

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники



Лабораторная работа №5

по дисциплине

«Программирование»

Вариант №367581

Выполнил студент группы Р3115
Федоров Егор Владимирович
Преподаватель:
Сорокин Роман Борисович

Содержание

1 Текст задания

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса `Vehicle`, описание которого приведено ниже.

Разработанная программа должна удовлетворять следующим требованиям:

- Класс, коллекцией экземпляров которого управляет программа, должен реализовывать сортировку по умолчанию.
- Все требования к полям класса (указанные в виде комментариев) должны быть выполнены.
- Для хранения необходимо использовать коллекцию типа `java.util.TreeMap`
- При запуске приложения коллекция должна автоматически заполняться значениями из файла.
- Имя файла должно передаваться программе с помощью: **переменная окружения**.
- Данные должны храниться в файле в формате `csv`
- Чтение данных из файла необходимо реализовать с помощью класса `java.io.InputStreamReader`
- Запись данных в файл необходимо реализовать с помощью класса `java.io.OutputStreamWriter`
- Все классы в программе должны быть задокументированы в формате `javadoc`.
- Программа должна корректно работать с неправильными данными (ошибки пользовательского ввода, отсутствие прав доступа к файлу и т.п.).

В интерактивном режиме программа должна поддерживать выполнение следующих команд:

- `help`: вывести справку по доступным командам
- `info`: вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.)
- `show`: вывести в стандартный поток вывода все элементы коллекции в строковом представлении
- `insert null {element}`: добавить новый элемент с заданным ключом
- `update id {element}`: обновить значение элемента коллекции, `id` которого равен заданному
- `remove_key id`: удалить элемент коллекции по его ключу
- `clear`: очистить коллекцию
- `save`: сохранить коллекцию в файл

- `execute_script file_name`: считать и исполнить скрипт из указанного файла. В скрипте содержатся команды в таком же виде, в котором их вводит пользователь в интерактивном режиме.
- `exit`: завершить программу (без сохранения в файл)
- `remove_greater {element}`: удалить из коллекции все элементы, превышающие заданный
- `remove_lower {element}`: удалить из коллекции все элементы, меньшие, чем заданный
- `replace_if_lower null {element}`: заменить значение по ключу, если новое значение меньше старого
- `min_by_id`: вывести любой объект из коллекции, значение поля `id` которого является минимальным
- `count_by_type type`: вывести количество элементов, значение поля `type` которых равно заданному
- `count_less_than_engine_power enginePower`: вывести количество элементов, значение поля `enginePower` которых меньше заданного

Формат ввода команд:

- Все аргументы команды, являющиеся стандартными типами данных (примитивные типы, классы-оболочки, `String`, классы для хранения дат), должны вводиться в той же строке, что и имя команды.
- Все составные типы данных (объекты классов, хранящиеся в коллекции) должны вводиться по одному полю в строку.
- При вводе составных типов данных пользователю должно показываться приглашение к вводу, содержащее имя поля (например, "Введите дату рождения:")
- Если поле является `enum`'ом, то вводится имя одной из его констант (при этом список констант должен быть предварительно выведен).
- При некорректном пользовательском вводе (введена строка, не являющаяся именем константы в `enum`'е; введена строка вместо числа; введённое число не входит в указанные границы и т.п.) должно быть показано сообщение об ошибке и предложено повторить ввод поля.
- Для ввода значений `null` использовать пустую строку.
- Поля с комментарием "Значение этого поля должно генерироваться автоматически" не должны вводиться пользователем вручную при добавлении.

Описание хранимых в коллекции классов:

```

public class Vehicle {
    private Integer id; //Поле не может быть null, Значение поля должно быть больше 0,
    // Значение этого поля должно быть уникальным, Значение этого поля должно
    // генерироваться автоматически
    private String name; //Поле не может быть null, Строка не может быть пустой
    private Coordinates coordinates; //Поле не может быть null
    private java.time.LocalDate creationDate; //Поле не может быть null,
    // Значение этого поля должно генерироваться автоматически
    private double enginePower; //Значение поля должно быть больше 0
    private VehicleType type; //Поле может быть null
    private FuelType fuelType; //Поле не может быть null
}
public class Coordinates {
    private Integer x; //Значение поля должно быть больше -523,
    // Поле не может быть null
    private long y;
}
public enum VehicleType {
    PLANE,
    SUBMARINE,
    BOAT,
    BICYCLE;
}
public enum FuelType {
    GASOLINE,
    ELECTRICITY,
    MANPOWER,
    PLASMA,
    ANTIMATTER;
}

```

2 Исходный код программы

Исходный код доступен в git-репозитории по адресу <https://github.com/FEgor04/labs/tree/main/programming/lab5>



Рис. 1: UML-диаграмма классов

3 Вывод

Во время выполнения данной лабораторной работы я научился работать с классами `java.io.InputStreamReader`, `java.io.OutputStreamWriter`, закрепил знания ООП и SOLID на практике, изучил возможности `gradle` по тестированию с помощью библиотек `JUnit5` и `mockk`, реализовал генерацию отчета о покрытии с помощью `jacoco` и генерацию HTML-документации с помощью плагина `dokka`.