Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

# ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование на языке Java»

на тему «Разработка многомодульного приложения на языке Java»

Выполнил:

студент группы 21ВВП1

Сущёв М.В.

Принял:

Юрова О.В.

Пенза 2024

**Содержание**

# Введение

В современном мире растет интерес к многопользовательским играм, что приводит к увеличению предложений различных игр с клиент-серверной архитектурой. Одной из таких игр является многомодульное приложение "Танчики", в котором пользователи могут сражаться друг с другом в режиме реального времени. Эта игра не только обеспечивает захватывающий игровой процесс, но и требует надежного и эффективного обмена данными между клиентами и сервером для синхронизации прогресса игроков.

Архитектура клиент-сервер представляет собой распределенную систему, где задачи и сетевая нагрузка распределяются между поставщиками услуг (серверами) и их потребителями (клиентами). Взаимодействие между клиентами и серверами происходит через компьютерные сети и может включать как различные физические устройства, так и программное обеспечение. Пример простой технологии клиент-сервера заключается в том, что пользователь делает запрос (например, подключение к игровому серверу), а сервер предоставляет ответ в виде данных о состоянии игры.

Целью данного курсового проекта является разработка многомодульного приложения с клиент-серверной архитектурой для игры "Танчики", состоящего из сервера и клиентов, взаимодействующих по сети. В рамках проекта будут реализованы механизмы управления игровым процессом, синхронизации прогресса игроков, а также поддержка сетевой безопасности и устойчивости к нагрузкам.

# 1. Постановка задачи

В курсовой работе необходимо разработать многомодульную программу на языке Java – игровое приложение "Танчики".

Необходимо создать сервер и клиент. Клиент – игровое приложение, которое будет отправлять серверу запросы. Сервер должен отвечать на эти запросы.

Функции клиента:

- Интерфейс пользователя
- Взаимодействие с сервером

Функции сервера:

- Взаимодействие с программами-клиентами

Операционная система – Windows.

Язык программирования – Java.

Среда программирования: IntelliJ IDEA Community Edition.

## 2. Выбор решения

Важной частью курсовой работы является передача сообщений от клиента к серверу. Для реализации этого используются TCP протокол. TCP расшифровывается как Transmission Control Protocol. По сравнению с UDP, сетевой протокол TCP более надежен, но медленнее и сложнее. Он часто используется в ситуациях, когда важна надёжность.

В отличие от UDP, протокол TCP устанавливает предварительную связь между двумя сторонами. Ввиду этого иногда данные не могут потеряться в пути, взамен получения более низкой скорости передачи пакетов.

Протокол TCP был выбран, потому что нам необходимо передавать данные пользователя для авторизации и его прогресс прохождения игры, эти данные не большого размера, поэтому скорость не так важна как надёжность. Для взаимодействия между компьютерами используются адреса и порты.

## 3. Описание программы

Разработанная программа состоит основывается на вышеописанных протоколах, а также из нескольких модулей, которые реализовывались в следующем порядке:

1. Game.java – основной файл.
2. Level.java, GifType,java, GlobalConstance.java, Direction.java, TankType.java, WallType.java, SceneObject.java, TankSide.java, RandGenerate.java, GameStatus.java – вспомогательные модули (перечисляемые типы, классы для хранения неизменяемых данных, интерфейсы).
3. GameEnvironmentHelper.java, LevelService.java – отвечают за считывание карт из файла, карты находятся в папке recources/maps, от их названия зависит то к какому по счёту уровню они относятся и сколько противников будет map2,8.txt (2 уровень, 8 противников).
4. ServerTask.java – отвечает за отправку запросов на сервер и получение ответов.
5. OperatorGift.java, Tank.java, Wall.java, Flag.java, Shot.java, GameObjectHelper.java, Player.java, AddEnemy.java, GiveGift.java, EnemyTankMovement.java – эти модули отвечают за реализацию игровых механик.
6. GameData.java – отвечает за хранение игровых данных (противники, игрок, бонусы и т.д.).
7. GamePage.java, LoginPage.java, StartMenu.java, StartMenuHelper.java, EndGameScene.java, SceneHelper.java – отвечают за отрисовку меню и игрового интерфейса.
8. EventHandler.java, PlayerController.java – отвечают за управление игрока.

При запуске программа загружает игровые карты, создаёт несколько нитей для реализации игровых механик и отрисовывает интерфейс. При

попытке входа или регистрации, отправляется запрос на сервер, если введены верные данные, то загружается сцена с выбором уровней. После прохождения уровня, если есть следующий, он становится доступным для пользователя и отправляется запрос на сервер для обновления данных о прохождение уровня, затем открывается сцена с выбором уровня.

Для того, чтобы авторизация и обновление данных о прохождение работало, необходимо запустить сервер, который также состоит из нескольких модулей:

1. Server.java – основной файл.
2. ClientHandler.java – нить, которая обрабатывает запросы клиента.
3. User.java – класс для хранения данных о пользователе.
4. UserDatabase.java – класс для получения данных из текстового файла и сохранения их.

Отправка сообщений от клиента серверу.

```
public static String sendRequest(String request) {  2 usages
    try {
        Socket socket = new Socket( host: "localhost",  port: 12345);

        PrintWriter out = new PrintWriter(socket.getOutputStream(),  autoFlush: true);
        BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

        out.println(request);

        String response = in.readLine();

        in.close();
        out.close();
        socket.close();

        return response;
    } catch (IOException e) {
        showAlert( title: "Ошибка",  message: "Произошла ошибка при попытке подключения к серверу.");
        e.printStackTrace();
    }
    return "error_connect";
}
```

**Рисунок 1 – Отправка сообщений от клиента серверу.**

Отправка сообщений от сервера клиенту.

```
private void handleClient() throws IOException {  1 usage
    BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
    PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),  autoFlush: true);

    String request = in.readLine();
    String[] parts = request.split( regex: "/");
    if (parts.length >= 3) {
        String operation = parts[0];
        String username = parts[1];
        String password = parts[2];
        boolean result = false;

        if ("login".equalsIgnoreCase(operation)) {...} else if ("signup".equalsIgnoreCase(operation)) {...} else if ("levelup".e
            int level = Integer.parseInt(parts[3]);
            result = userDatabase.levelUp(username, password, level);
            if (result) {...} else {...}
        } else {...}
    } else {...}

    in.close();
    out.close();
}
```

**Рисунок 2 – Отправка сообщений от сервера клиенту.**

Взаимодействие пользователя-клиента-сервера отображено в Приложение В.3 Диаграмма последовательности. Работа клиента и сервера показаны в Приложение В.2 Диаграмма развёртывания и в Приложение В.4 Диаграмма деятельности.

## 4. Описание способа реализации пользовательского интерфейса

Пользовательский интерфейс присутствует только в клиентской части приложения с использованием JavaFX. Приложение состоит из трёх экранов, реализованных с помощью Scene. Первый экран – регистрация, второй экран – выбор уровня, третий экран – игровой.

При запуске программы пользователь увидит перед собой два поля “Логин”, “Пароль” и две кнопки “Войти”, “Зарегистрироваться”. После входа или регистрации можно выбрать уровень. После выбора уровня запускается игра. Управление реализовано при помощи кнопок: W, A, S, D, SPACE. Игра может завершиться победой или поражением. После этого опять откроется меню выбора уровня. Для того чтобы выйти из приложения требуется нажать крестик в правом верхнем углу программы.

Возможности клиента отображены в Приложение В.5 Диаграмма вариантов использования приложения.

## 5. Описание результатов работы программы

Ниже представлена работа клиент-серверного приложения.

Для начала необходимо запустить сервер.

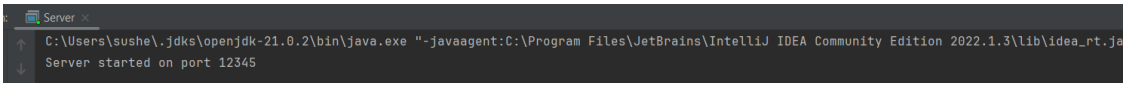


**Рисунок 3 – Запуск сервера.**

Далее запускаем клиент.

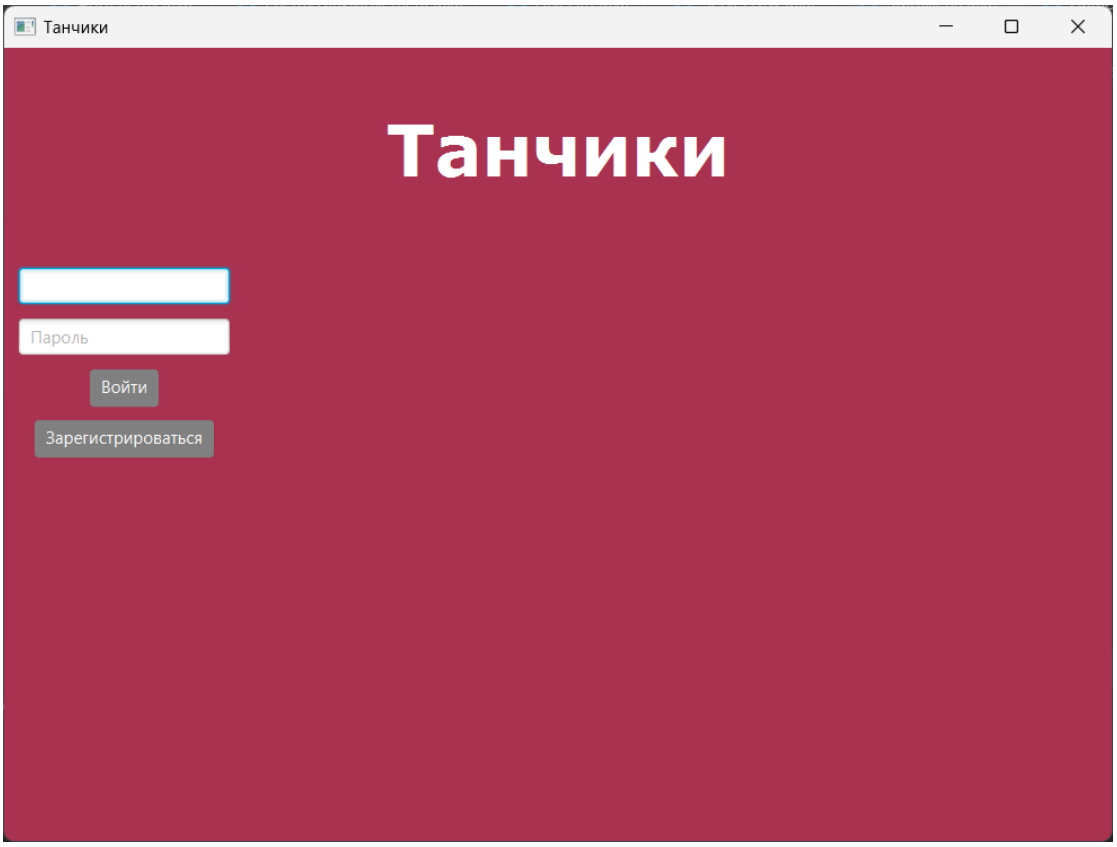


**Рисунок 4 – Сцена авторизации.**

Не верно ввели логин или пароль, срабатывает исключение.

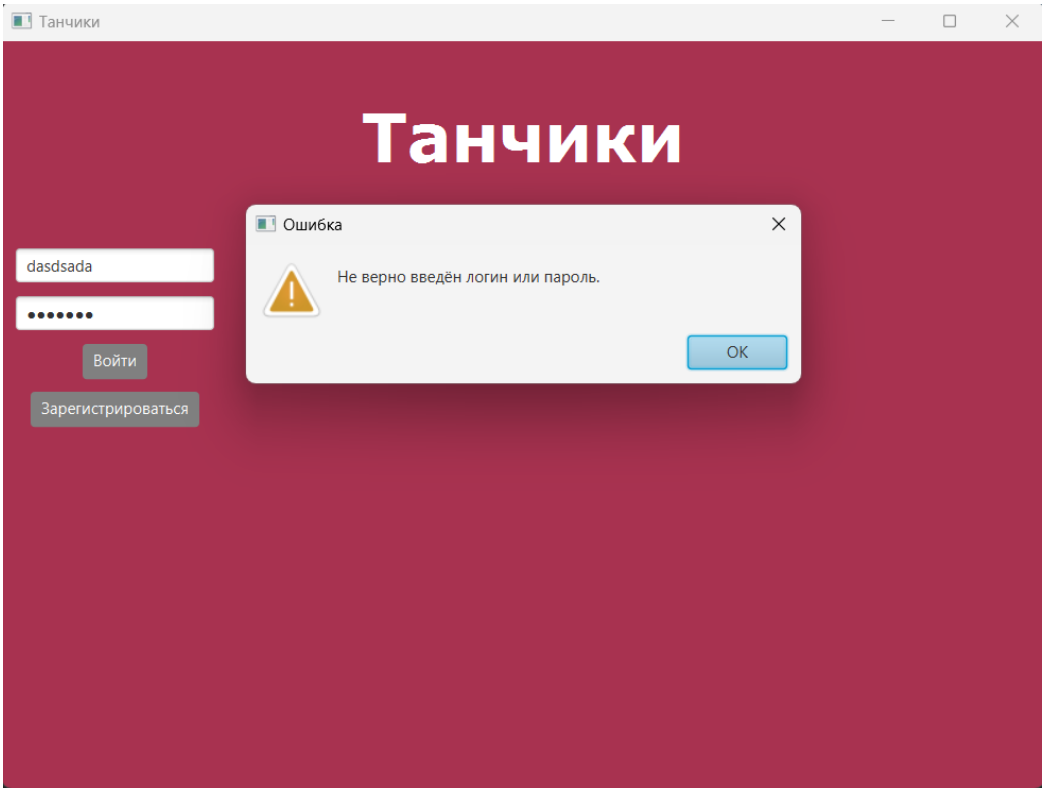


**Рисунок 5 – Сцена авторизации.**

Выполняем регистрацию и попадаем на сцену выбора уровня.

