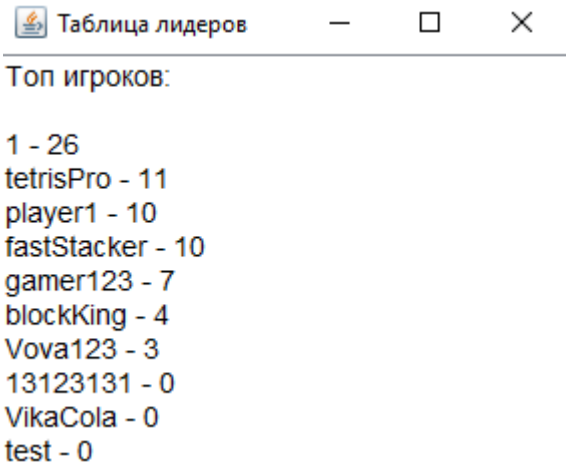

Рисунок 20 – Окно с таблицей лидеров

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы было разработано клиент-серверное приложение «Игра «Тетрис» с использованием технологии RMI.

В первом разделе проведен анализ предметной области. На основе этого анализа была сформулирована постановка задачи.

Во втором разделе была разработана структурная схема системы и с использованием языка UML созданы канонические диаграммы UML.

В третьем разделе был описан комплекс программных средств, который будет использоваться в процессе разработки.

В четвертом разделе описаны пользовательский интерфейс и диаграмма компонентов системы.

Разработанное приложение выполняет все поставленные задачи и выполняет весь необходимый функционал.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Тетрис [Электронный ресурс] // Википедия: [сайт]. URL: https://ru.wikipedia.org/wiki/Тетрис (дата обращения: 27.02.2024).

2 Описание и пример RMI технологии [Электронный ресурс] // Java-online.ru: [сайт]. URL: https://java-online.ru/java-rmi.xhtml (дата обращения: 27.02.2024).

3 Методология объектно-ориентированного анализа и проектирования [Электронный ресурс] // НАО Интуит: [сайт]. URL: https://intuit.ru/studies/courses/32/32/lecture/1000?page=2 (дата обращения: 28.02.2024).

4 Анализ предметной области. [Электронный ресурс] // НАО Интуит: [сайт]. https://intuit.ru/studies/courses/574/430/lecture/9749 (дата обращения: 28.02.2024).

5 Тетрис [Электронный ресурс] // Механико-математический факультет МГУ имени М.В. Ломоносова: [сайт]. URL: http://mech.math.msu.su/~shvetz/54/inf/perl-problems/chTetris.xhtml (дата обращения: 11.03.2024).

6 История «Тетриса» и его изобретателя [Электронный ресурс] // vc.ru : [сайт]. URL: https://vc.ru/story/78735-pridumat-populyarnuyu-igru-no-ne-zarabotat-na-ney-istoriya-tetrisa-i-ego-izobretatelya (дата обращения: 11.03.2024).

7 История создания Тетриса [Электронный ресурс] // Сервер Информационных Технологий Cit Forum: [сайт]. URL: https://cloudteh.ru/2019/08/su-tetris/ (дата обращения: 11.03.2024).

8 Java [Электронный ресурс] // SkillFactory Media: [сайт]. URL: https://blog.skillfactory.ru/glossary/java/

9 IntelliJ IDEA [Электронный ресурс] // SkillFactory Media: [сайт]. URL: https://blog.skillfactory.ru/glossary/intellij-idea/

10 RMI (Remote Method Invocation) [Электронный ресурс] // Хабр: [сайт]. URL: https://habr.com/ru/articles/74639/

11 Windows 10 в простых вопросах и понятных ответах [Электронный ресурс] // Юга: [сайт]. URL: https://www.yuga.ru/articles/society/7230.html

