Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по курсу «Программирование на языке Java»

на тему «Разработка модульного приложения “2048”»

Выполнил:

студент группы 19ВВ2

Отставнов А.М.

Принял:

Юрова О.В.

Пенза 2022

# СОДЕРЖАНИЕ

[**СОДЕРЖАНИЕ 4**](#_Toc104725858)

[**ВВЕДЕНИЕ 5**](#_Toc104725859)

[**1 ПОСТАНОВКА ЗАДАЧИ 6**](#_Toc104725860)

[**2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ ЗАДАНИЯ 7**](#_Toc104725861)

[**2.1 Среда разработки 7**](#_Toc104725862)

[**2.2 Игра «2048» 7**](#_Toc104725863)

[**3 ОПИСАНИЕ АЛГОРИТМА РАБОТЫ 9**](#_Toc104725864)

[**4 ОПИСАНИЕ ПРОГРАММЫ 10**](#_Toc104725865)

[**4.1 Сервер 10**](#_Toc104725866)

[**4.2 Клиент 11**](#_Toc104725867)

[**5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 18**](#_Toc104725868)

[**5.1 Сервер 18**](#_Toc104725869)

[**5.2 Запуск клиента 18**](#_Toc104725870)

[**5.3 Окно игры 19**](#_Toc104725871)

[**5.4 Окно с результатами 20**](#_Toc104725872)

[**ЗАКЛЮЧЕНИЕ 21**](#_Toc104725873)

[**СПИСОК ЛИТЕРАТУРЫ 22**](#_Toc104725874)

[**Приложение A. Листинг программы 23**](#_Toc104725875)

[**Приложение A.1. Файл «server.java» для Сервера 23**](#_Toc104725876)

[**Приложение A.2. Файл «toptable.java» для Клиента 31**](#_Toc104725877)

[**Приложение A.3. Файл «menu.java» для Клиента 32**](#_Toc104725878)

[**Приложение A.4. Файл «visual.java» для Клиента 39**](#_Toc104725879)

[**Приложение A.5. Файл «myException.java» для Клиента 47**](#_Toc104725880)

[**Приложение A.6. Файл «algorithm.java» для Клиента 48**](#_Toc104725881)

[**Приложение Б. UML-диаграммы 55**](#_Toc104725882)

[**Приложение Б.1. UML-диаграмма вариантов использования 55**](#_Toc104725883)

[**Приложение Б.2. UML-диаграмма классов 56**](#_Toc104725885)

[**Приложение Б.3. UML- диаграмма деятельности 57**](#_Toc104725886)

[**Приложение Б.4. UML-диаграмма последовательности 1 58**](#_Toc104725887)

[**Приложение Б.5. UML-диаграмма последовательности 2 59**](#_Toc104725888)

[**Приложение Б.6. UML-диаграмма развертывания 60**](#_Toc104725889)

# ВВЕДЕНИЕ

На сегодняшний момент язык Java является одним из самых распространенных и популярных языков программирования. Первая версия языка появилась еще в 1996 году в недрах компании Sun Microsystems, впоследствии поглощенной компанией Oracle. Язык Java активно применяется для создания программного обеспечения для множества устройств: обычных ПК, планшетов, смартфонов и мобильных телефонов и даже бытовой техники. Достаточно вспомнить популярность мобильной ОС Android, большинство программ для которой пишутся именно на Java.

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем этот байт-код выполняется виртуальной машиной JVM (Java Virtual Machine). В этом плане Java отличается от стандартных интерпретируемых языков как PHP или Perl, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как С или С++.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Mac OS и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код.

Java является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

# ПОСТАНОВКА ЗАДАЧИ

Основной задачей данного курсового проекта является программная реализация модульного приложения «2048». Язык программирования Java, среда разработки NetBeans. Приложение должно обладать графическим интерфейсом и использовать следующие технологии:

1. Java Collection Framework;

2. Механики обработки исключительных ситуаций;

3. Java Stream API;

4. Java Multithreading;

5. Сетевое взаимодействие.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ ЗАДАНИЯ

## **Среда разработки**

NetBeans IDE — [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [интегрированная среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) приложений на [языках программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) Java, Python, PHP, JavaScript, C, C++, [Ада](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) и ряда других. В нём содержится множество API для более легкой работы с окнами, действиями, файлами и т.п.

Последние версии NetBeans IDE поддерживают [рефакторинг](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3), [профилирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету и множество предопределённых шаблонов кода.

Для разработки программ в среде NetBeans и для успешной инсталляции и работы самой среды NetBeans должен быть предварительно установлен Sun [JDK](https://ru.wikipedia.org/wiki/JDK) или J2EE SDK подходящей версии. Среда разработки NetBeans

по-умолчанию поддерживала разработку для платформ J2SE и J2EE. Начиная с версии 6.0 NetBeans поддерживает разработку для мобильных платформ J2ME, [C++](https://ru.wikipedia.org/wiki/C%2B%2B) (только g++) и PHP без установки дополнительных компонентов.

Каждая отдельная возможность в приложение на Netbeans IDE может быть представлена отдельным модулем, которые, в свою очередь, сравнимы с плагином. Модуль Netbeans — это группа Java классов, которая предоставляет приложению определенный функционал.

## **Игра «2048»**

В каждом раунде появляется плитка номинала «2».

Нажатием стрелки [игрок](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%B0%D0%B3%D0%BE%D0%BD%D0%B8%D1%81%D1%82) может скинуть все плитки игрового поля в одну из 4 сторон. Если при сбрасывании две плитки одного номинала «налетают» одна на другую, то они превращаются в одну, номинал которой равен сумме соединившихся плиток. После каждого хода на свободной секции поля появляется новая плитка номиналом «2». Если при нажатии кнопки местоположение плиток или их номинал не изменится, то ход не совершается.

Если в одной строчке или в одном столбце находится более двух плиток одного номинала, то при сбрасывании они начинают соединяться с той стороны, в которую были направлены. Например, находящиеся в одной строке плитки (4, 4, 4) после хода влево превратятся в (8, 4), а после хода вправо — в (4, 8). Данная обработка неоднозначности позволяет более точно формировать стратегию игры.

За каждое соединение игровые очки увеличиваются на номинал получившейся плитки.

Игра заканчивается поражением, если после очередного хода невозможно совершить действие.

# ОПИСАНИЕ АЛГОРИТМА РАБОТЫ

* 1. **Описание клиент-серверной части**

Разработанная программа состоит из клиента и сервера.

Вначале нужно запустить сервер с уникальным адресом для обслуживания TCP/IP, определяемый комбинацией IP-адреса хоста с номером порта обслуживания, который создает конечную точку для обслуживания.

Далее сокет привязывается к локальной конечной точке. Чтобы сокет клиента мог идентифицировать потоковый сокет TCP, серверная программа присваивает имя своему сокету.

Затем сокет переходит в состояние прослушивания, в котором он будет ожидать входящие попытки соединения от клиентов. В случаи обнаружения входящего соединения, сервер предоставляет клиенту нить для обслуживания. На этом установление соединения между клиентом и сервером заканчивается.

После установления соединения сервер ожидает запросы от клиентов.

Как только клиенты отправляют запросы, сервер их обрабатывает и присылает ответы на данные запросы.