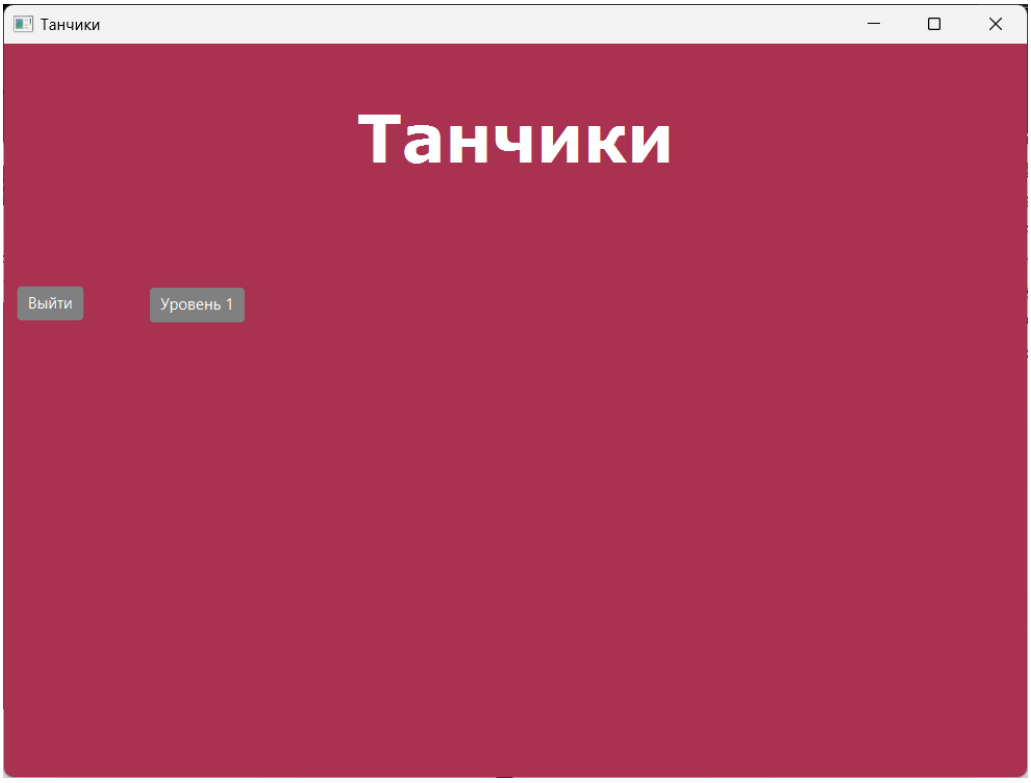


**Рисунок 6 – Сцена выбора уровня.**

Запускаем уровень.


**Рисунок 7 – Геймплей игры.**

Проходим уровень и возвращаемся к сцене выбора уровня.

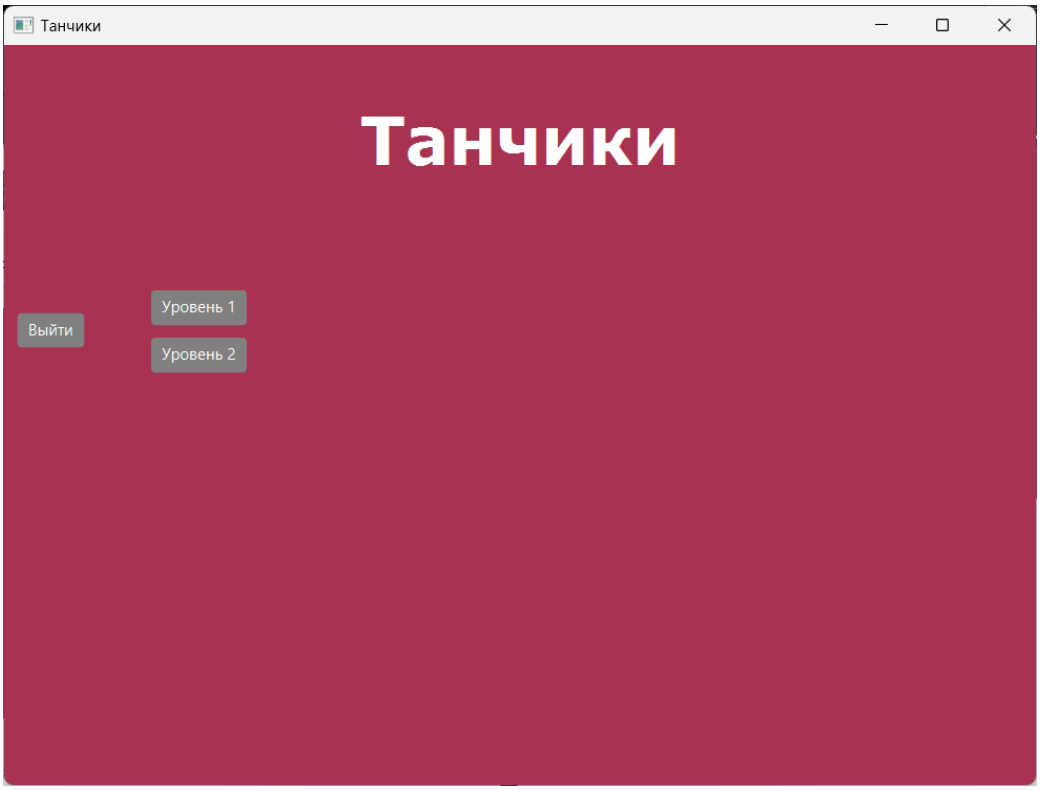


**Рисунок 8 – Сцена выбора уровня.**

## Заключение

При выполнении данного курсового проекта были изучены принципы работы разработки многомодульного приложения на языке Java. Также изучили принципы разработки графического интерфейса и следующие технологии: Java Collections Framework, Механизм обработки исключений, Java Stream API, Java Multithreading, Сетевое взаимодействие.

В результате выполнения данной курсовой работы была разработана система программ клиент-серверной архитектуры, позволяющая осуществлять взаимодействие между клиентом и сервером.

Программа написана на языке Java с использование среды программирования IntelliJ IDEA Community Edition.

## Список используемых источников


1. code-basics.com Сайт о программировании [Электронный ресурс]. Режим доступа: https://code-basics.com/, свободный (Дата обращения 22.05.2024).
2. javarush.com Сайт о программировании [Электронный ресурс]. Режим доступа: https://javarush.com/, свободный (Дата обращения 22.05.2024).
3. sentry.io Сайт о программировании [Электронный ресурс]. Режим доступа: https://sentry.io/, свободный (Дата обращения 22.05.2024).

# Приложение А.

# Листинг программы

## Приложение А.1.

## Файл «Server.java» Server

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
    private static final int PORT = 12345;
    private UserDatabase userDatabase;

    public Server() {
        userDatabase = new UserDatabase();
    }

    public void start() {
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Server started on port " + PORT);

            while (true) {
                try {
                    Socket clientSocket = serverSocket.accept();
                    new Thread(new ClientHandler(clientSocket, userDatabase)).start();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Server server = new Server();
        server.start();
    }
}
```

# Приложение А.2.

## Файл «ClientHandler.java» Server

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ClientHandler implements Runnable {
    private Socket clientSocket;
    private UserDatabase userDatabase;

    public ClientHandler(Socket clientSocket, UserDatabase userDatabase) {
        this.clientSocket = clientSocket;
        this.userDatabase = userDatabase;
    }

    @Override
    public void run() {
        try {
            handleClient();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                clientSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    private void handleClient() throws IOException {
        BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

        String request = in.readLine();
        String[] parts = request.split("/");
        if (parts.length >= 3) {
            String operation = parts[0];
            String username = parts[1];
            String password = parts[2];
            boolean result = false;

            if ("login".equalsIgnoreCase(operation)) {
                result = userDatabase.login(username, password);
                if (result) {
                    int level = userDatabase.getLevel(username);
                    out.println("success/" + level);
                } else {
                    out.println("error");
                }
            } else if ("signup".equalsIgnoreCase(operation)) {
                result = userDatabase.signUp(username, password);
                if (result) {
                    out.println("success");
                } else {
                    out.println("error");
                }
            } else if ("levelup".equalsIgnoreCase(operation) && parts.length == 4) {
                int level = Integer.parseInt(parts[3]);
                result = userDatabase.levelUp(username, password, level);
                if (result) {
                    out.println("success");
                } else {
                    out.println("error");
                }
            } else {
                out.println("error");
            }
        } else {
            out.println("error");
        }

        in.close();
```

```
            out.close();
        }
    }
```

## Приложение А.3.

## Файл «User.java» Server

```
public class User {
    private String username;
    private String password;
    private int level;

    public User(String username, String password, int level) {
        this.username = username;
        this.password = password;
        this.level = level;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public int getLevel() {
        return level;
    }

    public void setLevel(int level) {
        this.level = level;
    }
}
```

# Приложение А.4.

## Файл «UserDatabase.java» Server

```
import java.io.*;
import java.util.HashMap;
import java.util.Map;
import java.util.stream.Collectors;

public class UserDatabase {
    private static final String FILE_NAME = "users.txt";
    private Map<String, User> users;

    public UserDatabase() {
        users = new HashMap<>();
        loadUsers();
    }

    private void loadUsers() {
        try (BufferedReader reader = new BufferedReader(new FileReader(FILE_NAME))) {
            users = reader.lines()
                    .map(line -> line.split("/"))
                    .filter(parts -> parts.length == 3)
                    .collect(Collectors.toMap(
                            parts -> parts[0],
                            parts -> new User(parts[0], parts[1], Integer.parseInt(parts[2]))
                    ));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void saveUsers() {
        try (PrintWriter writer = new PrintWriter(new FileWriter(FILE_NAME))) {
            users.values().forEach(user ->
                    writer.println(user.getUsername() + "/" + user.getPassword() + "/" +
user.getLevel())
            );
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public boolean login(String username, String password) {
        return users.containsKey(username) && users.get(username).getPassword().equals(password);
    }

    public boolean signUp(String username, String password) {
        if (!users.containsKey(username)) {
            users.put(username, new User(username, password, 1));
            saveUsers();
            return true;
        }
        return false;
    }

    public int getLevel(String username) {
        if (users.containsKey(username)) {
            return users.get(username).getLevel();
        }
        return 1;
    }

    public boolean levelUp(String username, String password, int level) {
        if (users.containsKey(username) && users.get(username).getPassword().equals(password)) {
            users.get(username).setLevel(level);
            saveUsers();
            return true;
        }
        return false;
    }
}
```

## Приложение А.5.

## Файл «Game.java» Client

```
package org.example.tanchiki;

import org.example.tanchiki.GUI.SceneHelper;
import org.example.tanchiki.GUI.start.StartMenu;
import org.example.tanchiki.services.LevelService;
import org.example.tanchiki.services.threads.AddEnemy;
import org.example.tanchiki.services.threads.EnemyTankMovement;
import org.example.tanchiki.services.threads.GiveGift;
import org.example.tanchiki.services.Level;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;

import java.util.List;

public class Game extends Application {

    private static final Pane PANE = new Pane();
    private static final Scene SCENE = new Scene(PANE);

    private EnemyTankMovement enemyTankMover;
    private GiveGift giveGift;
    private AddEnemy addEnemy;
    private static List<Level> levels;

    public void start(Stage stage) {
        levels = LevelService.getLevels();
        SceneHelper.conformStage(stage, PANE, SCENE);
        startEnemyTankThread();
        startGiveGiftThread();
        startAddEnemyThread();
        StartMenu.makeMenuScene(stage, PANE, SCENE);
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }

    public static List<Level> getLevels() {
        return levels;
    }

    public void startEnemyTankThread() {
        enemyTankMover = new EnemyTankMovement();
        enemyTankMover.start();
    }

    public void startGiveGiftThread() {
        giveGift = new GiveGift();
        giveGift.start();
    }

    public void startAddEnemyThread() {
        addEnemy = new AddEnemy();
        addEnemy.start();
    }

    @Override
    public void stop() {
        if (enemyTankMover != null) enemyTankMover.stopRunning();
        if (giveGift != null) giveGift.stopRunning();
        if (addEnemy != null) addEnemy.stopRunning();
        try {
            if (enemyTankMover != null) enemyTankMover.join();
            if (giveGift != null) giveGift.join();
            if (addEnemy != null) addEnemy.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

# Приложение А.6.

## Файл «GlobalConstance.java» Client

```
package org.example.tanchiki;

public class GlobalConstance {
    public final static int WINDOWS_HEIGHT = 600;

    public final static int WINDOWS_WIDTH = 800;

    public final static int MAP_FIRST_X = 50;

    public final static int MAP_FIRST_Y = 25;

    public static int mapHeight = 500;

    public static int mapSize = 20;

    public static double scale = (double) mapHeight / mapSize;

    public final static int NORMAL_TANK_HEALTH = 1;

    public final static int STRONG_TANK_HEALTH = 2;

    public final static int PLAYER_TANK_HEALTH = 3;

    public final static int DEFAULT_PLAYER_SHOT_DAMAGE = 1;

    public static int playerShotDamage = DEFAULT_PLAYER_SHOT_DAMAGE;

    public static final String DEFAULT_MAP = "src/main/resources/maps/map1,6.txt";

    public static String pathMap = "src/main/resources/maps/";

    public static final String BACKGROUND_COLOR = "-fx-background-color: #a83250;";
    public static final String FONT_FAMILY = "Verdana";
    public static final int FONT_SIZE = 50;
    public static final String GAME_TITLE = "Танчики";

    public static final String BUTTON_STYLE = "-fx-background-color: #808080; -fx-text-fill:
white;";

    public static void updateSize() {
        if (mapHeight % mapSize != 0) {
            mapHeight = ((mapHeight / mapSize) + 1) * mapSize;
        }
        scale = (double) mapHeight / mapSize;
    }

    public static void resetPlayerShotDamage() {
        GlobalConstance.playerShotDamage = DEFAULT_PLAYER_SHOT_DAMAGE;
    }

    public static void applyPowerPlayerShotDamage() {
        playerShotDamage *= 2;
    }
}
```

## Приложение А.7.

## Файл «Level.java» Client

