

Федеральное государственное автономное
образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет
Системы управления и робототехники



Вариант №9851
Лабораторная работа №3-4
по дисциплине
«Программирование»

Выполнил Студент группы R3180

Романенко Ю.Ю.

Преподаватель:

Кобелев Р.П.

г. Санкт-Петербург
2025г.

Текст задания:

Лабораторная работа #3-4

Этапы выполнения работы:

1. Получить вариант
2. Нарисовать UML-диаграмму, представляющую классы и интерфейсы объектной модели и их взаимосвязи;
3. Придумать сценарий, содержащий действия персонажей, аналогичные приведенным в исходном тексте;
4. Согласовать диаграмму классов и сценарий с преподавателем.
5. Написать программу на языке Java, реализующую разработанные объектную модель и сценарий взаимодействия и изменения состояния объектов. При запуске программа должна проигрывать сценарий и выводить в стандартный вывод текст, отражающий изменение состояния объектов, приблизительно напоминающий исходный текст полученного отрывка.
6. Для одного из методов класса персонажа (или объекта), согласованного с преподавателем, создать промпты для 2-3 ИИ-ассистентов ([Gigachat](#), [DeepSeek](#), [Qwen](#), или др.), получить от них код метода, **реализовать метод самостоятельно**, и включить в отчет сравнительный анализ кода, написанного ИИ-ассистентами, и своего варианта. Оценить качество кода, полученного от ИИ-ассистентов, указать на возможные недоработки.
7. Продемонстрировать выполнение программы на сервере helios.
8. Ответить на контрольные вопросы и выполнить дополнительное задание.

Текст, выводящийся в результате выполнения программы не обязан дословно повторять текст, полученный в исходном задании. Также не обязательно реализовывать грамматическое согласование форм и падежей слов выводимого текста.

Стоит отметить, что цель разработки объектной модели состоит не в выводе текста, а в эмуляции объектов предметной области, а именно их состояния (поля) и поведения (методы). Методы в разработанных классах должны изменять состояние объектов, а выводимый текст должен являться побочным эффектом, отражающим эти изменения.

Требования к объектной модели, сценарию и программе:

1. В модели должны быть представлены основные персонажи и предметы, описанные в исходном тексте. Они должны иметь необходимые

атрибуты и характеристики (состояние) и уметь выполнять свойственные им действия (поведение), а также должны образовывать корректную иерархию наследования классов.

2. Объектная модель должна реализовывать основные принципы ООП - инкапсуляцию, наследование и полиморфизм. Модель должна соответствовать принципам SOLID, быть расширяемой без глобального изменения структуры модели.
3. Сценарий должен быть вариативным, то есть при изменении начальных характеристик персонажей, предметов или окружающей среды, их действия могут изменяться и отклоняться от базового сценария, приведенного в исходном тексте. Кроме того, сценарий должен поддерживать элементы случайности (при генерации персонажей, при задании исходного состояния, при выполнении методов).
4. Объектная модель должна содержать как минимум один корректно использованный элемент каждого типа из списка:
 - абстрактный класс как минимум с одним абстрактным методом;
 - интерфейс;
 - перечисление (enum);
 - запись (record);
 - массив или ArrayList для хранения однотипных объектов;
 - проверяемое исключение.
5. В созданных классах основных персонажей и предметов должны быть корректно переопределены методы equals(), hashCode() и toString(). Для классов-исключений необходимо переопределить метод getMessage().
6. Созданные в программе классы-исключения должны быть использованы и обработаны. Кроме того, должно быть использовано и обработано хотя бы одно unchecked исключение (можно свое, можно из стандартной библиотеки).
7. При необходимости можно добавить внутренние, локальные и анонимные классы.

Жители Змеевки с веселыми лицами ходили по улицам. Все были рады тому, что Бублик и Шурупчик нашлись, и в особенности тому, что Гвоздик перевоспитался. Некоторые, правда, не верили в это перевоспитание, боясь, как бы он опять не начал бить стекла. Через некоторое время Гвоздика на берегу. Работа по подготовке к балу была в полном разгаре. Беседка для оркестра и палатки вокруг танцевальной площадки строились. Тюбик расписывал беседку самыми затейливыми узорами, а остальные малыши окрашивали палатку во все цвета радуги. Малышки украшали площадку цветами, разноцветными фонариками и флажками. Незнайка кричал, суеился и только другим мешал. К счастью, каждый и без него знал, что нужно делать.