ПРИЛОЖЕНИЕ А

Листинг модулей программы

TetrisServer.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

public interface TetrisServer extends Remote {
    boolean register(String login, String password) throws RemoteException;
    boolean login(String login, String password) throws RemoteException;
    int getBestScore(String login) throws RemoteException;
    void updateBestScore(String login, int score) throws RemoteException;
    List<String> getLeaderboard() throws RemoteException;
}
```

TetrisServerImpl.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class TetrisServerImpl extends UnicastRemoteObject implements TetrisServer {
    private static final String URL = "jdbc:postgresql://localhost:5432/tetris_users_db";
    private static final String USER = "postgres";
    private static final String PASSWORD = "33585900";

    protected TetrisServerImpl() throws RemoteException {
        super();
    }

    private Connection connect() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }

    @Override
    public boolean register(String login, String password) throws RemoteException {
        String sql = "INSERT INTO tetris_players (login, password_tetris, best_score) VALUES (?, ?, 0)";
        try (Connection conn = connect(); PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, login);
            stmt.setString(2, password);
            stmt.executeUpdate();
            return true;
        } catch (SQLException e) {
            System.err.println("Ошибка при регистрации: " + e.getMessage());
            return false;
        }
    }

    @Override
    public boolean login(String login, String password) throws RemoteException {
        String sql = "SELECT * FROM tetris_players WHERE login = ? AND password_tetris = ?";
        try (Connection conn = connect(); PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, login);
            stmt.setString(2, password);
            ResultSet rs = stmt.executeQuery();
            return rs.next();
        } catch (SQLException e) {
            System.err.println("Ошибка при входе: " + e.getMessage());
            return false;
        }
    }

    @Override
    public int getBestScore(String login) throws RemoteException {
```

```
        String sql = "SELECT best_score FROM tetris_players WHERE login = ?";
        try (Connection conn = connect(); PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, login);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                return rs.getInt("best_score");
            }
        } catch (SQLException e) {
            System.err.println("Ошибка при получении лучшего счета: " + e.getMessage());
        }
        return 0;
    }

    @Override
    public void updateBestScore(String login, int score) throws RemoteException {
        String sql = "UPDATE tetris_players SET best_score = ? WHERE login = ? AND best_score < ?";
        try (Connection conn = connect(); PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setInt(1, score);
            stmt.setString(2, login);
            stmt.setInt(3, score);
            stmt.executeUpdate();
        } catch (SQLException e) {
            System.err.println("Ошибка при обновлении счета: " + e.getMessage());
        }
    }

    @Override
    public List<String> getLeaderboard() throws RemoteException {
        List<String> leaderboard = new ArrayList<>();
        String sql = "SELECT login, best_score FROM tetris_players ORDER BY best_score DESC";
        try (Connection conn = connect(); PreparedStatement stmt = conn.prepareStatement(sql); ResultSet rs =
stmt.executeQuery()) {
            while (rs.next()) {
                leaderboard.add(rs.getString("login") + " - " + rs.getInt("best_score"));
            }
        } catch (SQLException e) {
            System.err.println("Ошибка при получении рейтинга: " + e.getMessage());
        }
        return leaderboard;
    }
}
```

