**Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский ядерный университет «МИФИ»**

# ОТЧЕТ
# по лабораторной работе №5 по дисциплине: «Теория и технология программирования»

**Выполнила: студентка группы Б23-902 Деребас Л. И.**

**Проверил: Смирнов Д. С.**

**Москва 2025 г**

## Оглавление



# Постановка задачи

## Описание модельной ситуации.

Вы новый сотрудник перспективной инди-гейм-студии «GaMEPhIcation». Ваша компания известна своей успешной игрой – инди-файтингом «Смертельная Битва», и в этом году планирует сделать крупное обновление. К сожалению, изначальный разработчик данной игры больше не работает в Вашей компании, поэтому разработка обновления была поручена Вам. Из материалов разработчика остался git-репозиторий и отчет о выполненных работах, содержащий изначальное ТЗ.

## Задание

1. Ознакомится с отчетом о прошлой работе
2. Сделать форк исходного репозитория
3. Проверить код на работоспособность, в случае неработоспособности кода – восстановить ее
4. Внести в игру следующие изменения:
    a. В начале игры игрок задает, какое количество локаций он хочет пройти. В каждой локации появляется случайное количество врагов, зависящее от уровня игрока, в финале каждой локации игрока ждет босс
    b. При получении нового уровня игрок должен выбрать какую характеристику он хочет улучшить – урон или максимальное количество здоровья
    c. Добавить новое действие (игроку и противнику-магу) – ослабление противника. Если противник в этот момент уходит в защиту – с вероятностью 75% он получает временный дебаф на n ходов (n – уровень игрока), что увеличивает урон по нему

на 25% и сокращает на 50% уровень атаки игрока. Если противник атакует – ослабление срывается, а ослабитель получает +15% к дополнительному урону. Метка ослабления должна иметь визуальный эффект и отдельный атрибут у объектов-бойцов.

d. Добавить Боссу новое действие – босс в случайный момент боя будет пытаться регенерировать здоровье. Если игрок в этот момент уходит в защиту – босс восстанавливает 50% от полученного на текущий момент урона, если игрок в это время атакует – процесс регенерации прерывается и босс получает двойной урон от атаки игрока.

5. Компания будет Вам крайне-признательна, если Вы предложите и реализуете более адаптивную систему поведения противников, учитывающую поведение игрока
6. Так же Вы получите бонус, если оформите документацию на Ваши изменения и основные классы с использованием генератора документации javadoc