```
package org.example.tanchiki.services;

public record Level(String name, int num, int enemies) {
}
```

# Приложение А.8.

## Файл «LevelService.java» Client

```
package org.example.tanchiki.services;

import java.io.File;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

import static org.example.tanchiki.GlobalConstance.pathMap;

public class LevelService {

    public static List<Level> getLevels() {
        List<Level> levels = new ArrayList<>();
        File directory = new File(pathMap);

        if (directory.exists() && directory.isDirectory()) {
            File[] files = directory.listFiles((dir, name) -> name.startsWith("map"));

            if (files != null) {
                for (File file : files) {
                    String fileName = file.getName();
                    String[] parts = fileName.split("[,.]");
                    if (parts.length == 3 && parts[0].startsWith("map")) {
                        try {
                            int num = Integer.parseInt(parts[0].substring(3));
                            int enemies = Integer.parseInt(parts[1]);
                            Level level = new Level(fileName, num, enemies);
                            levels.add(level);
                        } catch (NumberFormatException e) {
                        }
                    }
                }
            }
        }
        levels.sort(Comparator.comparingInt(Level::num));
        return levels;
    }
}
```

## Приложение А.9.

## Файл «RandGenerate.java» Client

```
package org.example.tanchiki.services;

import java.util.Random;

import static org.example.tanchiki.GlobalConstance.mapSize;

public class RandGenerate {

    private final static RandGenerate INSTANCE = new RandGenerate();

    private final Random random;

    private RandGenerate() {
        random = new Random();
    }

    public int getRanBetween(int from, int to) {
        int result;
        do {
            result = random.nextInt(to);
        } while (!(result >= from && result < to));
        return result;
    }

    public int getRanEnemyLoc() {
        return switch (random.nextInt(4)) {
            case 0 -> 0;
            case 1 -> mapSize / 4 + 1;
            case 2 -> mapSize / 4 * 3;
            default -> mapSize - 1;
        };
    }

    public static RandGenerate getINSTANCE() {
        return INSTANCE;
    }
}
```

# Приложение А.10.

## Файл «ServerTask.java» Client

```
package org.example.tanchiki.services;

import java.io.*;
import java.net.Socket;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;

public class ServerTask {

    public static String loginOrSignUp(String operation, String username, String password){
        return sendRequest(operation + '/' + username + '/' + password);
    }

    public static void levelUp(String operation, String username, String password, int level){
        sendRequest(operation + '/' + username + '/' + password + '/' + level);
    }

    public static String sendRequest(String request) {
        try {
            Socket socket = new Socket("localhost", 12345);

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

            out.println(request);

            String response = in.readLine();

            in.close();
            out.close();
            socket.close();

            return response;
        } catch (IOException e) {
            showAlert("Ошибка", "Произошла ошибка при попытке подключения к серверу.");
            e.printStackTrace();
        }
        return "error_connect";
    }

    private static void showAlert(String title, String message) {
        Alert alert = new Alert(AlertType.WARNING);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```

## Приложение А.11.

## Файл «EventHandler.java» Client

```
package org.example.tanchiki.services.eventHandler;

import javafx.scene.Scene;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;

public class EventHandler {
    private static final EventHandler INSTANCE = new EventHandler();

    public static EventHandler getInstance() {
        return INSTANCE;
    }

    private EventHandler() {}

    public void attachEventHandlers(Scene scene){
        scene.addEventFilter(KeyEvent.KEY_PRESSED, keyEvent -> {
            KeyCode code = keyEvent.getCode();
            PlayerController.getInstance().handlePlayerMovements(code);
        });
        scene.addEventFilter(KeyEvent.KEY_RELEASED, keyEvent -> {
        });
    }
}
```

## Приложение А.12.

## Файл «PlayerController.java» Client

```
package org.example.tanchiki.services.eventHandler;

import org.example.tanchiki.services.GameData;
import org.example.tanchiki.models.gameObjects.Direction;
import org.example.tanchiki.models.gameObjects.tank.Tank;
import javafx.scene.input.KeyCode;

import static org.example.tanchiki.GlobalConstance.scale;

public class PlayerController {

    private static final PlayerController INSTANCE = new PlayerController();

    public static PlayerController getInstance() {
        return INSTANCE;
    }

    private PlayerController() {
    }

    public void handlePlayerMovements(KeyCode keyCode) {
        Tank player = GameData.getInstance().getPlayerTank();
        if (player != null) handlePlayerButtons(keyCode, player);
    }

    private void handlePlayerButtons(KeyCode keyCode, Tank player) {
        switch (keyCode) {
            case S:
                player.move(scale / 2, Direction.Down);
                break;
            case W:
                player.move(scale / 2, Direction.Up);
                break;
            case D:
                player.move(scale / 2, Direction.Right);
                break;
            case A:
                player.move(scale / 2, Direction.Left);
                break;
            case SPACE:
                player.fire();
                break;
            default:
                break;
        }
    }
}
```

# Приложение А.13.

## Файл «AddEnemy.java» Client

```
package org.example.tanchiki.services.threads;

import org.example.tanchiki.models.GameObjectHelper;
import org.example.tanchiki.services.GameData;
import org.example.tanchiki.services.RandGenerate;
import org.example.tanchiki.models.GameStatus;
import org.example.tanchiki.models.gameObjects.Flag;
import org.example.tanchiki.models.SceneObject;
import org.example.tanchiki.models.gameObjects.operatorGift.OperatorGift;
import org.example.tanchiki.models.gameObjects.tank.Tank;
import org.example.tanchiki.models.gameObjects.tank.TankSide;
import org.example.tanchiki.models.gameObjects.tank.TankType;
import org.example.tanchiki.models.gameObjects.wall.Wall;
import org.jetbrains.annotations.NotNull;

import java.awt.*;
import java.util.ArrayList;
import java.util.List;

import static org.example.tanchiki.GlobalConstance.*;
import static org.example.tanchiki.GlobalConstance.scale;

public class AddEnemy extends Thread {
    private volatile boolean running = true;

    public AddEnemy() {
    }

    @Override
    public void run() {
        TankType[] tankTypes = {TankType.NormalEnemy, TankType.RandomEnemy,
TankType.StrongEnemy};
        while (running) {
            try {
                sleep(1000); // wait for 1 second
                if (canAddEnemy()) {
                    Tank sceneObject = createEnemy(tankTypes);
                    addEnemyToGame(sceneObject);
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public void stopRunning() {
        running = false;
    }

    @NotNull
    private Tank createEnemy(TankType[] tankTypes) {
        Point point = findEmptyPoint();
        TankType tankType = tankTypes[RandGenerate.getINSTANCE().getRanBetween(0,
tankTypes.length)];
        return new Tank(tankType, TankSide.Enemy, point.x, point.y, scale);
    }

    private static void addEnemyToGame(Tank sceneObject) {
        synchronized (GameData.getInstance().getSceneObjects()) {
            GameData.getInstance().getSceneObjects().add(sceneObject);
            synchronized (GameData.getInstance().getEnemyTank()) {
                GameData.getInstance().getEnemyTank().add(sceneObject);
            }
        }
        GameData.getInstance().minusEnemyNumber();
    }

    private static boolean canAddEnemy() {
        return GameData.getInstance().getEnemyTank().size() < 4
                && GameData.getInstance().getEnemyNumber() > 0
                && GameData.getInstance().getGameStatus() == GameStatus.Running;
    }

    private Point findEmptyPoint() {
```

```
        int[][] nowayMap = new int[mapSize][mapSize];
        List<SceneObject> copyOfSceneObjects = new
ArrayList<>(GameData.getInstance().getSceneObjects());
        updateNoWayMap(nowayMap, copyOfSceneObjects);
        int x, y;
        do {
            x = RandGenerate.getINSTANCE().getRanEnemyLoc();
            y = 0;
        } while (nowayMap[y][x] == 1);
        x = (int) (MAP_FIRST_X + x * scale);
        y = (int) (MAP_FIRST_Y + y * scale);
        return new Point(x, y);
    }

    private void updateNoWayMap(int[][] nowayMap, List<SceneObject> sceneObjects) {
        for (SceneObject sceneObject : sceneObjects) {
            if (sceneObject instanceof Wall wall) {
                GameObjectHelper.applyWall(nowayMap, wall);
            } else if (sceneObject instanceof Tank) {
                GameObjectHelper.applyObjectToMap(nowayMap, sceneObject);
            } else if (sceneObject instanceof Flag) {
                GameObjectHelper.applyObjectToMap(nowayMap, sceneObject);
            } else if (sceneObject instanceof OperatorGift) {
                GameObjectHelper.applyObjectToMap(nowayMap, sceneObject);
            }
        }
    }
}
```

# Приложение А.14.

## Файл «EnemyTankMovement.java» Client

```
package org.example.tanchiki.services.threads;

import org.example.tanchiki.models.GameStatus;
import org.example.tanchiki.models.gameObjects.Flag;
import org.example.tanchiki.services.GameData;
import org.example.tanchiki.services.RandGenerate;
import org.example.tanchiki.models.gameObjects.Direction;
import org.example.tanchiki.models.gameObjects.tank.Tank;

import java.util.ArrayList;

public class EnemyTankMovement extends Thread {
    private volatile boolean running = true;

    private static final int MOVE_INTERVAL = 100;
    private static final int FIRE_INTERVAL = 1500;

    public EnemyTankMovement() {
    }

    @Override
    public void run() {
        long lastMoveTime = System.currentTimeMillis();
        long lastFireTime = System.currentTimeMillis();

        while (running) {
            try {
                long currentTime = System.currentTimeMillis();

                if (!GameData.getInstance().isEnemyFreezing()) {
                    if (currentTime - lastMoveTime >= MOVE_INTERVAL) {
                        moveEnemyTanks();
                        lastMoveTime = currentTime;
                    }

                    if (currentTime - lastFireTime >= FIRE_INTERVAL) {
                        fireEnemyTanks();
                        lastFireTime = currentTime;
                    }
                } else if (GameData.getInstance().getGameStatus() == GameStatus.Running) {
                    sleep(5000);
                    GameData.getInstance().setEnemyFreezing(false);
                }

                sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public void stopRunning() {
        running = false;
    }

    private void moveEnemyTanks() {
        ArrayList<Tank> enemyTanks = new ArrayList<>(GameData.getInstance().getEnemyTank());
        for (Tank enemyTank : enemyTanks) {
            Direction[] directions = Direction.values();
            Direction direction = Direction.Up;
            if (GameData.getInstance().getPlayerTank() != null) {
                if (isPlayerNear(enemyTank)) {
                    direction = findDirection(GameData.getInstance().getPlayerTank(), enemyTank);
                } else {
                    direction = findDirectionFlag(GameData.getInstance().getPlayersFlag(),
enemyTank);
                }
            }
            if (!enemyTank.move(5, direction)) {
                direction = directions[RandGenerate.getINSTANCE().getRanBetween(0,
directions.length)];
                enemyTank.move(5, direction);
            }
        }
```

```
    }

    private void fireEnemyTanks() {
        ArrayList<Tank> enemyTanks = new ArrayList<>(GameData.getInstance().getEnemyTank());
        for (Tank enemyTank : enemyTanks) {
            enemyTank.fire();
        }
    }

    private Direction findDirection(Tank target, Tank self) {
        if (Math.abs(target.getX() - self.getX()) > Math.abs(target.getY() - self.getY())) {
            return (target.getX() > self.getX()) ? Direction.Right : Direction.Left;
        } else {
            return (target.getY() > self.getY()) ? Direction.Down : Direction.Up;
        }
    }

    private Direction findDirectionFlag(Flag target, Tank self) {
        if (Math.abs(target.getX() - self.getX()) > Math.abs(target.getY() - self.getY())) {
            return (target.getX() > self.getX()) ? Direction.Right : Direction.Left;
        } else {
            return (target.getY() > self.getY()) ? Direction.Down : Direction.Up;
        }
    }

    private boolean isPlayerNear(Tank self) {
        Tank target = GameData.getInstance().getPlayerTank();
        double distance1 = Math.pow(Math.abs(target.getX() - self.getX()), 2) +
                Math.pow(Math.abs(target.getY() - self.getY()), 2);
        Flag target2 = GameData.getInstance().getPlayersFlag();
        double distance2 = Math.pow(Math.abs(target2.getX() - self.getX()), 2) +
                Math.pow(Math.abs(target2.getY() - self.getY()), 2);

        return distance1 < distance2;
    }
}
```

