

**Оглавление**

[Введение 4](#_Toc166750019)

[1. Постановка задачи 5](#_Toc166750020)

[2. Выбор решения 6](#_Toc166750021)

[3. Описание программы 7](#_Toc166750022)

[3.1 Серверная часть 7](#_Toc166750023)

[3.2 Клиентская часть 8](#_Toc166750024)

[3.2.1 Файл *MainMenu.java* 9](#_Toc166750025)

[3.2.2 Файл *ImageGen.java* 10](#_Toc166750026)

[3.2.3 Файл *GameResult.java* 13](#_Toc166750027)

[4. Описание способа организации пользовательского интерфейса 15](#_Toc166750028)

[5. Описание результатов работы программы 16](#_Toc166750029)

[Заключение 19](#_Toc166750030)

[Список используемых источников 20](#_Toc166750031)

[Приложение А Листинг программы 21](#_Toc166750032)

[Приложение А.1 Файл *Server.java* 21](#_Toc166750033)

[Приложение А.2 Файл *MainMenu.java* 24](#_Toc166750034)

[Приложение А.3 Файл *ImageGen.java* 31](#_Toc166750035)

[Приложение А.4 Файл *GameField.java* 36](#_Toc166750036)

[Приложение А.5 Файл *GameResult.java* 41](#_Toc166750037)

[Приложение А.6 Файл *myException.java* 44](#_Toc166750038)

[Приложение Б *UML*-диаграммы приложения 45](#_Toc166750039)

[Приложение Б.1 *UML*-диаграмма вариантов использования 45](#_Toc166750040)

[Приложение В.2 *UML*-диаграмма классов 46](#_Toc166750041)

[Приложение Б.3 *UML*-диаграмма деятельности 47](#_Toc166750042)

[Приложение Б.4 *UML*-диаграмма развертывания 48](#_Toc166750043)

[Приложение Б.5 *UML*-диаграмма использования 49](#_Toc166750044)

# **Введение**

У человека во все времена всегда была подробность в развлечении, общении и обмене какими-либо данными. Постепенно люди придумывали способы, которые могли удовлетворить данные потребности. Сейчас, в двадцать первом веке, благодаря современным технологиям, появилась возможность обмена различными видами информации, такими как видео, фото, музыка, звуки, и так далее по всему миру.

Для реализации приложений, которые бы соответствовали всем вышесказанным требованиям, есть очень популярный язык программирования – *Java*. *Java* является мощным и гибким языком программирования, который широко используется для разработки различных приложений, включая веб-приложения, мобильные приложения, игры и многое другое.

В рамках данной курсовой работы была разработана реализация игры «2048». Изначально, 2048 – браузерная игра, написанная 19-летним итальянским разработчиком Габриэле Чирулли на языке программирования JavaScript. Игровое поле имеет форму 4 на 4 квадрата. Целью игры является получение плитки номинала «2048». Девятнадцатилетний Габриэле Чирулли создал игру за одни выходные в качестве теста, чтобы проверить, сможет ли он запрограммировать игру с нуля.

«Это был способ скоротать время», – сказал он. Он описал ее как «концептуально похожую» на недавно выпущенную *iOS*-игру *Threes*, и клон другой игры, 1024. Разработанная *Veewo Studio*, 1024 сама по себе является клоном *Threes*, а ее описание в *App Store* однажды гласило «нет необходимости платить за *Threes*».

Игра является бесплатной, Чирулли сказал, что не хочет зарабатывать деньги «на концепции, которую он не изобретал». В мае 2014 года он выпустил порты для *iOS* и *Android*.

# **1. Постановка задачи**

Разработать систему многомодульных программ, клиент-серверной архитектуры, позволяющую играть в игру «2048» пользователями в рамках локальной вычислительной сети.

Функции сервера: прием и передача информации от пользователя, сбор статистики по пользователям.

Функции клиента: графический интерфейс пользователю, сетевое взаимодействие с сервером.

Операционная система: *Microsoft Windows* 10.

Среда разработки: *Apache NetBeans IDE* 16.

Язык программирования *Java*.

Используемые технологии:

* *Java Collections Framework*;
* Механизм обработки исключительных ситуаций;
* *Java Stream API*;
* *Java Multithreading*;
* Сетевое взаимодействие;
* *Java Swing*.

# **2. Выбор решения**

Для реализации курсового проекта «2048» используется язык программирования *Java*, библиотека *Swing*, *Java* коллекции *ArrayList* и *LinkedList*, *Java Stream Api*. В качестве среды разработки выбрана *IDE NetBeans*.

Основной задачей курсового проекта является реализация клиент-сервера. Для взаимодействия клиента и сервера были выбраны библиотеки *java.net.Socket* и *java.net.ServerSocket*.

На стороне сервера имеется класс, который обрабатывает запросы пользователей по принципу «Запрос-Ответ».

Со стороны клиента исходят запросы на получение данных у сервера или передачу данных серверу. Клиент выбирает нужное действие, а сервер это действие обрабатывает.

Пользовательский интерфейс реализован с помощью Java Swing.

*Java Swing* – это набор библиотек и классов, предоставляющих разработчикам возможность создания графических интерфейсов пользователя (*GUI*) на языке *Java*.

Были использованы следующие компоненты:

* кнопки (*Button*), поля ввода (*TextField*);
* панели (*Panel*), которые могут быть использованы для создания композитных компонентов, таких как таблицы и списки;
* метки (*Label*) для отображения текста, значков и изображений;
* диалоги и окна (*Dialogs*) для отображения сообщений пользователю, запрашивающих информацию или подтверждения;

# **3. Описание программы**

Программа, состоит из клиентской и серверной части. Сначала запускается сервер, который входит в состояние ожидания запроса клиента. После чего на устройствах запускается клиентская часть, которая по нажатию определенных кнопок, посылает определённый запросы к серверу.