* 1. **Описание работы приложения**

Игра будет представлять собой набором картинок, которые будут меняться в зависимости от нажатой клавиши. Для отображения картинок будут отведены определенные координаты. Изменение будет происходить с помощью таймера, который будет обновлять картинки в определенных координатах.

# ОПИСАНИЕ ПРОГРАММЫ

## **Сервер**

Программа сервера является консольным приложением (рис. 1) и хранится в файле *server.java*. Данный файл содержит описание нити, которая будет создаваться при появлении нового клиента, создание сокета по которому будут подключаться клиенты, выделение нити клиента, обслуживание запросов клиента нитью. Пример работы сервера представлен на рисунке 1.

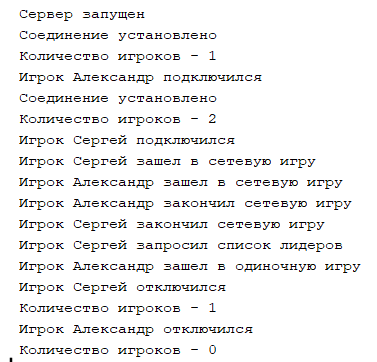


Рисунок 1 – Консольный вывод сервера.

Сервер обслуживает следующие запросы:

1. Запрос с кодом «0» - клиент отправляет запрос, что он будет играть в режим «Одиночной игры». Сервер принимает данный запрос, уменьшает переменную players, отвечающая за количество игроков, которые будут участвовать в многопользовательской игре.
2. Запрос с кодом «1» - клиент отправляет запрос, что он готов к многопользовательской игре. Сервер принимает данный запрос, захватывает мьютекс и ждет пока все игроки, которые будут играть в многопользовательском режиме, не захватят мьютекс. Во время ожидания проверяется, если мьютекс будет пересоздан. Тогда клиент захватит мьютекс заново.
3. Запрос с кодом «2» – клиент отправляет запрос на отправку результатов игры. Сервер принимает данный запрос и ожидает результат игры другого клиента, если такой есть. После получения добавляет результат игрока и его имя в файл, после чего захватывает мьютекс и ждет пока все игроки, которые будут играть в многопользовательском режиме, не захватят мьютекс. После этого каждая нить направляет своего клиента в главное меню, и проверяет все ли нити освободили мьютекс кроме нее. Если да, то эта нить очищает коллекцию.
4. Запрос с кодом «3» – клиент отправляет серверу запрос, что он хочет участвовать в сетевой игре. Если в это время игра уже идет, то идет ожидание окончания игры, после чего переменную players, отвечающая за количество игроков, которые будут участвовать в многопользовательской игре, увеличится и мьютекс будет пересоздан, а клиенту будет отправлено сообщение, о добавление его в участники.
5. Запрос с кодом «4» - клиент отправляет серверу запрос на просмотр списка лидеров. Сервер принимает данный запрос и извлекает информацию об игроках, набравших наибольшее количество очков в сетевой игре, после чего выводит на экран окно с таблицей лучших игроков.

## **Клиент**

Программа клиента разбита на несколько модулей. *menu.java* – главный файл проекта, файл *visual.java* – это отрисовка игры, файл *algorithm.java* – алгоритм игры, а файл *toptable.java* отвечает за вывод таблицы лидеров.

* + 1. **menu.java**

Данный файл отвечает за подключение к серверу, отправление запросов на и получение ответов от него, а также за запуск игры. При запуске данного файла появляется окно (рис. 2), в которой есть четыре кнопки:

1. «Одиночная игра»,
2. «Сетевая игра»,
3. «Подключится»,
4. «Список лидеров»

а также три поля для заполнения:

1. «Имя».
2. «Сервер»,
3. «Порт»,

Эти данные нужны для подключения к серверу. Если были введены неверные данные или сервер был выключен, то возникает ошибка и выводится окно с соответствующей надписью (рис. 3).

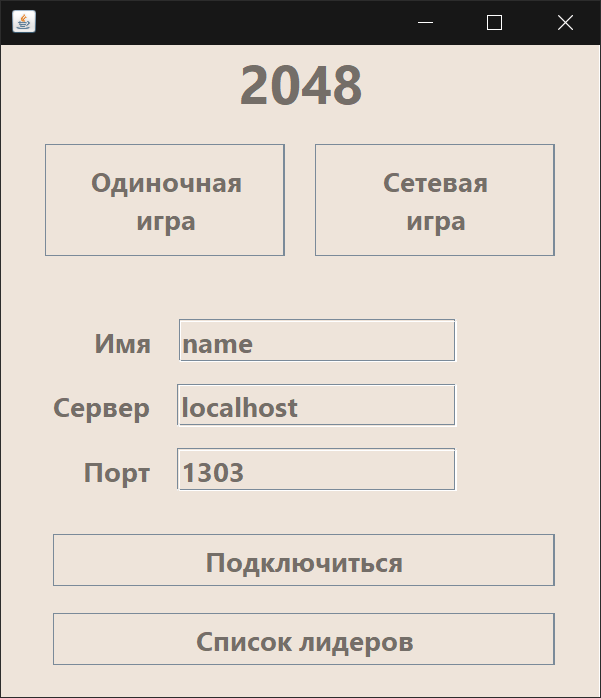


Рисунок 2 – Окно при запуске файла menu.java.

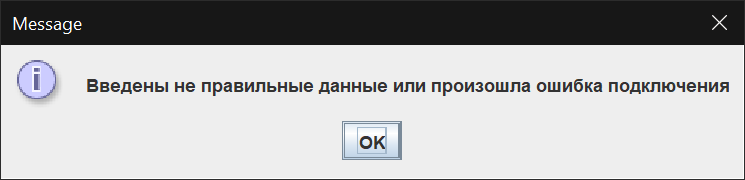


Рисунок 3 – Окно с ошибкой.

Для осуществления подключения к серверу надо нажать кнопку «Подключится». После этого данные из текстовых полей будут использованы для подключения. Далее сервер получит имя игрока, которое было введено в поле «Имя». После подключения будет создана нить, которая будет принимать сообщения от сервера. Нить обрабатывает следующие запросы от сервера:

1. Запрос с кодом «1» - означает, что все игроки готовы к игре. Тогда запускается файл *visual.java*, отправляя данному файлу фрейм текущего окна и цифру 1, означающая, что после игры надо отправить результаты серверу.
2. Запрос кодом «2» - означает, что результаты всех игроков отправлены. Нить получает результаты и имена всех игроков, добавляя эти данные в соответствующие коллекции, после чего данные добавляются в соответствующий файл, параллельно проходя сортировку по убыванию.
3. Запрос с кодом «3» - означает, что клиент был добавлен в участники. При нажатии кнопки «Одиночная игра», серверу отправляется сообщение с кодом «0», после чего запускается окно с игрой.
4. Запрос с кодом «4» - означает, что клиент запросил список лидеров. Сервер считывает из файла данные об игроках и набранных ими очках, после чего отправляет их отправляет их в файл *toptable.java*, который выведет их в виде нового окна.

При нажатии кнопки «Сетевая игра» серверу отправится один из двух запросов. Запрос с кодом «3», если игрок запустил одиночную игру или запрос с кодом «1». При отправке запроса с кодом «3» идет ожидание ответа от сервера, означающая что клиент был добавлен в участники. Если с кодом «1», то идет ожидания остальных игроков.

