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

ОТЧЕТ
по лабораторной работе №2 по дисциплине: «Теория и технология программирования»

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

Проверил: Смирнов Д. С.

Москва 2025 г

Оглавление

Постановка задачи

Саурон готовится к решающей битве за Средиземье. Ваша задача — собрать армию орков из разных племен (Мордор, Дол Гулдур, Мглистые Горы, Серые горы). Каждое племя имеет уникальное снаряжение (оружие, броня, знамёна), а орки могут быть разных типов (базовый, командир, разведчик).

1. Класс Ork
    1. Создайте класс Ork с полями: name, weapon, armor, banner, и базовыми атрибутами.
    2. Реализуйте паттерн Строитель (OrkBuilder), который позволяет конфигурировать орка шаг за шагом.
    3. Дополнительно: Используйте библиотеку Faker для генерации «аутентичных» имён орков, если имя не задано явно.
    4. Каждый орк имеет следующие базовые атрибуты (значения зависят от племени): o Сила (1-100) o Ловкость (1-100) o Интеллект (1-50) o Здоровье (50-200)
    5. Пример для племен:
        i. Мордор: Высокая сила (+30%), низкая ловкость.
        ii. Дол Гулдур: Сбалансированные характеристики.
        iii. Мглистые Горы: Высокая ловкость (+30%), низкий интеллект.
2. Снаряжение для племен
    1. Реализуйте паттерн Абстрактная фабрика для создания снаряжения:
    2. Интерфейс OrcGearFactory с методами: createWeapon(), createArmor(), createBanner().

3. Конкретные фабрики для племен:
    i. MordorGearFactory → Тяжелые мечи, стальная броня, знамя с Красным Оком.
    ii. DolGuldurGearFactory → Копья, кольчуги, знамя с пауком.
    iii. MistyMountainsGearFactory → Топоры, кожаная броня, знамя с Луной.

3. Фабричный метод для строителей
    1. Создайте интерфейс OrkBuilderFactory с методом createOrkBuilder().
    2. Для каждого племени реализуйте свою фабрику строителей (например, MordorOrkBuilderFactory), которая:
    3. Связывает OrkBuilder с конкретной OrcGearFactory.
    4. Позволяет создавать строителей, готовых к сборке орков определенного племени.
4. Директор
    1. Реализуйте класс OrcDirector, который управляет процессом сборки:
    2. Методы для создания «типовых» орков: createBasicOrk(), createLeaderOrk(), createScoutOrk().
    3. Пример: Командир получает знамя и горн, разведчик — лук вместо меча.
5. Графический интерфейс (Swing)
    1. Элементы для создания нового орка – основная кнопка создания и элементы для выбора племени и роли в армии
    2. JTree для иерархии армии: Корневой узел — Армия Мордора. Дочерние узлы — племена (Мордор, Изенгард и т.д.). Внутри племен — орки (отображаются по имени).
    3. Панель информации об орке: При выборе орка в JTree отображается:
        i. Имя и племя.
        ii. Снаряжение (оружие, броня, знамя).
        iii. Характеристики (прогресс-бары или текстовые поля)

