Федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(ФПИиКТ)

ОТЧЕТ ПО ЛАБАРОТОРНОЙ РАБОТЕ №5

Вариант 406650

Выполнил:

Студент группы P3115

Зыков Иван Евгеньевич

Проверил:

Вербовой Александр Александрович

Санкт-Петербург 2023

Оглавление

Оглавление 2
Задание 3
Выполнение 4
Диаграмма классов 4
Вывод 7

Задание

Лабораторная работа #5

Введите вариант: 406650

Внимание! У разных вариантов разный текст задания!

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса `SpaceMarine`, 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.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.io.InputStreamReader`
- 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_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 6 команд (без их аргументов)
- `count_less_than_chapter chapter` : вывести количество элементов, значение поля chapter которых меньше заданного
- `count_less_than_chapter chapter` : вывести количество элементов, значение поля chapter которых меньше заданного
- `filter_contains_name name` : вывести элементы, значение поля name которых содержит заданную подстроку
- `print_field_descending_melee_weapon` : вывести значения поля meleeWeapon всех элементов в порядке убывания

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 SpaceMarine {
    private long 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.LocalDateTime creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически
    private Double health; //Поле не может быть null, Значение поля должно быть больше 0
    private Integer height; //Поле может быть null
    private AstartesCategory category; //Поле не может быть null
    private MeleeWeapon meleeWeapon; //Поле может быть null
    private Chapter chapter; //Поле может быть null
}
public class Coordinates {
    private float x; //Максимальное значение поля: 938
    private Double y; //Значение поля должно быть больше -841, Поле не может быть null
}
public class Chapter {
    private String name; //Поле не может быть null, Строка не может быть пустой
    private String parentLegion;
    private Integer marinesCount; //Поле не может быть null, Значение поля должно быть больше 0, Максимальное значение поля: 1000
    private String world; //Поле может быть null
}
public enum AstartesCategory {
```

```
public class Chapter {
    private String name; //Поле не может быть null, Строка не может быть пустой
    private String parentLegion;
    private Integer marinesCount; //Поле не может быть null, Значение поля должно быть больше 0, Максимальное значение поля: 1000
    private String world; //Поле может быть null
}
public enum AstartesCategory {
    SCOUT,
    CHAPLAIN,
    HELIX;
}
public enum MeleeWeapon {
    CHAIN_SWORD,
    POWER_SWORD,
    CHAIN_AXE,
    MANREAPER,
    POWER_BLADE;
}
```

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:

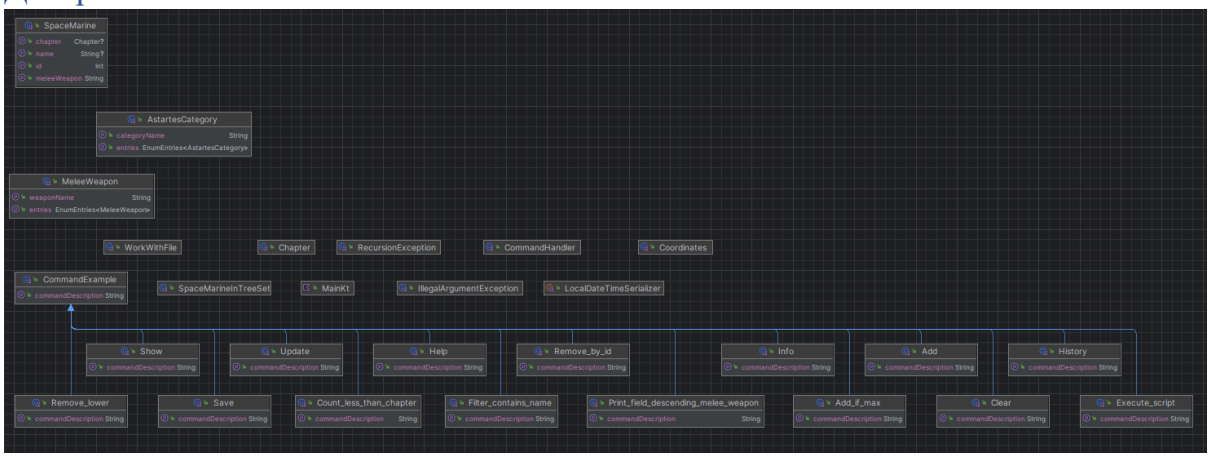
1. Коллекции. Сортировка элементов коллекции. Интерфейсы `java.util.Comparable` и `java.util.Comparator`.
2. Категории коллекций - списки, множества. Интерфейс `java.util.Map` и его реализации.
3. Параметризованные типы. Создание параметризуемых классов. Wildcard-параметры.
4. Классы-оболочки. Назначение, 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.

Лабораторная работа #6

Введите вариант:

Выполнение

Диаграмма классов



Исходный код программы

https://github.com/net0pyr/prog_labs.git

```kotlin
package com.net0pyr.commands

import ...