## Приложение А.15.

## Файл «GiveGift.java» Client

```
package org.example.tanchiki.services.threads;

import org.example.tanchiki.models.GameObjectHelper;
import org.example.tanchiki.services.GameData;
import org.example.tanchiki.services.RandGenerate;
import org.example.tanchiki.models.gameObjects.Flag;
import org.example.tanchiki.models.SceneObject;
import org.example.tanchiki.models.gameObjects.operatorGift.GifType;
import org.example.tanchiki.models.gameObjects.operatorGift.OperatorGift;
import org.example.tanchiki.models.gameObjects.tank.Tank;
import org.example.tanchiki.models.gameObjects.wall.Wall;
import org.jetbrains.annotations.NotNull;

import java.awt.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

import static org.example.tanchiki.GlobalConstance.*;
import static org.example.tanchiki.GlobalConstance.scale;

public class GiveGift extends Thread {
    private volatile boolean running = true;

    public GiveGift() {
    }

    @Override
    public void run() {
        while (running) {
            try {
                sleep(1000); // wait for 1 second
                if (GameData.getInstance().sendGift) {
                    OperatorGift operatorGift = createOperatorGift();
                    GameData.getInstance().getSceneObjects().add(operatorGift);
                    GameData.getInstance().sendGift = false;
                    sleep(5000);
                    operatorGift.setExpired(true);
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public void stopRunning() {
        running = false;
    }

    @NotNull
    private OperatorGift createOperatorGift() {
        Point point = findEmptyPoint();
        GifType[] gifTypes = GifType.values();
        GifType selected = gifTypes[new Random().nextInt(gifTypes.length)];
        OperatorGift operatorGift = new OperatorGift(point.x, point.y, scale, selected);
        return operatorGift;
    }

    private Point findEmptyPoint() {
        int[][] nowayMap = new int[mapSize][mapSize];
        List<SceneObject> copyOfSceneObjects = new
ArrayList<>(GameData.getInstance().getSceneObjects());
        updateNoWayMap(nowayMap, copyOfSceneObjects);
        int x, y;
        do {
            x = RandGenerate.getINSTANCE().getRanBetween(0, mapSize);
            y = RandGenerate.getINSTANCE().getRanBetween(0, mapSize);
        } while (nowayMap[y][x] == 1);
        x = (int) (MAP_FIRST_X + x * scale);
        y = (int) (MAP_FIRST_Y + y * scale);
        return new Point(x, y);
    }

    private void updateNoWayMap(int[][] nowayMap, List<SceneObject> sceneObjects) {
```

```
        for (SceneObject sceneObject : sceneObjects) {
            if (sceneObject instanceof Wall wall) {
                GameObjectHelper.applyWall(nowayMap, wall);
            } else if (sceneObject instanceof Tank) {
                GameObjectHelper.applyObjectToMap(nowayMap, sceneObject);
            } else if (sceneObject instanceof Flag) {
                GameObjectHelper.applyObjectToMap(nowayMap, sceneObject);
            } else if (sceneObject instanceof OperatorGift) {
                GameObjectHelper.applyObjectToMap(nowayMap, sceneObject);
            }
        }
    }
}
```

## Приложение А.16.

## Файл «GameData.java» Client

```
package org.example.tanchiki.services;

import org.example.tanchiki.Game;
import org.example.tanchiki.GlobalConstance;
import org.example.tanchiki.models.GameStatus;
import org.example.tanchiki.models.Player;
import org.example.tanchiki.models.gameObjects.Flag;
import org.example.tanchiki.models.SceneObject;
import org.example.tanchiki.models.gameObjects.operatorGift.GifType;
import org.example.tanchiki.models.gameObjects.tank.Tank;
import org.example.tanchiki.GUI.game.GameEnvironmentHelper;

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

import static org.example.tanchiki.GlobalConstance.*;

public class GameData {

    private final static GameData INSTANCE = new GameData();

    private GameData() {

    }

    public static GameData getInstance() {
        return INSTANCE;
    }

    private List<SceneObject> sceneObjects = new ArrayList<>();

    private GameStatus gameStatus = GameStatus.Stop;

    private Tank playerTank = null;

    private Flag playersFlag = null;

    private ArrayList<Tank> enemyTank = new ArrayList<>();

    private String[][] map;

    private int score = 0;

    private boolean enemyFreezing = false;

    public boolean sendGift = false;

    private Player currentPlayer = null;

    private int level;

    private int enemyNumber = 4;

    public Tank getPlayerTank(){
        return playerTank;
    }

    public void setPlayerTank(Tank tank) {
        playerTank = tank;
    }

    public ArrayList<Tank> getEnemyTank() {
        return enemyTank;
    }

    public List<SceneObject> getSceneObjects() {
        return sceneObjects;
    }

    public int getScore() {
        return score;
    }

    public void setScore(int score) {
```

```
        this.score = score;
    }

    public boolean isEnemyFreezing() {
        return enemyFreezing;
    }

    public void setEnemyFreezing(boolean enemyFreezing) {
        this.enemyFreezing = enemyFreezing;
    }

    public GameStatus getGameStatus() {
        return gameStatus;
    }

    public void setGameStatus(GameStatus gameStatus) {
        this.gameStatus = gameStatus;
    }

    public String[][] getMap() {
        return map;
    }

    public void resetGame(int level) {
        this.level = level;
        sceneObjects = new ArrayList<>();
        gameStatus = GameStatus.Running;
        playerTank = null;
        enemyTank = new ArrayList<>();
        score = 0;
        enemyFreezing = true;
        sendGift = false;
        applyLevel(level);
        map = GameEnvironmentHelper.readMapFile(pathMap + Game.getLevels().get(level -
1).name());
    }

    public void resetAll(int level) {
        sceneObjects = new ArrayList<>();
        gameStatus = GameStatus.Running;
        playerTank = null;
        enemyTank = new ArrayList<>();
        score = 0;
        enemyFreezing = true;
        sendGift = false;
        applyLevel(level);
    }

    public Player getCurrentPlayer() {
        return currentPlayer;
    }

    public void setCurrentPlayer(Player currentPlayer) {
        this.currentPlayer = currentPlayer;
    }

    public int getLevel() {
        return level;
    }

    public int getEnemyNumber() {
        return enemyNumber;
    }

    public void minusEnemyNumber() {
        enemyNumber--;
    }

    public void applyGift(GifType gifType) {
        switch (gifType) {
            case Enemy_Freezing -> {
                enemyFreezing = true;
            }
            case Extra_Shot -> {
                GlobalConstance.applyPowerPlayerShotDamage();
            }
            default -> {
                playerTank.setHealth(playerTank.getHealth() + 1);
            }
        }
```

```
    }

    private void applyLevel(int level) {
        if (level == 0) {
            enemyNumber = 0;
        } else {
            enemyNumber = Game.getLevels().get(level - 1).enemies();
        }
    }

    public Flag getPlayersFlag() {
        return playersFlag;
    }

    public void setPlayersFlag(Flag playersFlag) {
        this.playersFlag = playersFlag;
    }
}
```

## Приложение А.17.

## Файл «SceneObject.java» Client

```
package org.example.tanchiki.models;

import javafx.scene.Node;

public interface SceneObject {
    double getX();

    double getY();

    void update();

    Node getNode();

    boolean collidesWith(SceneObject object);

    boolean isVisible();
}
```

## Приложение А.18.

## Файл «GifType.java» Client

```
package org.example.tanchiki.models.gameObjects.operatorGift;

public enum GifType {
    Extra_Health, Enemy_Freezing, Extra_Shot
}
```

## Приложение А.19.

## Файл «OperatorGift.java» Client

```
package org.example.tanchiki.models.gameObjects.operatorGift;

import org.example.tanchiki.services.GameData;
import org.example.tanchiki.models.GameObjectHelper;
import org.example.tanchiki.models.SceneObject;
import org.example.tanchiki.models.gameObjects.tank.Tank;
import org.example.tanchiki.models.gameObjects.tank.TankSide;
import javafx.scene.Node;
import javafx.scene.image.ImageView;

public class OperatorGift implements SceneObject {

    private final ImageView imageView;

    private final GifType gifType;

    private final double x;

    private final double y;

    private final double scale;

    private boolean isCollision = false;

    private boolean expired = false;

    public OperatorGift(double x, double y, double scale, GifType gifType) {
        imageView = new ImageView(GameObjectHelper.attachGiftImage(gifType));
        this.scale = scale;
        imageView.setFitWidth(scale);
        imageView.setFitHeight(scale);
        imageView.setY(y);
        imageView.setX(x);
        this.x = x;
        this.y = y;
        this.gifType = gifType;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public void setExpired(boolean expired) {
        this.expired = expired;
    }

    @Override
    public void update() {

    }

    @Override
    public Node getNode() {
        return imageView;
    }

    @Override
    public boolean collidesWith(SceneObject object) {
        double shotLeft = x;
        double shotRight = x + scale;
        double shotTop = y;
        double shotBottom = y + scale;
        Tank tank = (Tank) object;
        if (tank.getTankSide() != TankSide.Player) {
            return false;
        }
        double tankScale = tank.getScale();
        double tankLeft = tank.getX();
        double tankRight = tank.getX() + tankScale;
        double tankTop = tank.getY();
```

```
        double tankBottom = tank.getY() + tankScale;
        if (shotLeft < tankRight && shotRight > tankLeft && shotTop < tankBottom && shotBottom >
tankTop) {
            isCollision = true;
            GameData.getInstance().applyGift(gifType);
            return true;
        }
        return false;
    }

    @Override
    public boolean isVisible() {
        if (isCollision) {
            return false;
        }
        return !expired;
    }
}
```

## Приложение А.20.

## Файл «TankSide.java» Client

```
package org.example.tanchiki.models.gameObjects.tank;

public enum TankSide {
    Player, Enemy
}
```

Приложение А.21.

Файл «TankType.java» Client

```
package org.example.tanchiki.models.gameObjects.tank;

public enum TankType {
    Player, NormalEnemy, StrongEnemy, RandomEnemy
}
```

## Приложение А.22.

## Файл «Tank.java» Client

```
package org.example.tanchiki.models.gameObjects.tank;

import org.example.tanchiki.services.GameData;
import org.example.tanchiki.services.RandGenerate;
import org.example.tanchiki.models.gameObjects.Shot;
import org.example.tanchiki.models.gameObjects.Direction;
import org.example.tanchiki.models.GameObjectHelper;
import org.example.tanchiki.models.SceneObject;
import javafx.scene.Node;
import javafx.scene.image.ImageView;

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

import static org.example.tanchiki.GlobalConstance.*;

public class Tank implements SceneObject {

    private final ImageView imageView;

    private final TankType tankType;

    private final TankSide tankSide;

    private int health;

    private int firstHealth;

    private Direction direction = Direction.Up;

    private double x;

    private double y;

    private final double scale;

    private long lastFireTime = 0;
    private static final int FIRE_INTERVAL = 300;

    private long lastMoveTime = 0;
    private static final int MOVE_INTERVAL = 100;

    public Tank(TankType tankType, TankSide tankSide, double x, double y, double scale) {
        imageView = new ImageView(GameObjectHelper.attachTankImage(tankType));
        this.scale = scale;
        imageView.setFitWidth(scale);
        imageView.setFitHeight(scale);
        this.tankType = tankType;
        this.tankSide = tankSide;
        this.x = x;
        this.y = y;
        switch (tankType) {
            case Player -> health = PLAYER_TANK_HEALTH;
            case NormalEnemy -> health = NORMAL_TANK_HEALTH;
            case StrongEnemy -> health = STRONG_TANK_HEALTH;
            default -> health = RandGenerate.getINSTANCE().getRanBetween(NORMAL_TANK_HEALTH,
STRONG_TANK_HEALTH + 1);
        }
        firstHealth = health;
    }

    public double getScale() {
        return scale;
    }

    public int getHealth() {
        return health;
    }

    public void setHealth(int health) {
        this.health = health;
        firstHealth = health;
    }
```

```
    public boolean move(double speed, Direction direction) {
        long currentTime = System.currentTimeMillis();
        if (currentTime - lastMoveTime < MOVE_INTERVAL) {
            return false;
        }
        lastMoveTime = currentTime;

        this.direction = direction;
        double oldY = y;
        double oldX = x;
        if (direction == Direction.Up) {
            y -= speed;
        }
        if (direction == Direction.Down) {
            y += speed;
        }
        if (direction == Direction.Right) {
            x += speed;
        }
        if (direction == Direction.Left) {
            x -= speed;
        }
        List<SceneObject> copyOfSceneObjects = new
ArrayList<>(GameData.getInstance().getSceneObjects());
        for (SceneObject sceneObject : copyOfSceneObjects) {
            if (sceneObject.collidesWith(this) && this != sceneObject) {
                y = oldY;
                x = oldX;
                return false;
            }
        }
        return true;
    }

    public void fire() {
        long currentTime = System.currentTimeMillis();
        if (currentTime - lastFireTime >= FIRE_INTERVAL) {
            double shotX = x + scale / 3;
            double shotY = y + scale / 3;

            switch (direction) {
                case Up -> shotY -= scale / 3;
                case Down -> shotY += scale / 3;
                case Left -> shotX -= scale / 3;
                case Right -> shotX += scale / 3;
            }

            GameData.getInstance().getSceneObjects().add(new Shot(tankSide, shotX, shotY, scale /
3, playerShotDamage, direction));
            lastFireTime = currentTime;
        }
    }

    public boolean collidesWith(SceneObject object) {
        if (object == this) {
            return false;
        }
        if (object instanceof Tank) {
            return collisionWithTank(object);
        }
        return false;
    }

    public boolean collisionWithTank(SceneObject object) {
        Tank tank = (Tank) object;
        //if (tank.getTankSide() == tankSide) {
        //    return false;
        //}
        double tankScale = tank.getScale();
        double thisLeft = x;
        double thisRight = x + scale;
        double thisTop = y;
        double thisBottom = y + scale;
        double tankLeft = tank.getX();
        double tankRight = tank.getX() + tankScale;
        double tankTop = tank.getY();
        double tankBottom = tank.getY() + tankScale;
        return thisLeft < tankRight && thisRight > tankLeft && thisTop < tankBottom && thisBottom
> tankTop;
    }
```

```
    @Override
    public boolean isVisible() {
        if (isDead()) {
            if (tankSide == TankSide.Player) {
                GameData.getInstance().setPlayerTank(null);
            } else {
                if (firstHealth == NORMAL_TANK_HEALTH) {
                    GameData.getInstance().setScore(GameData.getInstance().getScore() + 100);
                } else {
                    GameData.getInstance().setScore(GameData.getInstance().getScore() + 200);
                }
                if (tankType == TankType.RandomEnemy) {
                    GameData.getInstance().sendGift = true;
                }
                GameData.getInstance().getEnemyTank().remove(this);
            }
            return false;
        }
        return true;
    }

    public void takeDamage(int damage) {
        health -= damage;
    }

    public boolean isDead() {
        return (health <= 0);
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public TankSide getTankSide() {
        return tankSide;
    }

    @Override
    public void update() {
        imageView.setX(x);
        imageView.setY(y);
        switch (direction) {
            case Down -> imageView.setRotate(180);
            case Right -> imageView.setRotate(90);
            case Left -> imageView.setRotate(270);
            default -> imageView.setRotate(0);
        }
    }

    @Override
    public Node getNode() {
        return imageView;
    }
}
```