TetrisServerMain.java

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class TetrisServerMain {
    public static void main(String[] args) {
        try {
            TetrisServer server = new TetrisServerImpl();
            Registry registry = LocateRegistry.createRegistry(1099);
            registry.rebind("TetrisServer", server);
            System.out.println("Сервер Tetris запущен!");
        } catch (Exception e) {
            System.err.println("Ошибка запуска сервера: " + e.getMessage());
        }
    }
}
```

LoginWindow.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class LoginWindow extends JFrame {
```

```
    private JTextField loginField;
    private JPasswordField passwordField;
    private JButton loginButton, registerButton;
    private TetrisServer server;

    public LoginWindow() {
        try {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
            SwingUtilities.updateComponentTreeUI(this);
        } catch (Exception e) {
            System.err.println("Не удалось установить Windows Look and Feel: " + e.getMessage());
        }

        try {
            ImageIcon icon = new ImageIcon(getClass().getClassLoader().getResource("icon.png"));
            setIconImage(icon.getImage());
        } catch (Exception e) {
            System.err.println("Не удалось загрузить значок: " + e.getMessage());
        }

        setTitle("Вход в Tetris");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);

        JPanel panel = new JPanel(new GridLayout(3, 2, 5, 5));
        panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10)); // Отступы

        panel.add(new JLabel("Логин:"));
        loginField = new JTextField();
        panel.add(loginField);

        panel.add(new JLabel("Пароль:"));
        passwordField = new JPasswordField();
        panel.add(passwordField);

        loginButton = new JButton("Войти");
        registerButton = new JButton("Регистрация");

        loginButton.addActionListener(new LoginAction());
        registerButton.addActionListener(new RegisterAction());

        panel.add(loginButton);
        panel.add(registerButton);

        add(panel);

        // Подключаемся к серверу
        try {
            Registry registry = LocateRegistry.getRegistry("localhost", 1099);
            server = (TetrisServer) registry.lookup("TetrisServer");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Ошибка подключения к серверу!", "Ошибка",
JOptionPane.ERROR_MESSAGE);
            System.exit(1);
        }
    }

    private class LoginAction implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String login = loginField.getText();
            String password = new String(passwordField.getPassword());

            try {
                if (server.login(login, password)) {
                    JOptionPane.showMessageDialog(LoginWindow.this, "Вход выполнен!");
                    dispose();
                    new GameSetupWindow(server, login).setVisible(true);
                } else {
```

```
                JOptionPane.showMessageDialog(LoginWindow.this, "Неверный логин или пароль!", "Ошибка",
JOptionPane.ERROR_MESSAGE);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(LoginWindow.this, "Ошибка соединения!", "Ошибка",
JOptionPane.ERROR_MESSAGE);
        }
    }
  }

  private class RegisterAction implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      dispose();
      new RegistrationWindow(server).setVisible(true);
    }
  }

  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new LoginWindow().setVisible(true));
  }
}
```

RegistrationWindow.java

```
import javax.swing.*;
import javax.swing.text.PlainDocument;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class RegistrationWindow extends JFrame {
  private JTextField loginField;
  private JPasswordField passwordField, confirmPasswordField;
  private JButton registerButton, backButton;
  private TetrisServer server;

  public RegistrationWindow(TetrisServer server) {
    this.server = server;

    try {
      UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
      SwingUtilities.updateComponentTreeUI(this);
    } catch (Exception e) {
      System.err.println("Не удалось установить Windows Look and Feel: " + e.getMessage());
    }

    try {
      ImageIcon icon = new ImageIcon(getClass().getClassLoader().getResource("icon.png"));
      setIconImage(icon.getImage());
    } catch (Exception e) {
      System.err.println("Не удалось загрузить значок: " + e.getMessage());
    }

    setTitle("Регистрация");
    setSize(350, 250);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setLocationRelativeTo(null);
    setResizable(false);

    JPanel panel = new JPanel(new GridLayout(4, 2, 5, 5));
    panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10)); // Отступы

    panel.add(new JLabel("Логин:"));
    loginField = new JTextField();
    loginField.setDocument(new JTextFieldLimit(20)); // Ограничение длины
    panel.add(loginField);

    panel.add(new JLabel("Пароль:"));
    passwordField = new JPasswordField();
```

```
        passwordField.setDocument(new JTextFieldLimit(20));
        panel.add(passwordField);

        panel.add(new JLabel("Повторите пароль:"));
        confirmPasswordField = new JPasswordField();
        confirmPasswordField.setDocument(new JTextFieldLimit(20));
        panel.add(confirmPasswordField);

        registerButton = new JButton("Зарегистрироваться");
        backButton = new JButton("Назад");

        registerButton.addActionListener(new RegisterAction());
        backButton.addActionListener(e -> {
            dispose();
            new LoginWindow().setVisible(true);
        });

        panel.add(registerButton);
        panel.add(backButton);

        add(panel);
    }

    private class RegisterAction implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String login = loginField.getText().trim();
            String password = new String(passwordField.getPassword());
            String confirmPassword = new String(confirmPasswordField.getPassword());

            if (login.length() < 4 || password.length() < 4) {
                JOptionPane.showMessageDialog(RegistrationWindow.this, "Логин и пароль должны содержать от 4 до 20
символов!", "Ошибка", JOptionPane.ERROR_MESSAGE);
                return;
            }

            if (!password.equals(confirmPassword)) {
                JOptionPane.showMessageDialog(RegistrationWindow.this, "Пароли не совпадают!", "Ошибка",
JOptionPane.ERROR_MESSAGE);
                return;
            }

            try {
                if (server.register(login, password)) {
                    JOptionPane.showMessageDialog(RegistrationWindow.this, "Регистрация успешна! Теперь войдите.");
                    dispose();
                    new LoginWindow().setVisible(true);
                } else {
                    JOptionPane.showMessageDialog(RegistrationWindow.this, "Ошибка регистрации. Логин уже занят.",
"Ошибка", JOptionPane.ERROR_MESSAGE);
                }
            } catch (Exception ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(RegistrationWindow.this, "Ошибка соединения с сервером!", "Ошибка",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    // Класс для ограничения ввода в JTextField
    private static class JTextFieldLimit extends PlainDocument {
        private final int limit;

        public JTextFieldLimit(int limit) {
            this.limit = limit;
        }

        @Override
        public void insertString(int offset, String str, javax.swing.text.AttributeSet attr) throws
javax.swing.text.BadLocationException {
            if (str == null) return;
```

```
            if ((getLength() + str.length()) <= limit) {
                super.insertString(offset, str, attr);
            }
        }
    }
}
```