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

## **3.1 Серверная часть**

Программа сервера является консольным приложением 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.

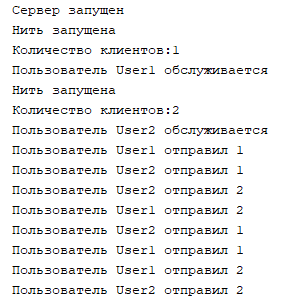


Рисунок 1 – вывод консоли программы сервера

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 получение результатов игры. Сервер принимает данный запрос и ожидает результат игры клиента. После получения добавляет результат игрока и его имя в специальные коллекции, после чего захватывает мьютекс и ждет пока все игроки, которые будут играть в многопользовательском режиме, не захватят мьютекс. После этого каждая нить отправляет своему клиенту результаты игроков, после чего отправляет имена всех клиентов, 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 многопользовательской 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.

## **3.2 Клиентская часть**

Программа клиента разбита на несколько модулей. *MainMenu.java* – главный файл проекта, файл *ImageGen.java* – это сама игра, файл *GameResult.java* отвечает за вывод результата.

## 3.2.1 Файл *MainMenu.java*

Файл *MainMenu.java* отвечает за подключение к серверу, отправку ему команд и принятие ответов от него, а также за запуск игры. При запуске данного файла появляется окно, в которой есть три кнопки: «Одиночная», «Многопользовательская», «Подключится», а также три поля для заполнения: «*IP*-адрес», «Порт», «Имя», что продемонстрированно на рисунке 2. Эти данные нужны для подключения к серверу. При возникновении ошибки с подключение выводится окно с ошибкой, показанное на рисунке 3: «Введены неправильные данные или произошла ошибка подключения».

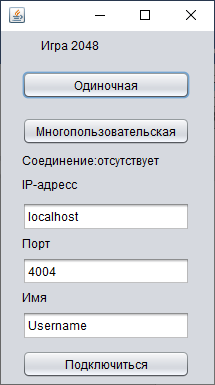


Рисунок 2 – окно главного меню игры «2048»

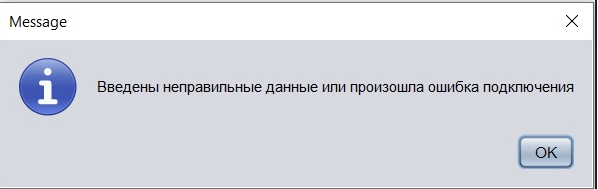


Рисунок 3 – сообщение об ошибке подключения

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

1. Запрос с кодом «1» - означает, что все игроки готовы к игре. Тогда запускается файл *ImageTest.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 получены и надо теперь их получить. Нить получает результаты и имена всех игроков, добавляя эти данные в соответствующие коллекции, после чего данные в этих коллекция сортируются по убыванию. После этого запускается файл *result.java*, отправляя данному файлу фрейм текущего окна и данные коллекции.

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.

При нажатии кнопки «Мультиплеер» 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.

## 3.2.2 Файл *ImageGen.java*

Файл *ImageGen.java* отвечает за отрисовку игровых действий. При запуске происходит загрузка всех изображений, добавление в коллекцию и их отображение на экране, что продемонстрировано на рисунке 4. Кроме этого запускается таймер, который по истечению времени перерисовывает изображения на экране. Тайл с определенной цифрой выбирается в зависимости от индекса, находящегося внутри массива *Id* класса *GameField*. При старте игры в массиве задаются случайные координаты двух элементов «два» с помощью функции данного класса – *NewElemen*t.

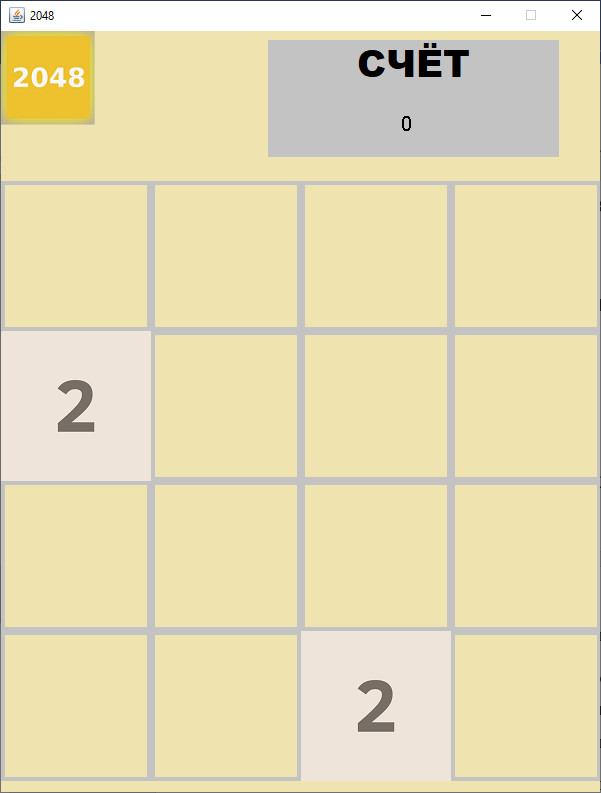


Рисунок 4 – окно игры при запуске

Основа программы строится вокруг класса *GameField*. При запуске файла *ImageGen.java* вызывается конструктор класса *GameField*, в котором массив *Id* заполняется цифрами «0». Значение всех чисел указан ниже:

* «0» - пустое поле;
* «1» - цифра два;
* «2» - цифра четыре;
* «3» - цифра восемь;
* «4» - цифра шестнадцать;
* «5» - цифра тридцать два;
* «6» - цифра шестьдесят четыре;
* «7» - цифра сто двадцать восемь;
* «8» - двести пятьдесят шесть;
* «9» -пятьсот двенадцать;
* «10» - тысяча двадцать четыре;
* «11» - две тысячи сорок восемь.