При ожидании других игроков после нажатия кнопки «Сетевая игра» будет выводится сообщение: «Ожидаем остальных игроков» (рис. 4).

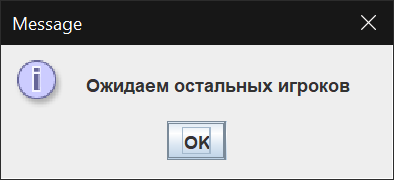


Рисунок 4 – Окно с сообщением.

Кроме этого, в данном файле присутствует функция *res*, отвечающая за отправку серверу количество набранных очков для игроков.

* + 1. **visual.java**

Данный файл отвечает за саму игру, включая ее отрисовку. При запуске происходит загрузка всех картинок, добавление их в коллекцию и отображение игрового поля (рис. 5). Также запускается таймер, который по истечению времени заново отрисовывает картинки. Картинка выбирается в зависимости от индекса, находящегося внутри массива Id класса *algorithm*. В начале игры два блока с числом «2» получают случайные координаты на игровом поле с помощью функции данного класса – *NewElement*. Остальные поля изначально заполнены пустыми блоками.

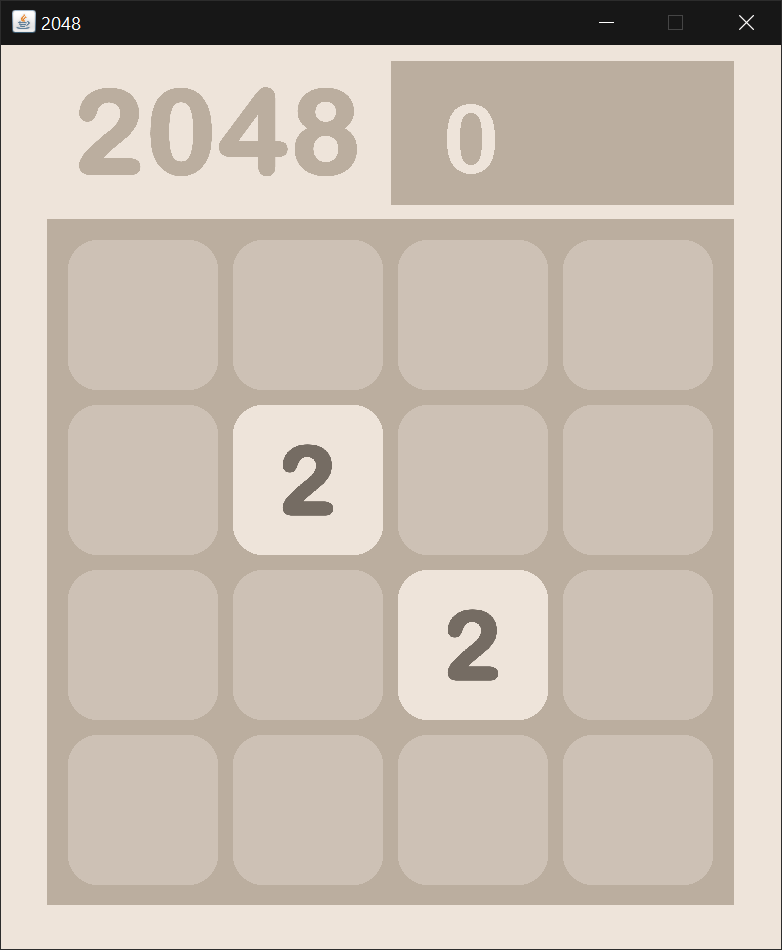


Рисунок 5 – Окно с игрой при запуске.

Вся игра строится вокруг класса *algorithm*. При запуске файла *visual.java* вызывается конструктор класса *algorithm*, в котором массив *Id* заполняется числом «0». Значение всех чисел указан ниже (рис. 6):

1. «0» - пустое поле
2. «1» - число 2;
3. «2» - число 4;
4. «3» - число 8;
5. «4» - число 16;
6. «5» - число 32;
7. «6» - число 64;
8. «7» - число 128;
9. «8» - число 256;
10. «9» - число 512;
11. «10» - число 1024;
12. «11» - число 2048;
13. «12» - число 4096;
14. «13» - число 8192;
15. «14» - число 16384.

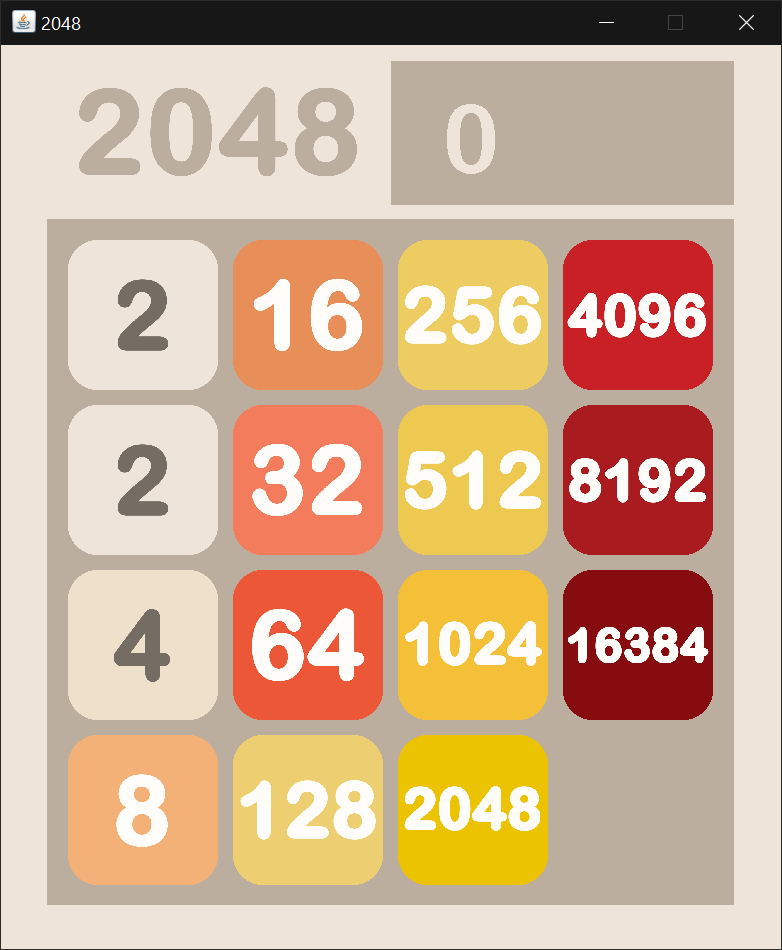


Рисунок 6 – Все доступные плитки.

Можно использовать одну из четырех кнопок: стрелка вверх, стрелка вниз, стрелка влево, стрелка вправо. При нажатии любой из этих кнопок происходит сдвиг блоков в соответствующую сторону одной из четырех функций внутри класса *algorithm.java*: *UP, DOWN, LEFT, RIGHT*. После выполнения сдвига случайным образом выбирается координата свободного блока, куда будет помещен блок «2», а также происходит тестовый сдвиг во все четыре стороны с помощью функции *test*, который проверяет возможность следующего хода. Если по результату не было сдвига ни в одну сторону, то у игрока больше нету ходов и игра окончена (рис. 7).

