

Национальный исследовательский университет информационных технологий,
механики и оптики

Факультет Программной Инженерии и Компьютерной Техники

Вариант № 3311706
Лабораторная работа №5
По дисциплине:
«Программирование»

Работу выполнила:
Студентка группы Р3112
Никонова Наталья Игоревна
Преподаватель:
Гаврилов Антон Валерьевич

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.LinkedList`
- При запуске приложения коллекция должна автоматически заполняться значениями из файла.
- Имя файла должно передаваться программе с помощью: **аргумент командной строки**.
- 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.BufferedWriter`
- 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)
- `insert_at index {element}` : добавить новый элемент в заданную позицию
- `sort` : отсортировать коллекцию в естественном порядке
- `history` : вывести последние 8 команд (без их аргументов)
- `remove_any_by_transport transport` : удалить из коллекции один элемент, значение поля transport которого эквивалентно заданному
- `average_of_number_of_rooms` : вывести среднее значение поля numberOfRooms для всех элементов коллекции
- `filter_less_than_view view` : вывести элементы, значение поля view которых меньше заданного

Описание хранимых в коллекции классов:

```
public class Flat {
    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.ZonedDateTime creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически
    private long area; //Значение поля должно быть больше 0
    private Integer numberOfRooms; //Поле может быть null, Значение поля должно быть больше 0
    private boolean centralHeating;
    private View view; //Поле не может быть null
    private Transport transport; //Поле не может быть null
    private House house; //Поле не может быть null
}

public class Coordinates {
    private double x;
    private Double y; //Поле не может быть null
}

public class House {
    private String name; //Поле может быть null
    private Long year; //Поле может быть null, Значение поля должно быть больше 0
    private Integer numberOfFloors; //Поле может быть null, Значение поля должно быть больше 0
}

public enum View {
    YARD,
    PARK,
    BAD,
    TERRIBLE;
}

public enum Transport {
    FEW,
    NONE,
    LITTLE,
    NORMAL;
}
```

Исходный код: <https://github.com/nanikon/FlatCollection>

Диаграмма:



```

public static void parseObject(ObjectArgument<?> arg, Scanner scr, boolean isConsole, boolean isPartly) throws ScriptException {
    HashMap<String[], ThrowConsumer<String>> params = arg.getParams();
    arg.clear();
    String[][] fields = params.keySet().toArray(new String[0][0]);
    Arrays.sort(fields, (line1, line2) -> Integer.parseInt(line1[0]) - Integer.parseInt(line2[0]));
    offer:
    for (String[] param : fields) {
        boolean right;
        if (isPartly) {
            do {
                if (isConsole) {
                    System.out.println("Хотите изменить поле " + param[1] + "? Если да, введите +, иначе -");
                }
                String line;
                try {
                    line = scr.nextLine();
                } catch (NoSuchElementException e) {
                    System.out.println("Скрипт закончился некорректно");
                    return;
                }
                if (line.equals("+")) {
                    break;
                } else if (line.equals("-")) {
                    continue offer;
                } else {
                    if (isConsole) {
                        System.out.println("Ожидалось +/-, встречено " + line + ". Попробуйте ещё раз.");
                    } else {
                        throw new ScriptException("Ошибка в скрипте!" + "Ожидалось +/-, встречено " + line);
                    }
                }
            } while (true);
        }
        right = true;
        if (isConsole) {
            System.out.println(param[0] + " Введите поле " + param[1] + ". " + param[2]);
        };
        try {
            params.get(param).accept(scr.nextLine());
        } catch (NumberFormatException e) {
            if (isConsole) {
                System.out.println("Ошибка! Введенная строка не является числом! Попробуйте ещё раз");
                right = false;
            } else {
                throw new ScriptException("Ошибка в скрипте! Ожидалось число, а встречена строка");
            }
        } catch (NullPointerException | IllegalArgumentException e) {
            if (isConsole) {
                System.out.println("Ошибка! " + e.getMessage() + " Попробуйте ещё раз.");
                right = false;
            } else {
                throw new ScriptException("Ошибка в скрипте! " + e.getMessage());
            }
        } catch (NotPositiveNumberException | BooleanInputException e) {
            if (isConsole) {
                System.out.println("Ошибка! " + e.getMessage() + " Попробуйте ещё раз.");
                right = false;
            } else {
                throw new ScriptException("Ошибка в скрипте!" + e.getMessage());
            }
        } catch (NoSuchElementException e) {
            System.out.println("Скрипт закончился некорректно");
        }
    }
    return;
} while (right);
}
arg.setValue("");
}