Для управления используются следующие кнопки: стрелка вверх, стрелка вниз, стрелка влево, стрелка вправо. При нажатии любой стрелки происходит сдвиг блоков в соответствующую сторону с использованием одной из четырех функций внутри класса *GameField.java*: *UP*, *DOWN*, *LEFT*, *RIGHT*. После выполнения сдвига генерируются случайные координаты следующего свободного блока с цифрой «два», а также происходит тестовый сдвиг во все четыре стороны с помощью функции *test*, если по результату не было сдвига ни в одну сторону у игрока больше нет возможных ходов, игра выдаст сообщение о проигрыше показанное на рисунке 5. Так же проверяется максимально значение блока, если оно равно «2048» – игра окончена и игрок видит сообщение продемонстрированное на рисунке 6. Если по результату нажатия кнопки сдвига не произошло, то описанные выше действия не происходят.



Рисунок 5 – сообщение «Вы проиграли»

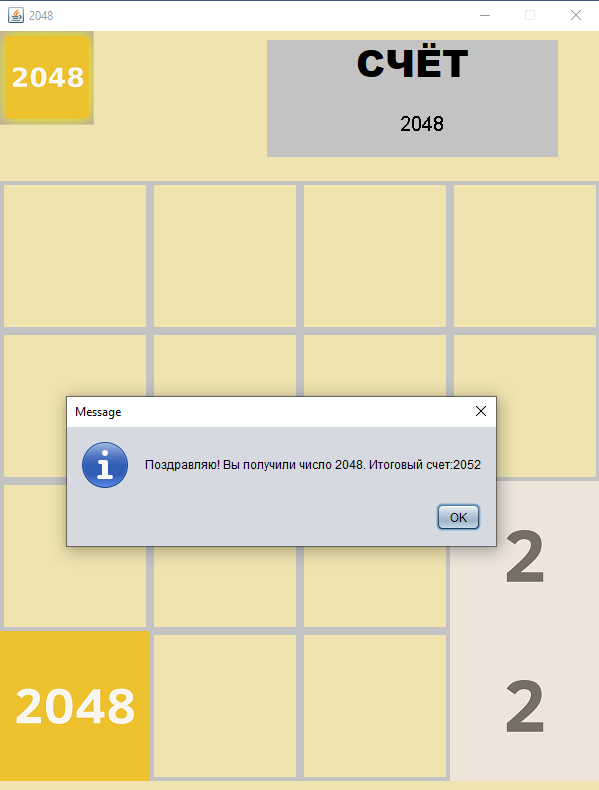


Рисунок 6 – сообщение «Вы выиграли»

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

## 3.2.3 Файл *GameResult.java*

Файл *GameResult.java* отвечает за получение коллекции имен игроков и их количества очков, после чего они выводятся в таблице на рисунке 7.

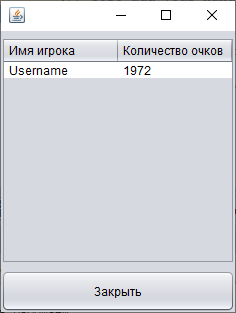


Рисунок 7 – таблица результатов

# **4. Описание способа организации пользовательского интерфейса**

В качестве среды разработки была выбрана программа *NetBeans IDE*. Программа предоставляет все средства, необходимые при разработке графического приложения.

Для реализации пользовательского интерфейса была использована библиотека *Swing*. Эта библиотека содержит более богатый и удобный набор элементов пользовательского интерфейса, обеспечивает одинаковое восприятие конечными пользователями приложений на разных платформах.

Для вставки изображений и текста был использован *jLabel*, который позволяет это реализовать.

Для корректного отображения всех элементов был выбран компонент *jPanel*.

# **5. Описание результатов работы программы**

Среда разработки *NetBeans* предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы.

Ниже продемонстрирован результат тестирования функционала программы.

При запуске программы появляется окно Главного меню игры, продемонстрированное на рисунке 8. Здесь имеется возможность начать одиночную игру, многопользовательскую, ввести *IP*-адрес и порт для подключения к серверу и сменить имя пользователя.

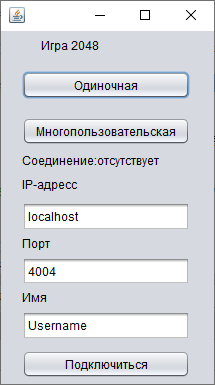


Рисунок 8 - запуск клиента

После того, как пользователь начнет игру, его встречает интерфейс игры, показанный на рисунке 9. На данном этапе пользователь начинает игру, управление осуществляется с помощью стрелок на клавиатуре. Игра заканчивается в случае, если игрок собирает число «2048», либо если у него отсутствует возможность для дальнейших действий. Также на экране отображается текущий счёт баллов игрока.