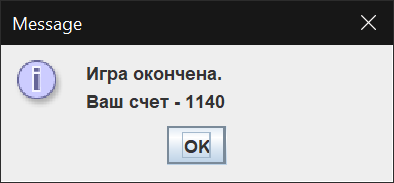


Рисунок 7 – Сообщение об окончании игры.

После окончания игры проверяется значение, которое было передано модулем *menu.java*. Если оно равно «1», то количество набранных очков надо отправить на сервер с помощью функции модуля *menu.java* - res. Если «0», то окно с игрой закрывается и модуль menu.java снова становится виден.

* + 1. **toptable.java**

Данный модуль отвечает за вывод окна в виде таблицы, которое отображает топ-10 игроков по набранным очкам в сетевой игре (рис. 8).



Рисунок 8 – Окно с таблицей лидеров.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## **Сервер**

Сервер не имеет графического интерфейса. Действия каждого клиента можно будет увидеть в консоли (рис. 1). Требуется только запустить сервер, дальше он сам будет принимать клиентов и обслуживать их.

## **Запуск клиента**

При запуске клиента первым окном является меню (рис. 9). В нем присутствует 4 кнопки:

1. Одиночная игра – отвечает за запуск одиночной версии игры;
2. Мультиплеер – отвечает за отправку серверу сообщения о готовности и запуска сетевой игры;
3. Подключится – отвечает за подключение клиента к серверу на основе данных из трех текстовых полей.
4. Список лидеров – отвечает за вывод окна в виде таблицы со списком игроков, набравших наибольшее количество очков в сетевой игре.

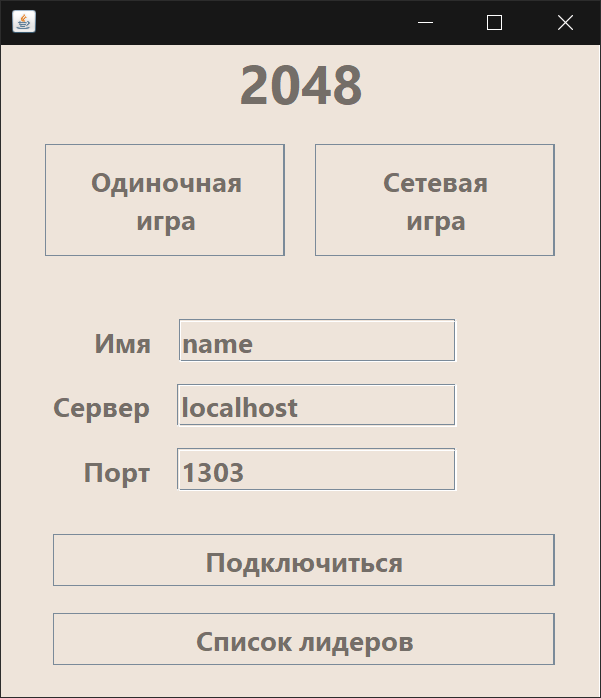


Рисунок 9 – Окно при запуске файла menu.java

## **Окно игры**

Данное окно является игрой «2048» (рис. 10). Что бы передвигать блоки с цифрами надо нажать одну из следующих кнопок:

1. Стрелка влево – сдвиг блоков влево.
2. Стрелка вверх – сдвиг блоков вверх;
3. Стрелка вправо - сдвиг блоков вправо;
4. Стрелка вниз – сдвиг блоков вниз;

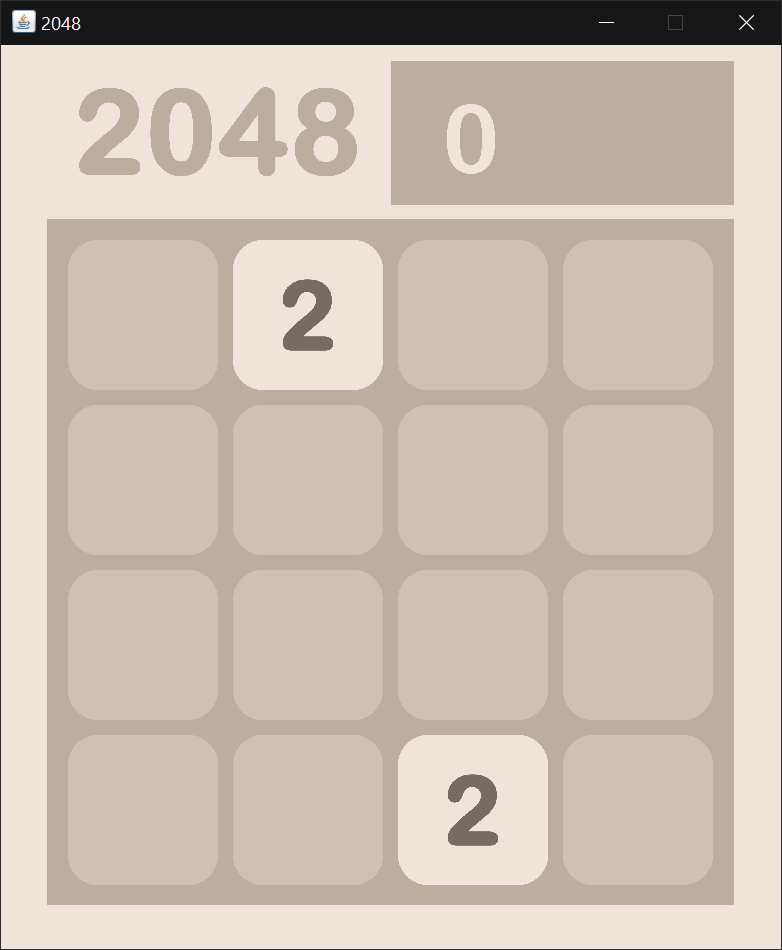


Рисунок 10 – Окно с игрой «2048».

## **Окно с результатами**

При нажатии на кнопку «Список лидеров» в главном меню открывается окно, отображающее имя и количество очков игроков, которые участвовали в сетевой игре и набрали наибольшее количество очков (рис. 11).



Рисунок 11 – Окно со списком лидеров

# ЗАКЛЮЧЕНИЕ

При выполнении данной курсовой работы были закреплены знания, полученные при прослушании курса и приобретены практические навыки разработки многомодульных приложений на языке Java.

В результате выполнения данного курсового проекта было разработано модульное приложение игры «2048», позволяющая проверить навыки пользователя в данной игре, а также предоставляющая возможность сравнивать свой результат с другими игроками с помощью клиент-серверной архитектуры.

# СПИСОК ЛИТЕРАТУРЫ

1. Игра «2048», <https://play2048.co/>
2. Swing. Руководство для начинающих / Герберт Шилдт. Пер. с англ. – М. : ООО «И.Д. Вильямс», 2007. – 704 с.
3. Java. Полное руководство, 10-е изд. / Герберт Шилдт. Пер. с англ. –СПб. : ООО «Альфа-книга», 2018. – 1488 с.
4. "Head First Java, Изучаем Java", Кэти Сьерра, Берт Бэйтс – ЭКСМО, 2012 – 708 с.
5. Java. Библиотека профессионала, том 1 / Кей Хорстманн, Гари Корнелл. Пер. с англ. –М. : ООО «И.Д. Вильямс», 2016. – 864 с.

