

## Содержание

[Введение 5](#_Toc113045)

[1 Постановка задачи 6](#_Toc113046)

[2 Выбор решения 7](#_Toc113047)

[3 Описание программы 10](#_Toc113048)

[4 Руководство пользователя 16](#_Toc113049)

[Заключение 25](#_Toc113052)

[Список используемых источников 26](#_Toc113053)

[Приложение А Листинги программы 27](#_Toc113054)

Приложение A.1. Файл «First.java»……………………………………………..27

Приложение A.2. Файл «RegistrationForm.java»….……………………………33

Приложение A.3. Файл «GameForm.java»…...…………………………………44

Приложение A.4. Файл «SoundManager.java»………………………………….64

Приложение A.5. Файл «ServerSaper.java»..……………………………………67

Приложение A.6. Файл «Database.java»…..…………………………………….70

Приложение A.7. Файл «ClientHandler.java»...…………………………………75

[Приложение Б UML-диаграммы…………………………………………………...78](#_Toc113055)

Приложение Б.1. UML-диграмма вариантов использования………………….78

Приложение Б.2. UML-диграмма последовательности…….………………….79

Приложение Б.3. UML-диграмма деятельности..…………...………………….80

Приложение Б.4. UML-диграмма классов.………………….………………….81

Приложение Б.5. UML-диграмма развертывания….………..………………... 82

# Введение

Клиент-серверная архитектура является одной из ключевых моделей взаимодействия в современных приложениях. Она предполагает разделение системы на две части: сервер, который предоставляет услуги, и клиент, который их запрашивает. В рамках данной курсовой работы разрабатывается клиент-серверное приложение компьютерной игры "Сапер" на языке Java. Игра "Сапер" представляет собой классическую головоломку, где игроку необходимо открывать ячейки на поле, избегая мин. В данном проекте игра реализуется в виде клиент-серверного приложения, что позволяет организовать многопользовательский режим. Серверная часть приложения отвечает за обработку запросов от клиентов и синхронизацию данных, а клиентская часть предоставляет пользовательский интерфейс для взаимодействия с игрой и логику игры.

Клиент-серверная архитектура обладает рядом преимуществ, таких как масштабируемость, централизованное управление данными и возможность распределения нагрузки между сервером и клиентами. Для реализации данного проекта выбран язык программирования Java, который предоставляет широкие возможности для разработки сетевых приложений благодаря встроенным библиотекам, таким как java.net для работы с сетью и java.util.concurrent для многопоточного программирования.

Графический интерфейс клиентской части будет реализован с использованием библиотеки Swing, что позволит создать удобный и интуитивно понятный интерфейс для пользователя. Серверная часть будет отвечать за обработку игровой логики, включая генерацию игрового поля, проверку ходов игроков и определение победителя. В ходе разработки будут изучены принципы клиент-серверного взаимодействия, сетевые протоколы и методы синхронизации данных между клиентами и сервером. Также будут освоены навыки работы с многопоточностью, что необходимо для обеспечения одновременного взаимодействия нескольких клиентов с сервером.

# Постановка задачи

Разработать клиент-серверное приложение для игры "Сапер", взаимодействующее по средствам сетевого протокола. Приложение должно обеспечивать многопользовательский режим, регистрацию и аутентификацию пользователей, а также ведение рейтинга лучших игроков.

**Функционал системы включает в себя:**

1. **Идентификацию пользователя** через регистрацию и аутентификацию;
2. **Логику игры "Сапер"**, включая генерацию игрового поля, обработку ходов игрока и определение победы или поражения;
3. **Ведение рейтинга** лучших игроков, с сохранением результатов в базу данных.

**Функционал сервера включает в себя:**

1. **Обработку запросов от клиентов**,включая регистрацию, аутентификацию;
2. **Сохранение результатов** в базу данных в случае победы игрока;
3. **Обеспечение синхронизации** данных между клиентами и сервером.

**Функционал клиента включает в себя:**

1. **Графический интерфейс пользователя** для взаимодействия с игрой;
2. **Логику игры "Сапер"**, включая обработку действий игрока и отображение игрового поля;
3. **Отправку запросов на сервер** для регистрации, аутентификации и передачи данных игре.

Полная диаграмма вариантов использования представлена в приложении Б.1. – UML диаграмма классов (рис. Б.1.).

# Выбор решения

Приложение реализовано на языке Java с использованием интерфейса Swing, в среде разработки NetBeans.

**Java** – это компилируемый, статически типизированный язык программирования общего назначения, который поддерживает такие парадигмы программирования, как объектно-ориентированное, функциональное и процедурное программирование. Java известен своей платформонезависимостью благодаря использованию виртуальной машины Java (JVM), которая позволяет выполнять скомпилированный байт-код на любом устройстве, поддерживающем JVM. Язык обладает богатой стандартной библиотекой, включающей средства для работы с коллекциями, вводом-выводом, сетевыми протоколами, многопоточностью, графическим интерфейсом и многим другим.

Java сочетает в себе простоту разработки и высокую производительность, что делает его универсальным инструментом для создания разнообразных приложений. В отличие от низкоуровневых языков, Java предоставляет встроенные механизмы для управления памятью (сборка мусора), что упрощает процесс разработки и снижает вероятность ошибок. При этом язык сохраняет гибкость и мощь, позволяя создавать как небольшие приложения, так и сложные корпоративные системы.

Java широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание веб-приложений (с использованием технологий Spring, Hibernate), мобильных приложений (Android), корпоративных систем, высоконагруженных серверов, а также приложений для встраиваемых систем и интернета вещей (IoT). Java также активно применяется в научных исследованиях, разработке игр и образовательных проектах.

NetBeans – это мощная и универсальная интегрированная среда разработки (IDE), созданная для разработки приложений на языке Java, включая приложения с графическим интерфейсом пользователя (GUI) с использованием библиотеки Swing. NetBeans предоставляет высокую степень удобства, кроссплатформенную совместимость и современные инструменты разработки, что делает её одним из лучших выборов для Java-разработчиков.

NetBeans отличается следующими ключевыми преимуществами:

1. Мощная система разработки GUI: NetBeans интегрирован с библиотекой Swing, которая предоставляет разработчику гибкие и многофункциональные компоненты для создания интерфейсов. Эти компоненты поддерживают сложные элементы, такие как таблицы, деревья, анимация и адаптивный дизайн для различных платформ.
2. Кроссплатформенность: NetBeans позволяет разрабатывать приложения, которые легко переносятся между Windows, Linux, macOS и другими платформами. Код, написанный на Java, может быть выполнен на любой платформе, поддерживающей виртуальную машину Java (JVM), без значительных изменений.
3. Удобство для программистов: NetBeans поддерживает программирование на Java и интеграцию с другими технологиями через плагины, что делает её гибкой для широкого спектра проектов. Благодаря мощному графическому редактору интерфейсов, разработчики могут сосредоточиться на проектировании UX/UI, не теряя времени на ручное управление кодом элементов GUI.
4. Интеграция с сетевыми библиотеками: NetBeans поддерживает разработку сетевых приложений и включает мощные инструменты для работы с протоколами TCP/IP и другими сетевыми технологиями. В данной работе использовался протокол TCP для реализации клиент-серверного взаимодействия, что позволяет эффективно передавать данные между клиентом и сервером.
5. Современные инструменты разработки: NetBeans включает поддержку:
   * Редактирования кода с автодополнением и подсветкой синтаксиса.
   * Графического интерфейса отладки и анализа производительности.
   * Инструментов сборки, таких как Maven и Ant.
6. Обширная экосистема: NetBeans поддерживает интеграцию с многочисленными библиотеками и технологиями, что делает его универсальным решением для сложных проектов, включая те, что требуют высокой производительности и современных сетевых технологий.

Swing – это мощная библиотека для создания графических интерфейсов на Java. Она предоставляет широкий набор компонентов, таких как кнопки, текстовые поля, таблицы и многое другое, что позволяет создавать интуитивно понятные и функциональные интерфейсы. Swing поддерживает кастомизацию внешнего вида и поведения компонентов, что делает его идеальным выбором для разработки кроссплатформенных приложений.

Логика системы программы будет написана на языке Java. Графическая часть будет реализована с использованием библиотеки Swing, а обмен данными между клиентом и сервером будет происходить с помощью протокола TCP, что обеспечивает надежную и эффективную передачу данных. NetBeans предоставляет программистам возможность сосредоточиться на создании мощных, красивых и удобных приложений, используя современные инструменты разработки и упрощённый доступ к сложным функциям. Его кроссплатформенные возможности делают его незаменимым для создания приложений, которые одинаково хорошо работают на Windows, macOS и других платформах.

# Описание программы

Разработанная программа состоит из клиента и сервера, они оба встроены в программу. Со стороны клиента за работу приложения отвечают следующие файлы: «Fist.java» – аутентификация, «RegistrationForm.java» – регистрация, «GameForm.java» – обработка игры, «SoundManager» – работа с музыкой. За серверную часть отвечают три файла: «ServerSaper.java» - порядок выполнения действий, «ClientHandler». – управление запросами, поступающих со стороны клиента и «Database.java» – работа с БД MySQL.

При запуске приложения пользователю открывается аутентификационное окно (рис. 1), поскольку процесс работы приложения стоит начинать рассматривать с регистрации, то описание аутентификации будет рассмотрено позже. Если пользователь не был зарегистрирован, то он может пройти процесс регистрации, нажав на кнопку “Регистрация”, при этом аутентификационное окно скроется, а регистрационное (рис. 2) откроется. Пользователь должен заполнить форму, состоящую из: имени, фамилии, логина, адреса электронной почты, пароля и подтверждения пароля. Копирование и отправку регистрационных данных на сервер выполняет метод «processRegistration», так же данный метод на стороне клиент проверяет корректность введенных параметров, а именно, наличие в имени и фамилии только букв, проверку логина на наличие только латинских букв и спец символов и максимальную длину в пятнадцать символов, наличия в электронной почте @ и домена, минимальную длину пароля в четыре символа и на совпадение пароля с подтверждающим паролем. В случае, если одна из проверок обнаружит ошибку в веденных пользователем данных, на клиенте высветиться ошибка. Только после успешно пройденных проверок на клиенте, на сервер поступят регистрационные данные по протоколу TCP с типом команды “1” (выполнения регистрации). В методе «run» осуществляется проверка на тип полученной команды. При регистрации в классе «Database» срабатывает метод «saveUser», он осуществляет SQL-запрос к БД. Полученные от пользователя регистрационные данные отправляются в БД, где логин и электронная почта проверяются на дубликат. В случае совпадения, на сервере формируется отрицательный ответ на регистрацию с указанием ошибки. Пользователь увидит на клиенте почему не получилось пройти процесс регистрации. Если логин и почта являются уникальными, то регистрационные данные пользователя занесутся в таблицу “users” и пользователь получит уведомление о успешно пройденной регистрации.

Процесс аутентификации заключается в следующем. Пользователю необходимо заполнить два поля для ввода, логин и пароль и нажать кнопку “Войти”. После нажатия на кнопку на сервер поступит код команды “2” (Аутентификация). Подробное взаимодействие классов и методов было расписано при прохождение процессе регистрации поэтому сейчас и в последующих описаниях алгоритма будет использована укороченная форма. В классе «Database» за авторизацию отвечает метод «checkUserCredentials» в нем формируется запрос к БД, в которой происходит поиск на совпадение логина и пароля в таблице “users”. В случае успешной аутентификации на клиент поступает соответствующие сообщение и происходит открытие игрового окна «GameForm» (рис. 3). Если логин и пароль не найден, то пользователь получит сообщение о ошибке.

Далее необходимо рассмотреть интерфейс игрового окна (рис. 3) и как устроен алгоритм игры “Сапер”, файл «GameForm.java». Игровое поле состоит из 225 ячеек, т.е его размерность составляет 15Х15. Количество бомб 45, а пустых ячеек 30. Состав игрового поля: jPanell, с 225 кнопками размером 40Х40 пикселей, расположенных по всей “панели”, за создание игрового поля отвечает метод «createGameField». Для того, чтобы игровое поле перестало было “пустым изображением” используется метод «initializeGameField», который отвечает за его инициализацию, устроена она следующим образом, за аналог игрового поля отвечает двумерный массив, в котором для обозначения бомбы используется

“-1”, пустая ячейка “0” и числа, отображающие бомбы вокруг, от 1 до 8. Заполнение игрового поля начинается с расставления 45 бомб в случайном порядке. Затем расставляются числа в соответствии с бомбами вокруг. Вокруг одной ячейки может быть максимум 8 бомб, для отображения чисел разным цветом используется метод «getColorForNumber», в котором для каждого числа, отвечающего за отображения бомб вокруг ячейки, используется свой цвет. За «отлавливание» нажатия на ячейку отвечает класс «CellClickListener», в методе «actionPerformed», при нажатии на первую ячейку свою работу начинает игровой секундомер (рис. 12), он сделан для того, чтобы при победе пользователя его результат можно было сравнить с другими пользователями, но данный алгоритм будет рассмотрен позже. Затем стоит рассмотреть еще одну функциональную особенность игры “Сапер”, флаг – механизм, который используется для упрощения игрового механизма, с помощью него можно отмечать бомбы. При нажатии на флаг (рис. 6), поле “flagMode” устанавливается в активный режим (true), вместо привычного курсора изображения мыши пользователь начинает видеть флаг, это было реализовано для того, чтобы было понятно, какой режим в данный момент используется. В том же методе «actionPerformed» происходит проверка на установку флага, если “flagMode” активен, то вместо открытия ячейки происходит установка (рис. 15) или снятия флага (рис. 16) из выбранной ячейки, при этом счетчик бомб (рис. 5) будет уменьшен на соответствующие количество установленных флагов (рис. 15) или увеличен, при снятии флага (рис. 16). Если “flagMode” неактивен, то происходит открытие ячейки (рис. 16), при этом значение счетчика открытых ячеек инкрементируется, если она уже не была открыта, то в этом случае ничего не произойдет. Если пользователь нажмет на бомбу, то будет вызван метод, отвечающий за обработку поражения, «gameOver», он останавливает секундомер, открывает все ячейки игрового поля (рис. 18) за это отвечает метод «revealCell», запускает “звук поражения” и показывает изображение проигрыша (рис. 17). Для того чтобы запустить новую игру, необходимо нажать на кнопку “Новая игры” (рис. 13). За этот функционал отвечает метод «resetGame», рассмотрим его работу, он отвечает за очищение двумерного массива, для нового процесса инициализации игрового поля, обнуление секундомера, восстановление изначального количества бомб, обнуления счетчика открытых ячеек и перерисовывание игрового поля. Так же стоит рассмотреть алгоритм открытия “пустых ячеек”, которые характеризуются “0” в массиве, за это отвечает метод «revealEmptyCells». Его функционал устроен следующим образом, при попадании в пустую ячейку она открывается и начинается процесс рекурсивного вызова метода «revealEmptyCells», пока не будут открыты все ближние “пустышки”, при этом соседние ячейки, которые отображают бомбы, то же открываются и каждая открытая ячейка инкрементирует значения счетчика открытых ячеек.