Рисунок 1 – Задание варианта «9851».

Диаграмма

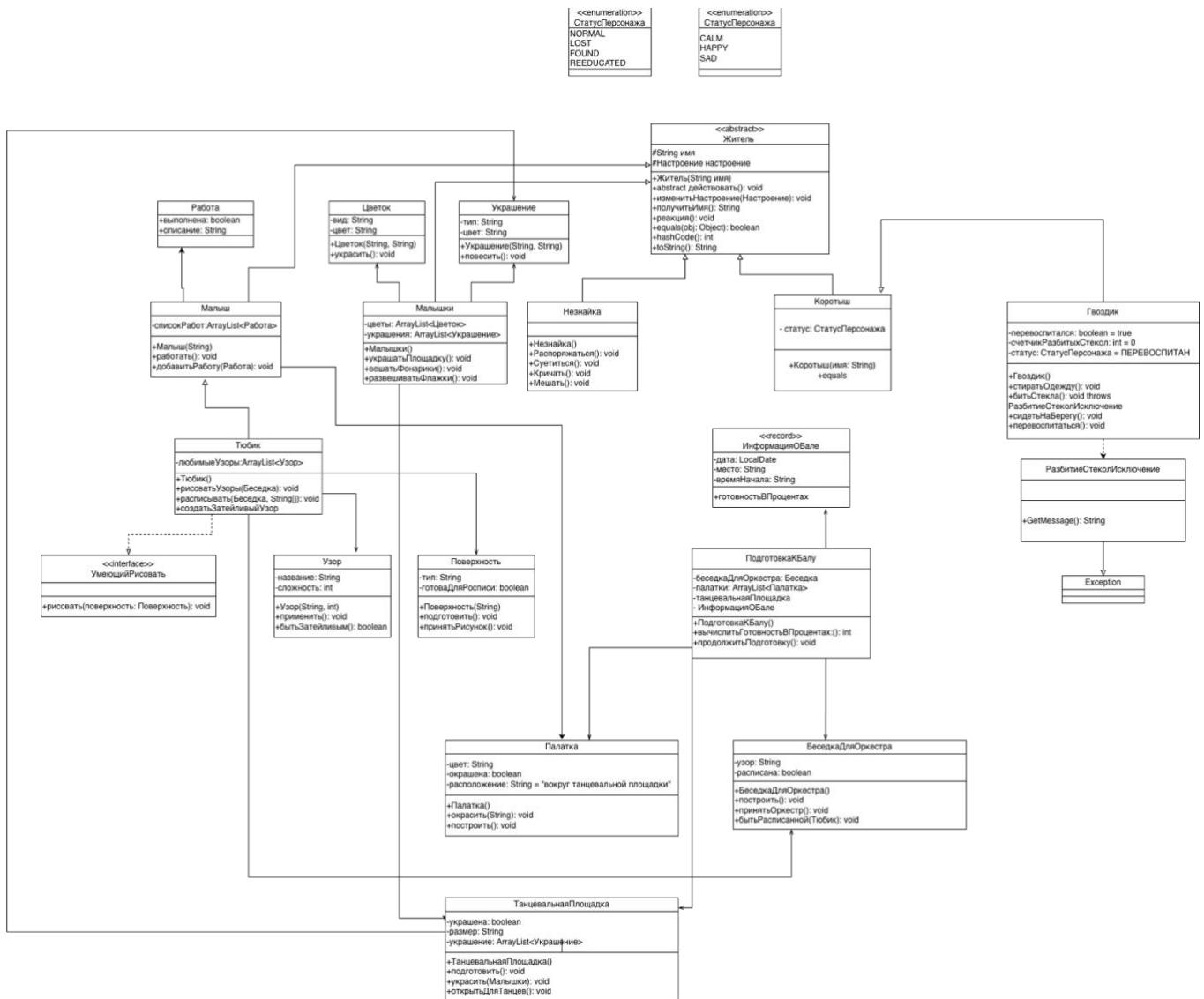


Рисунок 2 – UML-диаграмма

Исходный код программы

Ссылка: <https://github.com/xapork/lab3-4>

Работа с искусственным интеллектом

Промт:

Напиши метод reactToFound для класса Citizen на Java. Метод должен учитывать настроение персонажа (Mood) и выводить разную реакцию в зависимости от него.