Ссылка на GitHub

https://github.com/Shirouky/lab2-Java

Диаграммы

Рис 1. DFD-диаграмма 0 уровня

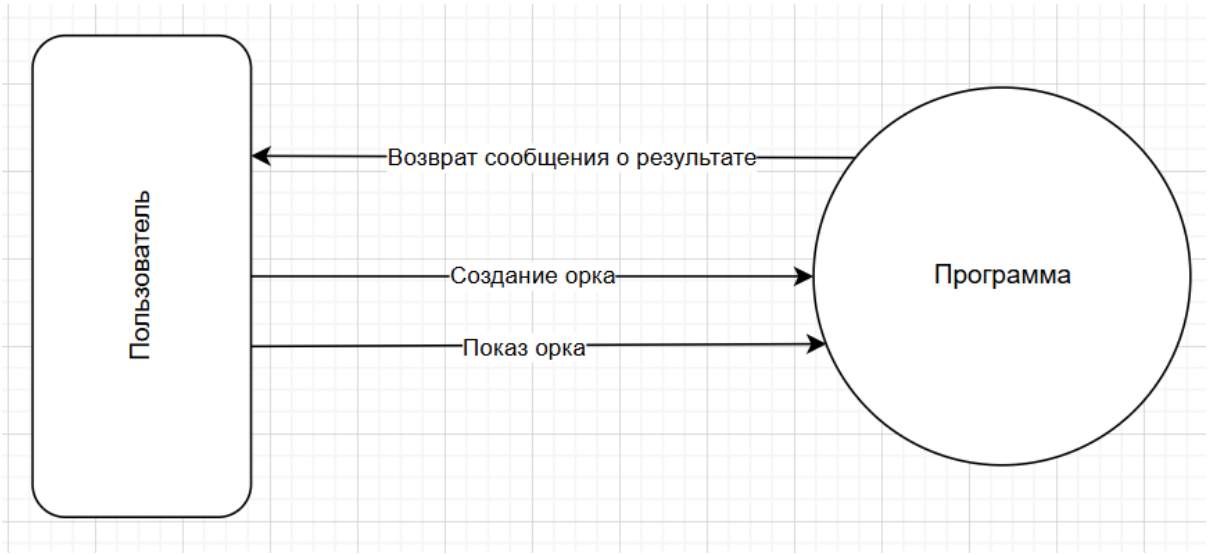


Рис 2. DFD-диаграмма 1 уровня