## Приложение А.23.

## Файл «WallType.java» Client

```
package org.example.tanchiki.models.gameObjects.wall;

public enum WallType {
    Normal, Iron
}
```

## Приложение А.24.

## Файл «Wall.java» Client

```
package org.example.tanchiki.models.gameObjects.wall;

import org.example.tanchiki.models.GameObjectHelper;
import org.example.tanchiki.models.SceneObject;
import org.example.tanchiki.models.gameObjects.tank.Tank;
import javafx.scene.Node;
import javafx.scene.image.ImageView;

public class Wall implements SceneObject {

    private final ImageView imageView;

    private int health = 2;

    private final WallType wallType;

    private final double x;

    private final double y;

    private final double scale;

    public Wall(WallType wallType, double x, double y, double scale) {
        this.wallType = wallType;
        this.x = x;
        this.y = y;
        this.scale = scale;
        imageView = new ImageView(GameObjectHelper.attachWallImage(wallType));
        imageView.setFitWidth(scale);
        imageView.setFitHeight(scale);
    }


    public void takeDamage(int damage) {
        if (wallType != WallType.Iron) {
            health -= damage;
            imageView.setImage(GameObjectHelper.normalWallDamaged);
        }
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public double getScale() {
        return scale;
    }

    @Override
    public void update() {
        imageView.setX(x);
        imageView.setY(y);
    }

    @Override
    public Node getNode() {
        return imageView;
    }

    @Override
    public boolean collidesWith(SceneObject object) {
        if (object instanceof Tank tank) {
            double tankScale = tank.getScale();
            double wallRight = x + scale;
            double wallBottom = y + scale;
            double tankLeft = tank.getX();
            double tankRight = tank.getX() + tankScale;
            double tankTop = tank.getY();
            double tankBottom = tank.getY() + tankScale;
```

```
            return x < tankRight && wallRight > tankLeft && y < tankBottom && wallBottom >
tankTop;
        }
        return false;
    }

    @Override
    public boolean isVisible() {
        return health > 0;
    }
}
```

Приложение А.25.

Файл «Direction.java» Client

```
package org.example.tanchiki.models.gameObjects;

public enum Direction {
    Up, Down, Right, Left
}
```

## Приложение А.26.

## Файл «Flag.java» Client

```
package org.example.tanchiki.models.gameObjects;

import org.example.tanchiki.models.GameObjectHelper;
import org.example.tanchiki.models.SceneObject;
import javafx.scene.Node;
import javafx.scene.image.ImageView;

public class Flag implements SceneObject {

    private int health = 3;

    private final double x;

    private final double y;

    private final ImageView imageView;

    private final double scale;

    public Flag(double x, double y, double scale) {
        imageView = new ImageView(GameObjectHelper.flag1);
        this.scale = scale;
        imageView.setFitWidth(scale);
        imageView.setFitHeight(scale);
        this.x = x;
        this.y = y;
    }

    public double getScale() {
        return scale;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public void takeDamage(int damage) {
        health -= damage;
    }

    @Override
    public void update() {
        imageView.setX(x);
        imageView.setY(y);

    }

    @Override
    public Node getNode() {
        return imageView;
    }

    @Override
    public boolean collidesWith(SceneObject object) {
        return false;
    }

    @Override
    public boolean isVisible() {
        return health > 0;
    }
}
```

# Приложение А.27.

## Файл «Shot.java» Client

```
package org.example.tanchiki.models.gameObjects;

import org.example.tanchiki.models.GameObjectHelper;
import org.example.tanchiki.models.SceneObject;
import org.example.tanchiki.services.GameData;
import org.example.tanchiki.GlobalConstance;
import org.example.tanchiki.models.gameObjects.tank.Tank;
import org.example.tanchiki.models.gameObjects.tank.TankSide;
import org.example.tanchiki.models.gameObjects.wall.Wall;
import javafx.scene.Node;
import javafx.scene.image.ImageView;

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

public class Shot implements SceneObject {

    private final ImageView imageView;

    private final TankSide tankSide;

    private final int damage;

    private Direction direction;

    private double x;

    private double y;

    private final double scale;

    private boolean isCollision = false;

    public Shot(TankSide tankSide, double x, double y, double scale, int damage, Direction
direction) {
        this.tankSide = tankSide;
        this.x = x;
        this.y = y;
        this.scale = scale;
        this.damage = damage;
        this.direction = direction;
        this.imageView = new ImageView(GameObjectHelper.attachShotImage());
        imageView.setFitWidth(scale);
        imageView.setFitHeight(scale);
    }

    private void checkCollisions() {
        List<SceneObject> sceneObjects = new
ArrayList<>(GameData.getInstance().getSceneObjects());
        for (SceneObject sceneObject : sceneObjects) {
            if (collidesWith(sceneObject)) {
                isCollision = true;
                break;
            }
        }
    }

    public void move(double speed, Direction direction) {
        this.direction = direction;
        switch (direction) {
            case Up -> y -= speed;
            case Down -> y += speed;
            case Right -> x += speed;
            case Left -> x -= speed;
        }
        checkCollisions();
    }

    public boolean collidesWith(SceneObject object) {
        if (object instanceof Tank tank) {
            return collisionWithTank(tank);
        } else if (object instanceof Wall wall) {
            return collisionWithWall(wall);
        } else if (object instanceof Flag flag) {
```

```
            return collisionWithFlag(flag);
        }
        return false;
    }

    private boolean collisionWithFlag(Flag flag) {
        if (tankSide == TankSide.Player) {
            return false;
        }
        return checkCollision(flag.getX(), flag.getY(), flag.getScale(), flag::takeDamage);
    }

    private boolean collisionWithTank(Tank tank) {
        if (tank.getTankSide() == tankSide) {
            return false;
        }
        boolean collision = checkCollision(tank.getX(), tank.getY(), tank.getScale(),
tank::takeDamage);
        if (collision && tank.getTankSide() == TankSide.Player) {
            GlobalConstance.resetPlayerShotDamage();
        }
        return collision;
    }

    private boolean collisionWithWall(Wall wall) {
        return checkCollision(wall.getX(), wall.getY(), wall.getScale(), wall::takeDamage);
    }

    private boolean checkCollision(double objX, double objY, double objScale, Damageable
damageable) {
        double shotLeft = x;
        double shotRight = x + scale;
        double shotTop = y;
        double shotBottom = y + scale;

        double objLeft = objX;
        double objRight = objX + objScale;
        double objTop = objY;
        double objBottom = objY + objScale;

        if (shotLeft < objRight && shotRight > objLeft && shotTop < objBottom && shotBottom >
objTop) {
            damageable.takeDamage(damage);
            return true;
        }
        return false;
    }

    @Override
    public boolean isVisible() {
        return !isCollision;
    }

    @Override
    public double getX() {
        return x;
    }

    @Override
    public double getY() {
        return y;
    }

    @Override
    public void update() {
        move(5, direction);
        imageView.setX(x);
        imageView.setY(y);
        switch (direction) {
            case Down -> imageView.setRotate(180);
            case Right -> imageView.setRotate(90);
            case Left -> imageView.setRotate(270);
            default -> imageView.setRotate(0);
        }
    }

    @Override
    public Node getNode() {
        return imageView;
    }
```

```
    @FunctionalInterface
    private interface Damageable {
        void takeDamage(int damage);
    }
}
```

## Приложение А.28.

## Файл «Player.java» Client

```
package org.example.tanchiki.models;

import org.example.tanchiki.Game;
import org.example.tanchiki.services.ServerTask;

import java.util.Objects;

public class Player {
    private final String username;

    private final String password;

    private int lastLevel;

    public Player(String username, String password) {
        this.username = username;
        this.password = password;
        this.lastLevel = 1;
    }

    public Player(String username, String password, int lastLevel) {
        this.username = username;
        this.password = password;
        this.lastLevel = lastLevel;
    }

    public void levelUp() {
        if (lastLevel < Game.getLevels().getLast().num()) {
            ServerTask.levelUp("levelup", username, password, ++lastLevel);
        }
    }

    public String getUsername() {
        return username;
    }

    public int getLastStage() {
        return lastLevel;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || getClass() != o.getClass()) {
            return false;
        }
        Player player = (Player) o;
        return username.equals(player.username) && password.equals(player.password);
    }

    @Override
    public int hashCode() {
        return Objects.hash(username, password);
    }
}
```

## Приложение А.29.

## Файл «GameStatus.java» Client

```
package org.example.tanchiki.models;

public enum GameStatus {
    Start, Stop, Running
}
```

# Приложение А.30.

## Файл «GameObjectHelper.java» Client

```
package org.example.tanchiki.models;

import org.example.tanchiki.models.gameObjects.operatorGift.GifType;
import org.example.tanchiki.models.gameObjects.tank.TankType;
import org.example.tanchiki.models.gameObjects.wall.Wall;
import org.example.tanchiki.models.gameObjects.wall.WallType;
import javafx.scene.image.Image;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.Objects;

import static org.example.tanchiki.GlobalConstance.*;

public class GameObjectHelper {

    private static Image playerTank;

    private static Image normalTank = null;

    private static Image randomTank = null;

    private static Image strongTank = null;

    private static Image normalWall = null;

    public static Image normalWallDamaged = null;

    private static Image ironWall = null;

    private static Image shot = null;

    public static Image flag1 = null;

    public static Image extraHealth = null;

    public static Image enemyFreezing = null;

    public static Image extraShot = null;

    static {
        try {
            playerTank = new Image(new FileInputStream("src/main/resources/images/tank-
yellow/yellow-tank-up.gif"));
            normalTank = new Image(new FileInputStream("src/main/resources/images/tank-
white/white-tank-up.gif"));
            strongTank = new Image(new FileInputStream("src/main/resources/images/tank-
green/green-tank-up.gif"));
            randomTank = new Image(new FileInputStream("src/main/resources/images/tank-red/red-
tank-up.gif"));
            normalWall = new Image(new FileInputStream("src/main/resources/images/wall.png"));
            normalWallDamaged = new Image(new FileInputStream("src/main/resources/images/wall-
damaged.png"));
            ironWall = new Image(new FileInputStream("src/main/resources/images/wallIron.png"));
            shot = new Image(new FileInputStream("src/main/resources/images/shot/shot-up.gif"));
            flag1 = new Image(new FileInputStream("src/main/resources/images/home.gif"));
            extraHealth = new Image(new
FileInputStream("src/main/resources/images/gift/giftTank.jpeg"));
            enemyFreezing = new Image(new
FileInputStream("src/main/resources/images/gift/giftTime.png"));
            extraShot = new Image(new
FileInputStream("src/main/resources/images/gift/giftStar.jpeg"));
        } catch (FileNotFoundException e) {
            System.out.println("Error get image " + e);
        }
    }

    public static Image attachTankImage(TankType tankType) {
        switch (tankType) {
            case Player -> {
                return playerTank;
            }
            case StrongEnemy -> {
                return strongTank;
```

```
            }
            case RandomEnemy -> {
                return randomTank;
            }
            default -> {
                return normalTank;
            }
        }
    }

    public static Image attachWallImage(WallType wallType) {
        if (Objects.requireNonNull(wallType) == WallType.Iron) {
            return ironWall;
        }
        return normalWall;
    }

    public static Image attachShotImage() {
        return shot;
    }

    public static Image attachGiftImage(GifType gifType) {
        switch (gifType) {
            case Enemy_Freezing -> {
                return enemyFreezing;
            }
            case Extra_Shot -> {
                return extraShot;
            }
            default -> {
                return extraHealth;
            }
        }
    }

    public static void applyObjectToMap(int[][] nowayMap, SceneObject object) {
        int j = (int) ((object.getX() - MAP_FIRST_X) / scale);
        int i = (int) ((object.getY() - MAP_FIRST_Y) / scale);
        nowayMap[i][j] = 1;
    }

    public static void applyWall(int[][] nowayMap, Wall wall) {
        if (wall.getX() > MAP_FIRST_X && wall.getX() < MAP_FIRST_X + mapHeight
                && wall.getY() > MAP_FIRST_Y && wall.getY() < MAP_FIRST_Y + mapHeight) {
            int j = (int) ((wall.getX() - MAP_FIRST_X) / scale);
            int i = (int) ((wall.getY() - MAP_FIRST_Y) / scale);
            nowayMap[i][j] = 1;
        }
    }
}
```