## Ссылка на GitHub

https://github.com/Shirouky/lab5-Java

## Диаграммы

Рис 1. DFD-диаграмма 0 уровня

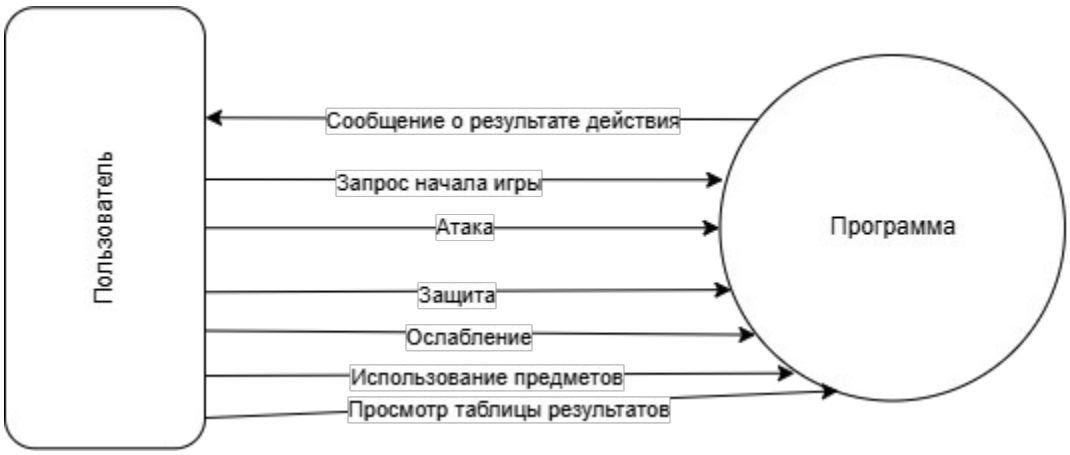



Рис 2. DFD-диаграмма 1 уровня

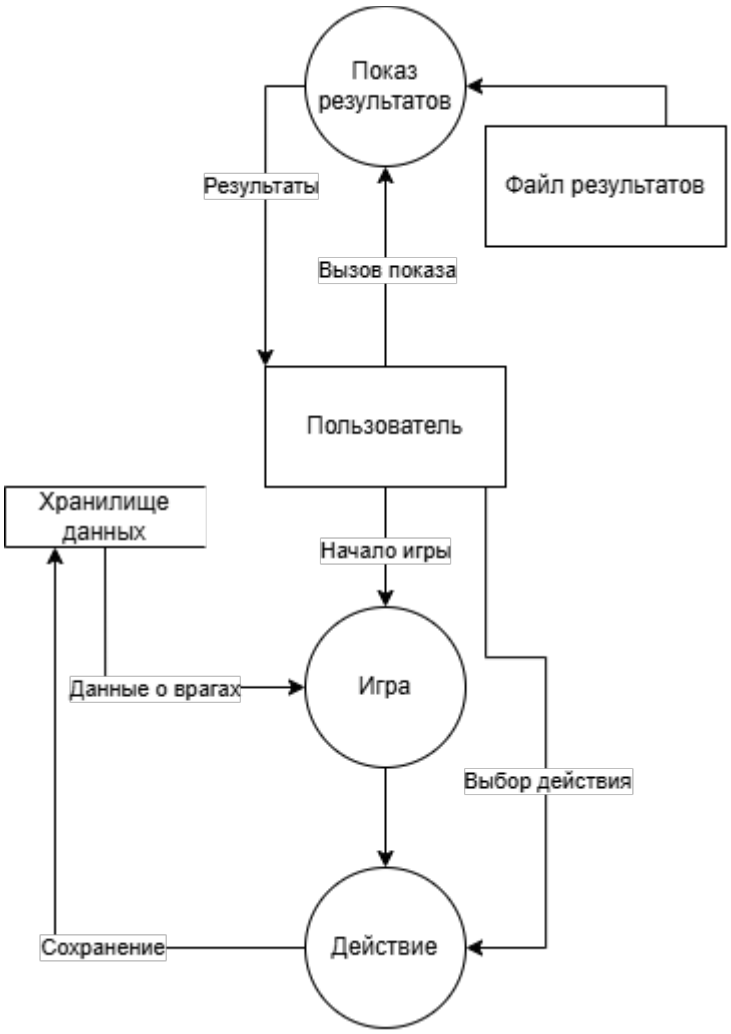

Показ результатов
Файл результатов
Результаты
Вызов показа
Пользователь
Хранилище данных
Начало игры
Данные о врагах
Игра
Выбор действия
Действие
Сохранение