GameSetupWindow.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class GameSetupWindow extends JFrame {
    private JComboBox<Integer> widthBox, heightBox;
    private JButton startButton, rulesButton;
    private String playerLogin;
    private TetrisServer server;

    public GameSetupWindow(TetrisServer server, String playerLogin) {
        this.server = server;
        this.playerLogin = playerLogin;

        setLookAndFeel();
        setTitle("Настройки игры");
        setSize(320, 220);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);
        setIcon();

        JPanel panel = new JPanel(new GridLayout(3, 2, 5, 5));
        panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        panel.add(new JLabel("Ширина поля:"));
        widthBox = new JComboBox<>(new Integer[]{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20});
        widthBox.setSelectedItem(12);
        panel.add(widthBox);

        panel.add(new JLabel("Высота поля:"));
        heightBox = new JComboBox<>(new Integer[]{10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30});
        heightBox.setSelectedItem(20);
        panel.add(heightBox);

        startButton = new JButton("Начать игру");
        startButton.addActionListener(new StartGameAction());
        panel.add(startButton);

        rulesButton = new JButton("Правила");
        rulesButton.addActionListener(new ShowRulesAction());
        panel.add(rulesButton);

        add(panel);
    }

    private void setLookAndFeel() {
        try {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
            SwingUtilities.updateComponentTreeUI(this);
        } catch (Exception e) {
            System.err.println("Не удалось установить Windows Look and Feel: " + e.getMessage());
        }
    }

    private void setIcon() {
        try {
            ImageIcon icon = new ImageIcon(getClass().getClassLoader().getResource("icon.png"));
            setIconImage(icon.getImage());
```

```
        } catch (Exception e) {
            System.err.println("Не удалось загрузить значок: " + e.getMessage());
        }
    }

    private class StartGameAction implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            int width = (int) widthBox.getSelectedItem();
            int height = (int) heightBox.getSelectedItem();

            dispose();
            new TetrisGameWindow(server, playerLogin, width, height).setVisible(true);
        }
    }

    private class ShowRulesAction implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            JFrame rulesFrame = new JFrame("Правила игры");
            rulesFrame.setSize(500, 450);
            rulesFrame.setResizable(false);
            rulesFrame.setLocationRelativeTo(null);
            rulesFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

            setIcon();
            setLookAndFeel();