/**
 * Класс команды execution_script file_name
 * @author net0pyr
 */
net0pyr *
class Execute_script: CommandExample() {
    /**Поле описания команды*/
    override val commandDescription = "execution_script file_name: считать и исполнить скрипт из указанного файла"

    /** Метод исполнения команды
     * @param commandArgument аргумент команды
     */
    net0pyr
    override fun commandExecution(commandArgument: String?) {

            try {
                var thisCommad = "execute_script $commandArgument"
                val inputStreamReader = InputStreamReader(FileInputStream(commandArgument))
                val reader = BufferedReader(inputStreamReader)
                var line: String?
                while (reader.readLine().also { line = it } != null) {
                    if(line.equals(thisCommad)) {
                        throw RecursionException("\u001B[31mОшибка:\u001B[0m Рекурсивный вызов скрипта")
                    }
                    var commandHandler = CommandHandler()
                    line?.let { commandHandler.handler(it) }
                }
            } catch (e: NullPointerException) {
                println("\u001B[31mОшибка:\u001B[0m Нет скрипта с таким названием")
            } catch (e: RecursionException) {
                println("\u001B[31mОшибка:\u001B[0m Рекурсивный вызов скрипта")
            }

    }
}
```

```kotlin
package com.net0pyr.entity

import ...

/**
 * Класс описывает космических десантников
 * @property name имя десантника
 * @property coordinates координаты десантника
 * @property health количество здоровья у десантника
 * @property height рост десантника
 * @property category вид войск
 * @property meleeWeapon вид оружия
 * @property chapter отряд
 * @property id
 * @author net0pyr
 */
net0pyr *
@Serializable
class SpaceMarine (private var name: String?, private var coordinates: Coordinates?, private var health: Double?, private var height: Int? = null,
                   private var category: AstartesCategory?, private var meleeWeapon: MeleeWeapon? = null, private var chapter: Chapter? = null, private var id: Int = 0): Comparable<SpaceMarine> {
    @Serializable(with = LocalDateTimeSerializer::class)
    /**Поле даты создания*/
    private val creationDate: LocalDateTime = LocalDateTime.now()
    net0pyr
    init {
        id = abs(creationDate.toString().hashCode())
        fun checkHealth(health: Double?) {
            if(health == null) {
                throw IllegalArgumentException("Пехотинец прошел жестокую подготовку и был воспитан войном. Его здоровье не может быть нулевым.")
            } else if (health <= 0) {
                throw IllegalArgumentException("Пехотинец прошел жестокую подготовку и был воспитан войном. Его здоровье не может быть отрицательным.")
            }
        }
        try {
            checkHealth(health!!);
        } catch (e: IllegalArgumentException) {
            println("Ошибка: ${e.message}")
        }
```

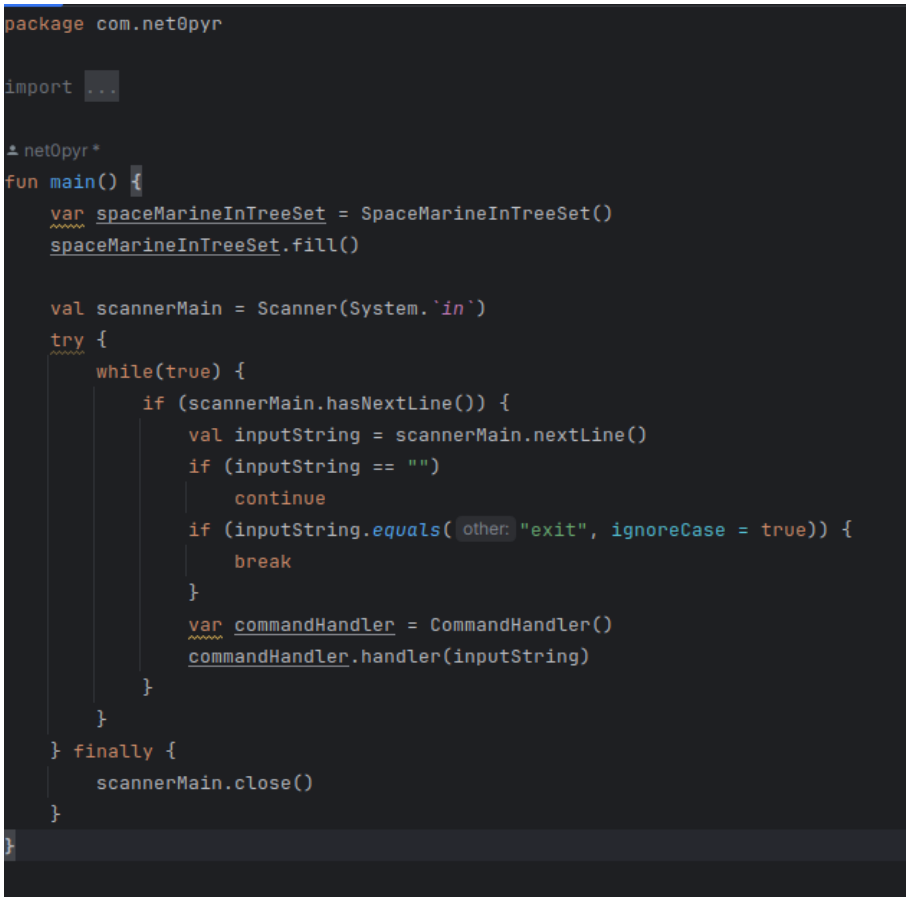
```
        fun checkName(name: String?) {
            if(name == "" || name == null) {
                throw IllegalArgumentException("Пехотинцу нужно имя, иначе его не смогут опознать после ранения.")
            }
        }
        try {
            checkName(name);
        } catch (e: IllegalArgumentException) {
            println("Ошибка: ${e.message}")
        }