Рис 3. UML-диаграмма на концептуальном уровне

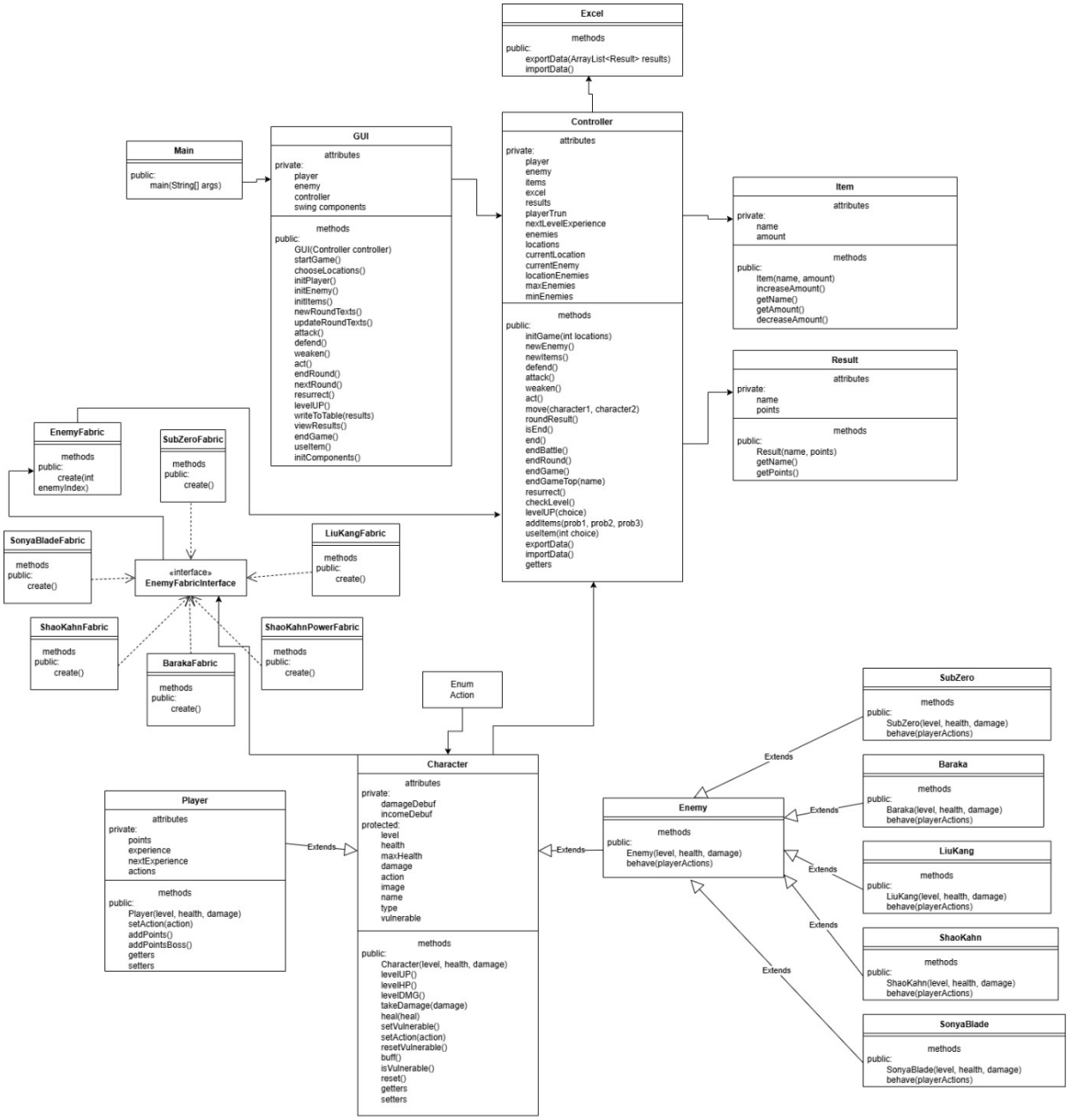
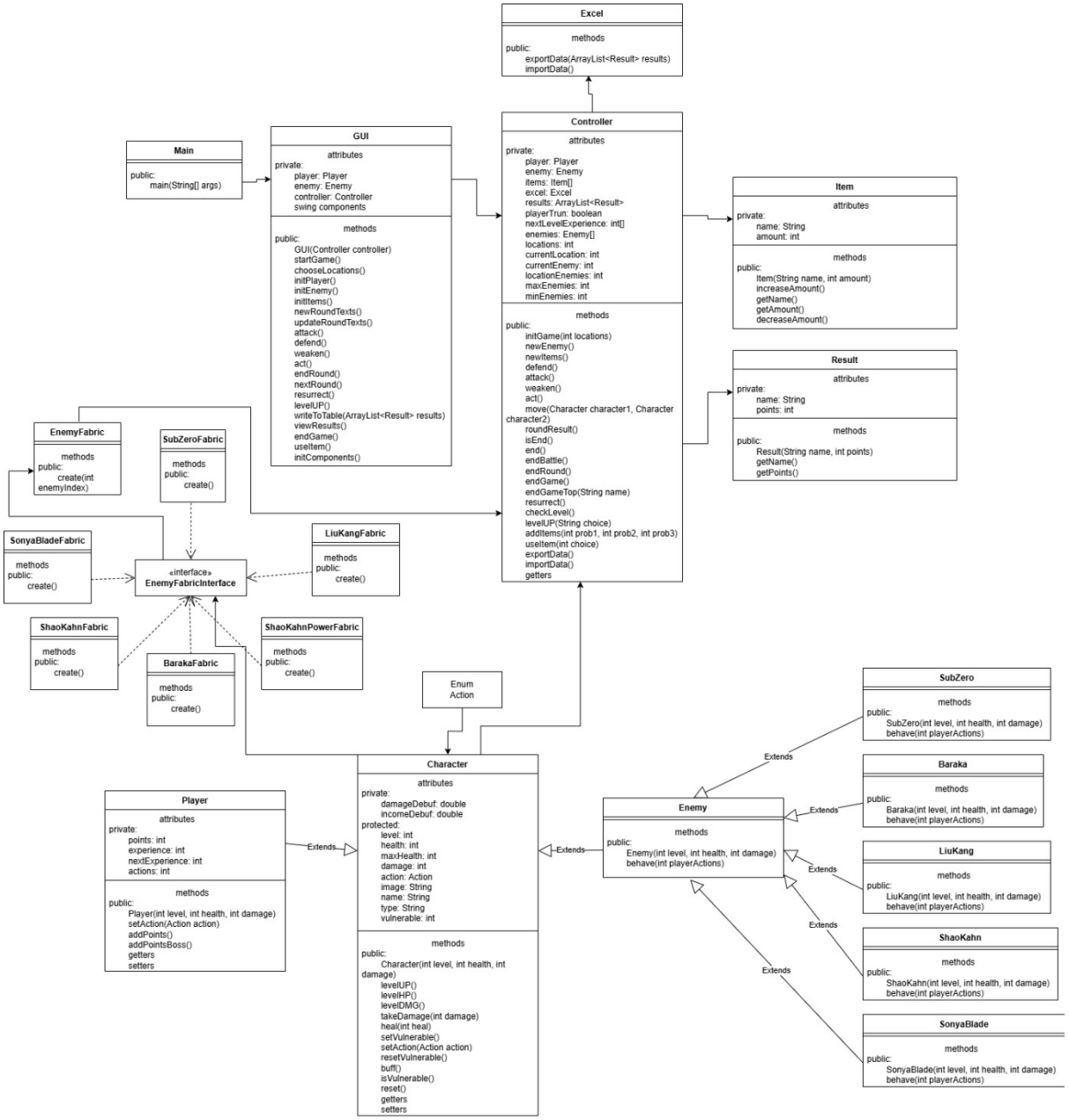