## Приложение А.31.

## Файл «SceneHelper.java» Client

```
package org.example.tanchiki.GUI;

import org.example.tanchiki.models.gameObjects.tank.Tank;
import org.example.tanchiki.services.GameData;
import org.example.tanchiki.services.eventHandler.EventHandler;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import org.jetbrains.annotations.NotNull;

import static org.example.tanchiki.GlobalConstance.*;

public class SceneHelper {

    public static void conformStage(Stage stage, Pane pane, Scene scene) {
        pane.setStyle("-fx-background-color: #708090;");
        EventHandler.getInstance().attachEventHandlers(scene);
        stage.setScene(scene);
        stage.setTitle(GAME_TITLE);
        stage.setHeight(WINDOWS_HEIGHT);
        stage.setWidth(WINDOWS_WIDTH);
        stage.setMaxHeight(WINDOWS_HEIGHT);
        stage.setMaxWidth(WINDOWS_WIDTH);
        stage.setMinHeight(WINDOWS_HEIGHT);
        stage.setMinWidth(WINDOWS_WIDTH);
    }

    public static void makeGameScene(Pane pane) {
        Rectangle square = new Rectangle(mapHeight, mapHeight);
        square.setFill(Color.BLACK);
        square.setStroke(Color.WHITE);
        pane.getChildren().add(square);
        square.setX(MAP_FIRST_X);
        square.setY(MAP_FIRST_Y);
        Text scoreTitle = new Text("Счёт: " + String.valueOf(GameData.getInstance().getScore()));
        scoreTitle.setFill(Color.WHITE);
        scoreTitle.setFont(Font.font("Verdana", FontWeight.BOLD, 20));
        scoreTitle.setX(WINDOWS_WIDTH - 225);
        scoreTitle.setY(250);
        pane.getChildren().add(scoreTitle);
        healthData(pane);
        userData(pane);
        tankShotData(pane);
        stageTitle(pane);
    }

    private static void stageTitle(Pane pane) {
        Text level = new Text("Уровень " + String.valueOf(GameData.getInstance().getLevel()));
        level.setFill(Color.WHITE);
        level.setFont(Font.font("Verdana", FontWeight.BOLD, 30));
        level.setX(WINDOWS_WIDTH - 225);
        level.setY(50);
        pane.getChildren().addAll(level);
    }

    private static void userData(Pane pane) {
        if (GameData.getInstance().getCurrentPlayer() != null) {
            Text username = createUsernameText();
            pane.getChildren().addAll(username);
        }
    }

    @NotNull
    private static Text createUsernameText() {
        Text username = new Text(GameData.getInstance().getCurrentPlayer().getUsername());
        username.setFill(Color.WHITE);
        username.setFont(Font.font("Verdana", FontWeight.BOLD, 20));
        username.setX(WINDOWS_WIDTH - 225);
        username.setY(100);
```

```
        return username;
    }

    private static void healthData(Pane pane) {
        if (GameData.getInstance().getPlayerTank() != null){
            Tank tank = GameData.getInstance().getPlayerTank();
            Text healthTitle = createHealthTitle(tank);
            pane.getChildren().add(healthTitle);
        }
    }

    @NotNull
    private static Text createHealthTitle(Tank tank) {
        Text healthTitle = new Text("Жизни: " + String.valueOf(tank.getHealth()));
        healthTitle.setFill(Color.WHITE);
        healthTitle.setFont(Font.font("Verdana", FontWeight.BOLD, 15));
        healthTitle.setX(WINDOWS_WIDTH - 225);
        healthTitle.setY(200);
        return healthTitle;
    }

    private static void tankShotData(Pane pane) {
        Text shotDamageTitle = shotDamageTitle();
        pane.getChildren().add(shotDamageTitle);
    }

    @NotNull
    private static Text shotDamageTitle() {
        Text healthTitle = new Text("Сила выстрела: " + String.valueOf(playerShotDamage));
        healthTitle.setFill(Color.WHITE);
        healthTitle.setFont(Font.font("Verdana", FontWeight.BOLD, 15));
        healthTitle.setX(WINDOWS_WIDTH - 225);
        healthTitle.setY(150);
        return healthTitle;
    }
}
```

## Приложение А.32.

## Файл «EndGameScene.java» Client

```
package org.example.tanchiki.GUI;

import org.example.tanchiki.GUI.start.StartMenu;
import org.example.tanchiki.services.GameData;
import javafx.animation.AnimationTimer;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.Stage;

import static org.example.tanchiki.GlobalConstance.*;

public class EndGameScene {
    public static void makeEndGame(boolean win, Pane pane, Stage stage, Scene scene) {
        if (win) {
            createMessage(win, pane);
            updateLevel();
        } else {
            createMessage(win, pane);
        }
        new AnimationTimer() {
            int timer = 3;
            private long lastUpdate = 0;

            public void handle(long currentNanoTime) {
                if (currentNanoTime - lastUpdate >= 1_000_000_000) {
                    timer--;

                    if (timer == 0) {
                        this.stop();
                        GameData.getInstance().resetAll(0);
                        StartMenu.makeLevelsMenuScene(stage, pane, scene);
                    }
                    lastUpdate = currentNanoTime;
                }
            }
        }.start();
    }

    private static void createMessage(boolean win, Pane pane) {
        Text gameOver;
        if (win) {
            gameOver = new Text("Победа!");
            gameOver.setFill(Color.GREEN);
        } else {
            gameOver = new Text("Поражение!");
            gameOver.setFill(Color.RED);
        }
        gameOver.setFont(new Font(50));
        gameOver.setX(MAP_FIRST_X + (double) mapHeight / 2);
        gameOver.setY(MAP_FIRST_Y + (double) mapHeight / 2);
        pane.getChildren().add(gameOver);
    }

    private static void updateLevel() {
        if (GameData.getInstance().getLevel() ==
GameData.getInstance().getCurrentPlayer().getLastStage()) {
            GameData.getInstance().getCurrentPlayer().levelUp();
        }
    }
}
```

# Приложение А.33.

## Файл «StartMenuHelper.java» Client

```
package org.example.tanchiki.GUI.start;

import org.example.tanchiki.Game;
import org.example.tanchiki.GUI.game.GamePage;
import org.example.tanchiki.services.GameData;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import org.example.tanchiki.services.Level;

import java.util.List;

import static org.example.tanchiki.GlobalConstance.*;

public class StartMenuHelper {

    public static void makeLevelsButton(Stage stage, Pane pane, Scene scene, HBox bottomOfPage) {
        HBox levelsButtons = new HBox();
        VBox optionButtons = createOptionsButton();
        VBox levelsButtonsLeft = new VBox();
        VBox levelsButtonsRight = new VBox();
        levelsButtons.setAlignment(Pos.CENTER);
        levelsButtons.setPadding(new Insets(30));
        List<Level> levels = Game.getLevels();
        configureVbox(levelsButtonsLeft);
        configureVbox(levelsButtonsRight);
        for (int i = 0; i < levels.size(); i++) {
            createLevelButton(levels, i, stage, pane, scene, levelsButtonsLeft,
levelsButtonsRight);
            if (levels.get(i).num() == GameData.getInstance().getCurrentPlayer().getLastStage())
{
                break;
            }
        }
        levelsButtons.setAlignment(Pos.CENTER_RIGHT);
        levelsButtons.getChildren().addAll(levelsButtonsLeft, levelsButtonsRight, optionButtons);
        bottomOfPage.getChildren().add(levelsButtons);
    }

    private static void createLevelButton(List<Level> levels, int i, Stage stage, Pane pane,
Scene scene,
                                          VBox levelsButtonsLeft, VBox levelsButtonsRight) {
        Button levelButton = LoginPage.createButton("Уровень " + levels.get(i).num(),
BUTTON_STYLE,
                e -> GamePage.startGame(levels.get(i).num(), stage, pane, scene,
GameData.getInstance(), PLAYER_TANK_HEALTH));
        if (i < levels.size() / 2) {
            levelsButtonsLeft.getChildren().add(levelButton);
        } else {
            levelsButtonsRight.getChildren().add(levelButton);
        }
    }

    private static VBox createOptionsButton() {
        VBox optionButtons = new VBox();
        configureVbox(optionButtons);
        return optionButtons;
    }

    public static VBox createLoginBox(Scene scene, Pane pane, Stage stage) {
        VBox loginBox = new VBox();
        TextField usernameField = new TextField();
        usernameField.setPromptText("Имя пользователя");
        PasswordField passwordField = new PasswordField();
        passwordField.setPromptText("Пароль");

        loginBox.getChildren().addAll(usernameField, passwordField);
        LoginPage.loginBoxButtons(loginBox, usernameField, passwordField, scene, pane, stage);
        configureVbox(loginBox);
```

```
        return loginBox;
    }

    public static VBox createSignOutBox(Scene scene, Pane pane, Stage stage) {
        VBox signOutBox = new VBox();
        LoginPage.signOutBoxButtons(signOutBox, scene, pane, stage);
        configureVbox(signOutBox);
        return signOutBox;
    }

    private static void configureVbox(VBox vBox) {
        vBox.setSpacing(10);
        vBox.setAlignment(Pos.CENTER);
        vBox.setPadding(new Insets(10));
    }
}
```

## Приложение А.34.

## Файл «StartMenu.java» Client

```
package org.example.tanchiki.GUI.start;

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.stage.Stage;

import static org.example.tanchiki.GlobalConstance.*;

public class StartMenu {

    public static void makeMenuScene(Stage stage, Pane pane, Scene scene) {
        BorderPane root = createBorderPane();
        VBox page = new VBox();
        HBox topOfPage = createTitle();
        HBox bottomOfPage = new HBox();
        VBox loginBox = StartMenuHelper.createLoginBox(scene, pane, stage);
        bottomOfPage.getChildren().add(loginBox);
        page.getChildren().addAll(topOfPage, bottomOfPage);
        root.setCenter(page);
        stage.setScene(new Scene(root));
    }

    public static void makeLevelsMenuScene(Stage stage, Pane pane, Scene scene) {
        BorderPane root = createBorderPane();
        VBox page = new VBox();
        HBox topOfPage = createTitle();
        HBox bottomOfPage = new HBox();
        VBox signOutBox = StartMenuHelper.createSignOutBox(scene, pane, stage);
        bottomOfPage.getChildren().add(signOutBox);
        page.getChildren().addAll(topOfPage, bottomOfPage);
        root.setCenter(page);
        stage.setScene(new Scene(root));
        StartMenuHelper.makeLevelsButton(stage, pane, scene, bottomOfPage);
    }

    private static BorderPane createBorderPane() {
        BorderPane root = new BorderPane();
        root.setStyle(BACKGROUND_COLOR);
        return root;
    }

    private static HBox createTitle() {
        HBox titleBox = new HBox();
        Label title = new Label(GAME_TITLE);
        title.setTextFill(Color.WHITE);
        title.setFont(Font.font(FONT_FAMILY, FontWeight.BOLD, FONT_SIZE));
        titleBox.getChildren().add(title);
        titleBox.setAlignment(Pos.CENTER);
        titleBox.setPadding(new Insets(40));
        return titleBox;
    }
}
```

# Приложение А.35.

## Файл «LoginPage.java» Client

```
package org.example.tanchiki.GUI.start;

import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.control.Alert;
import org.example.tanchiki.models.Player;
import org.example.tanchiki.services.GameData;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import org.example.tanchiki.services.ServerTask;
import org.jetbrains.annotations.NotNull;

import static org.example.tanchiki.GlobalConstance.*;

public class LoginPage {
    public static void loginBoxButtons(VBox loginBox, TextField usernameField, PasswordField
passwordField,
                                       Scene scene, Pane pane, Stage stage) {
        Button signUpConfirmButton = createButton("Зарегистрироваться", BUTTON_STYLE,
                e -> signUpEvent(usernameField, passwordField, scene, pane, stage));
        Button loginButton = createButton("Войти", BUTTON_STYLE, e -> loginEvent(usernameField,
                passwordField, scene, pane, stage));
        loginBox.getChildren().addAll(loginButton, signUpConfirmButton);
    }

    public static void signOutBoxButtons(VBox signOutBox, Scene scene, Pane pane, Stage stage) {
        Button signOutButton = LoginPage.createButton("Выйти", BUTTON_STYLE, e ->
LoginPage.signOutEvent(scene, pane, stage));
        signOutBox.getChildren().add(signOutButton);
    }

    @NotNull
    public static Button createButton(String text, String style, EventHandler<ActionEvent> event)
{
        Button button = new Button(text);
        button.setStyle(style);
        button.setOnAction(event);
        return button;
    }

    private static void loginEvent(TextField usernameField, PasswordField passwordField,
                                   Scene scene, Pane pane, Stage stage) {
        if (!usernameField.getText().isEmpty() && !passwordField.getText().isEmpty()){
            String response = ServerTask.loginOrSignUp("login", usernameField.getText(),
passwordField.getText());
            if (response.startsWith("success")){
                String[] parts = response.split("/");
                GameData.getInstance().setCurrentPlayer(new Player(usernameField.getText(),
passwordField.getText(),
                        Integer.parseInt(parts[1])));
                StartMenu.makeLevelsMenuScene(stage, pane, scene);
            } else if (response.equals("error")) {
                showAlert("Ошибка", "Не верно введён логин или пароль.");
            }
        }
    }

    private static void signUpEvent(TextField usernameField, PasswordField passwordField,
                                    Scene scene, Pane pane, Stage stage) {
        if (!usernameField.getText().isEmpty() && !passwordField.getText().isEmpty()){
            String response = ServerTask.loginOrSignUp("signup", usernameField.getText(),
passwordField.getText());
            if (response.startsWith("success")){
                GameData.getInstance().setCurrentPlayer(new Player(usernameField.getText(),
passwordField.getText()));
                StartMenu.makeLevelsMenuScene(stage, pane, scene);
            } else if (response.equals("error")) {
                showAlert("Ошибка", "Логин уже занят.");
            }
```

```
        }
    }

    public static void signOutEvent(Scene scene, Pane pane, Stage stage) {
        GameData.getInstance().setCurrentPlayer(null);
        StartMenu.makeMenuScene(stage, pane, scene);
    }

    private static void showAlert(String title, String message) {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```

# Приложение А.36.

## Файл «GameEnvironmentHelper.java» Client