# Приложение A. Листинг программы

## **Приложение A.1. Файл «server.java» для Сервера**

public class server {

static private Socket clientSocket; //сокет для общения

static private ServerSocket server; // серверсокет

static int NumberOfClients=0;//количество клиентов

static int players=0;//количество клиентов которые участвуют в игре

static Semaphore sem;

static Object lock=new Object();//объект синхронизации

static ArrayList<String> res=new ArrayList();

static ArrayList<String> nm=new ArrayList();

static int game=0;

static class Service extends Thread {

static class Record{

public Record(){

}

public Record(String n, int m){

name = n;

score = m;

}

public String name = "";

public int score = 0;

}

private BufferedReader in; // поток чтения из сокета

private BufferedWriter out; // поток записи в сокет\

Socket Client;

String word;

String name;

boolean wait;

Service(Socket Client){

this.Client=Client;

try {

System.out.println(" Соединение установлено");

out = new BufferedWriter(new OutputStreamWriter(Client.getOutputStream()));

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

} catch (IOException ex) {

System.out.println(" Ошибка при подключении ");

}

}

@Override

public void run(){

try {

int number;

word=in.readLine();//получаем

name=word;

System.out.println(" Игрок "+name+" подключился ");

while(true){

word=in.readLine();

switch(Integer.parseInt(word)){

case 0://пользователь играет один

System.out.println(" Игрок "+name+" зашел в одиночную игру");

synchronized(lock){

players--;

sem=new Semaphore(players);

}

break;

case 1:

try {

System.out.println(" Игрок "+name+" зашел в сетевую игру");

synchronized(lock){

if(game!=1)

game=1;

}

number=players;//запоминаем количество игроков

sem.acquire();

while(sem.availablePermits()!=0){

if(number!=players){

sem.acquire();

number=players;

}//если семафор был пересоздан то захватываем заново

}

this.sleep(10);

out.write("1\n");

out.flush();

sem.release();

} catch (InterruptedException ex) {

Logger.getLogger(server.class.getName()).log(Level.SEVERE, null, ex);

}

break;

case 2://пользователь готов получить результат

try {

//клиент готов к игре онлайн

word=in.readLine();//получаем результат игрока

System.out.println(" Игрок "+name+" закончил сетевую игру");

ArrayList<Record> mas = new ArrayList<>();

Record temp = new Record();

Boolean flag = false;

try(FileReader reader = new FileReader("toplist.txt"))

{

BufferedReader buf = new BufferedReader(reader);

temp.name = buf.readLine();

while( temp.name != null && !temp.name.equals("") && mas.size() < 10){

temp.score = Integer.valueOf(buf.readLine());

if ( Integer.valueOf( word ) > temp.score && !flag)

{

Record rec = new Record();

rec.name = name;

rec.score = Integer.valueOf(word);

mas.add(rec);

flag = true;

}

if ( mas.size() < 10 )

{

mas.add(new Record(temp.name, temp.score));

}

temp.name = buf.readLine();

}

}

catch(IOException ex){

}

if ( mas.size() < 10 && !flag )

{

Record rec = new Record();

rec.name = name;

rec.score = Integer.valueOf(word);

mas.add(rec);

}

try(PrintWriter writer = new PrintWriter("toplist.txt"))

{

for( var i = 0; i < mas.size(); i++ )

{

writer.write( mas.get(i).name + "\n" + mas.get(i).score + "\n" );

}

writer.flush();

}

catch(IOException ex){

}

synchronized(lock){

res.add(word);

nm.add(name);

}

sem.acquire();

while(sem.availablePermits()!=0){

}

Service.sleep(10);

out.write("2\n");

out.flush();

res.stream().forEach(s->{

try {

out.write(s+"\n");

out.flush();

} catch (IOException ex) {

Logger.getLogger(server.class.getName()).log(Level.SEVERE, null, ex);

}

});

out.write("0\n");

out.flush();

nm.stream().forEach(s->{

try {

out.write(s+"\n");

out.flush();

} catch (IOException ex) {

Logger.getLogger(server.class.getName()).log(Level.SEVERE, null, ex);

}

});

out.write("0\n");

out.flush();

if(sem.availablePermits()==players-1){

synchronized(lock){

if(game!=0)

game=0;

if(res.size()!=0){

res=new ArrayList();

nm=new ArrayList();

}

}

}

sem.release();

} catch (InterruptedException ex) {

ex.fillInStackTrace();

Logger.getLogger(server.class.getName()).log(Level.SEVERE, null, ex);

}

break;

case 3://пользователь хочет играть онлайн

while(game!=0){

try {

Service.sleep(10);

} catch (InterruptedException ex) {

Logger.getLogger(server.class.getName()).log(Level.SEVERE, null, ex);

}

}

synchronized(lock){

players++;

sem=new Semaphore(players);

}

out.write("3\n");

out.flush();

break;

case 4:

out.write("4\n");

System.out.println(" Игрок "+name+" запросил список лидеров");

try(FileReader reader = new FileReader("toplist.txt"))

{

BufferedReader buf = new BufferedReader(reader);

String str;

str = buf.readLine();

while( str != null ){

out.write( str + "\n" );

str = buf.readLine();

}

out.write("0\n");

out.flush();

}

catch(IOException ex){

System.out.println(ex.getMessage());

}

break;

}

}

} catch (IOException ex) {

System.out.println(" Игрок " + name + " отключился");

NumberOfClients--;

players--;

sem=new Semaphore( players);

System.out.println(" Количество игроков - "+NumberOfClients);

}

}

}

public static void main(String[] args) {

// TODO code application logic here

System.out.println(" Сервер запущен");

try {

server = new ServerSocket(1303); // серверсокет прослушивает порт 1303

while(true){

clientSocket = server.accept(); // accept() будет ждать пока

NumberOfClients++;

sem=new Semaphore(NumberOfClients);

players++;

new Service(clientSocket).start();

System.out.println(" Количество игроков - "+NumberOfClients);

}

} catch (IOException ex) {

System.out.println(" Произошла ошибка подключения игрока");

}

}

}

## **Приложение A.2. Файл «toptable.java» для Клиента**

public class toptable extends javax.swing.JFrame {

public static String s;

public toptable( String str ) {

initComponents();

s = str; }

public void view(){

DefaultTableModel model = (DefaultTableModel) jTable1.getModel();

String[] str = s.split(" ");

for (int i = 0; i < str.length; i+=2) {

model.addRow(new Object[]{str[i], str[i+1]}); } }

private void initComponents() {

}// </editor-fold>

public static void main() {

try { for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Metal".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break; } }

} catch (ClassNotFoundException | InstantiationException | IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(toptable.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

toptable kekW = new toptable(s);

kekW.setVisible(true);

kekW.view();

} }); } }

## **Приложение A.3. Файл «menu.java» для Клиента**

public class menu extends javax.swing.JFrame {

toptable tb = new toptable("");

public menu() {

initComponents();

}

@SuppressWarnings("unchecked")

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

private void initComponents() {

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

try{

if (game!=0){throw new myException("Ожидаем остальных игроков");}

if(clientSocket!=null){

try {

rejim=1;

out.write("0\n");

out.flush();