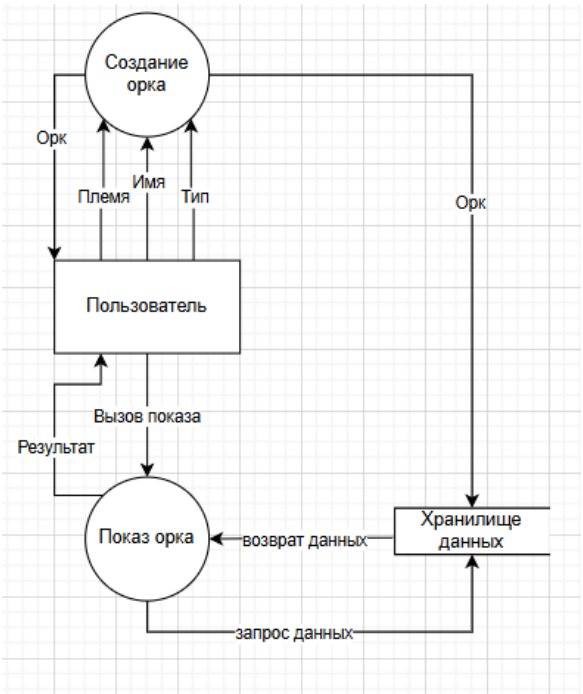


Рис 3. UML-диаграмма на концептуальном уровне

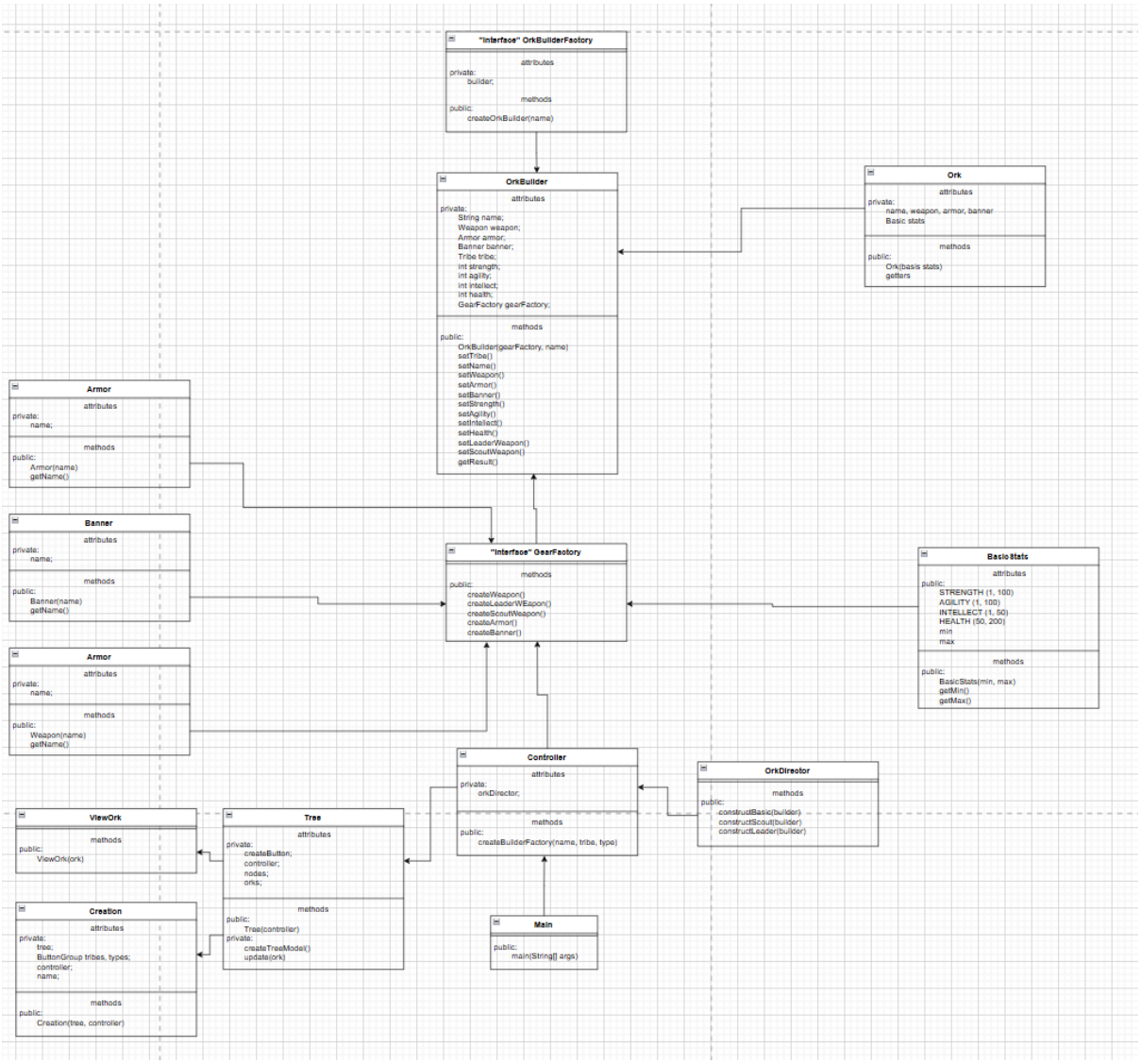
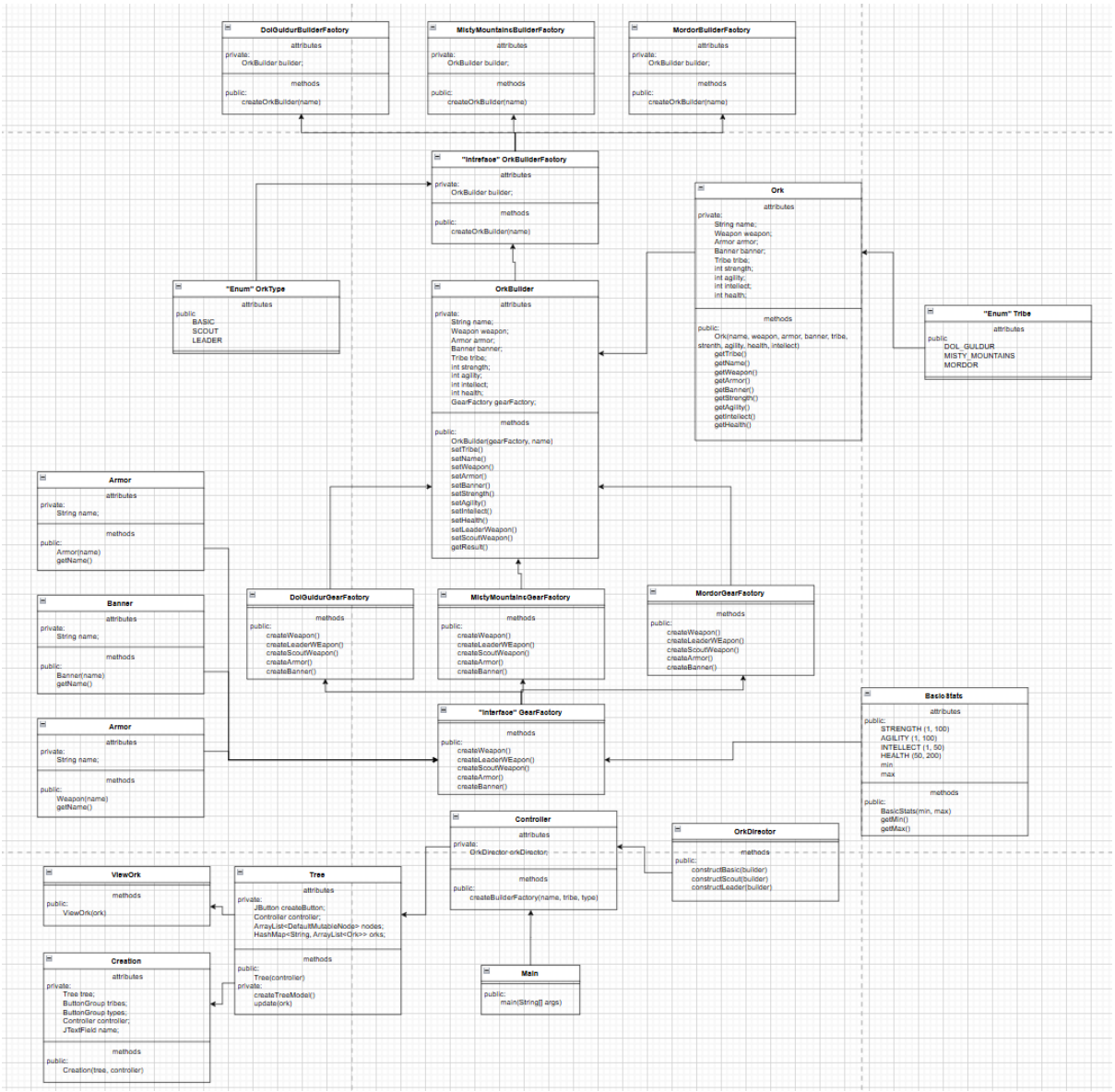