```
package org.example.tanchiki.GUI.game;

import org.example.tanchiki.GlobalConstance;
import org.example.tanchiki.models.gameObjects.Flag;
import org.example.tanchiki.models.SceneObject;
import org.example.tanchiki.models.gameObjects.tank.Tank;
import org.example.tanchiki.models.gameObjects.tank.TankSide;
import org.example.tanchiki.models.gameObjects.tank.TankType;
import org.example.tanchiki.models.gameObjects.wall.Wall;
import org.example.tanchiki.models.gameObjects.wall.WallType;
import org.example.tanchiki.services.GameData;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.List;

import static org.example.tanchiki.GlobalConstance.*;

public class GameEnvironmentHelper {
    public static void readMap(String[][] map, List<SceneObject> sceneObjects, int health) {
        mapSize = map.length;
        GlobalConstance.updateSize();
        for (int i = 0; i < mapSize; i++) {
            for (int j = 0; j < mapSize; j++) {
                if (map[i][j] != null) {
                    double x = MAP_FIRST_X + j * scale;
                    double y = MAP_FIRST_Y + i * scale;
                    switch (map[i][j]) {
                        case "B" -> sceneObjects.add(new Wall(WallType.Normal, x, y, scale));
                        case "M" -> sceneObjects.add(new Wall(WallType.Iron, x, y, scale));
                        case "P", "F" -> attachPlayers(map[i][j], x, y, sceneObjects, health);
                        case "O", "A", "c", "C" -> attachEnemy(map[i][j], x, y, sceneObjects);
                        default -> {

                        }
                    }

                }
            }
        }
        addMapWall(sceneObjects);
    }

    private static void attachPlayers(String s, double x, double y, List<SceneObject>
sceneObjects, int health) {
        switch (s) {
            case "P" -> {
                Tank tank = new Tank(TankType.Player, TankSide.Player, x, y, scale);
                tank.setHealth(health);
                sceneObjects.add(tank);
                GameData.getInstance().setPlayerTank(tank);
            }
            case "F" -> {
                Flag flag = new Flag(x, y, scale);
                sceneObjects.add(flag);
                GameData.getInstance().setPlayersFlag(flag);
            }
            default -> {

            }
        }
    }

    private static void attachEnemy(String s, double x, double y, List<SceneObject> sceneObjects)
{
        Tank sceneObject = null;
        switch (s) {
            case "O" -> {
                sceneObject = new Tank(TankType.NormalEnemy, TankSide.Enemy, x, y, scale);
                sceneObjects.add(sceneObject);
            }
            case "A" -> {
                sceneObject = new Tank(TankType.StrongEnemy, TankSide.Enemy, x, y, scale);
```

```
                sceneObjects.add(sceneObject);
            }
            case "c" -> {
                sceneObject = new Tank(TankType.RandomEnemy, TankSide.Enemy, x, y, scale);
                sceneObject.setHealth(NORMAL_TANK_HEALTH);
                sceneObjects.add(sceneObject);
            }
            case "C" -> {
                sceneObject = new Tank(TankType.RandomEnemy, TankSide.Enemy, x, y, scale);
                sceneObject.setHealth(STRONG_TANK_HEALTH);
                sceneObjects.add(sceneObject);
            }
            default -> {

            }
        }
        GameData.getInstance().getEnemyTank().add(sceneObject);
    }

    private static void addMapWall(List<SceneObject> sceneObjects) {
        int mapWallSize = 14;
        for (int i = 0; i < mapHeight / mapWallSize; i++) {
            //Left wall
            sceneObjects.add(new Wall(WallType.Iron, MAP_FIRST_X - mapWallSize,
                    (MAP_FIRST_Y) + (i) * mapWallSize, mapWallSize));
            //Right wall
            sceneObjects.add(new Wall(WallType.Iron, mapHeight + MAP_FIRST_X,
                    (MAP_FIRST_Y) + (i) * mapWallSize, mapWallSize));
            //Up wall
            sceneObjects.add(new Wall(WallType.Iron, (MAP_FIRST_X) + (i) * mapWallSize,
                    MAP_FIRST_Y - mapWallSize, mapWallSize));
            //Button
            sceneObjects.add(new Wall(WallType.Iron, (MAP_FIRST_X) + (i) * mapWallSize,
                    MAP_FIRST_Y + mapHeight, mapWallSize));
        }
    }

    public static String[][] readMapFile(String fileName) {
        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String line;
            int numRows = 0;
            int numCols = 0;
            while ((line = reader.readLine()) != null) {
                numRows++;
                numCols = Math.max(numCols, line.length());
            }
            String[][] array = new String[numRows][numCols];
            reader.close();

            BufferedReader read = new BufferedReader(new FileReader(fileName));
            int row = 0;
            while ((line = read.readLine()) != null) {
                for (int col = 0; col < line.length(); col++) {
                    array[row][col] = String.valueOf(line.charAt(col));
                }
                row++;
            }
            read.close();
            return array;
        } catch (IOException e) {
            return readMapFile(DEFAULT_MAP);
        }
    }
}
```

# Приложение А.37.

## Файл «GamePage.java» Client

```
package org.example.tanchiki.GUI.game;

import org.example.tanchiki.GUI.EndGameScene;
import org.example.tanchiki.GUI.SceneHelper;
import org.example.tanchiki.services.GameData;
import org.example.tanchiki.models.GameStatus;
import org.example.tanchiki.models.SceneObject;
import javafx.animation.AnimationTimer;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;

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

public class GamePage {

    public static void startGame(int level, Stage stage, Pane pane, Scene scene, GameData
gameData, int health) {
        conformationGame(level, stage, pane, scene, gameData, health);
        gameData.setEnemyFreezing(false);
        gameLoop(pane, gameData, stage, scene);
        gameData.setGameStatus(GameStatus.Running);
    }

    private static void conformationGame(int level, Stage stage, Pane pane, Scene scene,
                                         GameData gameData, int health) {
        SceneHelper.conformStage(stage, pane, scene);
        SceneHelper.makeGameScene(pane);
        gameData.resetGame(level);
        gameData.setGameStatus(GameStatus.Start);
        GameEnvironmentHelper.readMap(gameData.getMap(), gameData.getSceneObjects(), health);
    }

    private static void gameLoop(Pane pane, GameData gameData, Stage stage, Scene scene) {
        new AnimationTimer() {
            private long lastUpdate = 0;

            public void handle(long currentNanoTime) {
                update(gameData);
                if (currentNanoTime - lastUpdate >= 100_000) {
                    draw(gameData, pane);
                    if (gameData.getGameStatus() == GameStatus.Stop) {
                        this.stop();
                        endGameMassage(gameData, pane, stage, scene);
                    }

                    lastUpdate = currentNanoTime;
                }
            }
        }.start();
    }

    private static void endGameMassage(GameData gameData, Pane pane, Stage stage, Scene scene) {
        gameData.setEnemyFreezing(true);
        if (gameData.getEnemyNumber() == 0 && gameData.getEnemyTank().isEmpty()) {
            EndGameScene.makeEndGame(true, pane, stage, scene);
        } else {
            EndGameScene.makeEndGame(false, pane, stage, scene);
        }
    }

    private static void draw(GameData gameData, Pane pane) {
        pane.getChildren().clear();
        SceneHelper.makeGameScene(pane);
        List<SceneObject> copyOfSceneObjects = new ArrayList<>(gameData.getSceneObjects());
        for (SceneObject sceneObject : copyOfSceneObjects) {
            pane.getChildren().add(sceneObject.getNode());
        }
    }

    private static void update(GameData gameData) {
        List<SceneObject> copyOfSceneObjects = new ArrayList<>(gameData.getSceneObjects());
        for (SceneObject sceneObject : copyOfSceneObjects) {
```

```
            if (sceneObject.isVisible()) {
                sceneObject.update();
            } else {
                gameData.getSceneObjects().remove(sceneObject);
            }
        }
        checkEndGame(gameData);
    }

    private static void checkEndGame(GameData gameData) {
        if (gameData.getPlayerTank() == null ||
                !GameData.getInstance().getPlayersFlag().isVisible() ||
                (gameData.getEnemyTank().isEmpty()
                        && gameData.getEnemyNumber() == 0)) {
            gameData.setGameStatus(GameStatus.Stop);
        }
    }
}
```

## Приложение А.38.

## Файл «module-info.java» Client

```
module org.example.tanchiki {
    requires javafx.controls;
    requires javafx.fxml;
    requires org.jetbrains.annotations;
    requires java.datatransfer;
    requires java.desktop;
    requires jlayer;