} catch (IOException ex) {

Logger.getLogger(menu.class.getName()).log(Level.SEVERE, null, ex);

}}

setVisible(false);

visual.main(this,0);

}catch (myException ex) {

JOptionPane.showMessageDialog(null, ex.getMessage());

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

try{

if(rejim==1){

try {

rejim=0;

game=1;

out.write("3\n");

out.flush();

} catch (IOException ex) {

Logger.getLogger(menu.class.getName()).log(Level.SEVERE, null, ex);

}

throw new myException( "Ожидаем ответа от сервера");

}

if (game==1) {throw new myException( "Ожидаем остальных игроков");}

if(clientSocket==null){throw new myException( "Отсутствует подключение к серверу");}

try {

out.write("1\n");

out.flush();

} catch (IOException ex) {

Logger.getLogger(menu.class.getName()).log(Level.SEVERE, null, ex);

}

game=1;

}catch (myException e){

JOptionPane.showMessageDialog(null, e.getMessage());

return;

}// TODO add your handling code here:

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

try{

if(clientSocket!=null){throw new myException("Вы уже подключены к серверу");}

try {

clientSocket = new Socket( jTextField2.getText(),Integer.parseInt( jTextField3.getText()));

name=jTextField1.getText();

cl=new Client(this);

cl.start();

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

out = new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream()));

out.write(name+"\n");

out.flush();

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Введены не правильные данные или произошла ошибка подключения");

return;

}

// jLabel4.setText("Соединение:присутствует");

}catch (myException ex) {

JOptionPane.showMessageDialog(null, ex.getMessage());

return;

} // TODO add your handling code here:

}

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

try {

out = new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream()));

out.write("4\n");

out.flush();

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Введены не правильные данные или произошла ошибка подключения");

}

// TODO add your handling code here:

}

public void res(int score){

try {

out.write("2\n");

out.flush();

out.write(score+"\n");

out.flush();

} catch (IOException ex) {

Logger.getLogger(menu.class.getName()).log(Level.SEVERE, null, ex);

}

}

Client cl;

Socket clientSocket=null;

String name;

int game=0;

int rejim=0;

BufferedReader in; // поток чтения из сокета

BufferedWriter out;

class Client extends Thread {

ArrayList<String> res=new ArrayList();

ArrayList<String> nm=new ArrayList();

menu men;

BufferedWriter out;

String message;

String sss;

Client(menu men){

this.men=men;

try {

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

} catch (IOException ex) {

Logger.getLogger(menu.class.getName()).log(Level.SEVERE, null, ex);

}

}

@Override

public void run(){

try {

System.out.println("Успешное подключение к серверу");

while(true){

message=in.readLine();

switch(Integer.parseInt(message)){

case 1:

setVisible(false);

visual.main(men,1);

break;

case 2:

res=new ArrayList();

nm=new ArrayList();

message=in.readLine();

while(Integer.parseInt(message)!=0){

res.add(message);

message=in.readLine();

}

message=in.readLine();

while(message.equals("0")==false){

nm.add(message);

message=in.readLine();

}

for(int i=0;i<res.size();i++){

for(int j=0;j<res.size()-1;j++){

if(Integer.parseInt(res.get(j))<Integer.parseInt(res.get(j+1))){

String promej=res.get(j);

res.set(j, res.get(j+1));

res.set(j+1,promej);

promej=nm.get(j);

nm.set(j, nm.get(j+1));

nm.set(j+1,promej);

}

}

}

setVisible(false);

men.setVisible(true);

game=0;

break;

case 3:

game=0;

case 4:

// ImageIcon icon= new ImageIcon("src/resources/1.png");

sss = in.readLine();

String str = "";

while( sss.equals("") == false && sss.equals( "0") == false )

{

str += sss + " " + in.readLine() + " ";

sss = in.readLine();

}

tb = new toptable(str);

toptable.main() ;

break; } }

} catch (IOException ex) {

JOptionPane.showMessageDialog(null, "Вы были отключены от сервера");

} } }

public static void main(String args[]) {

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Metal".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

} } } catch (ClassNotFoundException | InstantiationException | IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(menu.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

java.awt.EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

new menu().setVisible(true);

} }); }}

## **Приложение A.4. Файл «visual.java» для Клиента**

public class visual {

static int up,down,left,right,shet,res,end,score=0;

static algorithm GM=new algorithm();

static ImageFrame IT;

static int rej;

static menu copy;

public static void main(menu args,int rejim) {

EventQueue.invokeLater(() -> {

rej=rejim;

GM=new algorithm();

copy=args;

ImageFrame frame = new ImageFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

GM.NewElement();

GM.NewElement();

}); }

static class ImageFrame extends JFrame

{

ImageComponent component = new ImageComponent();

public ImageFrame()

{

setTitle("2048");

setSize(DEFAULT\_WIDTH, DEFAULT\_HEIGHT);

setResizable(false);

add(component);

control();

IT=this;

}

private void control(){

addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e){

if (e.getKeyCode()==40){//стрелка вниз

if(left==0 & right==0 & up==0)

down=1;

}

if (e.getKeyCode()==39){//стрелка вправо

if(left==0 & up==0 & down==0)

right=1;

}

if (e.getKeyCode()==38){//стрелка вверх

if(left==0 & right==0 & down==0)

up=1;

}

if (e.getKeyCode()==37){//стрелка влево

if(right==0 & up==0 & down==0)

left=1;

} }

}

);

}

public static final int DEFAULT\_WIDTH = 535;

public static final int DEFAULT\_HEIGHT = 640;

@Override

public void hide(){

}

}

static class ImageComponent extends JComponent

{

ArrayList<Image> image=new ArrayList();

Image fon;

int \_2048;

public ImageComponent()

{

this.\_2048 = 0;

// Получаем изображения.

try

{

fon=ImageIO.read(new File("src/resources/back.png"));

image.add(ImageIO.read(new File("src/resources/0.png")));

image.add(ImageIO.read(new File("src/resources/2.png")));

image.add(ImageIO.read(new File("src/resources/4.png")));

image.add(ImageIO.read(new File("src/resources/8.png")));

image.add(ImageIO.read(new File("src/resources/16.png")));

image.add(ImageIO.read(new File("src/resources/32.png")));

image.add(ImageIO.read(new File("src/resources/64.png")));

image.add(ImageIO.read(new File("src/resources/128.png")));

image.add(ImageIO.read(new File("src/resources/256.png")));

image.add(ImageIO.read(new File("src/resources/512.png")));

image.add(ImageIO.read(new File("src/resources/1024.png")));

image.add(ImageIO.read(new File("src/resources/2048.png")));

image.add(ImageIO.read(new File("src/resources/4096.png")));

image.add(ImageIO.read(new File("src/resources/8192.png")));

image.add(ImageIO.read(new File("src/resources/16384.png")));

}

catch(IOException e)

{

}

Timer timer;

timer = new Timer(20, (ActionEvent e) -> {

//GM.NewElement();

if(up==1){//если была нажиты кнопка вверх

res+=GM.UP();//двигаем вверх и получаем информацию о сдвигах

shet++;//увеличиваем счетчик сдвигов

if(shet==3){//три раза сдвинули?