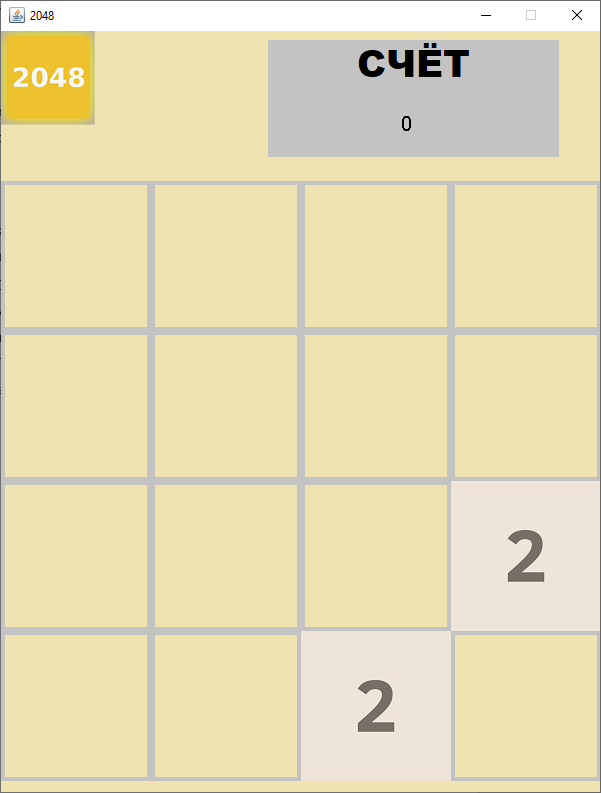


Рисунок 9 – запуск клиента

В случае победы игрока, или его проигрыша он получает соответствующие сообщения, что показано на рисунках 10-11.



Рисунок 10 – сообщение «Вы проиграли»

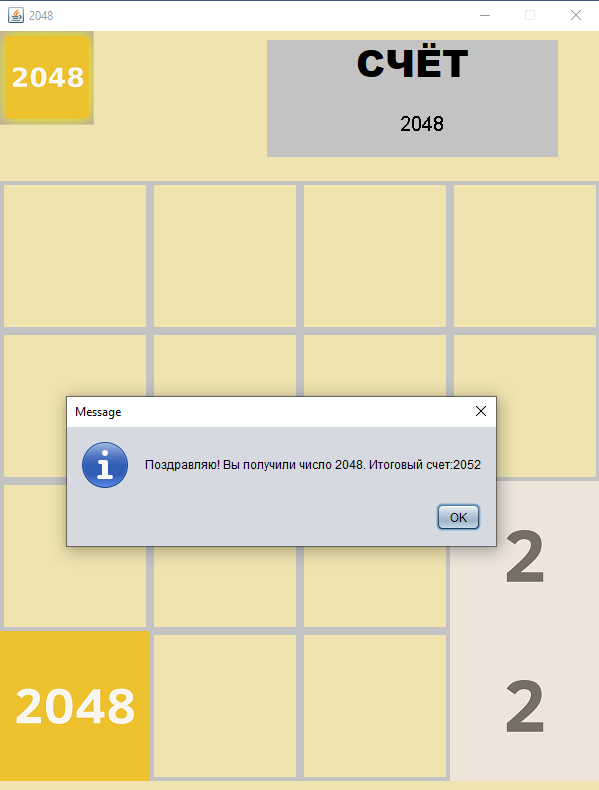


Рисунок 11 – сообщение «Вы выиграли»

В случае многопользовательской игры после ее окончания пользователь видит таблицу с результатами, показанную на рисунке 12.

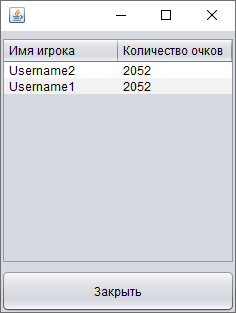


Рисунок 12 – таблица с количеством очков у игроков при многопользовательской игре

Серверная часть не имеет графического интерфейса. Получаемая информация отображается в консоли, что было продемонстрированно ранее на рисунке 1.

**Заключение**

При выполнении данной курсовой работы были получены навыки разработки программ на языке высокого уровня *Java*. Были освоены навыки разработки клиент серверных приложений на сокетах и графического интерфейса с помощью библиотеки *Swing*. Изучены возможности среды разработки *NetBeans IDE*.

В рамках данной работы было разработано приложение «2048», обладающий базовым функционалом, необходимым любому подобному приложению в наше время.

Список используемых источников

1. Университет Информационных Технологий (ИНТУИТ),   
   2016.— 285с.
2. Портянкин, *Java Swing*: Эффектные пользовательские интерфейсы - Издание второе, 2011, 200 с.
3. *Package javax.swing* // *docs.oracle.com* : [сайт]. – 2023. – *URL*: *https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html* (дата обращения: 05.05.2024).
4. Шилдт, Г. *Swing*. Руководство для начинающих. – М.: Вильямс, 2007. – 704 с.: ил.
5. Берд, Барри *Java* для чайников / Барри Берд. - М.: Диалектика / Вильямс, 2013. - 521 c.
6. Дубаков А.А. Сетевое программирование: учебное пособие / А.А. Дубаков – СП: НИУ ИТМО, 2013. – 248 с.
7. Лекции по курсу.