Рис 4. UML-диаграмма на имплементационном уровне



# Интерфейс

Рис 5. Главное окно

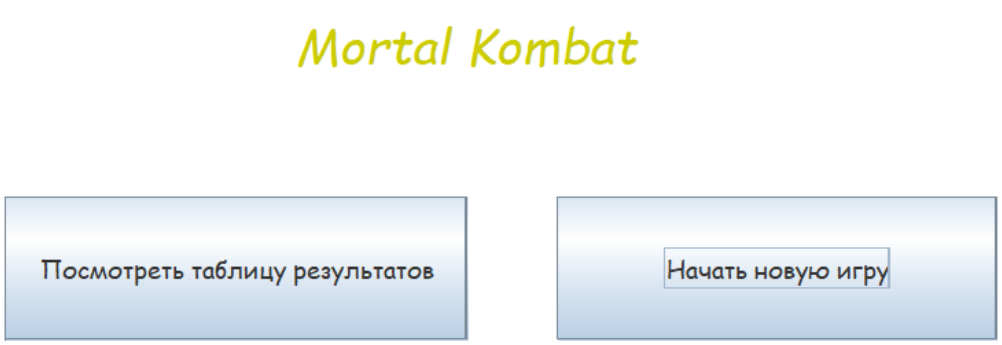



Рис 6. Окно игры

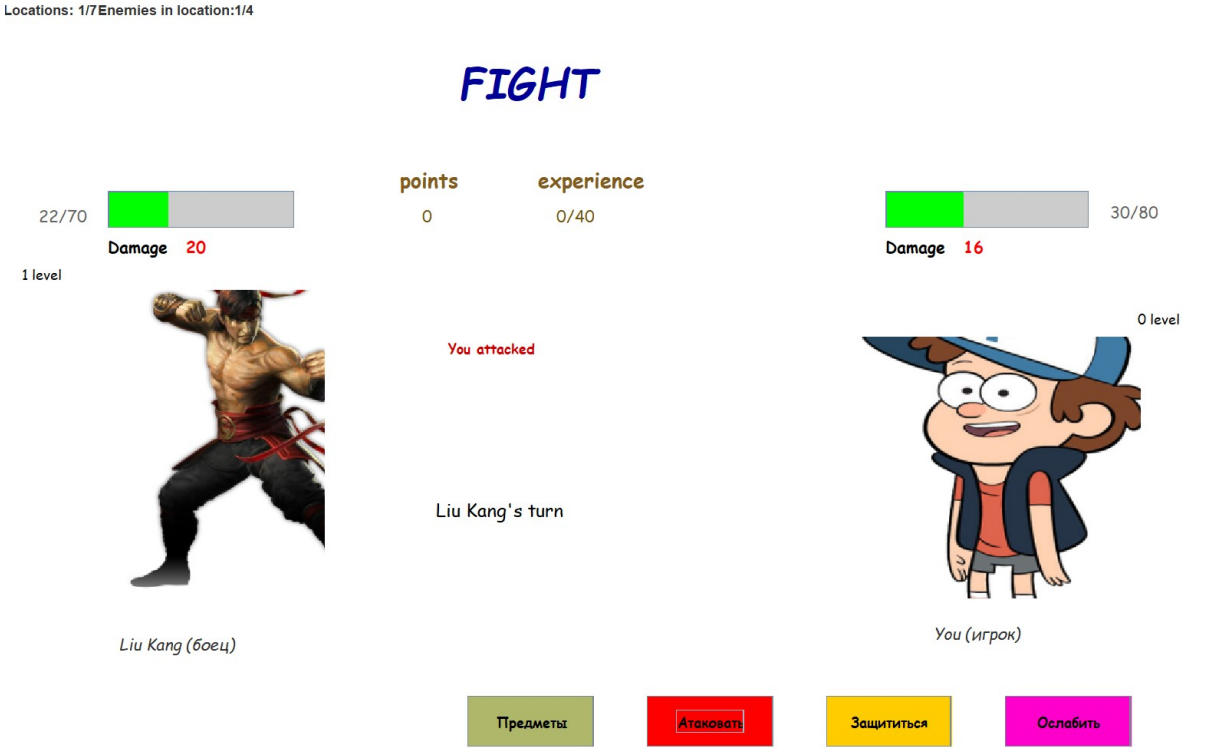
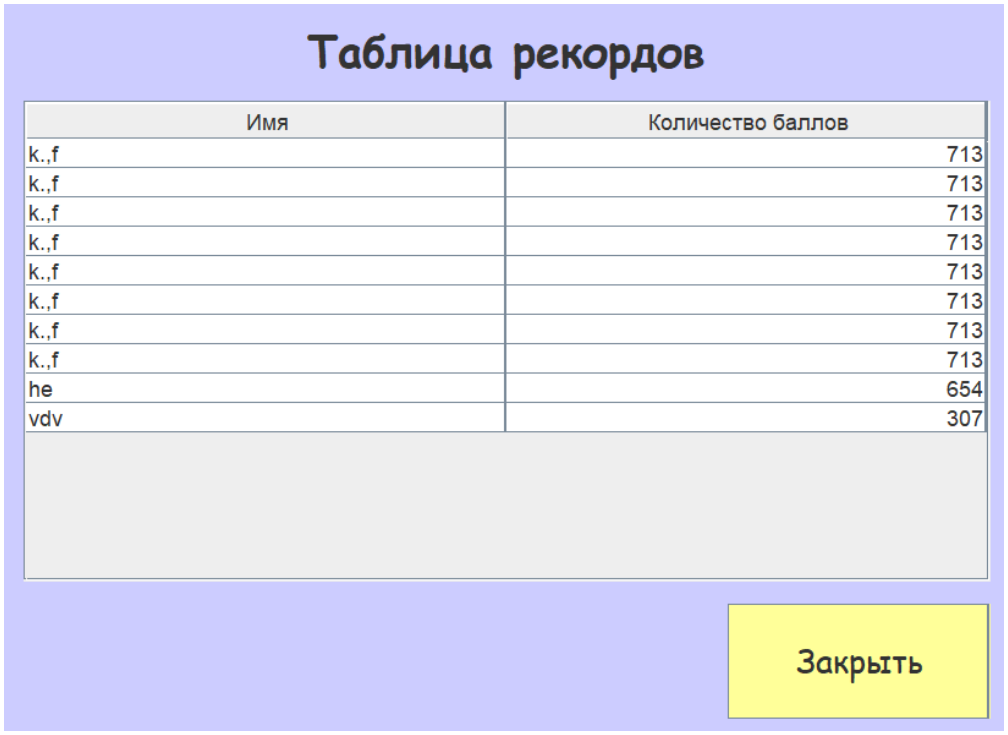


