МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА № 5

по дисциплине

‘ПРОГРАММИРОВАНИЕ’

Вариант №311322

Выполнили:
Студенты группы P3113
Орлов Егор Алексеевич
Свиридов Дмитрий Витальевич

Преподаватель:
Гаврилов Антон
Валерьевич

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2020

Задание:

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.TreeSet`
- 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.util.Scanner`
- Запись данных в файл необходимо реализовать с помощью класса `java.io.FileWriter`
- 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_min {element}` : добавить новый элемент в коллекцию, если его значение меньше, чем у наименьшего элемента этой коллекции
- `remove_greater {element}` : удалить из коллекции все элементы, превышающие заданный
- `history` : вывести последние 8 команд (без их аргументов)
- `sum_of_health` : вывести сумму значений поля health для всех элементов коллекции
- `max_by_melee_weapon` : вывести любой объект из коллекции, значение поля meleeWeapon которого является максимальным
- `filter_by_weapon_type weaponType` : вывести элементы, значение поля weaponType которых равно заданному

```
public class SpaceMarine {
    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.LocalDate creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически
    private double health; //Значение поля должно быть больше 0
    private AstartesCategory category; //Поле может быть null
    private Weapon weaponType; //Поле не может быть null
    private MeleeWeapon meleeWeapon; //Поле не может быть null
    private Chapter chapter; //Поле может быть null
}
public class Coordinates {
    private double x;
    private Float y; //Максимальное значение поля: 262, Поле не может быть null
}
public class Chapter {
    private String name; //Поле не может быть null, Строка не может быть пустой
    private long marinesCount; //Значение поля должно быть больше 0, Максимальное значение поля: 1000
}
public enum AstartesCategory {
    DREADNOUGHT,
    ASSAULT,
    TACTICAL,
    CHAPLAIN,
    APOTHECARY;
}
public enum Weapon {
    HEAVY_BOLTGUN,
    BOLT_PISTOL,
    GRAV_GUN;
}
public enum MeleeWeapon {
    CHAIN_SWORD,
    CHAIN_AXE,
    LIGHTING_CLAW,
    POWER_BLADE,
    POWER_FIST;
}
```

Исходный код доступен по ссылке или QR-коду:

https://github.com/slamach/prog-lab5

UML диаграмма классов доступна по ссылке или QR-коду:

https://github.com/slamach/prog-lab5/blob/master/doc/uml.png

Вывод:

Во время выполнения данной лабораторной работы мы закрепили принципы SOLID, собственные исключения и многое другое. Также мы научились использовать Javadoc, работать с потоками, файлами, интерфейсами Comparable и Comparator. Узнали что такое сериализация и десериализация.