Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление программная инженерия
Образовательная программа системное и прикладное программное обеспечение

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3–4

курса «Программирование»

по теме: «Принципы объектно-ориентированного программирования SOLID и STUPID»

Вариант № 68118

Выполнил студент:
Мухамедьяров Артур Альбертович
группа: P3109

Преподаватель:
Гаврилов А. В.,
Мустафаева А. В.

Санкт-Петербург, 2024 г.

Содержание

Лабораторная работа № 3–4 Принципы объектно-ориентированного программирования SOLID и STUPID

1. Задание варианта № 68118

В соответствии с выданным вариантом на основе предложенного текстового отрывка из литературного произведения создать объектную модель реального или воображаемого мира, описываемого данным текстом. Должны быть выделены основные персонажи и предметы со свойственным им состоянием и поведением. На основе модели написать программу на языке Java.

Описание предметной области, по которой должна быть построена объектная модель:

Незнайка махнул рукой и стал разглядывать красные туфли на ногах собеседника. Он заметил, что туфли застегивались на пряжки, которые были сделаны в виде полумесяца со звездой. Незнайка принялся рассказывать о том, как мечтал о волшебной палочке, как Кнопочка рассказала ему, что нужно совершать хорошие поступки, и как у него ничего не вышло, потому что он был способен совершать хорошие поступки только ради волшебной палочки, а не бескорыстно.

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. Продемонстрировать выполнение программы на сервере `helios`.

7. 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 принципе 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.

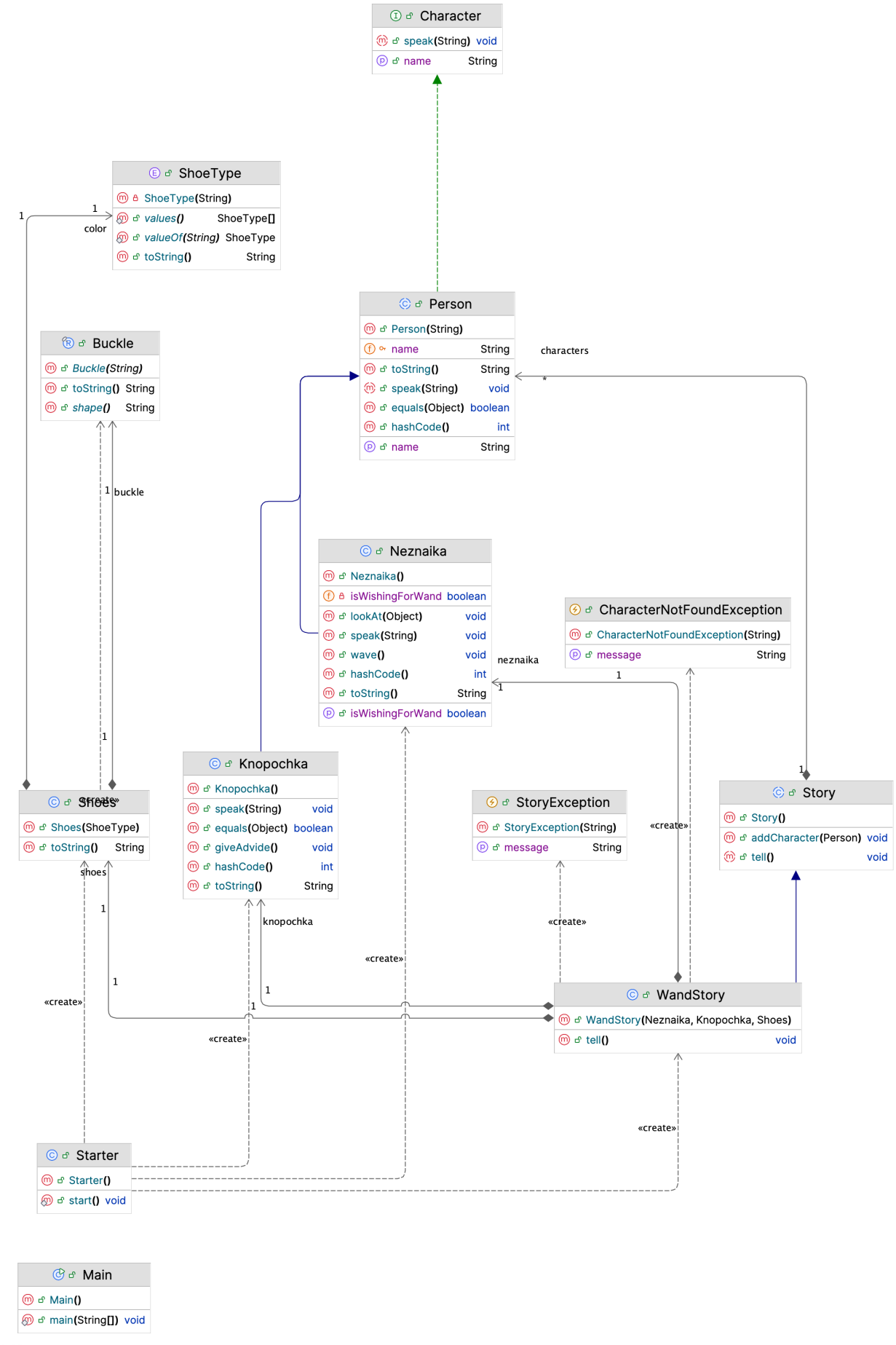

Рис. 1.1: UML диаграмма классов

2. Выполнение задания.

Задание было выполнено в редакторе кода IntejjiJ IDEA, собрано в `jar` файл `lab3.jar` и загружено в Git репозиторий на GitHub.

2. 1. Листинги кода

Листинг из файла 1.1

```
public class Main {
    public static void main(String[] args) {
        Starter.start();
    }
}
```

Листинг 1.1: Исходный код главного класса программы

3. Результат работы программы.

```
    Незнайка махнул рукой
    Незнайка разглядывает красные туфли с пряжкой в виде полумесяц со звездой
    Незнайка рассказывает о" том, как мечтал о волшебной палочке"
    Кнопочка говорит нужно" совершать хорошие поступки бескорыстно"
    но у Незнайка ничего не вышло, так как он делал добро не бескорыстно

```

Листинг 1.2: Результат выполнения программы

4. Вывод

Во время выполнения лабораторной работы я изучил основные принципы объектно-ориентированного программирования в Java. Освоил работу с наследованием, абстрактными классами, интерфейсами, изучил особенности использования различных модификаторов доступа и методов. Рассмотрел принципы SOLID, STUPID, особенности работы с коллекциями, системой обработки исключений и современными возможностями языка, такими как enum и record. Полученные знания создают надежную базу для дальнейшего изучения языка программирования и разработки программных решений.

Литература

[1] Ссылка на личный репозиторий GitHub: https://github.com/pozitp/itmo-labs/tree/main/prog/lab2