При достижение открытых ячеек в количестве 180(всего 225 клеток, 45 бомб, 225-45=180), вызывается метод «gameWin», который отвечает за обработку победы. Останавливается секундомер и его значение извлекается в переменную для отправки на сервер вместе с логином пользователя, на сервер поступает команда “4” (обработка победы). В методе «saveWinResult» формируются следующие SQL запросы:

1. Проверка на существование в таблице “leaderboard” логина текущего пользователя. Это необходимо для обновления результата на более лучший.
2. Добавление нового результата.
3. Запрос на обновление (Возвращаемся к 1 пункту).
4. Подсчет строк таблицы. Поскольку в таблице максимальное количество пользователей 100, нам необходимо знать достигла таблица 100 записей или нет.
5. Поиск худшего результата. Это необходимо, если таблица уже заполнена и нужно заменить одного пользователя на другого.
6. Удаление записи (5 пункт).

Далее происходит проверка на количество записей в таблице “leaderboard”. Существует ли запись с логином текущего пользователя, если запись уже есть, то происходит сравнение “нового” и “старого” времени, если новое время лучше, то происходит замена времени, если время хуже, то время остается без изменений, если записи ранее не было, то начинается проверка на количество записей в таблице, при достижении 100 записей начинается поиск более худшего результата, если он был найден, то происходит удаление соответствующий записи и на ее место устанавливается новая, если все “старые” результаты лучше нового, обновление таблицы не происходит. На клиенте запускается победные звук и изображение (рис. 20). Далее необходимо нажать на кнопку “Новая игра”, которая была описана ранее.

Важным элементом игры является ее соревновательная часть. Для этого была создана ранее затронутая таблица топ 100 лидеров «leaderboard». При открывании игрового окна (рис. 3) на сервер автоматически отправляется команда “3” (Получение таблицы лидеров). Таблицы лидеров храниться и в БД и локально на сервере в списке. Это было реализовано с той целью, чтобы пользователь, при подключении, получал такую же таблицу лидеров, как и у всех остальных подключенных клиентов. За рассылку содержимого таблицы лидеров отвечает метод «sendLeaderboard». Обновление таблицы лидеров на сервере происходит один раз в три минуты, это обеспечивает метод «updateLeaderboard» он формирует SQL запрос на получение отсортированного списка содержимого таблицы “leaderboard”, при каждом обновлении на сервере она отправляется всем подключенным клиентам с помощью метода «broadcastLeaderboard», для этого реализован метод «processServerMessage», который ожидает от сервера команду “ LEADERBOARD:”, в этом методе происходит копирование содержимого таблицы “leaderboard” с сервера в список на клиенте, с помощью метода «updateLeaderboardTable». За отображение на клиенте таблицы лидеров (рис. 11) отвечает метод «setupTopTable».

Настало время рассмотреть дополнительный функционал. Ранее уже затрагивались звуковое и музыкальное сопровождение. К звуковому сопровождению относятся следующие звуки: открытия ячейки, установки или снятия флага, взрыв бомбы(поражение) и торжественное пение(победа). К музыкальному сопровождению относятся 25 файлов с музыкой, которые воспроизводятся в случайно порядке, без повторений. Работа с музыкой и звуком, осуществляется в классе «SoundManager». По умолчанию, при открытии игрового окна (рис. 3) звук и музыка включены, однако пользователь может их выключить, нажав на кнопку “настроек” (рис. 7), при нажатии на которую, появятся две новые кнопки “звук” и “музыка” (рис. 8), нажатие на которые расценивается как отключение соответствующего функционала. При этом, если пользователь выключает “звук”, то иконка поменяет свой вид (рис. 9), или выключение “музыки” (рис. 10). Для того, чтобы вернуть “музыку” и “звук” необходимо снова нажать, на нужную кнопку, изображение вернется в исходное состояние (рис. 8). Программно это реализуется переключением состояний флагов при нажатии на соответствующие кнопки “звук” и “музыка”.

Рассмотрим заключительную часть работы приложения – отключение от сервера. За это отвечает метод “disconnectFromServer”, он отправляет на сервер команду “10” (Отключение от сервера). На сервере происходит выход из цикла обработки, метод «run», соответствующего пользователя и вызов метода «closeConnection», благодаря которому происходит удаление пользователя из списка подключенных клиентов.

Полная диаграмма классов представлена в приложении Б.4. – UML диаграмма классов (рис. Б.4.).

# Руководство пользователя

При запуске программы клиента пользователю открывается аутентификационное окно (рис. 1). Чтобы выполнить аутентификацию нужно ввести логин и пароль и нажать «Войти», если пользователь зарегистрирован, иначе нужно нажать на копку «Регистрация», откроется окно с регистраций (рис. 2).

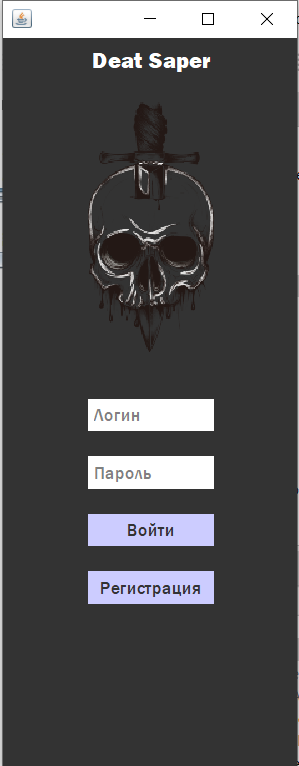


Рисунок 1 – Аутентификационное окно

Чтобы выполнить регистрацию необходимо ввести: имя, фамилию, логин, почту, пароль и подтверждение пароля и нажать кнопку “Готово”.

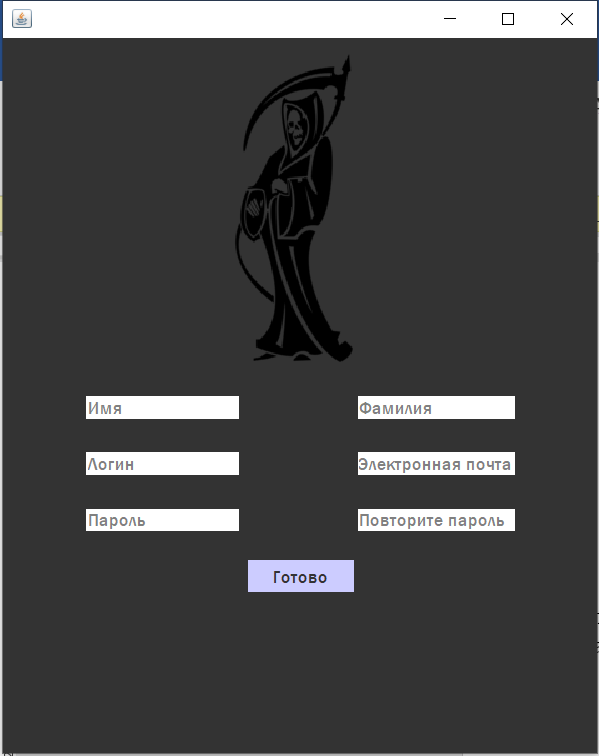


Рисунок 2 –Регистрационное окно

После успешной аутентификации пользователю открывается игровое окно (рис. 3).



Рисунок 3 – Игровое окно

Можно увидеть приветственное сообщение с логином пользователя (рис. 4).

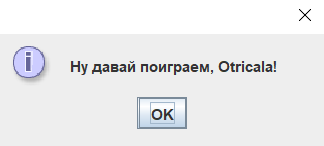


Рисунок 4 – Приветственное сообщение

Счетчик бомб (рис. 5).



Рисунок 5 – Счетчик бомб

Кнопку в виде флага, для отметки бомбы в ячейке (рис. 6).



Рисунок 6 – Кнопка переключения режима работы с ячейками

Кнопку “настройки” в виде шестеренок в правом верхнем углу окна (рис. 7).



Рисунок 7 – Кнопка настроек

При нажатии на которую начинают отображаться две дополнительные кнопки (рис. 8).



Рисунок 8 – Кнопки настройки

Кнопка с изображением динамика (рис. 8) отвечает за звуковое сопровождение, а именно: открытие ячейки, установка или снятие флага, звук победы и поражения. При нажатии на данную кнопку будет выключен звук и изображение поменяет свой вид (рис. 9). При повторном нажатии, пользователь снова начнет слышать звуки и изображение вернется в исходное состояние (рис. 8).



Рисунок 9 – Кнопка, отображающая выключенный звук

Кнопка с изображением виниловой пластинки (рис. 8) отвечает за музыкальное сопровождение, в момент открытия игрового окна начинает воспроизводиться музыка. При нажатии на данную кнопку будет выключена музыка и изображение поменяет свой вид (рис. 10). При повторном нажатии, музыка снова начнет воспроизводиться и изображение вернется в начальную форму (рис. 8).



Рисунок 10 – Кнопка, отображающая выключенную музыку

Справа, расположена таблица (рис. 11), в которой содержится актуальная информация о лучших ста игроках.

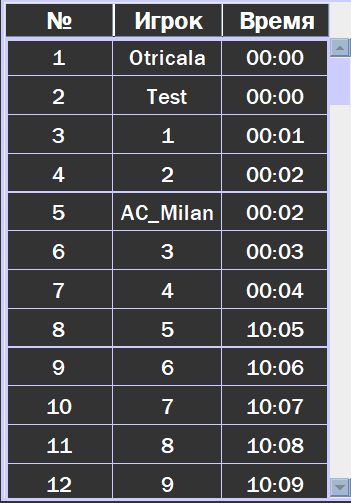


Рисунок 11 – Таблица лидеров

Секундомер игрового времени (рис. 12).



Рисунок 12 – Секундомер игрового времени

Кнопка “Новая игра” (рис. 13), отвечает за начало новый игры.



Рисунок 13 – Кнопка “Новая игра”

Для того, чтобы непосредственно начать игру, необходимо навестись курсором на нужную клетку и нажать левой кнопкой мыши, произойдет открытии ячейки (рис. 14), так же при нажатии на первую ячейку, счетчик игрового времени начинает свою работу (рис. 16).



Рисунок 14 – Запущенный секундомер при открытии первой клетки

При нажатии на изображение флага (рис. 6), меняться режим нажатия, вместо открытия ячейки, установка флага, при этом счетчик бомб уменьшается на соответствующие количество установленных флагов (рис. 15).



Рисунок 15 – Обновленное состояние счетчика бомб, при установке флага

Флаг можно убрать из ячейки, повторно нажав на нее с включенным режимом установки флага, счетчик бомб при этом измениться на соответствующие количество убранных флагов (рис. 16).



Рисунок 16 – Обновленное состояние счетчика бомб, при снятии флага

Если вы попали на бомбу, то игра будет закончена, вы увидите соответствующие изображение (рис. 17) и открытое игровое поле (рис. 18), чтобы удостовериться, что все честно.

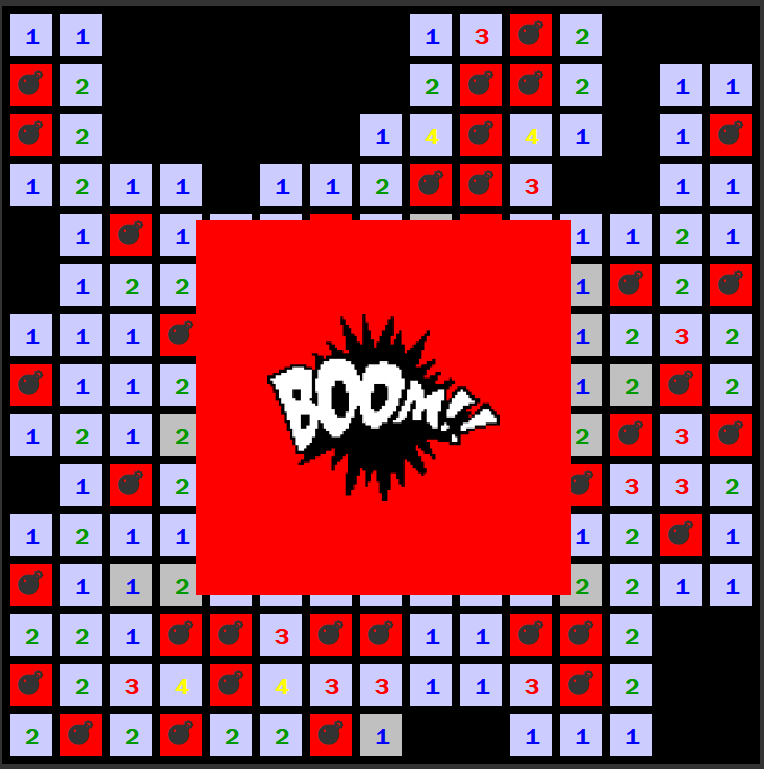


Рисунок 17 – Поражение

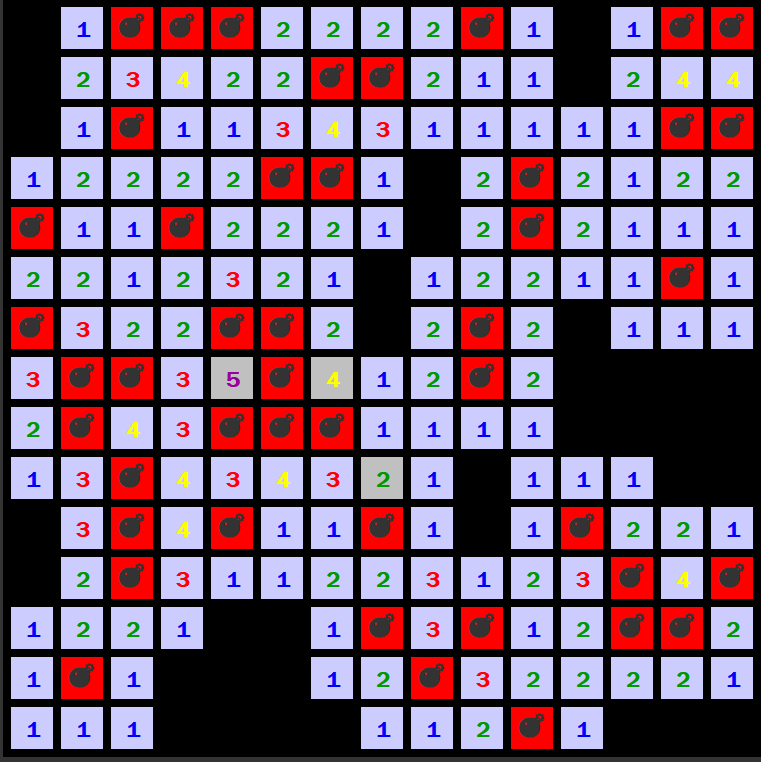


Рисунок 18 – Открытое игровое поле

После поражения необходимо нажать на кнопку “Новая игра” (рис. 13) произойдет обновление игрового поля (рис. 19).