up=0;//если да то заканчиваем сдвигать

shet=0;//обнуляем счетчик

if(res!=0){//если сдвинули

end=GM.NewElement();

res=GM.test();

if(res==0){JOptionPane.showMessageDialog

(null, "Игра окончена.\nВаш счет - "+GM.score);

if(rej==1){copy.res(GM.score);}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res=0;

} } }

if(down==1){

res+=GM.DOWN();

shet++;//увеличиваем счетчик сдвигов

if(shet==3){//три раза сдвинули?

down=0;//если да то заканчиваем сдвигать

shet=0;//обнуляем счетчик

if(res!=0){//если сдвинули

end=GM.NewElement();

res=GM.test();

if(res==0){JOptionPane.showMessageDialog(null, "Игра окончена.\nВаш счет - "+GM.score);

if(rej==1){copy.res(GM.score);}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res=0;

}

}

}

if(left==1){

res+=GM.LEFT();

shet++;//увеличиваем счетчик сдвигов

if(shet==3){//три раза сдвинули?

left=0;//если да то заканчиваем сдвигать

shet=0;//обнуляем счетчик

if(res!=0){//если сдвинули

end=GM.NewElement();

res=GM.test();

if(res==0){JOptionPane.showMessageDialog(null, "Игра окончена.\nВаш счет - "+GM.score);

if(rej==1){copy.res(GM.score);}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res=0;

}

}

}

if(right==1){

res+=GM.RIGHT();

shet++;//увеличиваем счетчик сдвигов

if(shet==3){//три раза сдвинули?

right=0;//если да то заканчиваем сдвигать

shet=0;//обнуляем счетчик

if(res!=0){//если сдвинули

end=GM.NewElement();

res=GM.test();

if(res==0){JOptionPane.showMessageDialog(null, "Игра окончена.\nВаш счет - "+GM.score);

if(rej==1){copy.res(GM.score);}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res=0;

}

}

}

repaint(); //To change body of generated methods, choose Tools | Templates.

} //запускаем таймер.

);

timer.start();

}

@Override

public void paintComponent(Graphics g)

{

if(image == null) return;

int imageWidth = 110;

int imageHeight = 110;

try {

InputStream inputStream = new BufferedInputStream(new FileInputStream("src/resources/arial.ttf"));

Font font = Font.createFont(Font.TRUETYPE\_FONT, inputStream);

Color txt = new Color(238,228,218);

g.setFont(font.deriveFont(Font.BOLD,63));

g.setColor(txt);

} catch (FontFormatException | IOException e) {

}

g.drawImage(fon,0, 0,520,650,null);

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

for(int j = 0; j<= 3; j++){

switch(GM.Id[i][j]){

case(0) -> g.drawImage(image.get(0),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(1) -> g.drawImage(image.get(1),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(2) -> g.drawImage(image.get(2),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(3) -> g.drawImage(image.get(3),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(4) -> g.drawImage(image.get(4),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(5) -> g.drawImage(image.get(5),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(6) -> g.drawImage(image.get(6),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(7) -> g.drawImage(image.get(7),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(8) -> g.drawImage(image.get(8),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(9) -> g.drawImage(image.get(9),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(10) -> g.drawImage(image.get(10),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(11) -> g.drawImage(image.get(11),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(12) -> g.drawImage(image.get(12),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(13) -> g.drawImage(image.get(13),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

case(14) -> g.drawImage(image.get(14),(i \* imageWidth)+45, (j \* imageHeight)+130,100,100, null);

}

g.drawString(""+GM.score, 295, 85);

}

} } }

## **Приложение A.5. Файл «myException.java» для Клиента**

package com.mycompany.THE\_GAME;

class myException extends Exception{

myException(String s) {

super(s);

}

}

## **Приложение A.6. Файл «algorithm.java» для Клиента**

package com.mycompany.THE\_GAME;

public class algorithm{

public int[][] Id = new int[4][4];

int max=1;

public int score=0;

algorithm(){//заполняем поле квадратами

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

Id[i][j]=0;

}

}

}

public int NewElement(){

int find=0;

int [][] popit=new int[4][4];

for(int i=0; i<4; i++){

for(int j=0; j<4; j++){

popit[i][j]=0; } }

while(find<15){

int a = 0 + (int) (Math.random() \* 4);

int b = 0 + (int) (Math.random() \* 4);

if(Id[a][b]==0){

Id[a][b]=1;

break;

}

else{

if(popit[a][b]==0){

popit[a][b]=1;

find++; } } }

return find;

}

public int UP(){//сдвиг блоков вверх

int sdvig=0;

for(int i=0;i<4;i++)

for(int j=0;j<4;j++){

if(j!=3 && Id[i][j]==Id[i][j+1] && Id[i][j]!=0){//проверяем можем ли объединить

Id[i][j]++;

score+=Math.pow(2, Id[i][j]);

if(Id[i][j]>max)max=Id[i][j];

Id[i][j+1]=0;

sdvig++;

}

if(j!=0 && Id[i][j]!=0 && Id[i][j-1]==0){

sdvig++;//проверяем можно ли сдвинуть

Id[i][j-1]=Id[i][j];

Id[i][j]=0;

}

}

return sdvig;

}

public int DOWN(){//сдвиг блоков вниз

int sdvig=0;

int chet=0;

for(int i=0;i<4;i++){

for(int j=3;j>-1;j--){

if(j!=0 && Id[i][j]==Id[i][j-1] && Id[i][j]!=0){//проверяем можем ли объединить

Id[i][j]++;

score+=Math.pow(2, Id[i][j]);

if(Id[i][j]>max)max=Id[i][j];

Id[i][j-1]=0;

sdvig++;

}

if(3!=j && Id[i][j]!=0 && Id[i][j+1]==0){

sdvig++;//проверяем можно ли сдвинуть

Id[i][j+1]=Id[i][j];

Id[i][j]=0; } } }

chet++;

return sdvig;

}

public int LEFT(){//сдвиг блоков влево

int sdvig=0;

int chet=0;

for(int j=0;j<4;j++){

for(int i=0;i<4;i++){

if(i!=3 && Id[i+1][j]==Id[i][j] && Id[i][j]!=0){//проверяем можем ли объединить

Id[i][j]++;

score+=Math.pow(2, Id[i][j]);

if(Id[i][j]>max)max=Id[i][j];

Id[i+1][j]=0;

sdvig++;

}

if(0!=i && Id[i][j]!=0 && Id[i-1][j]==0){

sdvig++;//проверяем можно ли сдвинуть

Id[i-1][j]=Id[i][j];

Id[i][j]=0; } } }

chet++;

return sdvig;

}

public int RIGHT(){//сдвиг блоков вправо

int sdvig=0;

int chet=0;

for(int j=0;j<4;j++){

for(int i=3;i>-1;i--){

if(i!=0 && Id[i-1][j]==Id[i][j] && Id[i][j]!=0){//проверяем можем ли объединить

Id[i][j]++;

score+=Math.pow(2, Id[i][j]);

if(Id[i][j]>max)max=Id[i][j];

Id[i-1][j]=0;

sdvig++;

}

if(3!=i && Id[i][j]!=0 && Id[i+1][j]==0){

sdvig++;//проверяем можно ли сдвинуть

Id[i+1][j]=Id[i][j];