            JTextArea rulesText = new JTextArea(
                    "Эту игру \"Тетрис\" выполнил студент Самарского университета Титов Артем Сергеевич " +
                            "группы 6401-020302D в 2025 году в рамках курсового проекта по дисциплине " +
                            "\"Объектно-распределенная обработка\".\n\n" +
                            "Приложение представляет из себя классическую игру \"Тетрис\". " +
                            "Основная задача - заполнить как можно больше линий снизу вверх с помощью случайных фигур, " +
                            "падающих сверху. Как только линия заполняется, то она пропадает и игроку добавляется один бал " +
                            "в счет.\n\n" +
                            "Фигурами можно управлять с помощью клавиш со стрелками и специальных клавиш:\n" +
                            "→ - передвинет фигуру вправо на 1 ячейку\n" +
                            "← - передвинет фигуру влево на 1 ячейку\n" +
                            "↓ - сбросит фигуру вниз\n" +
                            "↑ - повернет фигуру на 90° по часовой стрелке\n" +
                            "Esc - поставит игру на паузу\n\n" +
                            "В конце игры можно посмотреть статистику за текущую игру и свой рекорд за все время."
            );
            rulesText.setFont(new Font("Arial", Font.PLAIN, 14));
            rulesText.setWrapStyleWord(true);
            rulesText.setLineWrap(true);
            rulesText.setEditable(false);
            rulesText.setMargin(new Insets(10, 10, 10, 10));

            rulesFrame.add(new JScrollPane(rulesText));
            rulesFrame.setVisible(true);
        }
    }
}
```

TetrisBoard.java

```
import java.awt.Color;

public class TetrisBoard {
    private final int width, height;
    private final Color[][] grid;

    public TetrisBoard(int width, int height) {
        this.width = width;
        this.height = height;
        this.grid = new Color[height][width]; // Создаем пустое игровое поле
    }

    public boolean canMove(Tetromino piece, int newX, int newY) {
        int width = piece.getWidth();
```

```
    int height = piece.getHeight();
    Color[][] shape = piece.getShape();

    for (int y = 0; y < height; y++) {
        for (int x = 0; x < width; x++) {
            if (shape[y][x] != null) {
                int boardX = newX + x;
                int boardY = newY + y;

                // Проверяем выход за границы
                if (boardX < 0 || boardX >= this.width || boardY >= this.height) {
                    return false;
                }
                // Проверяем столкновение с уже установленными блоками
                if (grid[boardY][boardX] != null) {
                    return false;
                }
            }
        }
    }
    return true;
}

public boolean canPlace(Tetromino piece, int x, int y) {
    for (int i = 0; i < piece.getHeight(); i++) {
        for (int j = 0; j < piece.getWidth(); j++) {
            if (piece.getShape()[i][j] != null) {
                int newX = x + j;
                int newY = y + i;
                if (newX < 0 || newX >= width || newY >= height || (newY >= 0 && grid[newY][newX] != null)) {
                    return false;
                }
            }
        }
    }
    return true;
}

public void placePiece(Tetromino piece, int x, int y) {
    for (int i = 0; i < piece.getHeight(); i++) {
        for (int j = 0; j < piece.getWidth(); j++) {
            if (piece.getShape()[i][j] != null) {
                grid[y + i][x + j] = piece.getShape()[i][j];
            }
        }
    }
}

public int clearFullRows() {
    int cleared = 0;
    for (int r = 0; r < height; r++) {
        boolean full = true;
        for (int c = 0; c < width; c++) {
            if (grid[r][c] == null) {
                full = false;
                break;
            }
        }
        if (full) {
            clearRow(r);
            cleared++;
        }
    }
    return cleared;
}

private void clearRow(int row) {
    for (int r = row; r > 0; r--) {
        System.arraycopy(grid[r - 1], 0, grid[r], 0, width);
    }
```

```
        grid[0] = new Color[width]; // Очистить верхнюю строку
    }

    public Color[][] getGrid() {
        return grid;
    }
}
```

Tetromino.java

```
import java.awt.Color;

public class Tetromino {
    private Color[][] shape;