Qwen:

```
public void reactToFound(Citizen foundCitizen) {
    switch (mood) {
        case HAPPY:
            System.out.println(name + " радостно обнимает " + foundCitizen.getName());
            break;
        case SAD:
            System.out.println(name + " с облегчением вздыхает, увидев " + foundCitizen.getName());
            break;
        case CALM:
        default:
            System.out.println(name + " спокойно приветствует " + foundCitizen.getName());
            break;
    }
}
```

DeepSeek:

```
public void reactToFound(Citizen foundCitizen) {
    if (foundCitizen == null) {
        return;
    }

    if (mood == Mood.HAPPY) {
        System.out.println(name + " громко радуется, что нашёлся " + foundCitizen.getName());
    } else if (mood == Mood.SAD) {
        System.out.println(name + " тихо радуется находке " + foundCitizen.getName());
    } else {
        System.out.println(name + " отмечает, что " + foundCitizen.getName() + " найден");
    }
}
```

Мой вариант:

```
public void reactToFound(Citizen foundCitizen) {
    if (foundCitizen == null || foundCitizen == this) {
        return;
    }

    String reaction;

    switch (mood) {
        case HAPPY -> reaction = "искренне радуется, что нашёлся ";
        case SAD -> reaction = "с облегчением замечает, что найден ";
        case CALM -> reaction = "спокойно реагирует на находку ";
        default -> reaction = "реагирует на находку ";
    }

    System.out.println(name + " " + reaction + foundCitizen.getName());
}

public String getName() {
    return name;
}
```

Сравнение работы ИИ

| | | |
|---------------|-------------------------------------|--|
| Критерий | Qwen | DeepSeek |
| Стиль | Метод swith, код читаемый | Используется if-else, код более громоздкий |
| Проверка null | - | + |
| Расширяемость | Можно добавить новые состояния Mood | Добовление новых состояний усложнит код |
| Надежность | Нет проверок, низкая | Более надежная, так как есть null |

Результат выполнения:

```
[s502400@helios ~]$ java -jar lab3-4-1.0-SNAPSHOT.jar
Picked up _JAVA_OPTIONS: -XX:MaxHeapSize=1G -XX:MaxMetaspaceSize=128m
Беседка для оркестра построена
Поверхность подготовлена для росписи
Тюбик рисует Pattern{Цветочный, complexity =8}
Тюбик рисует Pattern{Музыкальный, complexity =6}
Беседка расписана затейливым узором
Оркестр разместился в беседке

  Проверка Гвоздика
Статус Гвоздика: REEDUCATED
Поймано исключение: Гвоздик перевоспитан и больше не бьет стекла.
Статус Гвоздика восстановлен: REEDUCATED

  Шурупчик и Бублик найдены!

Винтик с облегчением замечает, что найден Шурупчик
Винтик с облегчением замечает, что найден Бублик
Мальшки спокойно реагирует на находку Шурупчик
Мальшки спокойно реагирует на находку Бублик
Tubik искренне радуется, что нашёлся Шурупчик
Tubik искренне радуется, что нашёлся Бублик
Гвоздик с облегчением замечает, что найден Шурупчик
Гвоздик с облегчением замечает, что найден Бублик
🎉 Подготовка к балу началась
Танцевальная площадка (большая) готова
=> Готовность: 40%
Палатка построена и покрашена в синяя
=> Готовность: 65%
Malyshki украшают площадку
Flowers Ромашка (белый) украшает площадку
Украшение Фонарики (золотая) повешено
=> Готовность: 85%
Винтик выполняет работы
Работа выполнена: Установить фонари
=> Готовность: 100%

InfAboutBall[data=2026-01-14, place=Цветочный город, startTime=18:00, readyInPercent=100]
```

Вывод:

В ходе лабораторной работы я создала программу на Java по мотивам сюжета о жителях Змеевки. Программа моделирует поведение персонажей, их взаимодействие и подготовку к событиям, используя современные подходы объектно-ориентированного программирования.

Для сборки проекта был использован Maven, который позволил создать исполняемый JAR-файл. Этот файл был успешно загружен на сервер Helios, где программа была запущена и продемонстрировала стабильную работу.