Id[i][j]=0; } } }

chet++;

return sdvig;

}

public int test(){//данная функция проверяет смодет ли пользователь сделать шаг?если нет то пользователь проиграл

int[][] IdCopy=new int[4][4];

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

IdCopy[i][j]=Id[i][j];

}

}//копируем массив с элементами

int result=0;//проверяем получится ли сдвинуть

for(int i=0;i<4;i++){//проверяем сдвиг на вверх

for(int j=0;j<4;j++){

if(j!=3 && IdCopy[i][j]==IdCopy[i][j+1] && IdCopy[i][j]!=0){//проверяем можем ли объединить

IdCopy[i][j]++;

IdCopy[i][j+1]=0;

result++;

}

if(j!=0 && IdCopy[i][j]!=0 && IdCopy[i][j-1]==0){

IdCopy[i][j-1]=IdCopy[i][j];

IdCopy[i][j]=0;

result++;

}

}

}//конец проверки

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

IdCopy[i][j]=Id[i][j];

}

}

for(int i=0;i<4;i++){//проверяем на сдвиг вниз

for(int j=3;j>-1;j--){

if(j!=0 && IdCopy[i][j]==IdCopy[i][j-1] && IdCopy[i][j]!=0){//проверяем можем ли объединить

IdCopy[i][j]++;

IdCopy[i][j-1]=0;

result++;

}

if(3!=j && IdCopy[i][j]!=0 && IdCopy[i][j+1]==0){

result++;//проверяем можно ли сдвинуть

IdCopy[i][j+1]=IdCopy[i][j];

IdCopy[i][j]=0; } } }

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

IdCopy[i][j]=Id[i][j];

}

}

for(int j=0;j<4;j++){//проверяем можем ли сдвинуть влево

for(int i=0;i<4;i++){

if(i!=3 && IdCopy[i+1][j]==IdCopy[i][j] && IdCopy[i][j]!=0){//проверяем можем ли объединить

IdCopy[i][j]++;

IdCopy[i+1][j]=0;

result++;

}

if(0!=i && IdCopy[i][j]!=0 && IdCopy[i-1][j]==0){

result++;//проверяем можно ли сдвинуть

IdCopy[i-1][j]=IdCopy[i][j];

IdCopy[i][j]=0; } } }

for(int i=0;i<4;i++){

for(int j=0;j<4;j++){

IdCopy[i][j]=Id[i][j];

}

}

for(int j=0;j<4;j++){

for(int i=3;i>-1;i--){

if(i!=0 && IdCopy[i-1][j]==IdCopy[i][j] && IdCopy[i][j]!=0){//проверяем можем ли объединить

IdCopy[i][j]++;

IdCopy[i-1][j]=0;

result++;

}

if(3!=i && IdCopy[i][j]!=0 && IdCopy[i+1][j]==0){

result++;//проверяем можно ли сдвинуть

IdCopy[i+1][j]=IdCopy[i][j];

IdCopy[i][j]=0;

}

}

}

return result;

}

public int MAX(){return max;}

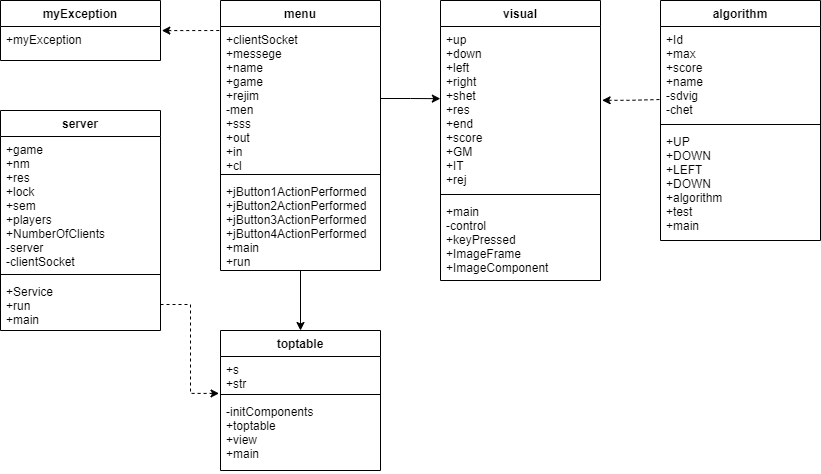
}

# Приложение Б. UML-диаграммы

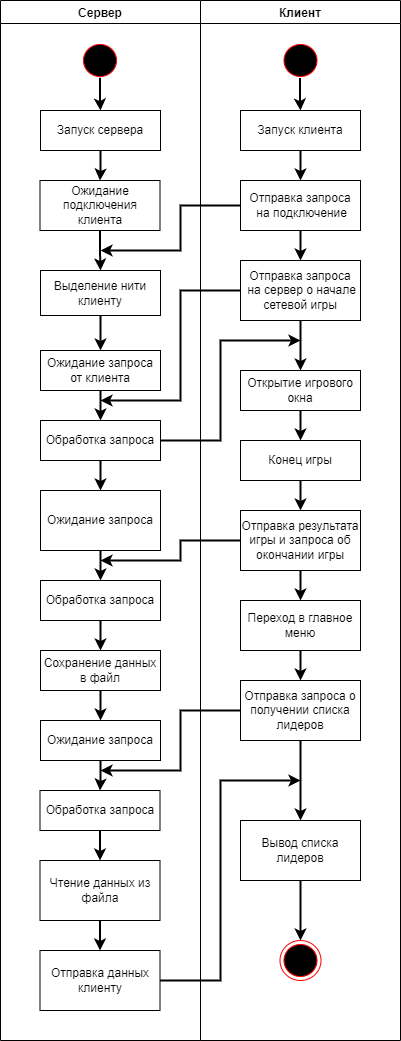
## **Приложение Б.1. UML-диаграмма вариантов использования**

## 

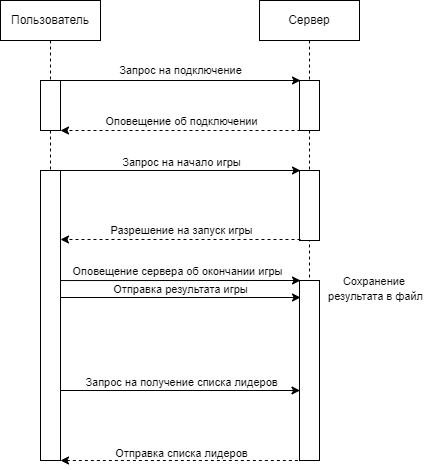
## **Приложение Б.2. UML-диаграмма классов**



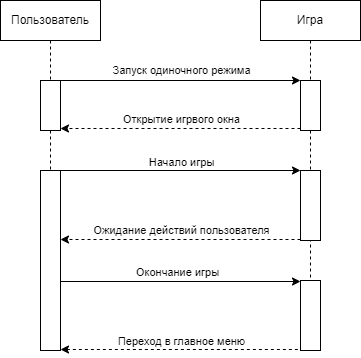
## **Приложение Б.3. UML- диаграмма деятельности**



## **Приложение Б.4. UML-диаграмма последовательности (сетевая игра)**



## **Приложение Б.5. UML-диаграмма последовательности (одиночная игра)**



## **Приложение Б.6. UML-диаграмма развертывания**

