МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
ИТМО»

Лабораторная работа №3

по дисциплине “Программирование”

Студент:
Дядев Владислав Александрович
Преподаватель:
Наумова Надежда Александровна

Санкт-Петербург
2024

Задание

Вариант №313114

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

> Малыш вернулся к телефону и сказал, что фрекен Бок не может подойти. Но Фрида хотела во что бы то ни стало узнать, чем так занята ее сестра, что не может даже подойти к телефону, и засыпала Малыша вопросами. В конце концов он сказал: Он положил трубку и поглядел; что делает Карлсон. Но Карлсона нигде не было. Малыш искал его всюду, в конце концов прибежал на кухню. Подошел к открытому окну и увидел на подоконнике странную фигуру. Там стояло верхом на маминой любимой метле таинственное существо, готовое к отлету. По всей вероятности, это и был Карлсон, хотя по виду существо это было похоже на маленькую ведьму: лицо, вымазанное черным, на голове косынка, а на плечах развевался ведьминский плащ -- мамин большой цветастый передник.

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.

Исходный код программы

Ссылка на github - https://github.com/Alvas07/ITMO/tree/main/1-2%20Programming/Lab3-4/src

Результат работы программы

Результат работы программы представлен на рисунке ниже.

```
vladislavdyadev@MacBook-Air-Vladislav-4 src % java -jar lab3.jar
Гостиная Гостиная создана
Кухня Кухня создана
Телефон и трубка созданы
Телефон появился в Гостиная
Трубка телефона появился в Гостиная
Окно создано
Подоконник создан
Метла создана
Метла появился на подоконнике

Environment Initialization done

Карлсон создан
Фрекен Бок Фрекен Бок создана
Фрида Фрида создана
Малыш Малыш создан
Мистическое существо создано
Фрекен Бок зашёл в Гостиная
Малыш зашёл в Гостиная
Мистическое существо с измазано зелёным лицом и косынка на голове зашёл в Кухня
Карлсон зашёл в Гостиная

Characters Initialization done

Малыш побежал к Телефон
Малыш поднял Трубка телефона
Трубка телефона уже поднята
Малыш сказал: 'Фрекен Бок не может подойти'
Фрида засыпала Малыш вопросами
Малыш посмотрел на Карлсон
Карлсон был здесь
Сценарий окончен
vladislavdyadev@MacBook-Air-Vladislav-4 src % java -jar lab3.jar
Гостиная Гостиная создана
Кухня Кухня создана
Телефон и трубка созданы
Телефон появился в Гостиная
Трубка телефона появился в Гостиная
Окно создано
Подоконник создан
Метла создана
Метла появился на подоконнике

Environment Initialization done

Карлсон создан
Фрекен Бок Фрекен Бок создана
Фрида Фрида создана
Малыш Малыш создан
Мистическое существо создано
Фрекен Бок зашёл в Гостиная
Малыш зашёл в Гостиная
Мистическое существо с чистое лицом и кепка на голове зашёл в Кухня
Карлсон зашёл в Гостиная

Characters Initialization done

Малыш побежал к Телефон
Малыш поднял Трубка телефона
Трубка телефона уже поднята
Малыш сказал: 'Фрекен Бок не может подойти'
Фрида засыпала Малыш вопросами
Малыш посмотрел на Карлсон
Карлсон был здесь
Сценарий окончен
```

Диаграмма классов

Диаграмма реализованных классов представлена в репозитории на github - https://github.com/Alvas07/ITMO/blob/main/1-2%20Programming/Lab3-4/Diagram.png

Выводы по работе

В ходе лабораторной работы я научился анализировать предметную область, составлять диаграмму классов и объектную модель, поработал с Enum, Record, абстрактными классами и интерфейсами.