    public Tetromino(int type) {
        shape = generateShape(type);
    }

    private Color[][] generateShape(int type) {
        switch (type) {
            case 0: // Квадрат
                return new Color[][]{
                        {Color.YELLOW, Color.YELLOW},
                        {Color.YELLOW, Color.YELLOW}
                };
            case 1: // Линия
                return new Color[][]{
                        {null, Color.CYAN, null, null},
                        {null, Color.CYAN, null, null},
                        {null, Color.CYAN, null, null},
                        {null, Color.CYAN, null, null}
                };
            case 2: // L-образная фигура
                return new Color[][]{
                        {null, Color.ORANGE},
                        {null, Color.ORANGE},
                        {Color.ORANGE, Color.ORANGE}
                };
            case 3: // Z-образная фигура
                return new Color[][]{
                        {Color.RED, Color.RED, null},
                        {null, Color.RED, Color.RED}
                };
            case 4: // Обратная L-образная фигура
                return new Color[][]{
                        {Color.BLUE, null},
                        {Color.BLUE, null},
                        {Color.BLUE, Color.BLUE}
                };
            case 5: // T-образная фигура
                return new Color[][]{
                        {Color.MAGENTA, Color.MAGENTA, Color.MAGENTA},
                        {null, Color.MAGENTA, null}
                };
            case 6: // Обратная Z-образная фигура
                return new Color[][]{
                        {null, Color.GREEN, Color.GREEN},
                        {Color.GREEN, Color.GREEN, null}
                };
            default:
                return new Color[][]{{Color.GRAY}};
        }
    }

    public Color[][] getShape() {
        return shape;
    }

    public int getWidth() {
```

```
        return shape[0].length;
    }

    public int getHeight() {
        return shape.length;
    }

    public Tetromino rotate() {
        int rows = shape.length;
        int cols = shape[0].length;
        Color[][] rotatedShape = new Color[cols][rows];