Приложение А  
Листинг программы

Приложение А.1   
Файл *Server.java*

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.concurrent.Semaphore;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Acer

\*/

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;

/\*\*

\* @param args the command line arguments

\*/

static class Service extends Thread {

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("Нить запущена");

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

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

} catch (IOException ex) {

System.out.println("Произошла ошибка получения буферов in и out");

}

}

@Override

public void run() {

try {

int number;

word = in.readLine();//получаем

name = word;

System.out.println("Пользователь " + name + " обслуживается");

while (true) {

word = in.readLine();

System.out.println("Пользователь " + name + " отправил " + word);

switch (Integer.parseInt(word)) {

case 0://пользователь играет один

synchronized (lock) {

players--;

sem = new Semaphore(players);

}

break;

case 1://пользователь готов играть с другими

try {

synchronized (lock) {

if (game != 1) {

game = 1;

}

}

//клиент готов к игре онлайн

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

synchronized (lock) {

res.add(word);

nm.add(name);

}

sem.acquire();

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

}

this.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) {

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

}

break;

case 3://пользователь хочет играть онлайн

while (game != 0) {

try {

this.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;

}

}

} catch (IOException ex) {

System.out.println("клиент отключился");

NumberOfClients--;

players--;

sem = new Semaphore(players);

System.out.println("Количество клиентов:" + NumberOfClients);

}

}

}

public static void main(String[] args) {

// TODO code application logic here

System.out.println("Сервер запущен");

try {

server = new ServerSocket(4004); // серверсокет прослушивает порт 4004

while (true) {

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

NumberOfClients++;

sem = new Semaphore(NumberOfClients);

players++;

new Service(clientSocket).start();

System.out.println("Количество клиентов:" + NumberOfClients);

}

} catch (IOException ex) {

System.out.println("Произошла ошибка добавления клиента");

}

}

}

Приложение А.2   
Файл *MainMenu.java*

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.net.Socket;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\*

\*/

public class MainMenu extends javax.swing.JFrame {

/\*\*

\* Creates new form menu

\*/

public MainMenu() {

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jTextField1 = new javax.swing.JTextField();

jTextField2 = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jButton3 = new javax.swing.JButton();

jLabel4 = new javax.swing.JLabel();

jTextField3 = new javax.swing.JTextField();

jLabel5 = new javax.swing.JLabel();

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

jLabel1.setText("Игра 2048");

jButton1.setText("Одиночная");

jButton1.addActionListener(new java.awt.event.ActionListener() {

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

jButton1ActionPerformed(evt);

}

});

jButton2.setText("Многопользовательская");

jButton2.addActionListener(new java.awt.event.ActionListener() {

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

jButton2ActionPerformed(evt);

}

});

jTextField1.setText("localhost");

jTextField2.setText("4004");

jLabel2.setText("IP-адресс");

jLabel3.setText("Порт");

jButton3.setText("Подключиться");

jButton3.addActionListener(new java.awt.event.ActionListener() {

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

jButton3ActionPerformed(evt);

}

});

jLabel4.setText("Соединение:отсутствует");

jTextField3.setText("Username");

jLabel5.setText("Имя");

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addGroup(layout.createSequentialGroup()

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

.addGroup(layout.createSequentialGroup()

.addGap(21, 21, 21)

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

.addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 95, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 95, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel5)

.addComponent(jButton3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jTextField1)

.addComponent(jTextField2)

.addComponent(jTextField3)))

.addGroup(layout.createSequentialGroup()

.addGap(40, 40, 40)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 110, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addContainerGap(24, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

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

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

.addGap(18, 18, 18)

.addComponent(jButton1)

.addGap(18, 18, 18)

.addComponent(jButton2)

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

.addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED\_SIZE, 19, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jLabel2)

.addGap(10, 10, 10)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(4, 4, 4)

.addComponent(jLabel3)

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

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, 28, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(4, 4, 4)

.addComponent(jLabel5)

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

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 10, Short.MAX\_VALUE)

.addComponent(jButton3)

.addContainerGap())

);

pack();

}// </editor-fold>

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

try {

if (game != 0) {

throw new myException("Ожидание остальных игроков");

}

if (clientSocket != null) {

try {

rejim = 1;

out.write("0\n");

out.flush();

} catch (IOException ex) {

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

}

}

setVisible(false);

ImageGen.main(this, 0);

} catch (myException ex) {

JOptionPane.showMessageDialog(null, ex.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(jTextField1.getText(), Integer.parseInt(jTextField2.getText()));

name = jTextField3.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 неправильные данные или произошла ошибка подключения");

return;

}

jLabel4.setText("Соединение: активно");

} catch (myException ex) {

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

return;

}// TODO add your handling code here:

}

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(MainMenu.class.getName()).log(Level.SEVERE, null, ex);

}

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

}

if (game == 1) {

throw new myException("Ожидание остальных игроков");

}

if (clientSocket == null) {

throw new myException("Отсутствует подключение к серверу");

}

try {

out.write("1\n");

out.flush();

} catch (IOException ex) {

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

}

game = 1;

} catch (myException e) {

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

return;

}

}

public void res(int score) {

try {

out.write("2\n");

out.flush();

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

out.flush();

} catch (IOException ex) {

Logger.getLogger(MainMenu.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();

MainMenu men;

BufferedWriter out;

String message;

Client(MainMenu men) {

this.men = men;

try {

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

} catch (IOException ex) {

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

}

}

@Override

public void run() {

try {

System.out.println("Нить работает");

while (true) {

message = in.readLine();

System.out.println("Получено: " + message);

switch (Integer.parseInt(message)) {

case 1:

setVisible(false);

ImageGen.main(men, 1);

break;

case 2:

res = new ArrayList();

nm = new ArrayList();

message = in.readLine();

System.out.println("Получено: " + message);

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

res.add(message);

message = in.readLine();

System.out.println("Получено: " + message);

}

message = in.readLine();