    opens org.example.tanchiki to javafx.fxml;
    exports org.example.tanchiki;
    exports org.example.tanchiki.services;
    opens org.example.tanchiki.services to javafx.fxml;
}
```

## Приложение А.39.

## Файл «pom.xml» Client

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>tanchiki</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>tanchiki</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<junit.version>5.10.0</junit.version>  </properties>

  <dependencies>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-controls</artifactId>
      <version>21</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-fxml</artifactId>
      <version>21</version>
    </dependency>

    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>${junit.version}</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>${junit.version}</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.jetbrains</groupId>
      <artifactId>annotations</artifactId>
      <version>23.0.0</version>
      <scope>compile</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/javazoom/jlayer -->
    <dependency>
      <groupId>javazoom</groupId>
      <artifactId>jlayer</artifactId>
      <version>1.0.1</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.11.0</version>
        <configuration>
          <source>21</source>
          <target>21</target>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.openjfx</groupId>
        <artifactId>javafx-maven-plugin</artifactId>
        <version>0.0.8</version>
        <executions>
          <execution>
            <!-- Default configuration for running with: mvn clean javafx:run -->
            <id>default-cli</id>
```

```
            <configuration>
              <mainClass>org.example.tanchiki/org.example.tanchiki.HelloApplication</mainClass>
              <launcher>app</launcher>
              <jlinkZipName>app</jlinkZipName>
              <jlinkImageName>app</jlinkImageName>
              <noManPages>true</noManPages>
              <stripDebug>true</stripDebug>
              <noHeaderFiles>true</noHeaderFiles>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

# Приложение В.

# UML - диаграммы

## Приложение В.1.

## Диаграммы классов

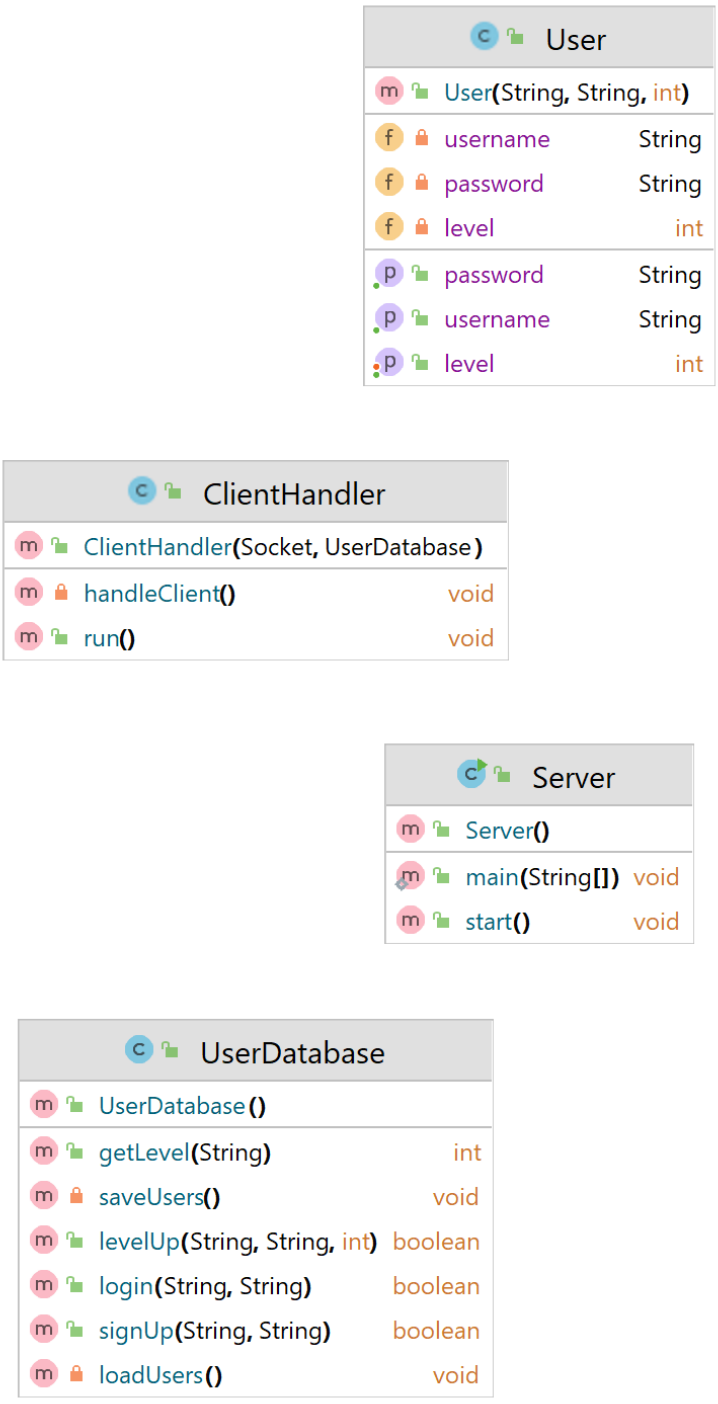


**Рисунок 9 – UML - диаграмма классов сервера.**

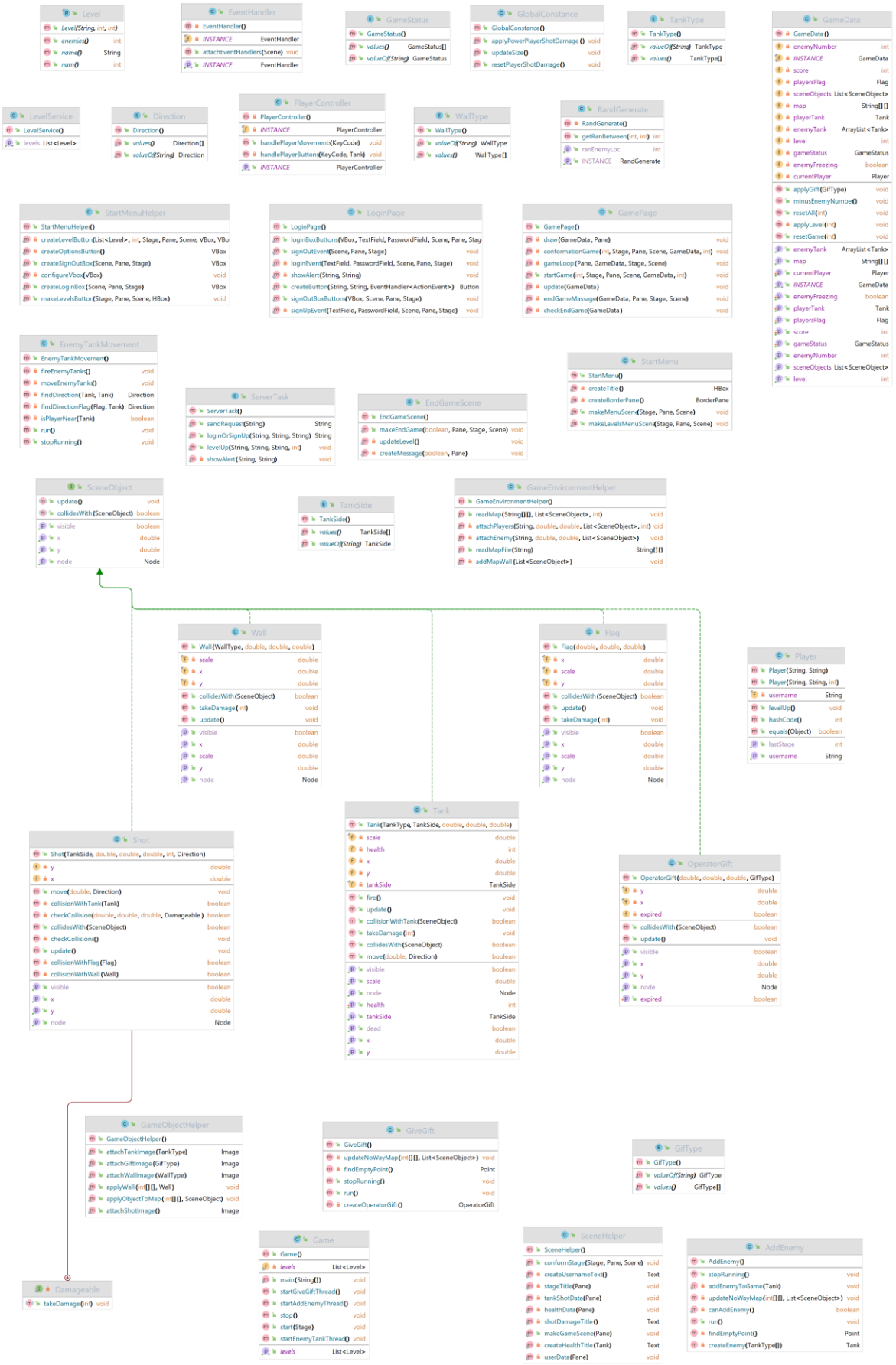


**Рисунок 10 – UML - диаграмма классов клиента.**

Приложение В.2.

Диаграмма развёртывания

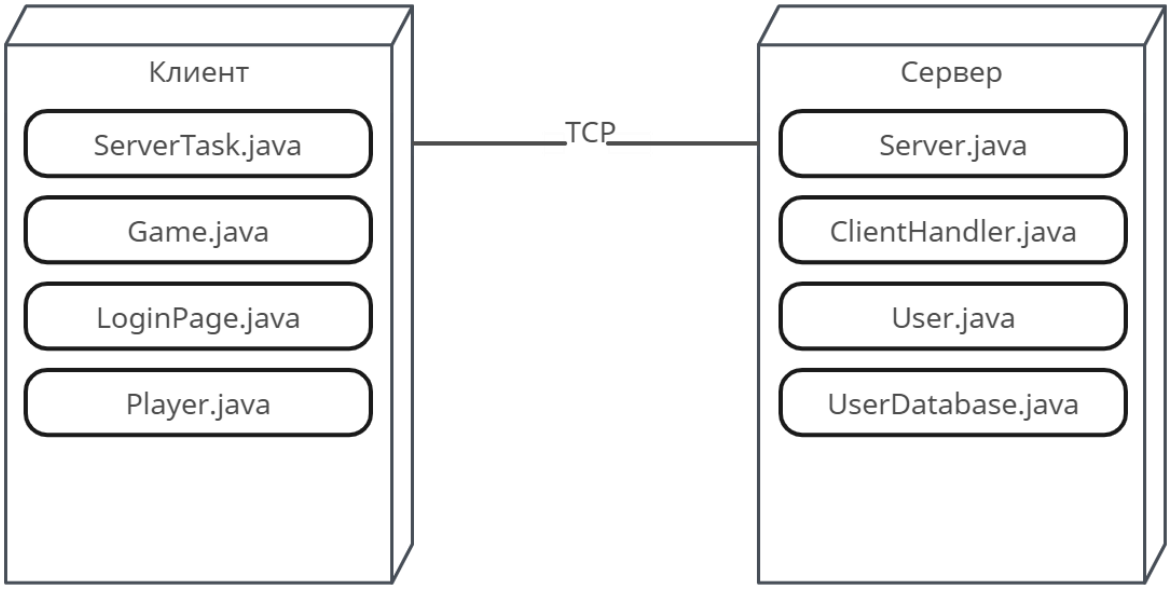


**Рисунок 11 – UML - диаграмма развёртывания.**

Приложение В.3.

Диаграмма последовательности

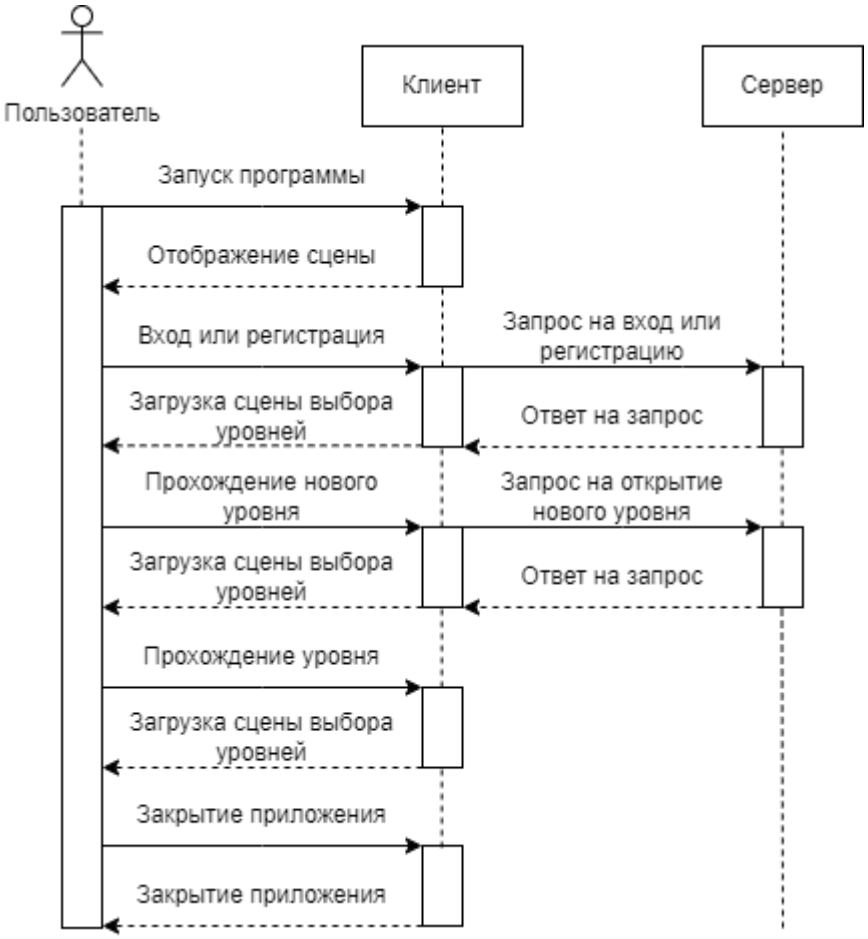


**Рисунок 12 – UML - диаграмма последовательности.**

Приложение В.4.

Диаграмма деятельности

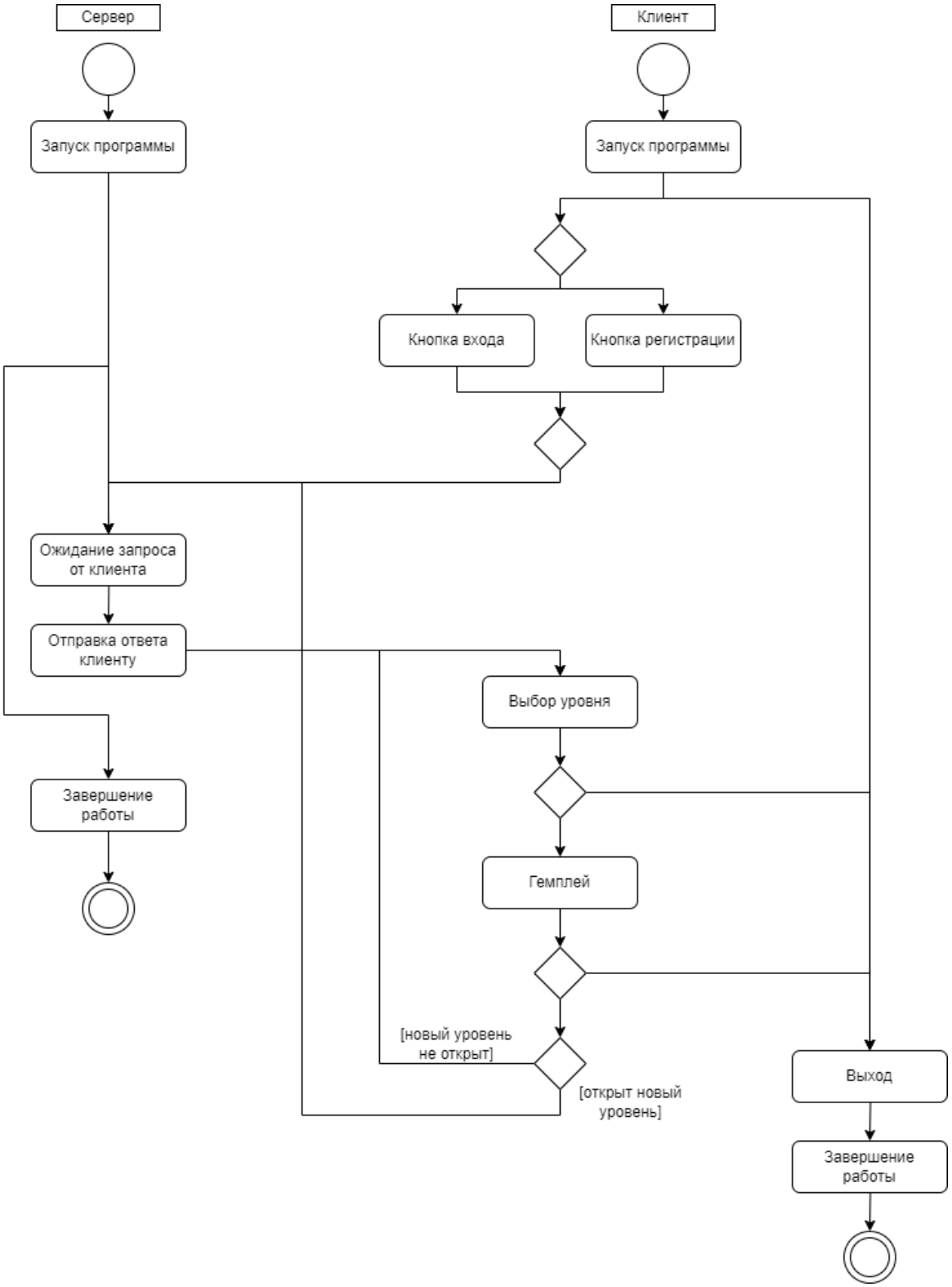


**Рисунок 13 – UML - диаграмма деятельности.**

Приложение В.5.

Диаграмма вариантов использования приложения

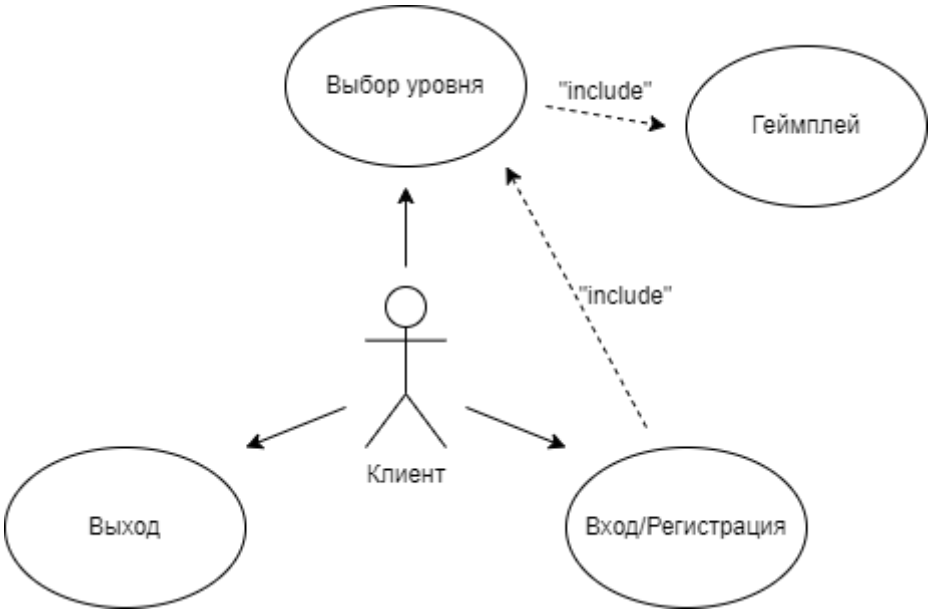


**Рисунок 14 – UML – диаграмма вариантов использования приложения.**