        for (int y = 0; y < rows; y++) {
            for (int x = 0; x < cols; x++) {
                rotatedShape[x][rows - 1 - y] = shape[y][x];  // Поворот на 90 градусов
            }
        }
        Tetromino rotatedPiece = new Tetromino(0);
        rotatedPiece.shape = rotatedShape;
        return rotatedPiece;
    }
}
```

TetrisGameWindow.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.List;

public class TetrisGameWindow extends JFrame {
    private final TetrisBoard board;
    private final TetrisServer server;
    private Tetromino currentPiece;
    private int pieceX, pieceY;
    private Timer timer;
    private BufferedImage buffer;
    private int score = 0;
    private final String playerLogin;
    private final int cellSize = 30;
    private final int boardWidth;
    private final int boardHeight;
    private final int topMargin = 60; // Отступ сверху для информации об игроке
    private final int borderOffset = 5; // Отступ от границы
    private boolean isPaused = false;
    private JLabel pauseLabel;

    public TetrisGameWindow(TetrisServer server, String playerLogin, int width, int height) {
        this.server = server;
        this.board = new TetrisBoard(width, height);
        this.currentPiece = new Tetromino((int) (Math.random() * 7));
        this.pieceX = width / 2 - 1;
        this.pieceY = 0;
        this.playerLogin = playerLogin;
        this.boardWidth = width * cellSize;
        this.boardHeight = height * cellSize;

        try {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
            SwingUtilities.updateComponentTreeUI(this);
        } catch (Exception e) {
            System.err.println("Не удалось установить Windows Look and Feel: " + e.getMessage());
        }
```

```
        try {
            ImageIcon icon = new ImageIcon(getClass().getClassLoader().getResource("icon.png"));
            setIconImage(icon.getImage());
        } catch (Exception e) {
            System.err.println("Не удалось загрузить значок: " + e.getMessage());
        }

        setTitle("Тетрис");
        setSize(boardWidth + 2 * borderOffset, boardHeight + topMargin + borderOffset + 5);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setResizable(false);

        buffer = new BufferedImage(getWidth(), getHeight(), BufferedImage.TYPE_INT_ARGB);

        // Метка паузы
        pauseLabel = new JLabel("Игра на паузе. Нажмите ESC для продолжения.", SwingConstants.CENTER);
        pauseLabel.setFont(new Font("Arial", Font.BOLD, 16));
        pauseLabel.setForeground(Color.RED);
        pauseLabel.setVisible(false);
        add(pauseLabel, BorderLayout.NORTH);

        addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_ESCAPE) {
                    togglePause();
                }
                if (!isPaused) {
                    switch (e.getKeyCode()) {
                        case KeyEvent.VK_LEFT:
                            moveLeft();
                            break;
                        case KeyEvent.VK_RIGHT:
                            moveRight();
                            break;
                        case KeyEvent.VK_DOWN:
                            moveDown();
                            break;
                        case KeyEvent.VK_UP:
                            rotatePiece();
                            break;
                    }
                    repaint();
                }
            }
        });

        timer = new Timer(500, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (!isPaused) {
                    moveDown();
                    repaint();
                }
            }
        });
        timer.start();

        setFocusable(true);
        setVisible(true);
    }

    private void togglePause() {
        isPaused = !isPaused;
        pauseLabel.setVisible(isPaused);

        if (isPaused) {
            timer.stop();
        } else {
```

```
            timer.start();
        }
        repaint();
    }

    private void moveLeft() {
        if (board.canPlace(currentPiece, pieceX - 1, pieceY)) {
            pieceX--;
        }
    }

    private void moveRight() {
        if (board.canPlace(currentPiece, pieceX + 1, pieceY)) {
            pieceX++;
        }
    }

    private void moveDown() {
        if (board.canPlace(currentPiece, pieceX, pieceY + 1)) {
            pieceY++;
        } else {
            board.placePiece(currentPiece, pieceX, pieceY);
            checkFullRows();

            currentPiece = new Tetromino((int) (Math.random() * 7));
            pieceX = board.getGrid()[0].length / 2 - 1;
            pieceY = 0;

            if (!board.canPlace(currentPiece, pieceX, pieceY)) {
                timer.stop();
                endGame();
            }
        }
    }

    public void rotatePiece() {
        Tetromino rotated = currentPiece.rotate();
        if (board.canMove(rotated, pieceX, pieceY)) {
            currentPiece = rotated;
            repaint();
        }
    }

    private void checkFullRows() {
        int clearedRows = board.clearFullRows();
        score += clearedRows;
    }

    private void endGame() {
        int best_score = 0;
        try {
            server.updateBestScore(playerLogin, score);
            best_score = server.getBestScore(playerLogin);
        } catch (RemoteException e) {
            System.err.println("Ошибка при обновлении счета в БД: " + e.getMessage());
        }

        // Создаем окно результата игры
        JDialog resultDialog = new JDialog(this, "Результаты игры", true);
        resultDialog.setLayout(new BorderLayout());
        resultDialog.setSize(175, 175);