        fun checkCoordinates(coordinates: Coordinates?) {
            if(coordinates == null) {
                throw IllegalArgumentException("Пехотинец славно служит империуму, нельзя выкидывать его в космос! Задайте координаты его корабля.")
            }
        }
        try {
            checkCoordinates(coordinates);
        } catch (e: IllegalArgumentException) {
            println("Ошибка: ${e.message}")
        }

        fun checkCategory(category: AstartesCategory?) {
            if(category == null) {
                throw IllegalArgumentException("Пехотинец определен как дезертир и не числится в составе армии, пока не будет задан вид войск, к которым он принадлежит! Задайте вид войск, к которым прин
            }
        }
        try {
            checkCategory(category);
        } catch (e: IllegalArgumentException) {
            println("Ошибка: ${e.message}")
        }
    }
    /** Метод переопределяющий компоратор
     * @param SpaceMarine объект с которым сравниваем
     */
    ♟ netOpyr
    override fun compareTo(other: SpaceMarine): Int {
        return (this.id - other.id).toInt()
    }
```

```
    ♟ netOpyr
    fun getId(): Int {
        return id
    }
    /** Метод для изменения поля id
     * @param newId новое значение id
     * */
    ♟ netOpyr
    fun setId(newId: Int) {
        id = newId
    }
    /** Метод для получения поля chapter*/
    ♟ netOpyr
    fun getChapter(): Chapter? {
        return chapter
    }
    /** Метод для получения поля name*/
    ♟ netOpyr
    fun getName(): String? {
        return name
    }
    /** Метод для получения поля meleeWeapon*/
    ♟ netOpyr
    fun getMeleeWeapon(): String {
        return meleeWeapon!!.weaponName
    }
    ♟ netOpyr
    override fun hashCode(): Int {
        return id
    }

    ♟ netOpyr
    override fun toString(): String {
        return "Имя: $name, id: $id"
    }
}
```



```
package com.net0pyr

import ...

net0pyr *
fun main() {
    var spaceMarineInTreeSet = SpaceMarineInTreeSet()
    spaceMarineInTreeSet.fill()

    val scannerMain = Scanner(System.`in`)
    try {
        while(true) {
            if (scannerMain.hasNextLine()) {
                val inputString = scannerMain.nextLine()
                if (inputString == "")
                    continue
                if (inputString.equals( other: "exit", ignoreCase = true)) {
                    break
                }
                var commandHandler = CommandHandler()
                commandHandler.handler(inputString)
            }
        }
    } finally {
        scannerMain.close()
    }
}
```

Вывод

В ходе выполнения работы я научился работать с потоками ввода и вывода, а также с файлами. Сделал документацию и познакомился с коллекциями.