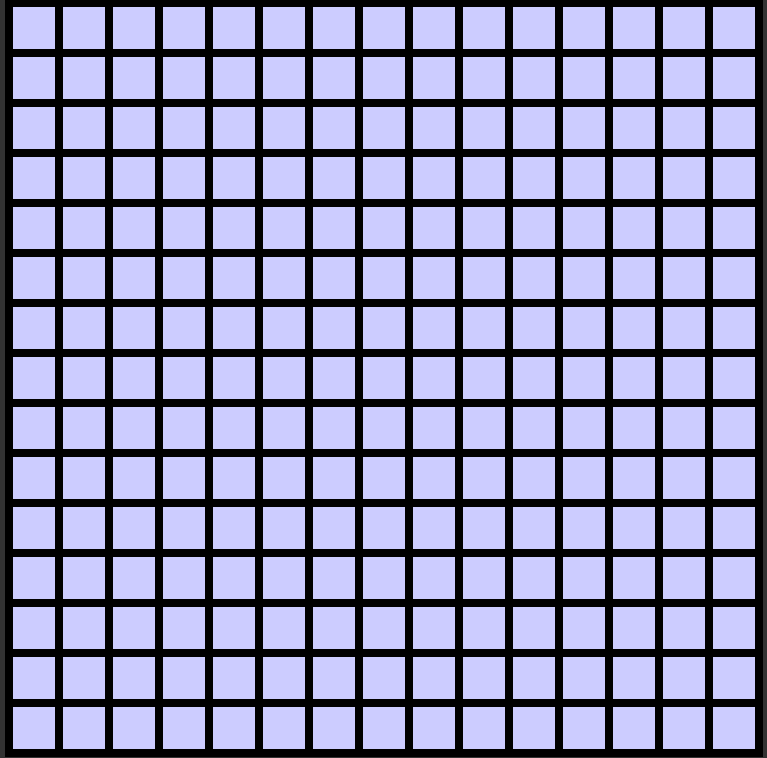


Рисунок 19 – Очищенное игровое поле

Если же вам удалось открыть все ячейки, то поздравляю, вы прошли игру “Сапер”, при этом появиться “победное” изображение (рис. 20). При этом возможно, что пользователю удастся попасть в таблицу лидеров (рис. 11). Чтобы начать новую игру, необходимо нажать на кнопку “Новая игра” (рис. 13) произойдет обновление игрового поля (рис. 19).



Рисунок 20 – “Победное изображение”

Полная диаграмма деятельности представлена в приложении Б.3. – UML диаграмма деятельности (рис. Б.3.).

# Заключение

В процессе выполнения курсовой работы было реализовано клиент-серверное приложение компьютерная игра “Сапер”, были получены следующие навыки: программирования на языке Java, работа с NetBeans, проработки архитектуры клиент-серверного программного обеспечения, разработки протоколов взаимодействия между клиентами и сервером.

# Список используемых источников

1. **Хорстманн К., Корнелл Г.** Java. Библиотека профессионала. Том 1. Основы. – 11-е изд. – М.: Диалектика, 2020. – 864 с.
2. **Хорстманн К., Корнелл Г.** Java. Библиотека профессионала. Том 2. Расширенные средства программирования. – 11-е изд. – М.: Диалектика, 2020. – 976 с.
3. **Эккель Б.** Философия Java. – 4-е изд. – СПб.: Питер, 2019. – 1168 с.
4. **Марк Л. Уивер, Джим Вайс, Джефф Старк.** MySQL 8. Справочник по администрированию и настройке. – М.: Диалектика, 2020. – 720 с.
5. **Дэвид М. Гери, Кэтрин С. Хорстманн.** Java Swing. Руководство для начинающих. – М.: Вильямс, 2018. – 480 с.

**Приложение А**

**Листинг программы**

**Приложение А.1.**

**Файл «First.java»**

package com.mycompany.clientsaper;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

import java.awt.Color;

import java.nio.charset.StandardCharsets;

// Класс аутентификационного окна

public class First extends javax.swing.JFrame {

public First() {

System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF\_8)); // Русификация

initComponents(); // UI

setupPlaceholders(); // Установка подcказок

Enter\_Login.setFocusable(false); // Убираем фокус в текcтовом поле(Мешало выводить подсказку)

// Передаем фокус окна в потоке

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

Enter\_Login.setFocusable(true); // Возвращаем фокус для текстового поля

First.this.requestFocusInWindow();

});

}

// Метод проверки для обычных полей

private boolean isPlaceholder(JTextField field) {

return field.getText().trim().isEmpty() || field.getForeground().equals(Color.GRAY);

}

// Метод проверки для полей с паролями

private boolean isPlaceholder(JPasswordField field) {

return new String(field.getPassword()).trim().isEmpty() || field.getForeground().equals(Color.GRAY);

}

@SuppressWarnings("unchecked")

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

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

Enter\_Login = new javax.swing.JTextField();

Button\_Enter = new javax.swing.JButton();

Button\_Reg = new javax.swing.JButton();

Enter\_Password = new javax.swing.JPasswordField();

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

setMinimumSize(new java.awt.Dimension(250, 620));

setPreferredSize(new java.awt.Dimension(250, 620));

jPanel1.setBackground(new java.awt.Color(51, 51, 51));

jPanel1.setMinimumSize(new java.awt.Dimension(100, 500));

jLabel1.setFont(new java.awt.Font("Franklin Gothic Heavy", 0, 18)); // NOI18N

jLabel1.setForeground(new java.awt.Color(255, 255, 255));

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel1.setText("Deat Saper"); // NOI18N

jLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/pngwing.com.png"))); // NOI18N

Enter\_Login.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Enter\_Login.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(255, 255, 255), 5));

Enter\_Login.addActionListener(new java.awt.event.ActionListener() {

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

Enter\_LoginActionPerformed(evt);

}

});

Button\_Enter.setBackground(new java.awt.Color(204, 204, 255));

Button\_Enter.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Button\_Enter.setText("Войти");

Button\_Enter.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(204, 204, 255), 5, true));

Button\_Enter.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

Button\_Enter.addActionListener(new java.awt.event.ActionListener() {

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

Button\_EnterActionPerformed(evt);

}

});

Button\_Reg.setBackground(new java.awt.Color(204, 204, 255));

Button\_Reg.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Button\_Reg.setText("Регистрация");

Button\_Reg.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(204, 204, 255), 5, true));

Button\_Reg.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

Button\_Reg.addActionListener(new java.awt.event.ActionListener() {

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

Button\_RegActionPerformed(evt);

}

});

Enter\_Password.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Enter\_Password.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(255, 255, 255), 5));

Enter\_Password.addActionListener(new java.awt.event.ActionListener() {

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

Enter\_PasswordActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(550, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(Enter\_Login)

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

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

.addComponent(Enter\_Password))

.addContainerGap(550, Short.MAX\_VALUE))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(7, 7, 7)

.addComponent(jLabel1)

.addGap(7, 7, 7)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 234, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(20, 20, 20)

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

.addGap(20, 20, 20)

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

.addGap(20, 20, 20)

.addComponent(Button\_Enter)

.addGap(20, 20, 20)

.addComponent(Button\_Reg)

.addContainerGap(134, Short.MAX\_VALUE))

);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

private void Enter\_LoginActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

// Обработка аутентификации

private void Button\_EnterActionPerformed(java.awt.event.ActionEvent evt) {

// Проверяем, что пользователь заполнил все поля

if (isPlaceholder(Enter\_Login) || isPlaceholder(Enter\_Password)) {

JOptionPane.showMessageDialog(this, "Все поля должны быть заполнены", "Ошибка", JOptionPane.ERROR\_MESSAGE);

return;

}

String login = Enter\_Login.getText(); // Получаем логин

String password = new String(Enter\_Password.getPassword()); // Получаем пароль

// Формируем запрос

try (Socket socket = new Socket("192.168.56.1", 1236);

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))) {

// Отправляем команду аутентификации

out.println("2");

out.println(login);

out.println(password);

// Получаем ответ от сервера

String response = in.readLine();

JOptionPane.showMessageDialog(this, response, "Ответ сервера", JOptionPane.INFORMATION\_MESSAGE);

// Если аутентификация прошла успешно

if ("Авторизация успешна".equals(response)) {

GameForm gameWindow = new GameForm(login); // Передаем логин в игровое окно

gameWindow.setVisible(true); // Открываем игровое окно и передаем логин

this.dispose(); // Закрываем аутентификационное окно

}

} catch (IOException e) {

JOptionPane.showMessageDialog(this, "Ошибка соединения с сервером", "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

private void Enter\_PasswordActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

// Обработка нажатия на регистрацию

private void Button\_RegActionPerformed(java.awt.event.ActionEvent evt) {

this.setVisible(false); // Скрываем текущее окно

new RegistrationForm(this).setVisible(true); // Передаём ссылку на First и открываем регистрационное окно

}

// Установка подсказки

private void setupPlaceholders() {

// Установка серого текста (подсказки)

setPlaceholder(Enter\_Login, "Логин");

setPasswordPlaceholder(Enter\_Password, "Пароль");

}

// Тоже самое что и в регистрации

private void setPlaceholder(JTextField field, String placeholder) {

field.setText(placeholder);

field.setForeground(Color.GRAY);

field.addFocusListener(new java.awt.event.FocusAdapter() {

public void focusGained(java.awt.event.FocusEvent evt) {

if (field.getText().equals(placeholder)) {

field.setText("");

field.setForeground(Color.BLACK);

}

}

public void focusLost(java.awt.event.FocusEvent evt) {

if (field.getText().isEmpty()) {

field.setText(placeholder);

field.setForeground(Color.GRAY);

}

}

});

}

// Тоже самое что и в регистрации

private void setPasswordPlaceholder(JPasswordField field, String placeholder) {

field.setForeground(Color.GRAY);

field.setEchoChar((char) 0); // Показываем текст как обычный

field.setText(placeholder);

field.addFocusListener(new java.awt.event.FocusAdapter() {

public void focusGained(java.awt.event.FocusEvent evt) {

if (String.valueOf(field.getPassword()).equals(placeholder)) {

field.setText("");

field.setForeground(Color.BLACK);

field.setEchoChar('●'); // Устанавливаем скрытый ввод

}

}

public void focusLost(java.awt.event.FocusEvent evt) {

if (field.getPassword().length == 0) {

field.setText(placeholder);

field.setForeground(Color.GRAY);

field.setEchoChar((char) 0); // Возвращаем текст как обычный

}

}

});

}

// Открытие окна

public static void main(String args[]) {

// Запускаем приложение

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

new First().setVisible(true);

});

}

// Variables declaration - do not modify

private javax.swing.JButton Button\_Enter;

private javax.swing.JButton Button\_Reg;

private javax.swing.JTextField Enter\_Login;

private javax.swing.JPasswordField Enter\_Password;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JPanel jPanel1;

// End of variables declaration

}

## Приложение A.2. Файл «RegistrationForm.java»

package com.mycompany.clientsaper;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

import java.awt.Color;

import java.util.regex.Pattern;

import java.nio.charset.StandardCharsets;

// Класс обработки регистрации

public class RegistrationForm extends javax.swing.JFrame {

private First parentWindow; // Аутентификационное окно являеться родителем

// Обработка регистрационного окна

public RegistrationForm(First parent) {

System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF\_8)); // Русификация

this.parentWindow = parent; // Указываем, что аутентификационное окно родитель

initComponents(); // UI

setupPlaceholders(); // Применяем подсказки

// Не закрываем окно с авторизацией, при закрытии окна с регистрацией

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

// Добавляем обработчик закрытия окна

addWindowListener(new java.awt.event.WindowAdapter() {

@Override

public void windowClosing(java.awt.event.WindowEvent e) {

if (parentWindow != null) {

parentWindow.setVisible(true); // Показываем главное окно

}

}

});

// Передаем фокус окна в потоке

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

RegistrationForm.this.requestFocusInWindow();

});

// Добавляем обработчик нажатия на кнопку регистрации

Button\_Done.addActionListener(e -> processRegistration());

}

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

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

Button\_Done = new javax.swing.JButton();

Name\_User = new javax.swing.JTextField();

Surname\_User = new javax.swing.JTextField();

Login\_User = new javax.swing.JTextField();

Password\_User\_First = new javax.swing.JPasswordField();

Mail\_User = new javax.swing.JTextField();

Password\_User\_Second = new javax.swing.JPasswordField();

jLabel1 = new javax.swing.JLabel();

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

setBackground(new java.awt.Color(51, 51, 51));

setMinimumSize(new java.awt.Dimension(490, 610));

setPreferredSize(new java.awt.Dimension(490, 610));

jPanel1.setBackground(new java.awt.Color(51, 51, 51));

jPanel1.setPreferredSize(new java.awt.Dimension(490, 610));

Button\_Done.setBackground(new java.awt.Color(204, 204, 255));

Button\_Done.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Button\_Done.setText("Готово");

Button\_Done.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(204, 204, 255), 5, true));

Button\_Done.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

Button\_Done.addActionListener(new java.awt.event.ActionListener() {

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

Button\_DoneActionPerformed(evt);

}

});

Name\_User.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Name\_User.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51), 5));

Name\_User.setMaximumSize(new java.awt.Dimension(130, 2147483647));

Name\_User.setMinimumSize(new java.awt.Dimension(130, 30));

Name\_User.addActionListener(new java.awt.event.ActionListener() {

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

Name\_UserActionPerformed(evt);

}

});

Surname\_User.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Surname\_User.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51), 5));

Surname\_User.setMaximumSize(new java.awt.Dimension(130, 2147483647));

Surname\_User.setMinimumSize(new java.awt.Dimension(130, 30));

Surname\_User.addActionListener(new java.awt.event.ActionListener() {

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

Surname\_UserActionPerformed(evt);

}

});

Login\_User.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Login\_User.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51), 5));

Login\_User.setMaximumSize(new java.awt.Dimension(120, 2147483647));

Login\_User.setMinimumSize(new java.awt.Dimension(120, 30));

Login\_User.addActionListener(new java.awt.event.ActionListener() {

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

Login\_UserActionPerformed(evt);

}

});

Password\_User\_First.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Password\_User\_First.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51), 5));

Password\_User\_First.setMaximumSize(new java.awt.Dimension(120, 2147483647));

Password\_User\_First.setMinimumSize(new java.awt.Dimension(120, 30));

Mail\_User.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Mail\_User.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51), 5));

Mail\_User.setMinimumSize(new java.awt.Dimension(64, 30));

Mail\_User.addActionListener(new java.awt.event.ActionListener() {

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

Mail\_UserActionPerformed(evt);

}

});

Password\_User\_Second.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

Password\_User\_Second.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51), 5));

Password\_User\_Second.setMinimumSize(new java.awt.Dimension(120, 30));

Password\_User\_Second.addActionListener(new java.awt.event.ActionListener() {

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

Password\_User\_SecondActionPerformed(evt);

}

});

jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Smert2.png"))); // NOI18N

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(326, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(Name\_User, javax.swing.GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

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

.addComponent(Password\_User\_First, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGap(87, 87, 87)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(Password\_User\_Second, javax.swing.GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

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

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

.addContainerGap(326, Short.MAX\_VALUE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(457, Short.MAX\_VALUE)