        resultDialog.setResizable(false); // Окно фиксированного размера
        resultDialog.setLocationRelativeTo(this);

        // Панель с текстом и кнопками
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        // Добавляем отступ сверху перед заголовком
        panel.add(Box.createVerticalStrut(10));
```

```
    JLabel resultLabel = new JLabel("<html><b>Игра окончена!</b><br>Игрок: " + playerLogin +
            "<br>Счет: " + score + "<br>Рекорд: " + best_score + "</html>", SwingConstants.CENTER);
    resultLabel.setAlignmentX(Component.CENTER_ALIGNMENT);

    JButton leaderboardButton = new JButton("Таблица лидеров");
    JButton okButton = new JButton("OK");

    okButton.setAlignmentX(Component.CENTER_ALIGNMENT);
    leaderboardButton.setAlignmentX(Component.CENTER_ALIGNMENT);

    // Обработчик кнопки "OK"
    okButton.addActionListener(e -> resultDialog.dispose());

    // Обработчик кнопки "Таблица лидеров"
    leaderboardButton.addActionListener(e -> {
        resultDialog.dispose(); // Закрываем окно результата
        new LeaderboardWindow(server).setVisible(true); // Открываем таблицу лидеров
    });

    // Добавляем элементы в панель
    panel.add(resultLabel);
    panel.add(Box.createVerticalStrut(10));
    panel.add(okButton);
    panel.add(Box.createVerticalStrut(5));
    panel.add(leaderboardButton);

    resultDialog.add(panel, BorderLayout.CENTER);
    resultDialog.setVisible(true);

    dispose();
    new GameSetupWindow(server, playerLogin).setVisible(true);
}


@Override
public void paint(Graphics g) {
    Graphics2D g2d = (Graphics2D) buffer.getGraphics();
    g2d.setColor(getBackground());
    g2d.fillRect(0, 0, getWidth(), getHeight());
    drawGameStats(g2d);
    drawBoard(g2d);
    drawCurrentPiece(g2d);
    drawBorders(g2d);
    g.drawImage(buffer, 0, 0, this);

    if (isPaused) {
        g.setColor(new Color(0, 0, 0, 150));
        g.fillRect(borderOffset - 1, topMargin + borderOffset - 1, boardWidth + 1, boardHeight + 1);
        g.setColor(Color.WHITE);
        g.setFont(new Font("Arial", Font.BOLD, 24));
        String pauseText = "Пауза";
        int textWidth = g.getFontMetrics().stringWidth(pauseText);
        g.drawString(pauseText, (boardWidth - textWidth) / 2, boardHeight / 2);
    }
}

private void drawBoard(Graphics g) {
    Color[][] grid = board.getGrid();

    for (int y = 0; y < grid.length; y++) {
        for (int x = 0; x < grid[0].length; x++) {
            if (grid[y][x] != null) {
                int drawX = x * cellSize + borderOffset;
                int drawY = y * cellSize + topMargin + borderOffset;
                g.setColor(grid[y][x]);
                g.fillRect(drawX, drawY, cellSize, cellSize);

                g.setColor(Color.BLACK);
                g.drawRect(drawX, drawY, cellSize, cellSize);
```

```
            }
        }
    }
}

private void drawCurrentPiece(Graphics g) {
    Color[][] shape = currentPiece.getShape();

    for (int i = 0; i < shape.length; i++) {
        for (int j = 0; j < shape[i].length; j++) {
            if (shape[i][j] != null) {
                int drawX = (pieceX + j) * cellSize + borderOffset;
                int drawY = (pieceY + i) * cellSize + topMargin + borderOffset;
                g.setColor(shape[i][j]);
                g.fillRect(drawX, drawY, cellSize, cellSize);

                g.setColor(Color.BLACK);
                g.drawRect(drawX, drawY, cellSize, cellSize);
            }
        }
    }
}

private void drawBorders(Graphics g) {
    Graphics2D g2d = (Graphics2D) g;
    g2d.setColor(Color.BLACK);
    g2d.setStroke(new BasicStroke(3));
    g2d.drawRect(borderOffset - 1, topMargin + borderOffset - 1, boardWidth + 1, boardHeight + 1);
}


private void drawGameStats(Graphics g) {
    g.setColor(Color.BLACK);
    g.setFont(new Font("Arial", Font.BOLD, 20));

    String playerText = "Игрок: " + playerLogin;
    String scoreText = "Счет: " + score;

    FontMetrics fm = g.getFontMetrics();
    int playerTextWidth = fm.stringWidth(playerText);
    int scoreTextWidth = fm.stringWidth(scoreText);

    int centerX = boardWidth / 2;

    g.drawString(playerText, centerX - playerTextWidth / 2, 25);
    g.drawString(scoreText, centerX - scoreTextWidth / 2, 50);
}
}
```

LeaderboardWindow.java

```
import javax.swing.*;
import java.awt.*;
import java.rmi.RemoteException;
import java.util.List;

public class LeaderboardWindow extends JFrame {
    public LeaderboardWindow(TetrisServer server) {
        setTitle("Таблица лидеров");
        setSize(300, 400);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        JTextArea leaderboardArea = new JTextArea();
        leaderboardArea.setEditable(false);
        leaderboardArea.setFont(new Font("Arial", Font.PLAIN, 14));

        try {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
            SwingUtilities.updateComponentTreeUI(this);
```

```
        } catch (Exception e) {
            System.err.println("Не удалось установить Windows Look and Feel: " + e.getMessage());
        }

        try {
            List<String> leaderboard = server.getLeaderboard();
            StringBuilder leaderboardText = new StringBuilder("Топ игроков:\n\n");
            for (String entry : leaderboard) {
                leaderboardText.append(entry).append("\n");
            }
            leaderboardArea.setText(leaderboardText.toString());
        } catch (RemoteException e) {
            leaderboardArea.setText("Ошибка загрузки рейтинга.");
        }

        JScrollPane scrollPane = new JScrollPane(leaderboardArea);
        add(scrollPane, BorderLayout.CENTER);
    }
}
```