Федеральное государственное автономное образовательное учреждение
высшего образования “Национальный исследовательский университет ИТМО”

Факультет Программной Инженерии И Компьютерной Техники

Лабораторная работа №5

Вариант 2875

Выполнила:

Абдуллаева София Улугбековна

Группа P3108

Проверил:

Миху Вадим Дмитриевич

Санкт-Петербург 2025

Задание

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса Organization, 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.LinkedHashSet
- 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.BufferedOutputStream
- Все классы в программе должны быть задокументированы в формате javadoc.
- Программа должна корректно работать с неправильными данными (ошибки пользовательского ввода, отсутсвие 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
- add {element} : добавить новый элемент в коллекцию
- update id {element} : обновить значение элемента коллекции, id которого равен заданному
- remove_by_id id : удалить элемент из коллекции по его id
- 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)
- add_if_max {element} : добавить новый элемент в коллекцию, если его значение превышает значение наибольшего элемента этой коллекции
- add_if_min {element} : добавить новый элемент в коллекцию, если его значение меньше, чем у наименьшего элемента этой коллекции
- history : вывести последние 15 команд (без их аргументов)

- remove_all_by_annual_turnover annualTurnover : удалить из коллекции все элементы, значение поля annualTurnover которого эквивалентно заданному
- sum_of_annual_turnover : вывести сумму значений поля annualTurnover для всех элементов коллекции
- max_by_postal_address : вывести любой объект из коллекции, значение поля postalAddress которого является максимальным

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 Organization {
    private Long 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.ZonedDateTime creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически
    private long annualTurnover; //Значение поля должно быть больше 0
    private OrganizationType type; //Поле может быть null
    private Address postalAddress; //Поле не может быть null
}
public class Coordinates {
    private Float x; //Значение поля должно быть больше -947, Поле не может быть null
    private Long y; //Поле не может быть null
}
public class Address {
    private String street; //Длина строки не должна быть больше 122, Поле не может быть null
}
public enum OrganizationType {
    COMMERCIAL,
    PUBLIC,
    TRUST,
    OPEN_JOINT_STOCK_COMPANY;
}
```

Диаграмма классов реализованной объектной модели

Исходный код программы

Можно посмотреть в репозитории:
https://github.com/LunarSonic/programming_lab5

Вывод

В процессе выполнения лабораторной работы я применила знания ООП и принципы SOLID и познакомилась с параметризованными типами. Кроме того, я изучила новые интерфейсы, такие как Map, Set, Queue, Deque, Iterable и Iterator. Впервые узнала об интерфейсах Comparator и Comparable, о потоках ввода-вывода и потоках-фильтрах и научилась работать с ними.