.addComponent(Button\_Done, javax.swing.GroupLayout.PREFERRED\_SIZE, 85, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(457, Short.MAX\_VALUE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(440, Short.MAX\_VALUE)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 119, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(440, Short.MAX\_VALUE))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

.addGap(22, 22, 22)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

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

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

.addGap(19, 19, 19)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

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

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

.addGap(19, 19, 19)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(Password\_User\_First, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(Password\_User\_Second, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(19, 19, 19)

.addComponent(Button\_Done)

.addContainerGap(128, Short.MAX\_VALUE))

);

Surname\_User.getAccessibleContext().setAccessibleParent(Name\_User);

Login\_User.getAccessibleContext().setAccessibleParent(Name\_User);

Password\_User\_First.getAccessibleContext().setAccessibleParent(Name\_User);

Mail\_User.getAccessibleContext().setAccessibleParent(Name\_User);

Password\_User\_Second.getAccessibleContext().setAccessibleParent(Name\_User);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

private void Button\_DoneActionPerformed(java.awt.event.ActionEvent evt) {

}

private void Surname\_UserActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void Name\_UserActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void Password\_User\_SecondActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void Login\_UserActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void Mail\_UserActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

/\*\*

\* @param args the command line arguments

\*/

// Устанавливаем подсказки

private void setupPlaceholders() {

// Установка серого текста (подсказки)

setPlaceholder(Login\_User, "Логин");

setPlaceholder(Mail\_User, "Электронная почта");

setPlaceholder(Name\_User, "Имя");

setPlaceholder(Surname\_User, "Фамилия");

setPasswordPlaceholder(Password\_User\_First, "Пароль");

setPasswordPlaceholder(Password\_User\_Second, "Повторите пароль");

}

// Обработка полей(обычных) для ввода

private void setPlaceholder(JTextField field, String placeholder) {

// Настраиваем поля для ввода

field.setText(placeholder);

field.setForeground(Color.GRAY);

// Слушатель событий фокуса

field.addFocusListener(new java.awt.event.FocusAdapter() {

// При нажатии на поле, убираем подсказку

@Override

public void focusGained(java.awt.event.FocusEvent evt) {

if (field.getText().equals(placeholder)) {

field.setText("");

field.setForeground(Color.BLACK);

}

}

// При потери фокуса

@Override

public void focusLost(java.awt.event.FocusEvent evt) {

// Если поле пустое, ставим подсказку обратно

if (field.getText().isEmpty()) {

field.setText(placeholder);

field.setForeground(Color.GRAY);

}

}

});

}

// Обработка полей(с паролем) для ввода

private void setPasswordPlaceholder(JPasswordField field, String placeholder) {

// Устанавливаем подсказки

field.setForeground(Color.GRAY);

field.setEchoChar((char) 0); // Показываем текст как обычный

field.setText(placeholder);

// Слушатель событий фокуса

field.addFocusListener(new java.awt.event.FocusAdapter() {

// При нажатии на поле

@Override

public void focusGained(java.awt.event.FocusEvent evt) {

if (String.valueOf(field.getPassword()).equals(placeholder)) {

field.setText("");

field.setForeground(Color.BLACK);

field.setEchoChar('●'); // Устанавливаем скрытый ввод

}

}

// При потери фокуса

@Override

public void focusLost(java.awt.event.FocusEvent evt) {

// Если поле пустое возвращаем подсказку

if (field.getPassword().length == 0) {

field.setText(placeholder);

field.setForeground(Color.GRAY);

field.setEchoChar((char) 0);

}

}

});

}

// Метод проверки для обычных полей

private boolean isPlaceholder(JTextField field) {

return field.getText().trim().isEmpty() || field.getForeground().equals(Color.GRAY);

}

// Метод проверки для полей с паролями

private boolean isPlaceholder(JPasswordField field) {

return new String(field.getPassword()).trim().isEmpty() || field.getForeground().equals(Color.GRAY);

}

// Запрос на сервер

private void processRegistration() {

// Очищаем регистрационные данные от пробелов

String login = Login\_User.getText().trim();

String mail = Mail\_User.getText().trim();

String name = Name\_User.getText().trim();

String surname = Surname\_User.getText().trim();

String password1 = new String(Password\_User\_First.getPassword()).trim();

String password2 = new String(Password\_User\_Second.getPassword()).trim();

Color defaultBorderColor = new Color(51, 51, 51); // Исходный цвет границы

Color errorBorderColor = Color.RED; // Цвет ошибки

StringBuilder errorMessage = new StringBuilder();

boolean hasErrors = false; // Флаг некорректного ввода

// Проверяем, что пользователь заполнил все поля

if (isPlaceholder(Login\_User) || isPlaceholder(Mail\_User) || isPlaceholder(Name\_User) ||

isPlaceholder(Surname\_User) || isPlaceholder(Password\_User\_First) ||

isPlaceholder(Password\_User\_Second)) {

JOptionPane.showMessageDialog(this, "Все поля должны быть заполнены", "Ошибка", JOptionPane.ERROR\_MESSAGE);

return;

}

// Проверка имени

if (!Pattern.matches("^[A-Za-zА-Яа-яЁё]+$", name)) {

Name\_User.setBorder(BorderFactory.createLineBorder(errorBorderColor, 5));

errorMessage.append("Имя должно содержать только буквы.\n");

hasErrors = true;

} else {

Name\_User.setBorder(BorderFactory.createLineBorder(defaultBorderColor, 5));

}

// Проверка фамилии

if (!Pattern.matches("^[A-Za-zА-Яа-яЁё]+$", surname)) {

Surname\_User.setBorder(BorderFactory.createLineBorder(errorBorderColor, 5));

errorMessage.append("Фамилия должна содержать только буквы.\n");

hasErrors = true;

} else {

Surname\_User.setBorder(BorderFactory.createLineBorder(defaultBorderColor, 5));

}

// Проверка почты

if (!Pattern.matches("^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$", mail)) {

Mail\_User.setBorder(BorderFactory.createLineBorder(errorBorderColor, 5));

errorMessage.append("Неверный формат email.\n");

hasErrors = true;

} else {

Mail\_User.setBorder(BorderFactory.createLineBorder(defaultBorderColor, 5));

}

// Проверка пароля (длина)

if (password1.length() < 4) {

Password\_User\_First.setBorder(BorderFactory.createLineBorder(errorBorderColor, 5));

errorMessage.append("Пароль должен содержать минимум 4 символа.\n");

hasErrors = true;

} else {

Password\_User\_First.setBorder(BorderFactory.createLineBorder(defaultBorderColor, 5));

}

// Проверка совпадения паролей

if (!password1.equals(password2)) {

Password\_User\_Second.setBorder(BorderFactory.createLineBorder(errorBorderColor, 5));

errorMessage.append("Пароли не совпадают.\n");

hasErrors = true;

} else {

Password\_User\_Second.setBorder(BorderFactory.createLineBorder(defaultBorderColor, 5));

}

// Проверка логина на корректность

if (!Pattern.matches("^[A-Za-z0-9!@#$%^&\*()\_+=-]+$", login)) {

Login\_User.setBorder(BorderFactory.createLineBorder(errorBorderColor, 5));

errorMessage.append("Логин может содержать только латинские буквы, цифры и спецсимволы.\n");

hasErrors = true;

} else {

Login\_User.setBorder(BorderFactory.createLineBorder(defaultBorderColor, 5));

}

// Проверка длины логина

if (login.length() > 15) {

Login\_User.setBorder(BorderFactory.createLineBorder(errorBorderColor, 5));

errorMessage.append("Логин не может содержать более 15 символов.\n");

hasErrors = true;

} else {

Login\_User.setBorder(BorderFactory.createLineBorder(defaultBorderColor, 5));

}

// Если есть ошибки, выводим их разом и прерываем процесс регистрации

if (hasErrors) {

// Показываем ошибки в диалоговом окне

JOptionPane.showMessageDialog(this, errorMessage.toString().trim(), "Ошибка регистрации", JOptionPane.ERROR\_MESSAGE);

return;

}

String serverAddress = "192.168.56.1"; // IP-адрес сервера

int port = 1236; // Порт сервера

// Формирование запроса

try (Socket socket = new Socket(serverAddress, port);

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

PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {

// Отправляем код команды регистрации

out.println("1");

// Отправляем регистрационные данные

out.println(name);

out.println(surname);

out.println(login);

out.println(mail);

out.println(password1);

// Получаем ответ от сервера

String response = in.readLine();

// Проверяем на дубликат email

if ("Пользователь с такой электронной почтой уже зарегистрирован!".equals(response)) {

Mail\_User.setBorder(BorderFactory.createLineBorder(errorBorderColor, 5));

System.out.println("Пользователь с таким логином уже зарегистрирован " + mail);

} else {

Mail\_User.setBorder(BorderFactory.createLineBorder(defaultBorderColor, 5));

}

// Проверяем на дубликат логина

if ("Пользователь с таким логином уже зарегистрирован!".equals(response)) {

Login\_User.setBorder(BorderFactory.createLineBorder(errorBorderColor, 5));

System.out.println("Пользователь с таким логином уже зарегистрирован " + login);

} else {

Login\_User.setBorder(BorderFactory.createLineBorder(defaultBorderColor, 5));

}

// Выводим ответ сервера

JOptionPane.showMessageDialog(this, response, "Ответ сервера", JOptionPane.INFORMATION\_MESSAGE);

// Если регистрация успешна, закрываем окно

if ("Регистрация прошла успешно!".equals(response)) {

this.dispose(); // Закрываем окно регистрации

// Показываем главное окно

if (parentWindow != null) {

parentWindow.setVisible(true);

}

}

} catch (IOException e) {

JOptionPane.showMessageDialog(this, "Ошибка подключения к серверу!", "Ошибка", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

}

}

// Variables declaration - do not modify

private javax.swing.JButton Button\_Done;

private javax.swing.JTextField Login\_User;

private javax.swing.JTextField Mail\_User;

private javax.swing.JTextField Name\_User;

private javax.swing.JPasswordField Password\_User\_First;

private javax.swing.JPasswordField Password\_User\_Second;

private javax.swing.JTextField Surname\_User;

private javax.swing.JLabel jLabel1;

private javax.swing.JPanel jPanel1;

// End of variables declaration

}

## Приложение A.3. Файл «GameForm.java»

package com.mycompany.clientsaper;

import java.io.\*;

import java.net.\*;

import java.awt.\*;

import javax.swing.\*;

import java.util.Random;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.awt.event.WindowEvent;

import java.awt.event.ActionEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.ActionListener;

import java.nio.charset.StandardCharsets;

import javax.swing.table.DefaultTableModel;

import javax.swing.plaf.basic.BasicScrollBarUI;

import javax.swing.table.DefaultTableCellRenderer;

// Класс игрового окна

public class GameForm extends javax.swing.JFrame {

private final int rows = 15; // Количество строк

private final int cols = 15; // Количество столбцов

private int[][] gameField = new int[rows][cols]; // Поле игры

private JButton[][] buttons = new JButton[rows][cols]; // Массив для хранения кнопок

private int bombCount = 45; // Количество бомб

private final int emptyCount = 30; // Количество пустых ячеек

private int openedCells = 0; // Счётчик открытых безопасных ячеек

private boolean flagMode = false; // Флаг режима установки флагов

private Timer gameTimer; // Таймер

private int secondsElapsed = 0; // Прошедшее время

private boolean timerStarted = false; // Флаг старта

private final String Flag\_Volume = "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Flag\_Volume.wav"; // Путь к звуку установки/снятия флага

private final String Cell\_Volume = "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Ratio.wav"; // Путь к звуку открытия ячейки

private final String Win\_Music = "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Win\_Music.wav"; // Путь к музыке победы

private final String Boom\_Bomb = "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Boom.wav"; // Путь к звуку взрыва бомбы

private final SoundManager soundManager = new SoundManager("D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Music\\"); // Файл с фоновой музыкой

private final String userLogin; // Поле для хранения логина

private Socket socket; // Сокет

private BufferedReader in; // Отправка

private PrintWriter out; // Получение

private Thread serverListenerThread; // Для прослушивания сервера

// Основной метод

public GameForm(String login) {

System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF\_8)); // Локализация для логов

setExtendedState(JFrame.MAXIMIZED\_BOTH); // Устанавливаем окно в максимизированное состояние

initComponents(); // Инициализация окна

this.userLogin = login; // Сохраняем логин пользователя

createGameField(); // Генерация игрового поля

initializeGameField(); // Инициализация игрового поля

soundManager.loadMusicFiles(); // Загрузка фоновой музыки из файла

DefaultTableModel model = new DefaultTableModel(new Object[]{"№", "Игрок", "Время"}, 0); // Создаем модель таблицы

Top\_Table.setModel(model); // Применяем новую модель таблицы

SwingUtilities.invokeLater(this::showWelcomeMessage); // Приветствие

if (soundManager.isMusicOn) {

soundManager.startMusicLoop(0.5f); // Запускаем музыку

}

// Обработчика кнопки настроек

Settings\_Button.addActionListener(this::Settings\_ButtonActionPerformed);

// Таймер игрового времени

gameTimer = new Timer(1000, e -> {

secondsElapsed++; // Увеличиваем счетчик секунд

// Проверяем, достигло ли время 59:59

if (secondsElapsed >= 3599) {

gameOver(); // Поражение по истечении времени

} else {

updateTimerDisplay(); // Обновляем отображение времени

}

});

// Стили кнопок

customizeButton(Settings\_Button);

customizeButton(New\_Game);

// Подключаемся к серверу

connectToServer();

// Обработчик закрытия соединения при нажатии на крестик

this.addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

disconnectFromServer(); // Отключаемся от сервера

}

});

}

// Обработка внешнего вида кнопок

private void customizeButton(JButton button) {

button.setBorderPainted(false);

button.setFocusPainted(false);

button.setContentAreaFilled(true);

button.setOpaque(true);

}

// Метод, подключения к серверу

private void connectToServer() {

try {

socket = new Socket("192.168.56.1", 1236);

in = new BufferedReader(new InputStreamReader(socket.getInputStream())); // Отправка

out = new PrintWriter(socket.getOutputStream(), true); // Получение

System.out.println("Успешно подключились к серверу!");

// Запрашиваем leaderboard при подключении

out.println("3");

// Запускаем поток для чтения сообщений от сервера

serverListenerThread = new Thread(() -> {

try {

String message;

while ((message = in.readLine()) != null) {

System.out.println("Сообщение от сервера: " + message);

processServerMessage(message); // Получаем сообщение

}

} catch (IOException e) {

System.err.println("Ошибка чтения данных от сервера: " + e.getMessage());

}

});

serverListenerThread.start(); // Слушаем дальше

} catch (IOException e) {

System.err.println("Ошибка подключения к серверу: " + e.getMessage());

}

}

// Метод, для обработки сообщения от сервера

private void processServerMessage(String message) {

if (message.startsWith("LEADERBOARD:")) {

// Убираем префикс "LEADERBOARD:" и разделяем строки по ";"

String[] rows = message.substring(12).split(";");

updateLeaderboardTable(rows);