System.out.println("Получено: " + message);

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

//System.out.println("Получено: "+message);

nm.add(message);

message = in.readLine();

System.out.println("Получено: " + message);

}

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);

GameResults.main(men, res, nm);

game = 0;

break;

case 3:

game = 0;

break;

}

}

} catch (IOException ex) {

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

jLabel4.setText("Соединение: активно");

}

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

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

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

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

break;

}

}

} catch (ClassNotFoundException ex) {

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

//</editor-fold>

/\* Create and display the form \*/

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

public void run() {

new MainMenu().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

private javax.swing.JTextField jTextField3;

// End of variables declaration

}

Приложение А.3   
Файл *ImageGen.java*

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.io.\*;

import java.util.ArrayList;

import javax.imageio.\*;

import javax.swing.\*;

public class ImageGen {

static JLabel jLabel1 = new javax.swing.JLabel("0");

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

static GameField GM = new GameField();

static ImageFrame IT;

static int mode;

static MainMenu copy;

// static ImageComponent component = new ImageComponent();

public static void main(MainMenu args, int gameMode) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

mode = gameMode;

GM = new GameField();

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);

// Добавление компонента к фрейму.

//ImageComponent component = new ImageComponent();

add(component);

creatGUI();

IT = this;

}

public void creatGUI() {

addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e) {

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

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

up = 1;

}

}

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() == 37) {//стрелка влево

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

left = 1;

}

}

}

}

);

}

public static final int DEFAULT\_WIDTH = 615;

public static final int DEFAULT\_HEIGHT = 800;

public void hide() {

}

}

static class ImageComponent extends JComponent {

ArrayList<Image> image = new ArrayList();

Image BGImage;

public ImageComponent() {

// Получаем изображения.

try {

BGImage = ImageIO.read(new File("src/2048 game.png"));

image.add(ImageIO.read(new File("src/tile\_0.png")));

image.add(ImageIO.read(new File("src/tile\_1.png")));

image.add(ImageIO.read(new File("src/tile\_2.png")));

image.add(ImageIO.read(new File("src/tile\_3.png")));

image.add(ImageIO.read(new File("src/tile\_4.png")));

image.add(ImageIO.read(new File("src/tile\_5.png")));

image.add(ImageIO.read(new File("src/tile\_6.png")));

image.add(ImageIO.read(new File("src/tile\_7.png")));

image.add(ImageIO.read(new File("src/tile\_8.png")));

image.add(ImageIO.read(new File("src/tile\_9.png")));

image.add(ImageIO.read(new File("src/tile\_10.png")));

image.add(ImageIO.read(new File("src/tile\_11.png")));

} catch (IOException e) {

e.printStackTrace();

}

Timer timer = new Timer(100, new ActionListener() {//запускаем таймер.Раз в секунду проеряет значение в массиве и перерисовывает

@Override

public void actionPerformed(ActionEvent e) {

//GM.NewElement();

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

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

ctn++;//увеличиваем счетчик сдвигов

if (ctn == 3) {//три раза сдвинули?

up = 0;//если да то заканчиваем сдвигать

ctn = 0;//обнуляем счетчик

if (res != 0) {//если сдвинули

end = GM.NewElement();

res = GM.test();

if (res == 0) {

JOptionPane.showMessageDialog(null, "Вы проиграли. Итоговый счет:" + GM.score);

if (mode == 1) {

copy.res(GM.score);

}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res = GM.MAX();

if (res == 11) {

JOptionPane.showMessageDialog(null, "Поздравляю! Вы получили число 2048. Итоговый счет:" + GM.score);

if (mode == 1) {

copy.res(GM.score);

}

res = 0;

IT.setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res = 0;

}

}

}

if (down == 1) {

res += GM.DOWN();

ctn++;

if (ctn == 3) {

down = 0;

ctn = 0;

if (res != 0) {

end = GM.NewElement();

res = GM.test();

if (res == 0) {

JOptionPane.showMessageDialog(null, "Вы проиграли. Итоговый счет:" + GM.score);

if (mode == 1) {

copy.res(GM.score);

}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res = GM.MAX();

if (res == 11) {

JOptionPane.showMessageDialog(null, "Поздравляю! Вы получили число 2048. Итоговый счет:" + GM.score);

res = 0;

if (mode == 1) {

copy.res(GM.score);

}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res = 0;

}

}

}

if (left == 1) {

res += GM.LEFT();

ctn++;

if (ctn == 3) {

left = 0;

ctn = 0;

if (res != 0) {

end = GM.NewElement();

res = GM.test();

if (res == 0) {

JOptionPane.showMessageDialog(null, "Вы проиграли. Итоговый счет:" + GM.score);

if (mode == 1) {

copy.res(GM.score);

}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res = GM.MAX();

if (res == 11) {

JOptionPane.showMessageDialog(null, "Поздравляю! Вы получили число 2048. Итоговый счет:" + GM.score);

res = 0;

if (mode == 1) {

copy.res(GM.score);

}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res = 0;

}

}

}

if (right == 1) {

res += GM.RIGHT();

ctn++;

if (ctn == 3) {

right = 0;

ctn = 0;

if (res != 0) {

end = GM.NewElement();

res = GM.test();

if (res == 0) {

JOptionPane.showMessageDialog(null, "Вы проиграли. Итоговый счет:" + GM.score);

if (mode == 1) {

copy.res(GM.score);

}

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res = GM.MAX();

if (res == 11) {

JOptionPane.showMessageDialog(null, "Поздравляю! Вы получили число 2048. Итоговый счет:" + GM.score);

if (mode == 1) {

copy.res(GM.score);

}

res = 0;

setVisible(false);

copy.setVisible(true);

IT.dispose();

}

res = 0;

}

}

}

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

}

});

timer.start();

}

@Override

public void paintComponent(Graphics g) {

if (image == null) {

return;

}

int imageWidth = 150;

int imageHeight = 150;

g.drawImage(BGImage, 0, 0, null);

// Многократный вывод изображения в панели.

