

Санкт-Петербургский Национальный Исследовательский  
Университет ИТМО  
Мегафакультет Компьютерных Технологий и Управления  
Факультет Программной Инженерии и Компьютерной Техники



**Вариант №255687**  
**Лабораторная работа №5**  
по дисциплине  
**Программирование**

Выполнил Студент группы Р3116  
**Student's name**  
Преподаватель:  
**Teacher's name**

г. Санкт-Петербург  
2021г.

# Содержание

1	Текст задания	3
2	Диаграмма классов объектной модели.	7
3	Исходный код программы	8
4	Выводы по работе	8

# 1 Текст задания

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса `Route`, 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.BufferedReader`
- 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
- `add {elemen}` : 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
- **remove\_lower {element}** : удалить из коллекции все элементы, меньшие, чем заданный
- **history** : вывести последние 10 команд (без их аргументов)
- **remove\_all\_by\_distance distance** : удалить из коллекции все элементы, значение поля distance которого эквивалентно заданному
- **min\_by\_distance** : вывести любой объект из коллекции, значение поля distance которого является минимальным
- **max\_by\_creation\_date** : вывести любой объект из коллекции, значение поля creationDate 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 Route {
    private int id; //Значение поля должно быть больше 0, Значение этого поля должно быть
    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 по
    private Location from; //Поле не может быть null
    private Location to; //Поле не может быть null
    private Long distance; //Поле не может быть null, Значение поля должно быть больше 1
}
```

```
public class Coordinates {  
    private Double x; //Максимальное значение поля: 669, Поле не может быть null  
    private double y;  
}  
public class Location {  
    private Integer x; //Поле не может быть null  
    private float y;  
    private double z;  
}  
public class Location {  
    private Float x; //Поле не может быть null  
    private Long y; //Поле не может быть null  
    private String name; //Поле не может быть null  
}
```



## 2 Диаграмма классов объектной модели.

### 3 Исходный код программы

<https://github.com/AaLexUser/ProgaLab5>

### 4 Выводы по работе

В этой лабораторной работе я познакомился с коллекциями в Java и с Java i/o, также я научился работать с Javadoc и интерфейсами Comparable и Comparator. Закрепил знание принципов SOLID и работы с исключениями.