}

}

// Метод обновления таблицы

private void updateLeaderboardTable(String[] rows) {

SwingUtilities.invokeLater(() -> {

DefaultTableModel model = (DefaultTableModel) Top\_Table.getModel();

model.setRowCount(0); // Очищаем таблицу перед обновлением

int rowNumber = 1; // Начинаем нумерацию с 1

for (String row : rows) {

// Разделяем строку по шаблону " - "

String[] data = row.split(" - ");

if (data.length == 2) {

String login = data[0]; // Логин игрока

String time = data[1]; // Время в формате ЧЧ:ММ:СС

// Преобразуем время в формат ММ:СС

String formattedTime = convertToMMSS(time);

// Добавляем данные в таблицу с порядковым номером

model.addRow(new Object[]{rowNumber, login, formattedTime});

rowNumber++; // Увеличиваем номер строки

}

}

// Повторно применяем стили к таблице

setupTopTable();

});

}

// Отключение от сервера

private void disconnectFromServer() {

try {

if (socket != null && !socket.isClosed()) {

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

out.println("10"); // Отправляем код 10

out.flush(); // Убедимся, что сервер получил данные

// Дадим серверу время обработать запрос

Thread.sleep(100);

socket.close(); // Закрываем сокет после отправки сигнала

System.out.println("Соединение с сервером закрыто.");

}

} catch (IOException | InterruptedException e) {

System.err.println("Ошибка при закрытии соединения: " + e.getMessage());

}

}

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

private void initComponents() {

jPanel2 = new javax.swing.JPanel();

jPanel1 = new javax.swing.JPanel();

Bomb\_img = new javax.swing.JLabel();

Flag\_img = new javax.swing.JLabel();

Timer = new javax.swing.JTextField();

Volume\_Bomb = new javax.swing.JTextField();

New\_Game = new javax.swing.JButton();

Settings\_Button = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

Top\_Table = new javax.swing.JTable();

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

setBackground(new java.awt.Color(51, 51, 51));

setMinimumSize(new java.awt.Dimension(1520, 980));

setPreferredSize(new java.awt.Dimension(1550, 654));

jPanel2.setBackground(new java.awt.Color(51, 51, 51));

jPanel1.setBackground(new java.awt.Color(51, 51, 51));

jPanel1.setBorder(javax.swing.BorderFactory.createMatteBorder(5, 1, 1, 5, new java.awt.Color(0, 0, 0)));

jPanel1.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 14)); // NOI18N

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 500, Short.MAX\_VALUE)

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 500, Short.MAX\_VALUE)

);

Bomb\_img.setIcon(new javax.swing.ImageIcon(getClass().getResource("/bomb2.png"))); // NOI18N

Flag\_img.setIcon(new javax.swing.ImageIcon(getClass().getResource("/red\_flag.png"))); // NOI18N

Flag\_img.setText("jLabel2");

Flag\_img.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

Flag\_imgMouseClicked(evt);

}

});

Timer.setEditable(false);

Timer.setBackground(new java.awt.Color(51, 51, 51));

Timer.setFont(new java.awt.Font("Franklin Gothic Medium", 1, 22)); // NOI18N

Timer.setForeground(new java.awt.Color(255, 255, 255));

Timer.setHorizontalAlignment(javax.swing.JTextField.CENTER);

Timer.setText("00:00");

Timer.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51)));

Timer.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));

Timer.setEnabled(false);

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

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

TimerActionPerformed(evt);

}

});

Volume\_Bomb.setEditable(false);

Volume\_Bomb.setBackground(new java.awt.Color(51, 51, 51));

Volume\_Bomb.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 18)); // NOI18N

Volume\_Bomb.setForeground(new java.awt.Color(255, 255, 255));

Volume\_Bomb.setText("45/45");

Volume\_Bomb.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51)));

Volume\_Bomb.setEnabled(false);

Volume\_Bomb.addActionListener(new java.awt.event.ActionListener() {

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

Volume\_BombActionPerformed(evt);

}

});

New\_Game.setBackground(new java.awt.Color(51, 51, 51));

New\_Game.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 20)); // NOI18N

New\_Game.setForeground(new java.awt.Color(204, 204, 255));

New\_Game.setText("Новая игра");

New\_Game.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51), 3));

New\_Game.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

New\_Game.addActionListener(new java.awt.event.ActionListener() {

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

New\_GameActionPerformed(evt);

}

});

Settings\_Button.setBackground(new java.awt.Color(51, 51, 51));

Settings\_Button.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Settings.png"))); // NOI18N

Settings\_Button.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 51)));

Settings\_Button.setBorderPainted(false);

Settings\_Button.addActionListener(new java.awt.event.ActionListener() {

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

Settings\_ButtonActionPerformed(evt);

}

});

jScrollPane1.setAutoscrolls(true);

jScrollPane1.setHorizontalScrollBar(null);

jScrollPane1.setMaximumSize(new java.awt.Dimension(1000, 400));

jScrollPane1.setPreferredSize(new java.awt.Dimension(1000, 402));

Top\_Table.setBackground(new java.awt.Color(51, 51, 51));

Top\_Table.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

Top\_Table.setFont(new java.awt.Font("Franklin Gothic Medium", 0, 18)); // NOI18N

Top\_Table.setForeground(new java.awt.Color(255, 255, 255));

Top\_Table.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

"№", "Логин", "Время"

}

) {

boolean[] canEdit = new boolean [] {

false, false, false

};

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit [columnIndex];

}

});

Top\_Table.setMaximumSize(new java.awt.Dimension(2000, 3148));

Top\_Table.setMinimumSize(new java.awt.Dimension(1100, 100));

Top\_Table.setName(""); // NOI18N

Top\_Table.setPreferredSize(new java.awt.Dimension(2000, 3098));

Top\_Table.setRequestFocusEnabled(false);

Top\_Table.setRowHeight(31);

Top\_Table.setSelectionBackground(new java.awt.Color(51, 51, 51));

Top\_Table.setSelectionForeground(new java.awt.Color(255, 255, 255));

jScrollPane1.setViewportView(Top\_Table);

if (Top\_Table.getColumnModel().getColumnCount() > 0) {

Top\_Table.getColumnModel().getColumn(0).setResizable(false);

Top\_Table.getColumnModel().getColumn(0).setPreferredWidth(10);

Top\_Table.getColumnModel().getColumn(1).setResizable(false);

Top\_Table.getColumnModel().getColumn(2).setResizable(false);

}

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap(390, Short.MAX\_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(Bomb\_img)

.addGap(0, 0, 0)

.addComponent(Volume\_Bomb, javax.swing.GroupLayout.PREFERRED\_SIZE, 71, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(450, 450, 450)

.addComponent(Flag\_img, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 110, Short.MAX\_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(Settings\_Button, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 43, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 283, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap(640, Short.MAX\_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(New\_Game)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(17, 17, 17)

.addComponent(Timer, javax.swing.GroupLayout.PREFERRED\_SIZE, 71, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addContainerGap(641, Short.MAX\_VALUE))

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(15, 15, 15)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(Flag\_img, javax.swing.GroupLayout.PREFERRED\_SIZE, 36, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(Bomb\_img))

.addGap(5, 5, 5)

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(10, 10, 10)

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

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(New\_Game)

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

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(Settings\_Button, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(116, 116, 116)

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

.addGap(0, 0, Short.MAX\_VALUE))

);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addComponent(jPanel2, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

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

.addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

private void Settings\_ButtonActionPerformed(java.awt.event.ActionEvent evt) {

// Создаем всплывающее меню

JPopupMenu popupMenu = new JPopupMenu();

// Убираем границы у всплывающего меню

popupMenu.setBorder(BorderFactory.createLineBorder(new Color(51, 51, 51)));

popupMenu.setOpaque(true);

popupMenu.setBackground(new Color(51, 51, 51));

// Загружаем изображения

ImageIcon volumeIcon = new ImageIcon(soundManager.isVolumeOn

? "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Volume\_On.png"

: "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Volume\_Off.png");

ImageIcon musicIcon = new ImageIcon(soundManager.isMusicOn

? "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Music\_On.png"

: "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Music\_Off.png");

// Создаем кнопки с изображениями

JMenuItem volumeButton = new JMenuItem("", volumeIcon);

JMenuItem musicButton = new JMenuItem("", musicIcon);

volumeButton.setPreferredSize(new Dimension(42, 42));

musicButton.setPreferredSize(new Dimension(42, 42));

Color customColor = new Color(51, 51, 51);

// Убираем границы у кнопок

volumeButton.setBorderPainted(false);

volumeButton.setContentAreaFilled(true);

volumeButton.setOpaque(true);

volumeButton.setBackground(customColor);

musicButton.setBorderPainted(false);

musicButton.setContentAreaFilled(true);

musicButton.setOpaque(true);

musicButton.setBackground(customColor);

// Добавляем кнопки в меню

popupMenu.add(volumeButton);

popupMenu.add(musicButton);

// Определяем позицию (ниже Settings\_Button)

popupMenu.show(Settings\_Button, -3, Settings\_Button.getHeight());

// Кнопка с изображением звука, обработчик нажатия

volumeButton.addActionListener(e -> {

soundManager.isVolumeOn = !soundManager.isVolumeOn; // Переключаем состояние

volumeButton.setIcon(new ImageIcon(soundManager.isVolumeOn

? "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Volume\_On.png" // Если звук включен

: "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Volume\_Off.png")); // Если звук выключен

});

// Кнопка с изображением музыки, обработчик нажатия

musicButton.addActionListener(e -> {

soundManager.isMusicOn = !soundManager.isMusicOn; // Переключаем состояние

musicButton.setIcon(new ImageIcon(soundManager.isMusicOn

? "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Music\_On.png" // Если музыка включена

: "D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Music\_Off.png")); // Если музыка выключена

if (soundManager.isMusicOn) {

soundManager.startMusicLoop(0.5f); // Включаем музыку, если она была выключена

} else {

soundManager.stopMusic(); // Выключаем музыку

}

});

}

// Метод отображения приветственного сообщения

private void showWelcomeMessage() {

JOptionPane.showMessageDialog(this, "Ну давай поиграем, " + userLogin + "!", "", JOptionPane.INFORMATION\_MESSAGE);

}

private void New\_GameActionPerformed(java.awt.event.ActionEvent evt) {

resetGame(); // Вызов новой игры

}

private void Volume\_BombActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

// Обработчик нажатия на режим флага

private void Flag\_imgMouseClicked(java.awt.event.MouseEvent evt) {

flagMode = !flagMode; // Переключаем режим

// Если реим флага активен

if (flagMode) {

Flag\_img.setBackground(Color.LIGHT\_GRAY); // Подсветим кнопку флага

// Иначе

} else {

Flag\_img.setBackground(null); // Убираем подсветку

}

// Если режим флага активен

if (flagMode) {

// Устанавливаем курсор в виде флага

ImageIcon flagIcon = getScaledIcon("D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\red\_flag.png", 32, 32);

Toolkit toolkit = Toolkit.getDefaultToolkit();

Cursor flagCursor = toolkit.createCustomCursor(flagIcon.getImage(), new Point(0, 0), "Flag Cursor");

this.setCursor(flagCursor); // Устанавливаем курсор для всего окна

// Иначе

} else {

// Возвращаем стандартный курсор

this.setCursor(Cursor.getDefaultCursor());

}

}

// Метод для создания игрового поля

private void createGameField() {

// Проверка на создание панели

if (jPanel1 == null) {

System.out.println("ERROR: jPanel1 is null!");

return;

}

jPanel1.setLayout(new GridLayout(rows, cols));

buttons = new JButton[rows][cols];

// Создание игрового поля

int buttonCount = 0;

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

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

JButton cell = new JButton();

cell.setPreferredSize(new Dimension(40, 40));

cell.setActionCommand(i + "," + j);

cell.addActionListener(new CellClickListener());

cell.setBackground(new java.awt.Color(204, 204, 255));

cell.setBorder(javax.swing.BorderFactory.createMatteBorder(1, 5, 5, 1, new java.awt.Color(0, 0, 0)));

cell.setMargin(new java.awt.Insets(1, 50, 1, 20));

jPanel1.add(cell);

buttons[i][j] = cell;

buttonCount++; // Увеличиваем обработанные ячейки

}

}

// Обновляем интерфейс

jPanel1.revalidate();

jPanel1.repaint();

}

// Метод для отображения игровых иконок

private ImageIcon getScaledIcon(String path, int width, int height) {

ImageIcon icon = new ImageIcon(path);

Image img = icon.getImage();

Image resizedImg = img.getScaledInstance(width, height, Image.SCALE\_SMOOTH);

return new ImageIcon(resizedImg);

}

// Метод, для отображения таблицы лидеров

void setupTopTable() {

SwingUtilities.invokeLater(() -> {

// Настройки цвета фона и границ

Top\_Table.getTableHeader().setBackground(new Color(51, 51, 51));

Top\_Table.getTableHeader().setForeground(Color.WHITE);

Top\_Table.setBackground(new Color(51, 51, 51));

jScrollPane1.getViewport().setBackground(new Color(51, 51, 51));

Color borderColor = new Color(204, 204, 255);

Top\_Table.getTableHeader().setBorder(BorderFactory.createLineBorder(borderColor, 1));

Top\_Table.getTableHeader().setFont(new Font("Franklin Gothic Medium", Font.BOLD, 20));

Top\_Table.setBorder(BorderFactory.createLineBorder(borderColor, 3));

jScrollPane1.setBorder(BorderFactory.createLineBorder(borderColor, 3));

// Центровка текста в ячейках

DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();

centerRenderer.setHorizontalAlignment(SwingConstants.CENTER);

centerRenderer.setVerticalAlignment(SwingConstants.CENTER);

// Применяем центровку ко всем столбцам

for (int i = 0; i < Top\_Table.getColumnCount(); i++) {

Top\_Table.getColumnModel().getColumn(i).setCellRenderer(centerRenderer);

}

// Установка границ ячеек

Top\_Table.setGridColor(borderColor);

Top\_Table.setShowGrid(true);

// Стилизация скроллбара

jScrollPane1.getVerticalScrollBar().setUI(new BasicScrollBarUI() {

@Override

protected void configureScrollBarColors() {

this.thumbColor = borderColor; // Цвет ползунка

}

});

// Убедимся, что таблица имеет три столбца

DefaultTableModel model = (DefaultTableModel) Top\_Table.getModel();

if (model.getColumnCount() != 3) {

// Если столбцов меньше или больше, пересоздаем модель

model.setColumnIdentifiers(new Object[]{"№", "Игрок", "Время"});

}

});

}

// Метод для инициализации поля

private void initializeGameField() {

Random random = new Random(); // Случайное значение для заполнения ячеек

// Проверяем, инициализирован ли массив gameField

if (gameField == null) {

System.out.println("ERROR: gameField is null!");

return;

}

// Расставляем бомбы

int bombs = bombCount; // Передаем количество бомб

int attempts = 0; // Для проверки зависания

// Пока не заполним поле 45 бомбами

while (bombs > 0) {

int x = random.nextInt(rows);

int y = random.nextInt(cols);

if (gameField[x][y] == 0) {

gameField[x][y] = -1; // Бомба

bombs--; // Уменьшаем счетчик бомб

}

attempts++; // Проверка на зависание

if (attempts > 10000) { // Если зависло

break;