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), (j \* imageHeight) + 150, null);

break;

case (1):

g.drawImage(image.get(1), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (2):

g.drawImage(image.get(2), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (3):

g.drawImage(image.get(3), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (4):

g.drawImage(image.get(4), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (5):

g.drawImage(image.get(5), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (6):

g.drawImage(image.get(6), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (7):

g.drawImage(image.get(7), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (8):

g.drawImage(image.get(8), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (9):

g.drawImage(image.get(9), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (10):

g.drawImage(image.get(10), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

case (11):

g.drawImage(image.get(11), (i \* imageWidth), (j \* imageHeight) + 150, null);

break;

}

g.setFont(new Font("TimesRoman", Font.PLAIN, 20));

g.drawString("" + GM.score, 400, 100);

}

}

}

}

}

Приложение А.4   
Файл *GameField.java*

public class GameField {

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

int max = 1;

public int score = 0;

GameField() {//заполняем поле квадратами

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

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

Id[i][j] = 0;

}

}

//Id[0][0] = 10;

//Id[0][1] = 10;

}

public int NewElement() {

int apopit = 0;

int bpopit = 0;

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;

int chet = 0;

// while(chet<3){

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

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

if (j != 3) {

if (Id[i][j] == Id[i][j + 1]) {

if (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 (0 != j) {

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

if (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 вверх

int sdvig = 0;

int chet = 0;

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

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

if (j != 0) {

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

if (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) {

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

if (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 вверх

int sdvig = 0;

int chet = 0;

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

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

if (i != 3) {

if (Id[i + 1][j] == Id[i][j]) {

if (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) {

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

if (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 вверх

int sdvig = 0;

int chet = 0;

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

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

if (i != 0) {

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

if (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) {

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

if (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) {

if (IdCopy[i][j] == IdCopy[i][j + 1]) {

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

IdCopy[i][j]++;

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

result++;

}

}

}

if (0 != j) {

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

if (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) {

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

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

IdCopy[i][j]++;

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

result++;

}

}

}

if (3 != j) {

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

if (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) {

if (IdCopy[i + 1][j] == IdCopy[i][j]) {

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

IdCopy[i][j]++;

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

result++;

}

}

}

if (0 != i) {

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

if (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) {

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

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

IdCopy[i][j]++;

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

result++;

}

}

}

if (3 != i) {

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

if (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 А.5   
Файл *GameResult.java*

import java.util.ArrayList;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Acer

\*/

public class GameResults extends javax.swing.JFrame {

static MainMenu copy;

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

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

/\*\*

\* Creates new form result

\*/

public GameResults() {

initComponents();

javax.swing.table.DefaultTableModel model = (javax.swing.table.DefaultTableModel) jTable1.getModel();

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

model.addRow(new Object[]{name.get(i), score.get(i)});

}

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

jButton1 = new javax.swing.JButton();

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

jTable1.setModel(new javax.swing.table.DefaultTableModel(

new Object[][]{},

new String[]{

"Имя игрока", "Количество очков"

}

) {

boolean[] canEdit = new boolean[]{

false, false

};

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit[columnIndex];

}

});

jScrollPane1.setViewportView(jTable1);

jButton1.setText("Закрыть");

jButton1.addActionListener(new java.awt.event.ActionListener() {

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

jButton1ActionPerformed(evt);

}

});

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE)

.addComponent(jButton1, javax.swing.GroupLayout.DEFAULT\_SIZE, 234, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

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

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

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

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 227, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

);

pack();

}// </editor-fold>

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

setVisible(false);

copy.setVisible(true);

dispose();

// TODO add your handling code here:

}

public static void main(MainMenu men, ArrayList<String> res, ArrayList<String> nm) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

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

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

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

break;

}

}

} catch (ClassNotFoundException ex) {

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

//</editor-fold>

/\* Create and display the form \*/

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

public void run() {

copy = men;

score = res;

name = nm;

new GameResults().setVisible(true);

}

});

}

static void table() {

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable jTable1;

// End of variables declaration

}

Приложение А.6   
Файл *myException.java*

class myException extends Exception {

myException(String s) {

super(s);