Рис 4. UML-диаграмма на имплементационном уровне



Интерфейс

Рис 5. Главная страница (скучная)

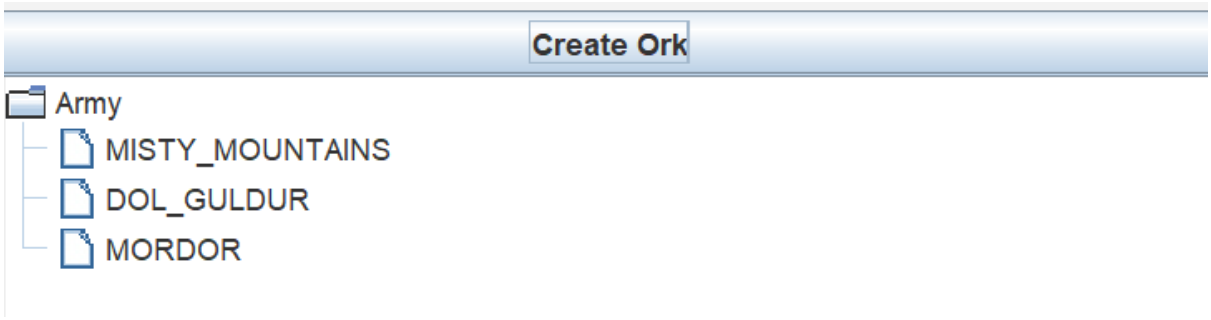


Рис 6. Создание орка (скучное)

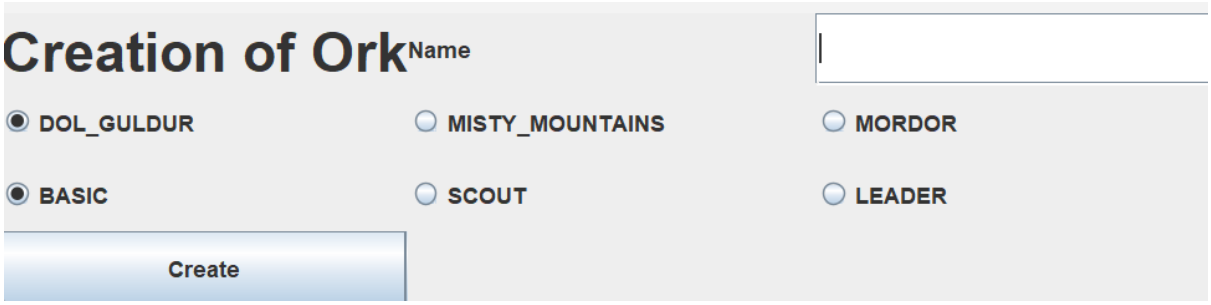
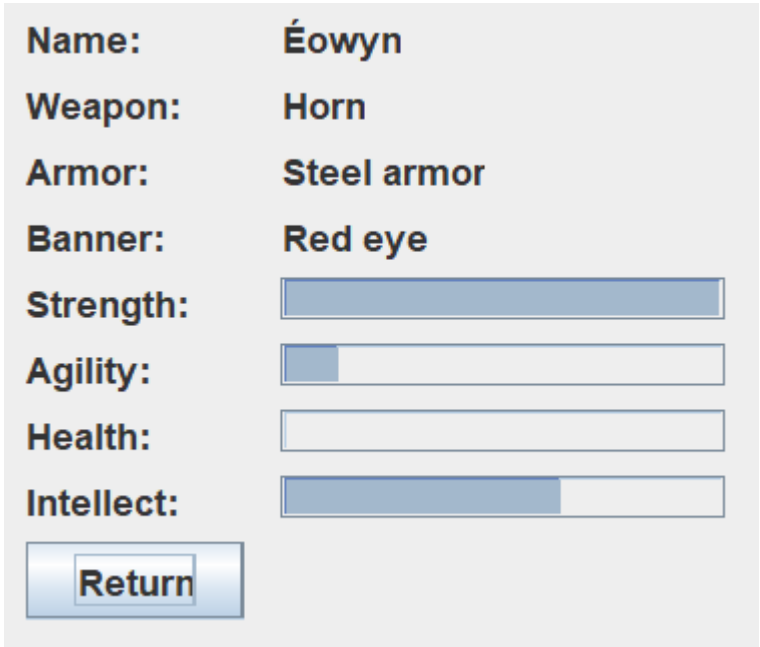