}

}

// Заполняем числами

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

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

if (gameField[i][j] == -1) continue; // Пропускаем бомбы

int count = 0;

for (int dx = -1; dx <= 1; dx++) {

for (int dy = -1; dy <= 1; dy++) {

int nx = i + dx, ny = j + dy;

if (nx >= 0 && nx < rows && ny >= 0 && ny < cols && gameField[nx][ny] == -1) {

count++;

}

}

}

gameField[i][j] = count;

}

}

// Заполняем пустыми ячейками

int emptyCells = emptyCount;

attempts = 0; // Проверка зависания

while (emptyCells > 0) {

int x = random.nextInt(rows);

int y = random.nextInt(cols);

if (gameField[x][y] == 0) {

gameField[x][y] = -2; // Пустая клетка

emptyCells--;

}

attempts++;

if (attempts > 10000) { // Если зависло

break;

}

}

}

// Метод обрабатывающий нажатие на ячейку

private class CellClickListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

JButton clickedButton = (JButton) e.getSource();

// Запуск игрового таймера

if (!timerStarted) {

gameTimer.start();

timerStarted = true;

}

// Проверяем, был ли уже поставлен флаг

if (clickedButton.getClientProperty("flagged") != null) {

if (flagMode) {

soundManager.playClickSound(Flag\_Volume, 0.3f); // Запуск звука установки/снятия флага

clickedButton.setIcon(null); // Убираем флаг

clickedButton.putClientProperty("flagged", null); // Помечаем ячейку, как без флага

bombCount++; // Увеличиваем счетчик бомб

updateBombCounter(); // Обновляем отображение

}

return; // Выход из метода

}

// Проверяем, была ли кнопка уже нажата

if (clickedButton.getClientProperty("clicked") != null) {

return; // Игнорируем повторное нажатие

}

// Получаем координаты кнопки

String position = clickedButton.getActionCommand();

String[] pos = position.split(",");

int x = Integer.parseInt(pos[0]);

int y = Integer.parseInt(pos[1]);

// Если режим флагов включен, ставим флаг и выходим

if (flagMode) {

soundManager.playClickSound(Flag\_Volume, 0.3f); // Запуск звука установки/снятия флага

// Ставим изображение флага в ячейку

ImageIcon flagIcon = getScaledIcon("D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\red\_flag.png", 30, 30);

clickedButton.setIcon(flagIcon);

clickedButton.putClientProperty("flagged", true); // Помечаем кнопку как с флагом

bombCount--; // Уменьшаем счетчик бомб

updateBombCounter(); // Обновляем отображение

return; // Выходим из метода

}

int value = gameField[x][y]; // Значение в ячейке

// Если попали на бомбу

if (value == -1) {

clickedButton.setBackground(Color.RED); // Задний фон ячейки с бомбой

clickedButton.setText("💣"); // Ставим бомбу

clickedButton.setFont(new Font("Segoe UI Emoji", Font.BOLD, 18)); // Уставнока шрифта

gameOver(); // Вызов метода проигрыша

// Если попали в ячейку с цифрой

} else if (value > 0) {

soundManager.playClickSound(Cell\_Volume, 1.4f); // Воспроизводим звук клика

// Показываем число

clickedButton.setFont(new Font("Franklin Gothic Medium", Font.BOLD, 18));

Color textColor = getColorForNumber(value);

clickedButton.setForeground(textColor);

clickedButton.setText(String.valueOf(value));

clickedButton.setBackground(Color.LIGHT\_GRAY);

openedCells++; // Увеличиваем количество открытых ячеек

// Если попали в пустую ячейку

} else {

soundManager.playClickSound(Cell\_Volume, 1.4f); // Воспроизводим звук клика

// Если пустая клетка, открываем соседние

revealEmptyCells(x, y); // Выхывем метод открытия соседних пустых ячеек

}

// Делаем кнопку некликабельной

clickedButton.putClientProperty("clicked", true);

clickedButton.setFocusable(false);

clickedButton.setOpaque(true);

// Проверяем победу

if (openedCells == 180) {

gameWin(); // Вызываем метод, обрабатывающий победу

}

}

}

// Метод для рекурсивного открытия пустых ячеек

private void revealEmptyCells(int x, int y) {

if (x < 0 || x >= rows || y < 0 || y >= cols || gameField[x][y] == -1) return;

JButton button = buttons[x][y]; // Получаем кнопку из массива

// Убираем флаг, если он есть

if (Boolean.TRUE.equals(button.getClientProperty("flagged"))) {

button.setIcon(null); // Убираем иконку флага

button.putClientProperty("flagged", null); // Удаляем свойство

bombCount++; // Увеличиваем счетчик бомб

updateBombCounter(); // Обновляем отображение

}

// Если ячейка уже была открыта, ничего не делаем

if (button.getClientProperty("clicked") != null) return;

// Устанавливаем значение ячейки, как открытой

button.setBackground(Color.LIGHT\_GRAY);

button.putClientProperty("clicked", true);

button.setFocusable(false);

button.setOpaque(true);

openedCells++; // Учитываем открытие пустой клетки

// Если в ячейке есть число (количество бомб), то отображаем его

if (gameField[x][y] > 0) {

button.setFont(new Font("Franklin Gothic Medium", Font.BOLD, 18)); // Задаем шрифт

button.setText(String.valueOf(gameField[x][y]));

button.setForeground(getColorForNumber(gameField[x][y]));

return;

}

// Если ячейка пустая, рекурсивно открываем все соседние пустые ячейки

for (int dx = -1; dx <= 1; dx++) {

for (int dy = -1; dy <= 1; dy++) {

if (dx != 0 || dy != 0) {

revealEmptyCells(x + dx, y + dy);

}

}

}

}

// Метод по отображении таймера

private void updateTimerDisplay() {

int minutes = secondsElapsed / 60; // Минуты

int seconds = secondsElapsed % 60; // Секнды

String timeFormatted = String.format("%02d:%02d", minutes, seconds);

Timer.setText(timeFormatted); // Установка времени в режиме ММ:СС

}

// Метод, отображающий текущие значение не отмеченных бомб

private void updateBombCounter() {

Volume\_Bomb.setText(String.valueOf(bombCount + "/45"));

}

// Метод для получения цвета по числу

private Color getColorForNumber(int number) {

switch (number) {

case 1: return new Color(0, 0, 255);

case 2: return new Color(0, 153, 0);

case 3: return new Color(255, 0, 0);

case 4: return new Color(255, 255, 0);

case 5: return new Color(153, 0, 153);

case 6: return new Color(153, 0, 0);

case 7: return new Color(0, 102, 102);

case 8: return new Color(255, 255, 255);

default: return Color.BLACK;

}

}

// Метод открытия игрового поля

private void revealCell(int i, int j) {

if (buttons[i][j] == null) return; // Проверка на null

JButton clickedButton = buttons[i][j]; // Получаем нажатие

// Убираем флаг, если он есть

if (clickedButton.getClientProperty("flagged") != null) {

clickedButton.setIcon(null); // Убираем иконку флага

clickedButton.putClientProperty("flagged", null); // Удаляем свойство

}

int value = gameField[i][j]; // Получаем содержимое ячейки

if (value == -1) {

clickedButton.setText("💣");

clickedButton.setFont(new Font("Segoe UI Emoji", Font.BOLD, 18)); // Шрифт для эмодзи

clickedButton.setBackground(Color.RED); // Красный фон

} else if (value > 0) {

// Если это число - показываем его с цветом и шрифтом

clickedButton.setText(String.valueOf(value));

clickedButton.setForeground(getColorForNumber(value));

clickedButton.setFont(new Font("Franklin Gothic Medium", Font.BOLD, 18));

} else {

// Если это пустая ячейка - черный фон

clickedButton.setText("");

clickedButton.setBackground(Color.BLACK);

}

// Делаем кнопку "плоской", чтобы показать, что она раскрыта

clickedButton.setBorder(BorderFactory.createMatteBorder(1, 5, 5, 1, Color.BLACK)); // Граница 5 пикселей

// Делаем кнопку "некликабельной" без потери стилей

clickedButton.putClientProperty("clicked", true);

clickedButton.setFocusable(false);

clickedButton.setOpaque(true);

}

// Метод обрабатывающий "Новая игра"

private void resetGame() {

// Перед пересозданием поля полностью обнуляем массив

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

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

gameField[i][j] = 0;

}

}

// Остановка таймера

if (gameTimer != null) {

gameTimer.stop();

}

// Обнуление значения таймера

secondsElapsed = 0;

timerStarted = false;

updateTimerDisplay();

// Восстанавливаем счетчик бомб

bombCount = 45;

updateBombCounter();

openedCells = 0; // Обнуляем счетчик открытых ячеек

// Очищаем игровую панель

jPanel1.removeAll();

jPanel1.setLayout(new GridLayout(rows, cols)); // Обновляем GridLayout

// Инициализируем кнопки заново

buttons = new JButton[rows][cols];

// Пересоздаем игровое окно

SwingUtilities.invokeLater(() -> {

createGameField();

initializeGameField();

// Обновляем интерфейс

jPanel1.revalidate();

jPanel1.repaint();

});

}

// Метод, обрабатывающий поражение

private void gameOver() {

gameTimer.stop(); // Останавливаем таймер

New\_Game.setEnabled(false);// Блокируем кнопку "Новая игра"

// Открываем все ячейки на поле

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

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

revealCell(i, j); // Вызов метода, открывающий ячейки

}

}

ImageIcon icon = new ImageIcon("D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Boom.png"); // Загружаем изображение взрыва

// Создаём диалоговое окно

JDialog dialog = new JDialog((Frame) null, "");

dialog.setUndecorated(true); // Убираем рамки окна

dialog.setSize(300, 300); // Размер окна

dialog.getContentPane().setBackground(new Color(255, 0, 0)); // Цвет фона

dialog.setLayout(new BorderLayout());

// Создаём метку с изображением

JLabel label = new JLabel(icon);

label.setHorizontalAlignment(SwingConstants.CENTER); // Выравнивание по центру

dialog.add(label, BorderLayout.CENTER);

dialog.setLocationRelativeTo(null); // Размещаем окно по центру экрана

dialog.setVisible(true); // Отображаем изображение

// Запускаем фоновый процесс для звука и задержки

new SwingWorker<Void, Void>() {

@Override

protected Void doInBackground() throws Exception {

// Если звук включен

if (soundManager.isVolumeOn) {

soundManager.playClickSound(Boom\_Bomb, 0.4f); // Воспроизводим звук проигрыша

Thread.sleep(2700); // И ждем конца воспроизведения

// Иначе

} else {

Thread.sleep(2700); // Ждем

}

return null;

}

@Override

protected void done() {

dialog.dispose(); // Закрываем окно

New\_Game.setEnabled(true); // Разблокируем кнопку "Новая игра"

}

}.execute();

}

// Метод, обрабатывающий победу

private void gameWin() {

gameTimer.stop(); // Останавливаем игровой таймер

// Получаем время из текстового поля таймера

String elapsedTime = Timer.getText();

if (socket != null && socket.isConnected()) {

try {

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

String login = this.userLogin; // Текущий логин игрока

out.println("4"); // Отправляем код команды

out.println(login);

out.println(elapsedTime);

System.out.println("Отправлены данные о победе: " + login + " | " + elapsedTime);

} catch (IOException e) {

System.err.println("Ошибка отправки данных о победе: " + e.getMessage());

}

} else {

System.err.println("Соединение с сервером не установлено.");

}

// Загружаем изображение

ImageIcon icon = new ImageIcon("D:\\ClientSaper\\ClientSaper\\src\\main\\resources\\Death\_Lose.png");

Image img = icon.getImage();

// Создаём диалоговое окно

JDialog dialog = new JDialog((Frame) null, "");

dialog.setUndecorated(true); // Убираем рамки

dialog.setSize(300, 525);

dialog.getContentPane().setBackground(new Color(51, 51, 51));

dialog.setLayout(new BorderLayout());

dialog.setModal(true);

New\_Game.setFocusable(false); // Блокируем фокус кнопки новая игра

// Создаём кастомную панель для отрисовки изображения и текста

JPanel imagePanel = new JPanel() {

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

g.drawImage(img, 0, 0, getWidth(), getHeight(), this);

// Настройки текста

g.setFont(new Font("Franklin Gothic Medium", Font.BOLD, 16));

g.setColor(Color.WHITE);

FontMetrics fm = g.getFontMetrics();

String text = "В следующий раз ты от меня не уйдешь";

// Координаты для центрирования текста по горизонтали

int x = (getWidth() - fm.stringWidth(text)) / 2;

int y = getHeight() - 500;

g.drawString(text, x, y);

}

};

imagePanel.setPreferredSize(new Dimension(300, 525));

dialog.add(imagePanel, BorderLayout.CENTER);

dialog.setLocationRelativeTo(null);

// Если музыка включена

if (soundManager.isMusicOn)

{

soundManager.playClickSound(Win\_Music, 0.7f); // Воспроизводим победную музыку

}

// Таймер для закрытия окна через 5 секунд

new Timer(5000, e -> {

dialog.dispose();

New\_Game.setFocusable(true); // Восстанавливаем фокус кнопки "Новая игра"

}).start();

dialog.setVisible(true);

}

// Метод запуска окна

public static void main(String args[]) {

java.awt.EventQueue.invokeLater(() -> new GameForm("Пустышка").setVisible(true));

}

// Метода для корректного отображения времени в таблице на клиенте

private String convertToMMSS(String time) {

try {

// Разделяем время на часы, минуты и секунды

String[] parts = time.split(":");

if (parts.length == 3) {

int hours = Integer.parseInt(parts[0]);

int minutes = Integer.parseInt(parts[1]);

int seconds = Integer.parseInt(parts[2]);

// Преобразуем часы в минуты

minutes += hours \* 60;

// Форматируем время в ММ:СС

return String.format("%02d:%02d", minutes, seconds);

}

} catch (NumberFormatException e) {

System.err.println("Ошибка при преобразовании времени: " + time);

}

// Если что-то пошло не так, возвращаем исходное время

return time;

}

// Variables declaration - do not modify

private javax.swing.JLabel Bomb\_img;

private javax.swing.JLabel Flag\_img;

private javax.swing.JButton New\_Game;

private javax.swing.JButton Settings\_Button;

private javax.swing.JTextField Timer;

private javax.swing.JTable Top\_Table;

private javax.swing.JTextField Volume\_Bomb;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JScrollPane jScrollPane1;

}

## Приложение A.4. Файл «SoundManager.java»

package com.mycompany.clientsaper;

import java.util.\*;

import javax.sound.sampled.\*;

import java.io.File;