```

```

public static void parseEnum(EnumArgument<?> arg, String value, Scanner scr, boolean isConsole) throws ScriptException {
    boolean right = true;
    do {
        try {
            arg.setValue(value);
            right = true;
        } catch (IllegalArgumentException | NullPointerException e) {
            if (isConsole) {
                System.out.println("Ошибка! " + e.getMessage() + "Повторите ввод ещё раз, ");
                System.out.println(arg.getConstants());
                right = false;
                try {
                    value = scr.nextLine();
                } catch (NoSuchElementException ex) {
                    System.out.println("Скрипт закончился некорректно");
                    return;
                }
            } else {
                throw new ScriptException("В скрипте обнаружена ошибка! " + e.getMessage());
            }
        }
    } while (!right);
}

```

Файл FlatArg.java

```

public class FlatArg extends ObjectArgument<Flat> {
    private FlatBuilder builder = new FlatBuilder();
    {
        params.put(new String[]{"1", "имя квартиры", "Это должна быть непустая строка"}, builder::setName);
        params.put(new String[]{"2", "координата x", "Это должно быть целое или вещественное число с точкой в виде десятичной дроби"}, builder::setX);
        params.put(new String[]{"3", "координата y", "Это должно быть целое или вещественное число с точкой в виде десятичной дроби"}, builder::setY);
        params.put(new String[]{"4", "площадь квартиры", "Это должно быть целое число больше нуля, не превышающее " + Long.MAX_VALUE}, builder::setArea);
        params.put(new String[]{"5", "количество комнат в квартире", "Это должно быть целое число больше нуля, не превышающее " + Integer.MAX_VALUE}, builder::setRooms);
        params.put(new String[]{"6", "наличие центрального отопления", "Если в квартире есть центральное отопление, введите +, иначе введите -"}, builder::setHeating);
        params.put(new String[]{"7", "вид из окон квартиры", "Это должен быть один из следующих вариантов: " + Arrays.toString(View.values())}, builder::setView);
        params.put(new String[]{"8", "загруженность транспорта рядом с квартирой", "Это должен быть один из следующих вариантов: " + Arrays.toString(Load.values())}, builder::setLoad);
        params.put(new String[]{"9", "имя дома, в котором находится квартира", "Это должна быть любая строка, в том числе пустая"}, builder::setNameHouse);
        params.put(new String[]{"10", "год строительства дома", "Это должно быть целое число больше нуля и не превышающее " + Long.MAX_VALUE}, builder::setYear);
        params.put(new String[]{"11", "количество квартир в доме", "Это должно быть целое число больше нуля, не превышающее " + Integer.MAX_VALUE}, builder::setApartments);
    }
    @Override
    public void setValue(String value) { this.value = builder.getResult(); }

    @Override
    public void clear() {
        builder.reset();
        value = null;
    }

    public FlatBuilder getBuilder() { return builder; }
}

```

Файл IdArg.java

```

public class IdArg extends AbstractArgument<Integer> {
    @Override
    public void setValue(String value) {
        try {
            this.value = Integer.valueOf(value);
        } catch (NumberFormatException e) {
            if (value.equals("")) {
                throw new NullPointerException("Аргумент id не найден");
            } else {
                throw new NumberFormatException("Аргумент id должен быть числом");
            }
        }
    }
}

```

Файл FlatBuilder.java

```

public void setId(int id) {
    this.id = id;
    changed.add("id");
}

public void setName(String name) {
    if (name.equals("")) {
        throw new NullPointerException("Поле имя квартиры не может быть пустым!");
    } else {
        this.name = name;
        changed.add("name");
    }
}

public void setX(String value) {
    coordsBuilder.setX(value);
    changed.add("x");
}

public void setY(String value) {
    coordsBuilder.setY(value);
    changed.add("y");
}

public Flat getResult() { return new Flat(id, name, coordsBuilder.getResult(), area, numberOfRooms, centralHeating, view, transport, houseBuild); }

```

Файл AddCommand.java

```

public class AddCommand implements Command {
    private CollectionManager collection;
    private AbstractArgument<>[] params = {new FlatArg()};
    private String information = "'add {element}' - добавить новый элемент в коллекцию";

    public AddCommand(CollectionManager collection) { this.collection = collection; }

    @Override
    public void execute(AbstractArgument<>[] params) {
        FlatBuilder builder = ((FlatArg) params[0]).getBuilder();
        builder.setId(collection.generateNextId());
        Flat flat = builder.getResult();
        collection.addLast(flat);
    }

    @Override
    public AbstractArgument<>[] getArgs() { return params; }

    @Override
    public String getInformation() { return information; }
}

```

Выводы

В ходе выполнения лабораторной работы я познакомилась с некоторыми коллекциями в Java, обобщенным программированием, некоторыми потоками ввода\вывода. Опробовала такие паттерны проектирования, как Команда и Билдер. Вспомнила принципы SOLID и снова попыталась применять их при проектировании классов, поработала с собственными исключениями, функциональными интерфейсами вместе с лямбда-выражениями и ссылками на методы. Попробовала подключить к своему проекту стороннюю библиотеку.