}

}

Приложение Б  
*UML*-диаграммы приложения

Приложение Б.1  
*UML*-диаграмма вариантов использования



Рисунок Б.1 – UML-диаграмма вариантов использования

Приложение В.2  
*UML*-диаграмма классов

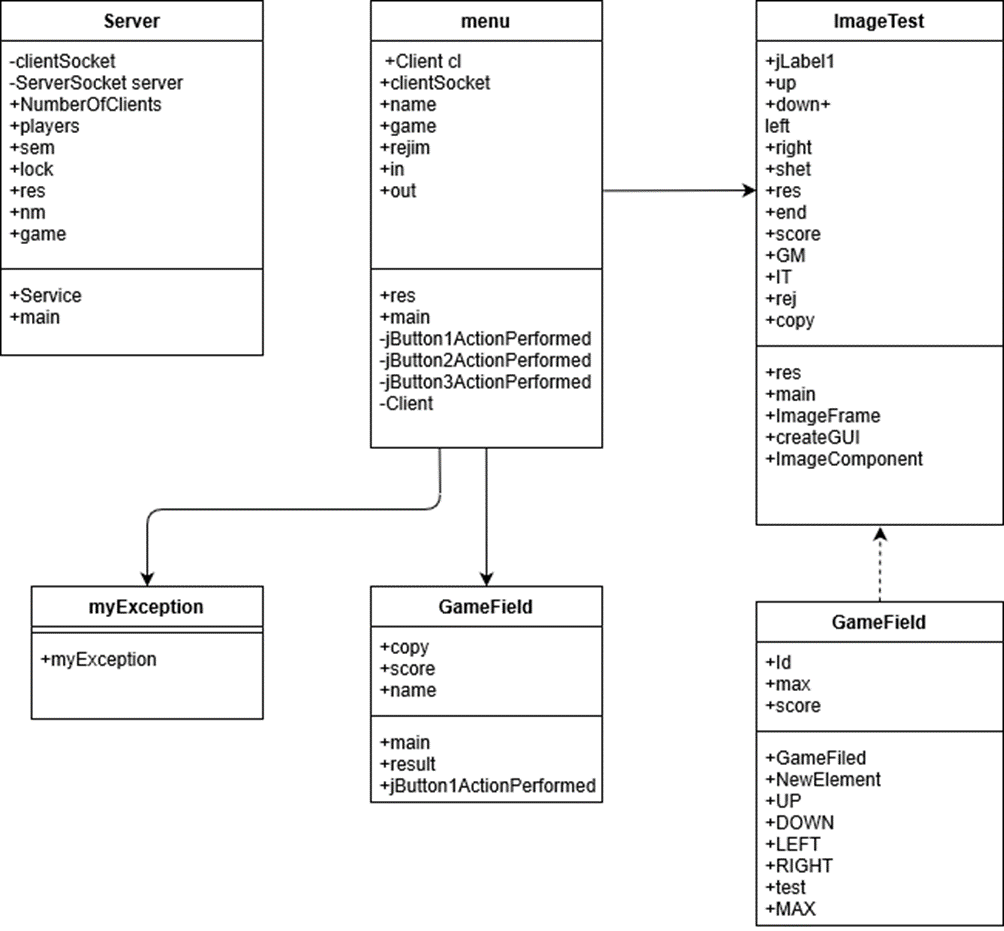


Рисунок Б.2 – UML-диаграмма классов

Приложение Б.3  
*UML*-диаграмма деятельности

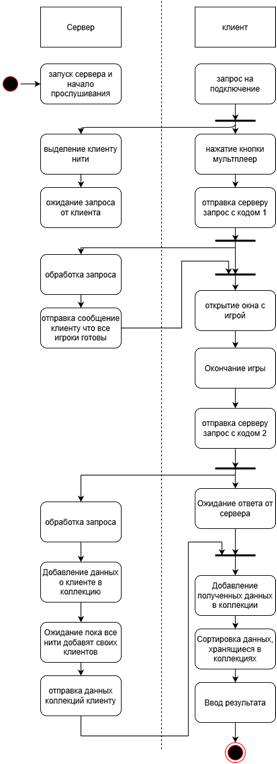


Рисунок Б.3 – UML-диаграмма деятельности

Приложение Б.4  
*UML*-диаграмма развертывания

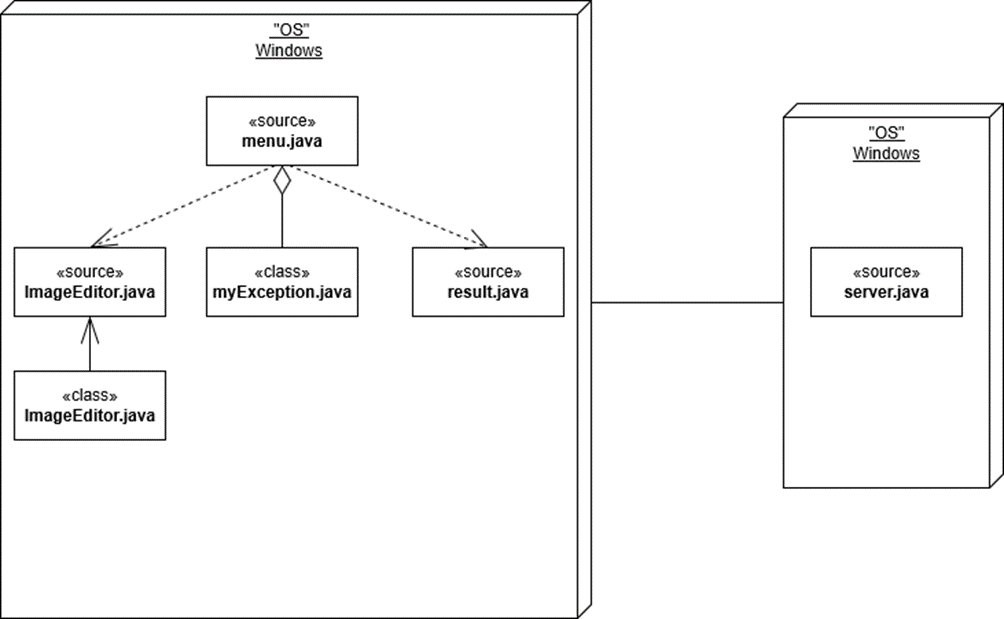


Рисунок Б.4 – UML-диаграмма развертывания

Приложение Б.5  
*UML*-диаграмма использования



Рисунок Б.5 – UML-диаграмма использования