Рис 7. Страница орка (скучная)



Важные части кода

Рис 8. Структура кода

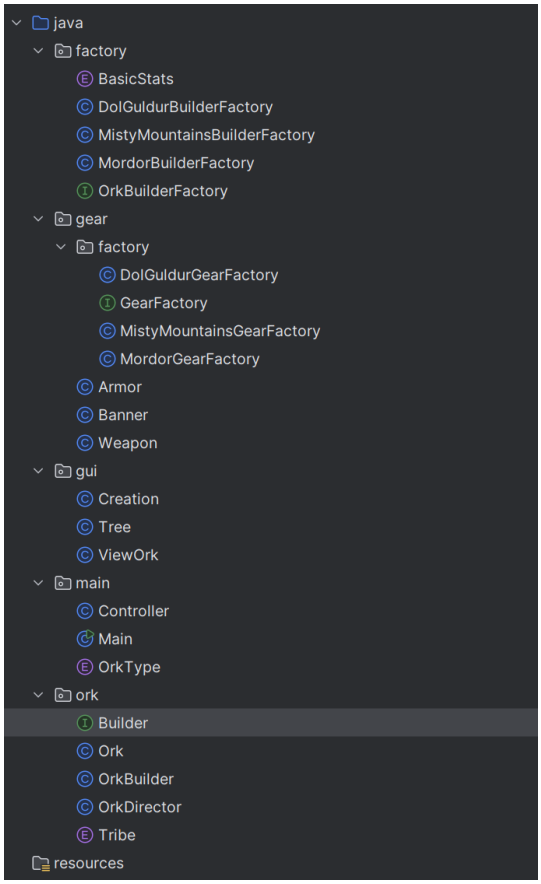


Рис 9. Класс OrkDirector

```
public class OrkDirector {  3 usages  Shirouky
    public void constructBasic(Builder builder) {  1 usage  Shirouky
        builder.setWeapon();
        builder.setArmor();
    }

    public void constructScout(Builder builder) {  1 usage  Shirouky
        builder.setScoutWeapon();
        builder.setArmor();
    }

    public void constructLeader(Builder builder) {  1 usage  Shirouky
        builder.setLeaderWeapon();
        builder.setArmor();
        builder.setBanner();
    }
}
```

Рис 10. Интерфейс Builder

```
public interface Builder {  4 usages  1 implementation  Shirouky
    void setTribe(Tribe tribe);  1 implementation  Shirouky

    void setWeapon();  1 usage  1 implementation  Shirouky

    void setLeaderWeapon();  1 usage  1 implementation  Shirouky

    void setScoutWeapon();  1 usage  1 implementation  Shirouky

    void setArmor();  3 usages  1 implementation  Shirouky

    void setBanner();  1 usage  1 implementation  Shirouky

    void setStrength(int strength);  3 usages  1 implementation  Shirouky

    void setAgility(int agility);  3 usages  1 implementation  Shirouky

    void setHealth(int health);  3 usages  1 implementation  Shirouky

    void setIntellect(int intellect);  3 usages  1 implementation  Shirouky
}
```

Рис 10. Интерфейс GearFactory

```
import gear.Armor;
import gear.Banner;
import gear.Weapon;

public interface GearFactory {  6 usages  3 implementations  Shirouky

    Weapon createWeapon();  1 usage  3 implementations  Shirouky

    Weapon createLeaderWeapon();  1 usage  3 implementations  Shirouky

    Weapon createScoutWeapon();  1 usage  3 implementations  Shirouky

    Armor createArmor();  1 usage  3 implementations  Shirouky

    Banner createBanner();  1 usage  3 implementations  Shirouky
}
```

Рис 11. Интерфейс OrkBuilderFactory

```
import ork.OrkBuilder;

public interface OrkBuilderFactory {  3 usages  3 implementations  Shirouky
    OrkBuilder createOrkBuilder(String name);  3 usages  3 implementations  Shirouky
}
```