Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №5

По программированию

Вариант 4443

Выполнил:

Ларионов Владислав Васильевич

Группа P3109

Проверил:

Мустафаева Айнур Вугар Кызы

Санкт-Петербург 2024

Содержание

[**Задание** 3](#_Toc192125971)

[**Разработанная программа должна удовлетворять следующим требованиям:** 3](#_Toc192125972)

[**В интерактивном режиме программа должна поддерживать выполнение следующих команд:** 4](#_Toc192125973)

[**Формат ввода команд:** 5](#_Toc192125974)

[**Описание хранимых в коллекции классов:** 6](#_Toc192125975)

[**Программа** 9](#_Toc192125976)

[**Вывод:** 9](#_Toc192125977)

# **Задание**

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса City, 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 json
* Чтение данных из файла необходимо реализовать с помощью класса java.io.FileReader
* 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, 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
* 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)
* remove\_greater {element} : удалить из коллекции все элементы, превышающие заданный
* remove\_lower {element} : удалить из коллекции все элементы, меньшие, чем заданный
* history : вывести последние 12 команд (без их аргументов)
* min\_by\_meters\_above\_sea\_level : вывести любой объект из коллекции, значение поля metersAboveSeaLevel которого является минимальным
* count\_less\_than\_governor governor : вывести количество элементов, значение поля governor которых меньше заданного
* filter\_greater\_than\_standard\_of\_living standardOfLiving : вывести элементы, значение поля standardOfLiving которых больше 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 City {

private int id; //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 int area; //Значение поля должно быть больше 0

private Integer population; //Значение поля должно быть больше 0, Поле не может быть null

private int metersAboveSeaLevel;

private Climate climate; //Поле может быть null

private Government government; //Поле не может быть null

private StandardOfLiving standardOfLiving; //Поле не может быть null

private Human governor; //Поле не может быть null

}

public class Coordinates {

private long x;

private Double y; //Значение поля должно быть больше -339, Поле не может быть null

}

public class Human {

private long height; //Значение поля должно быть больше 0

private java.time.LocalDateTime birthday;

}

public enum Climate {

MEDITERRANIAN,

SUBARCTIC,

TUNDRA;

}

public enum Government {

PUPPET\_STATE,

OLIGARCHY,

PATRIARCHY,

REPUBLIC,

THEOCRACY;

}

public enum StandardOfLiving {

ULTRA\_HIGH,

VERY\_HIGH,

VERY\_LOW,

NIGHTMARE;

}

**Отчёт по работе должен содержать:**

1. Текст задания.
2. Диаграмма классов разработанной программы.
3. Исходный код программы.
4. Выводы по работе.

**Вопросы к защите лабораторной работы:**

1. Коллекции. Сортировка элементов коллекции. Интерфейсы java.util.Comparable и java.util.Comparator.
2. Категории коллекций - списки, множества. Интерфейс java.util.Map и его реализации.
3. Параметризованные типы. Создание параметризуемых классов. Wildcard-параметры.
4. Классы-оболочки. Назначение, область применения, преимущества и недостатки. Автоупаковка и автораспаковка.
5. Потоки ввода-вывода в Java. Байтовые и символьные потоки. "Цепочки" потоков (Stream Chains).
6. Работа с файлами в Java. Класс java.io.File.
7. Пакет java.nio - назначение, основные классы и интерфейсы.
8. Утилита javadoc. Особенности автоматического документирования кода в Java.

# **Программа**

Ссылка на репозиторий, содержащий код, jar-архив, диаграмму классов:

[ITMO\_VT/prog/lab5\_dir at main · vladlenblch/ITMO\_VT](https://github.com/vladlenblch/ITMO_VT/tree/main/prog/lab5_dir)

# **Вывод:**

Во время выполнения данной лабораторной работы я научился:

1. Работать с коллекциями
2. Писать javadoc документации
3. Работать с файлами

Также я повторил свои знания в написании программ на языке программирования Java.