Рис 7. Таблица рекордов



# Важные части кода

Рис 8. Структура кода

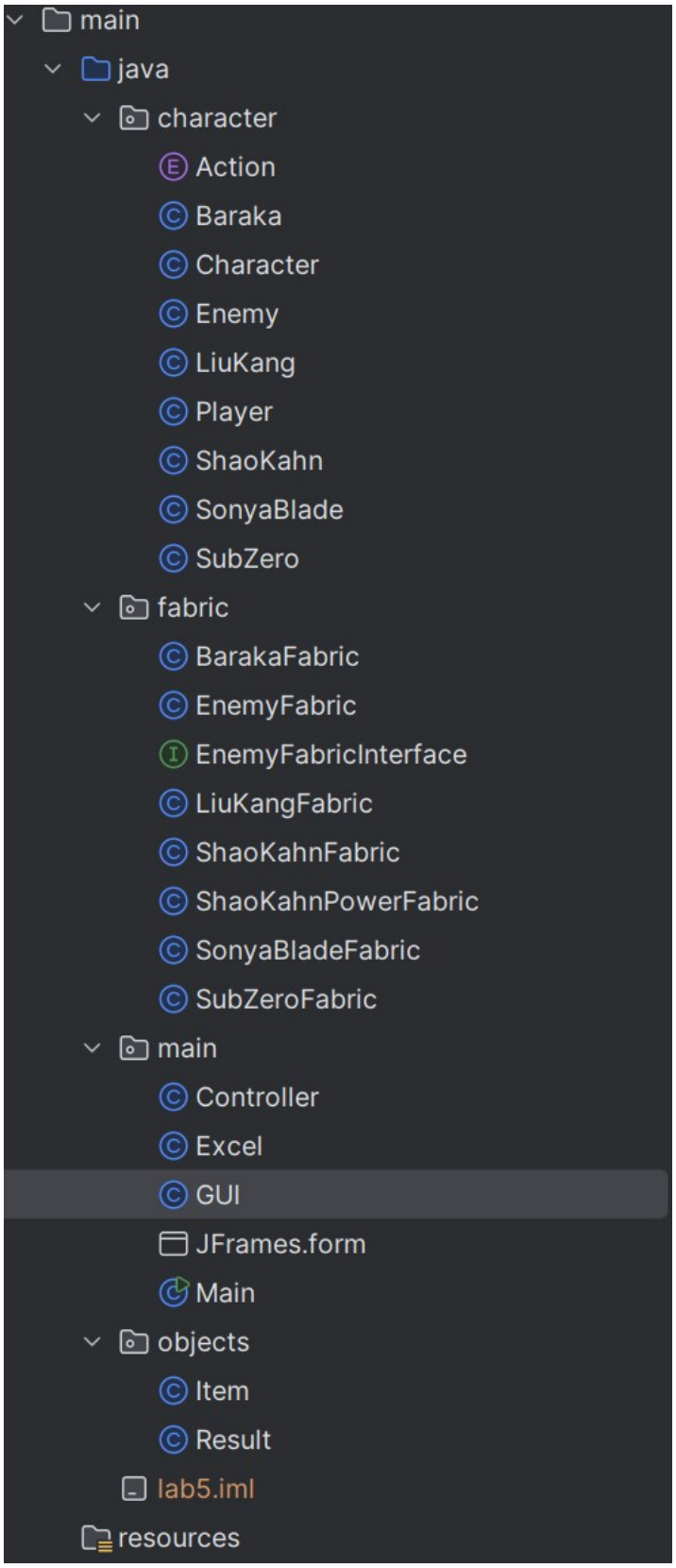


Рис 9. Пример документации

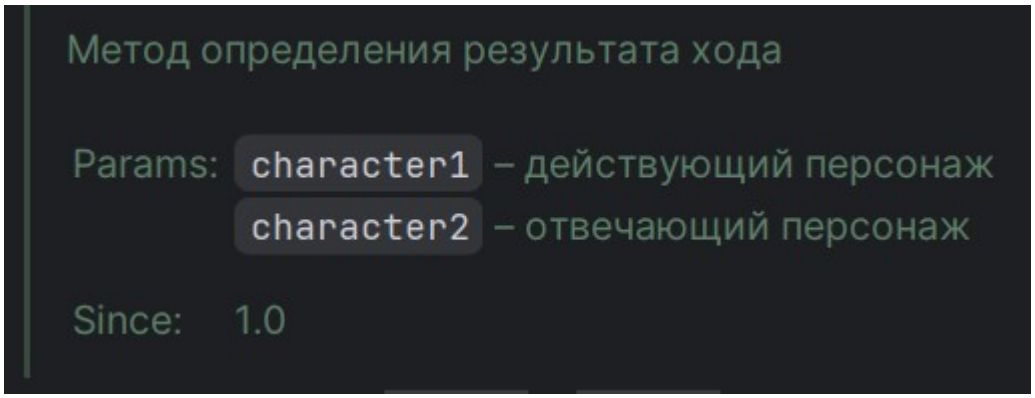



Рис 10. Часть метода move класса Controller

```
public HashMap<String, String> move(Character character1, Character character2) {  2 usages  Shirouky *
    HashMap<String, String> labels = new HashMap<>();
    labels.put("status", "");
    if (character1.isVulnerable() && character2.getAction() == ATTACK) {
        character1.resetVulnerable();
        character2.buff();
        labels.put("status", character1.getName() + " removed weakening and " + character2.getName() + " got buff");
        labels.put("progressbar", String.valueOf(character1 instanceof Player));
    }
    if (character2.isVulnerable() && character1.getAction() == ATTACK) {
        character2.resetVulnerable();
        character1.buff();
        labels.put("status", character2.getName() + " removed weakening and " + character1.getName() + " got buff");
        labels.put("progressbar", String.valueOf(character2 instanceof Player));
    }

    switch (character1.getAction()) {
        case REGENERATE:
            switch (character2.getAction()) {
                case DEFEND: {
                    character1.heal((int) ((character1.getMaxHealth() - character1.getHealth()) * 0.5));
                    labels.put("action", character1.getName() + " regenerated");
                    break;
                }
                case STUN: {