// Класс для управления звуками и музыкой в игре

public class SoundManager {

protected boolean isMusicOn = true; // Флаг включения/выключения музыки

protected boolean isVolumeOn = true; // Флаг включения/выключения звуковых эффектов

private final String MUSIC\_FOLDER; // Путь к папке с музыкальными файлами

private boolean isPlaying = false; // Флаг текущего состояния воспроизведения

private LinkedList<File> musicFiles = new LinkedList<>(); // Очередь музыкальных файлов

private boolean stopMusicRequested = false; // Флаг запроса на остановку музыки

private Clip clip; // Объект для воспроизведения аудио

private Thread musicThread; // Поток для циклического воспроизведения музыки

// Конструктор SoundManager

public SoundManager(String musicFolder) {

this.MUSIC\_FOLDER = musicFolder;

}

// Воспроизведение звукового эффекта

protected void playClickSound(String sound, float volumeFactor) {

// Если звук выключен, ничего не делаем

if (!isVolumeOn) return;

try {

// Загружаем звуковой файл

File soundFile = new File(sound);

AudioInputStream audioStream = AudioSystem.getAudioInputStream(soundFile);

Clip clip = AudioSystem.getClip();

clip.open(audioStream);

// Настраиваем громкость

FloatControl gainControl = (FloatControl) clip.getControl(FloatControl.Type.MASTER\_GAIN);

float dB = (float) (Math.log(volumeFactor) / Math.log(10) \* 20);

gainControl.setValue(dB);

// Запускаем воспроизведение

clip.start();

} catch (Exception e) {

e.printStackTrace();

}

}

// Загрузка музыкальных файлов из указанной папки

protected void loadMusicFiles() {

File folder = new File(MUSIC\_FOLDER);

// Проверяем существование папки

if (!folder.exists() || !folder.isDirectory()) {

return;

}

// Получаем все .wav файлы из папки

File[] files = folder.listFiles((dir, name) -> name.toLowerCase().endsWith(".wav"));

// Если файлов нет, выходим

if (files == null || files.length == 0) {

return;

}

// Сохраняем файлы и перемешиваем их

musicFiles = new LinkedList<>(Arrays.asList(files));

Collections.shuffle(musicFiles);

}

// Запуск циклического воспроизведения музыки

protected void startMusicLoop(float volumeFactor) {

// Проверяем условия для воспроизведения

if (!isMusicOn || isPlaying || musicFiles.isEmpty()) return;

// Сбрасываем флаг остановки и устанавливаем флаг воспроизведения

stopMusicRequested = false;

isPlaying = true;

// Создаем поток для воспроизведения музыки

musicThread = new Thread(() -> {

// Цикл воспроизведения треков

while (isMusicOn && !stopMusicRequested) {

File track = musicFiles.pollFirst();

if (track == null) break;

// Воспроизводим текущий трек

playTrack(track, volumeFactor);

// Возвращаем трек в конец очереди

musicFiles.addLast(track);

}

isPlaying = false;

});

musicThread.start();

}

// Воспроизведение одного музыкального трека

private void playTrack(File soundFile, float volumeFactor) {

try {

// Проверяем необходимость остановки

if (!isMusicOn || stopMusicRequested) return;

// Загружаем и открываем аудиофайл

AudioInputStream audioStream = AudioSystem.getAudioInputStream(soundFile);

clip = AudioSystem.getClip();

clip.open(audioStream);

// Настраиваем громкость

FloatControl gainControl = (FloatControl) clip.getControl(FloatControl.Type.MASTER\_GAIN);

float min = gainControl.getMinimum();

float max = gainControl.getMaximum();

float dB = (float) (Math.log(volumeFactor) / Math.log(10) \* 20);

dB = Math.max(min, Math.min(dB, max));

gainControl.setValue(dB);

// Запускаем воспроизведение

clip.start();

// Ожидаем завершения воспроизведения трека

synchronized (clip) {

clip.addLineListener(event -> {

if (event.getType() == LineEvent.Type.STOP) {

synchronized (clip) {

clip.notify();

}

}

});

clip.wait();

}

// Закрываем ресурсы

clip.close();

} catch (Exception e) {

e.printStackTrace();

}

}

// Остановка воспроизведения музыки

protected void stopMusic() {

// Устанавливаем флаги остановки

stopMusicRequested = true;

isMusicOn = false;

// Останавливаем и освобождаем ресурсы

if (clip != null && clip.isRunning()) {

clip.stop();

clip.close();

clip = null;

}

isPlaying = false;

}

// Геттер текущего состояния установки флага музыки

protected boolean isMusicOn() {

return isMusicOn;

}

// Сеттер установки флага активности музыки

protected void setMusicOn(boolean musicOn) {

isMusicOn = musicOn;

}

// Геттер текущего состояния установки флага звука

protected boolean isVolumeOn() {

return isVolumeOn;

}

// Сеттер установки флага активности звуа

protected void setVolumeOn(boolean volumeOn) {

isVolumeOn = volumeOn;

}

}

## Приложение A.5. Файл «ServerSaper.java»

package com.mycompany.serversaper;

import java.io.\*;

import java.net.\*;

import java.util.Set;

import java.util.List;

import java.nio.charset.StandardCharsets;

import java.util.concurrent.ConcurrentHashMap;

import java.util.concurrent.CopyOnWriteArrayList;

// Класс сервера

public class ServerSaper {

private static final int PORT = 1236; // Порт

private static final Set<ClientHandler> clients = ConcurrentHashMap.newKeySet(); // Подключенные клиенты

private static List<String> leaderboardList = new CopyOnWriteArrayList<>(); // Таблица лидеров

public static void main(String[] args) {

System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF\_8)); // Русификация

loadLeaderboardFromDatabase(); // Запрос к БД для получения таблицы лидеров

startLeaderboardUpdater(); // Обновление таблицы лидеров

startClientMonitor(); // Вывод подключенных клиентов

// Обработчик подключения

try (ServerSocket serverSocket = new ServerSocket(PORT)) {

System.out.println("Сервер запущен на порту " + PORT);

while (true) {

Socket clientSocket = serverSocket.accept();

ClientHandler clientHandler = new ClientHandler(clientSocket);

clients.add(clientHandler); // Добавляем нового клиента

clientHandler.start();

}

} catch (IOException e) {

e.printStackTrace();

}

}

// Поток для вывода списка подключенных клиентов раз в 20 секунд

private static void startClientMonitor() {

new Thread(() -> {

while (true) {

try {

Thread.sleep(20\_000); // Ждем 20 секунд

printConnectedClients(); // Выводим список подключенных пользователей

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

break;

}

}

}).start();

}

//Выводим список подключенных клиентов

private static void printConnectedClients() {

System.out.println("\nПодключенные клиенты:");

if (clients.isEmpty()) {

System.out.println("Нет активных подключений.");

} else {

for (ClientHandler client : clients) {

System.out.println("/ " + client.getClientAddress());

}

}

}

// Метод для удаления пользователя из списка подключенных клиентов при его отключении

static void removeClient(ClientHandler clientHandler) {

clients.remove(clientHandler);

}

// Загрузка leaderboard из БД

private static void loadLeaderboardFromDatabase() {

leaderboardList = Database.getLeaderboard(); // / Заполняем таблицу

System.out.println(" Таблица лидеров загружена из базы данных.");

logLeaderboard(); // Вывод таблицы

}

// Обновление leaderboard раз в 3 минуты

private static void startLeaderboardUpdater() {

new Thread(() -> {

while (true) {

try {

Thread.sleep(180\_000); // Ждем 3 минуты

leaderboardList = Database.getLeaderboard();

System.out.println(" Таблица лидеров обновлена.");

logLeaderboard(); // Вывод таблицы

broadcastLeaderboard(); // Отправляем актуальную таблицу лидеров всем подключенным клиентам

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

break;

}

}

}).start();

}

// Логирование текущего состояния leaderboard

private static void logLeaderboard() {

if (leaderboardList.isEmpty()) {

System.out.println("Таблица лидеров пустая!");

} else {

System.out.println("Текущая таблица лидеров:");

for (String entry : leaderboardList) {

System.out.println(" " + entry);

}

}

}

// Отправка актуального leaderboard всем клиентам

private static void broadcastLeaderboard() {

// Если пустая таблица

if (leaderboardList.isEmpty()) {

System.out.println("Нет данных для отправки. Таблица лидеров пустая.");

return;

}

// Формируем отправление таблицы

String leaderboardData = "LEADERBOARD:" + String.join(";", leaderboardList);

System.out.println("Подготовлены данные для отправки: " + leaderboardData);

// Отправляем всем подключенным пользователям

for (ClientHandler client : clients) {

client.sendMessage(leaderboardData);

System.out.println("Данные отправлены клиенту: " + client.getClientAddress());

}

}

// Добавление очередного пользователя в таблицу на сервере

public static String getLeaderboardData() {

return String.join(";", leaderboardList);

}

}

## Приложение A.6. Файл «Database.java»

package com.mycompany.serversaper;

import java.util.List;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.time.LocalTime;

import java.sql.Connection;

import java.util.Collections;

import java.sql.SQLException;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.time.format.DateTimeFormatter;

// Класс работы с БД

public class Database {

private static final String URL = "jdbc:mysql://localhost:3306/DatabaseSaper?useSSL=false&serverTimezone=UTC"; // Адресс БД

private static final String USER = "root"; // Логин админа

private static final String PASSWORD = "\_89Wersys"; // Пароль админа

private static final List<String> leaderboardData = Collections.synchronizedList(new ArrayList<>()); // Таблица лидеров

static {

// Проверка наличия драйвера

try {

Class.forName("com.mysql.cj.jdbc.Driver");

// Проверка соединения

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD)) {

System.out.println("Соединение с базой данных установлено.");

}

} catch (ClassNotFoundException e) {

System.err.println("Драйвер не найден: " + e.getMessage());

} catch (SQLException e) {

System.err.println("Ошибка соединения с базой данных: " + e.getMessage());

}

}

// Обработка регистрации в БД

protected static String saveUser(String name, String surname, String login, String email, String password) {

String checkSql = "SELECT MAX(id) FROM users"; // Запрос для получения максимального ID пользователя

String insertSql = "INSERT INTO users (id, name, surname, login, email, password) VALUES (?, ?, ?, ?, ?, ?)"; // Запрос для вставки данных нового пользователя

// Подключение к БД и выполнение запросов

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement checkStmt = conn.prepareStatement(checkSql);

ResultSet rs = checkStmt.executeQuery()) {

int newId = 1; // Начальное значение ID, если таблица пуста

// Если в таблице есть записи, увеличиваем максимальный ID на 1

if (rs.next()) {

newId = rs.getInt(1) + 1;

}

// Вставка данных нового пользователя

try (PreparedStatement insertStmt = conn.prepareStatement(insertSql)) {

insertStmt.setInt(1, newId); // Устанавливаем новый ID

insertStmt.setString(2, name); // Устанавливаем имя

insertStmt.setString(3, surname); // Устанавливаем фамилию

insertStmt.setString(4, login); // Устанавливаем логин

insertStmt.setString(5, email); // Устанавливаем email

insertStmt.setString(6, password); // Устанавливаем пароль

insertStmt.executeUpdate(); // Выполняем запрос на вставку

return "Регистрация прошла успешно!"; // Возвращаем сообщение об успешной регистрации

}

} catch (SQLException e) {

// Обработка ошибок при регистрации

if (e.getMessage().contains("Duplicate entry")) {

// Проверяем, какое поле вызвало ошибку дубликата

if (e.getMessage().contains("users.email")) {

return "Пользователь с такой электронной почтой уже зарегистрирован!";

} else if (e.getMessage().contains("users.login")) {

return "Пользователь с таким логином уже зарегистрирован!";

}

}

// Возвращаем общее сообщение об ошибке, если причина не в дубликате

return "Ошибка при регистрации! Попробуйте снова.";

}

}

// Обработка авторизации

protected static boolean checkUserCredentials(String login, String password) {

String sql = "SELECT COUNT(\*) FROM users WHERE login = ? AND password = ?"; // Запрос на получения логина и пароля

// Формируем подключение и отправляем запрос

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setString(1, login);

stmt.setString(2, password);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

return rs.getInt(1) > 0;

}

} catch (SQLException e) {

System.err.println("Ошибка при проверке авторизации: " + e.getMessage());

}

return false;

}

// Сохраняет результат победы игрока в базе данных

protected static void saveWinResult(String login, String time) {

String checkSql = "SELECT time FROM leaderboard WHERE login = ?"; // Проверка существования логина в таблице лидеров

String insertSql = "INSERT INTO leaderboard (login, time) VALUES (?, ?)"; // Добавление нового результата

String updateSql = "UPDATE leaderboard SET time = ? WHERE login = ?"; // Обновление результата(если уже был в таблице)

String countSql = "SELECT COUNT(\*) FROM leaderboard"; // Подсчет строк таблицы

String getSlowestSql = "SELECT id FROM leaderboard ORDER BY time DESC LIMIT 1"; // Поиск худшего результата

String deleteLastSql = "DELETE FROM leaderboard WHERE id = ?"; // Удаление записи

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement countStmt = conn.prepareStatement(countSql);

ResultSet rsCount = countStmt.executeQuery()) {

// Получаем текущее количество записей в таблице

int count = 0;

if (rsCount.next()) {

count = rsCount.getInt(1);

}

// Проверяем, есть ли уже запись для этого игрока

try (PreparedStatement checkStmt = conn.prepareStatement(checkSql)) {

checkStmt.setString(1, login);

ResultSet rs = checkStmt.executeQuery();

// Если запись существует, сравниваем время

if (rs.next()) {

String existingTime = rs.getString("time").trim();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss");

LocalTime existingLocalTime = LocalTime.parse(existingTime, formatter);

LocalTime newLocalTime = LocalTime.parse(time, formatter);

// Если новое время лучше, обновляем запись

if (newLocalTime.isBefore(existingLocalTime)) {

try (PreparedStatement updateStmt = conn.prepareStatement(updateSql)) {

updateStmt.setString(1, time);

updateStmt.setString(2, login);

updateStmt.executeUpdate();

}

}

// Если таблица заполнена (100 записей), находим худший результат

} else if (count >= 100) {

int slowestId = -1;

try (PreparedStatement getSlowestStmt = conn.prepareStatement(getSlowestSql);

ResultSet rsSlowest = getSlowestStmt.executeQuery()) {

if (rsSlowest.next()) {

slowestId = rsSlowest.getInt("id");

}

}

// Если нашли худший результат и новый лучше, заменяем

if (slowestId != -1) {

try (PreparedStatement deleteStmt = conn.prepareStatement(deleteLastSql)) {

deleteStmt.setInt(1, slowestId);

deleteStmt.executeUpdate();

}

// Добавляем новый результат

try (PreparedStatement insertStmt = conn.prepareStatement(insertSql)) {

insertStmt.setString(1, login);

insertStmt.setString(2, time);

insertStmt.executeUpdate();

}

}

// Если места достаточно, просто добавляем новый результат

} else {

try (PreparedStatement insertStmt = conn.prepareStatement(insertSql)) {

insertStmt.setString(1, login);

insertStmt.setString(2, time);

insertStmt.executeUpdate();

}

}

}

} catch (SQLException e) {

System.err.println("Ошибка при записи в leaderboard: " + e.getMessage());

}

}

// Обновляет данные таблицы лидеров в на сервере

protected static void updateLeaderboard() {

List<String> newLeaderboard = new ArrayList<>(); // Список для сохранения таблицы на сервере

String query = "SELECT login, time FROM leaderboard ORDER BY time ASC"; // Запрос для получения отсортированного списка лидеров

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement stmt = conn.prepareStatement(query);

ResultSet rs = stmt.executeQuery()) {

// Формируем список строк "логин - время"

while (rs.next()) {

String login = rs.getString("login");

String time = rs.getString("time");

newLeaderboard.add(login + " - " + time);

}

// Обновляем кеш данных

leaderboardData.clear();

leaderboardData.addAll(newLeaderboard);

} catch (SQLException e) {

System.err.println("Ошибка SQL: " + e.getMessage());

}

}

// Получает актуальную таблицу лидеров из базы данных

protected static List<String> getLeaderboard() {

List<String> leaderboard = new ArrayList<>();

String query = "SELECT login, time FROM leaderboard ORDER BY time ASC"; // Запрос для получения отсортированного списка лидеров

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement stmt = conn.prepareStatement(query);

ResultSet rs = stmt.executeQuery()) {

// Формируем список строк "логин - время"

while (rs.next()) {

String login = rs.getString("login");

String time = rs.getString("time");

leaderboard.add(login + " - " + time);

}

System.out.println("Данные из таблицы leaderboard успешно загружены.");

} catch (SQLException e) {

System.err.println("Ошибка при загрузке таблицы лидеров: " + e.getMessage());

}

return leaderboard;

}

}

## Приложение A.7. Файл «ClientHandler.java»

package com.mycompany.serversaper;

import java.io.\*;

import java.net.\*;

import java.util.List;

// Обработка клиента

class ClientHandler extends Thread {

private Socket clientSocket; // Сокет клиента

private BufferedReader in; // Отправляем

private PrintWriter out; // Получаем

public ClientHandler(Socket socket) {

this.clientSocket = socket; // Получаем сокет клиента

}

@Override

public void run() {

try {

in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream())); // Принимаем с клиента информацию

out = new PrintWriter(clientSocket.getOutputStream(), true); // Отправляем клиенту подтверждение получения

System.out.println("Клиент подключился: " + getClientAddress());

String command;

while ((command = in.readLine()) != null) { // Постоянно слушаем команды

System.out.println("Получена команда: " + command);

// Регистрация

if ("1".equals(command)) {

// Получаем регистрационные данных

System.out.println("Начался процесс регистрации");

String name = in.readLine();

String surname = in.readLine();

String login = in.readLine();

String email = in.readLine();

String password = in.readLine();

// Отправляем клиенту полученные данные

System.out.println("Получены регистрационные данные:");

System.out.println("Имя: " + name);

System.out.println("Фамилия: " + surname);

System.out.println("Логин: " + login);

System.out.println("Email: " + email);

System.out.println("Пароль: " + password);

// Сохранение данных в БД

String result = Database.saveUser(name, surname, login, email, password);

out.println(result); // Отправляем ответ клиенту

closeConnection(); // Закрываем соединение после регистрации

// Авторизация

} else if ("2".equals(command)) {

// Принимаем логин и пароль

System.out.println("Начался процесс авторизации");

String login = in.readLine();

String password = in.readLine();

// Отправляем обратно

System.out.println("Получены данные для авторизации:");

System.out.println("Логин: " + login);

System.out.println("Пароль: " + password);

// Проверяем логин и пароль в БД

boolean isValid = Database.checkUserCredentials(login, password);

// Если совпало

if (isValid) {

out.println("Авторизация успешна");

closeConnection(); // Закрываем соединение после авторизации

// Если не совпало

} else {

out.println("Ошибка: неверный логин или пароль");

closeConnection(); // Закрываем соединение после авторизации

}

// Запрос актуальной leaderboard от клиента

} else if ("3".equals(command)) {

System.out.println("Клиент запросил актуальную таблицу лидеров.");

sendLeaderboard(); // Отправляем таблицу лидеров клиенту

// Запрос обработки победной ситцации

} else if ("4".equals(command)) {

// Получаем логин и время

String login = in.readLine();

String time = in.readLine();

System.out.println("Игрок " + login + " победил с временем " + time);

Database.saveWinResult(login, "00:" + time); // Отправляем в БД

// Отключение клиента

} else if ("10".equals(command)) {

System.out.println("Клиент отправил команду отключения: " + getClientAddress());

break; // Выход из цикла -> закрываем соединение

} else {

System.out.println("Неизвестная команда от клиента: " + command);

out.println("Ошибка: неизвестная команда.");

}

}

} catch (IOException e) {

System.out.println("Ошибка при обработке клиента: " + getClientAddress());

} finally {

closeConnection(); // Закрываем соеденение

}

}

// Получение адреса клиента

public String getClientAddress() {

return clientSocket.getInetAddress().toString() + ":" + clientSocket.getPort();

}

// Метод для получения таблицы лидеров

private void sendLeaderboard() {

// Получаем данные из БД

List<String> leaderboard = Database.getLeaderboard();

// Проверяем, что в leaderboard

if (leaderboard.isEmpty()) {

System.out.println("Таблица лидеров пустая!");

} else {

System.out.println("Загруженная таблица лидеров:");

for (String entry : leaderboard) {

System.out.println(entry); // Логируем каждую строку

}

}

// Формируем строку для отправки

String leaderboardData = "LEADERBOARD:" + String.join(";", leaderboard);

// Логируем итоговую строку

System.out.println("Отправка данных клиенту " + getClientAddress() + ": " + leaderboardData);

// Отправляем данные клиенту

out.println(leaderboardData);

}

// Облегченная отправка ответа

public void sendMessage(String message) {

out.println(message);

}

// Метод закрытия соеденения

private void closeConnection() {

try {

clientSocket.close();

} catch (IOException ignored) {}

ServerSaper.removeClient(this);

}

}

**Приложение Б**

**UML-диаграммы**

**Приложение Б.1.**

**UML-диаграмма вариантов**

UML-диаграмма вариантов использования представлена на рисунке Б.1.

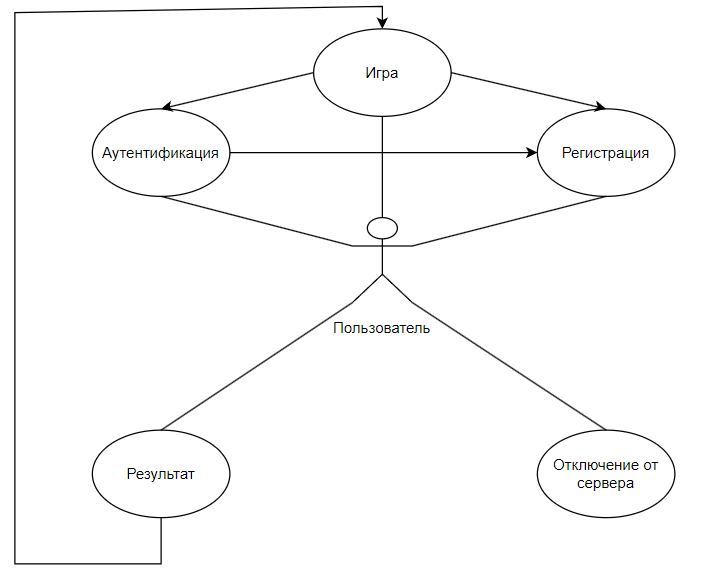


Рисунок Б.1. – UML-Диаграмма вариантов использования

## Приложение Б.2. UML- диаграмма последовательности

UML- диаграмма последовательности представлена на рисунке Б.2.

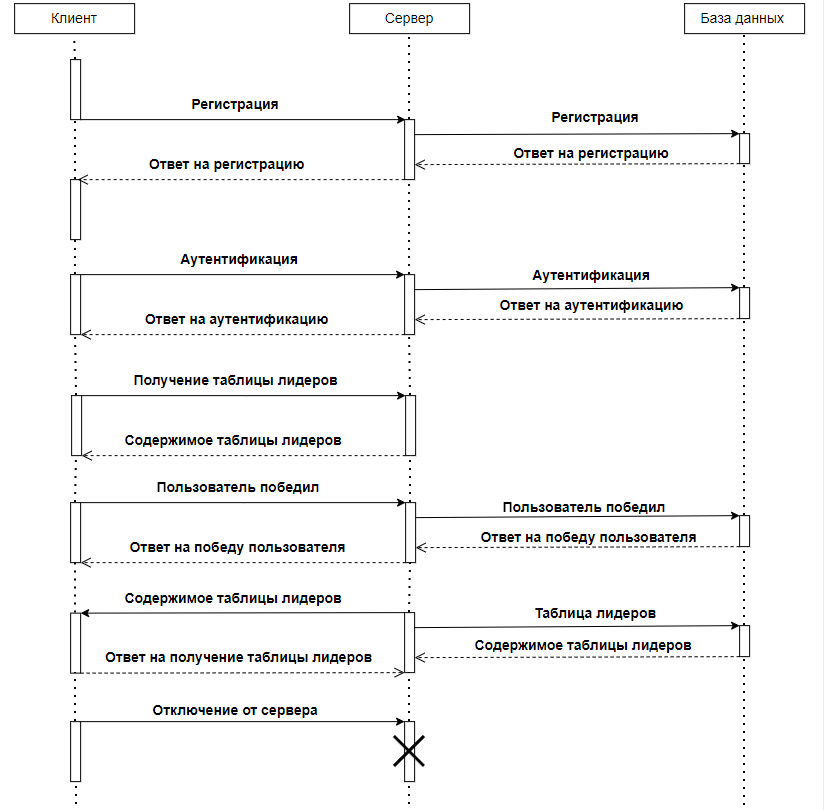


Рисунок Б.2. – UML-Диаграмма последовательности

## Приложение Б.3. UML- диаграмма деятельности

UML- диаграмма деятельности представлена на рисунке Б.3.

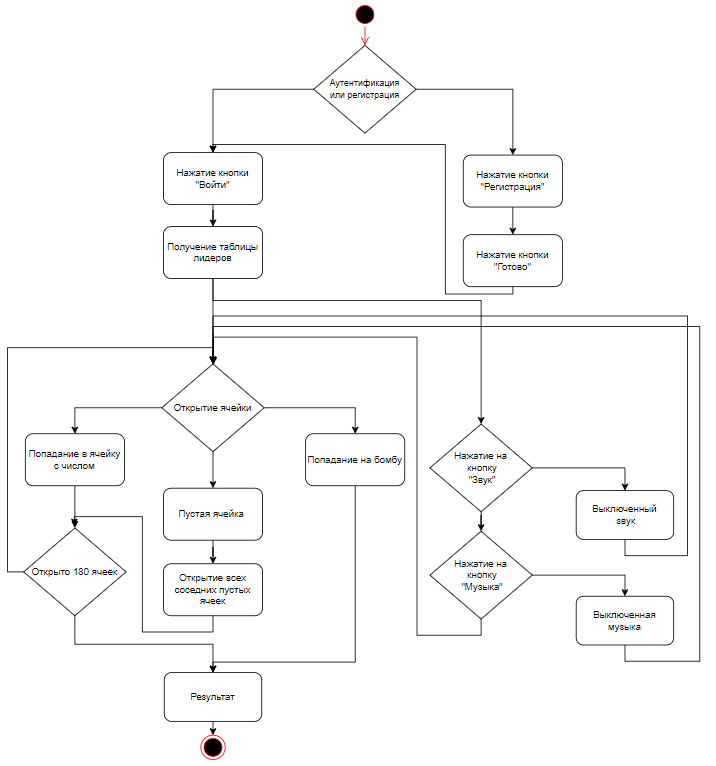


Рисунок Б.3. – UML-диаграмма деятельности

## Приложение Б.4. UML-диаграмма классов

UML-диаграмма развертывания представлена на рисунке Б.4.

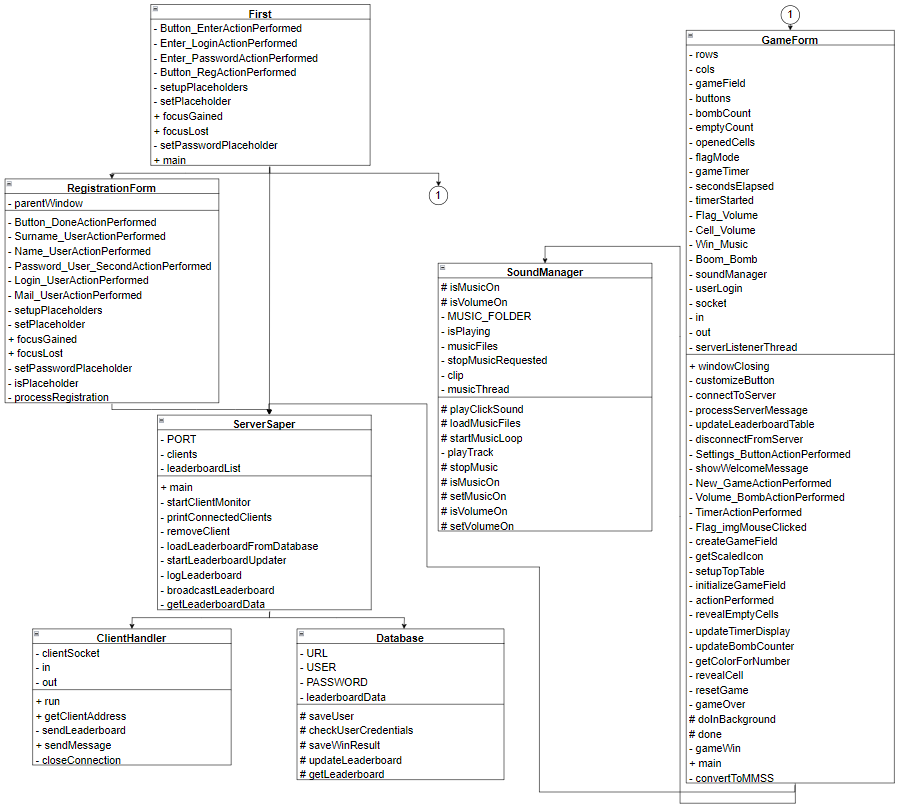


Рисунок Б.4. – UML-Диаграмма классов сервера и клиента.

## Приложение Б.5. UML-диаграмма развертывания

UML-диаграмма развертывания представлена на рисунке Б.5.

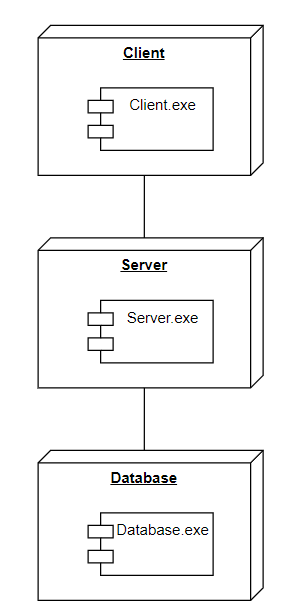


Рисунок Б.5. – UML-Диаграмма развертывания