                    character1.heal((int) ((character1.getMaxHealth() - character1.getHealth()) * 0.6));
                    labels.put("action", character1.getName() + " regenerated");
                    break;
                }
                case ATTACK: {
                    character1.takeDamage(character2.getDamage() * 2);
                    labels.put("action", character2.getName() + " stopped regeneration");
                    break;
                }
            }
            break;
```

Рис 11. Реализация влияния действий игрока на поведение Shao Kahn

```
@Override  1 usage  Shirouky
public void behave(int playerActions) {
    Random random = new Random();
    double randomN = random.nextDouble() * 100;

    if (randomN < 40 + playerActions) {
        action = ATTACK;
    } else if (randomN < 80){
        action = DEFEND;
    } else {
        action = REGENERATE;
    }
}
```

Рис 12. Улучшение характеристики в зависимости от решения игрока

```
public void levelUP(String choice) {  1 usage  Shirouky
    player.levelUP();
    if (Objects.equals(choice, b: "HP")) player.levelHP(player);
    else player.levelDMG(player);
    for (int i = 0; i < 5; i++) {
        if (nextLevelExperience[i] == player.getNextExperience()) {
            player.setNextExperience(nextLevelExperience[i + 1]);
            for (int j = 0; j < 4; j++) {
                enemies[j].levelUP();
                enemies[j].levelHP(player);
                enemies[j].levelDMG(player);
            }
            break;
        }
    }

}
```