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

Факультет программной инженерии и компьютерных технологий

Лабораторная работа №5

Вариант 407271

Выполнила:

Павличенко Софья Алексеевна, P3115

Проверил:

Вербовой Александр Александрович

Санкт-Петербург 2024г.

Оглавление

Задание....3
Диаграмма классов реализованной объектной модели 6
Решение 7
Исходный код программы....7
Результат работы программы ....**Ошибка! Закладка не определена.**
Заключение....9

Задание

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса StudyGroup, 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.PriorityQueue
- 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 xml
- Чтение данных из файла необходимо реализовать с помощью класса java.io.InputStreamReader
- 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, 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)
- head : вывести первый элемент коллекции
- remove_greater {element} : удалить из коллекции все элементы, превышающие заданный
- history : вывести последние 14 команд (без их аргументов)
- count_less_than_students_count studentsCount : вывести количество элементов, значение поля studentsCount которых меньше заданного
- print_descending : вывести элементы коллекции в порядке убывания

- print_field_ascending_form_of_education : вывести значения поля formOfEducation всех элементов в порядке возрастания

Формат ввода команд:

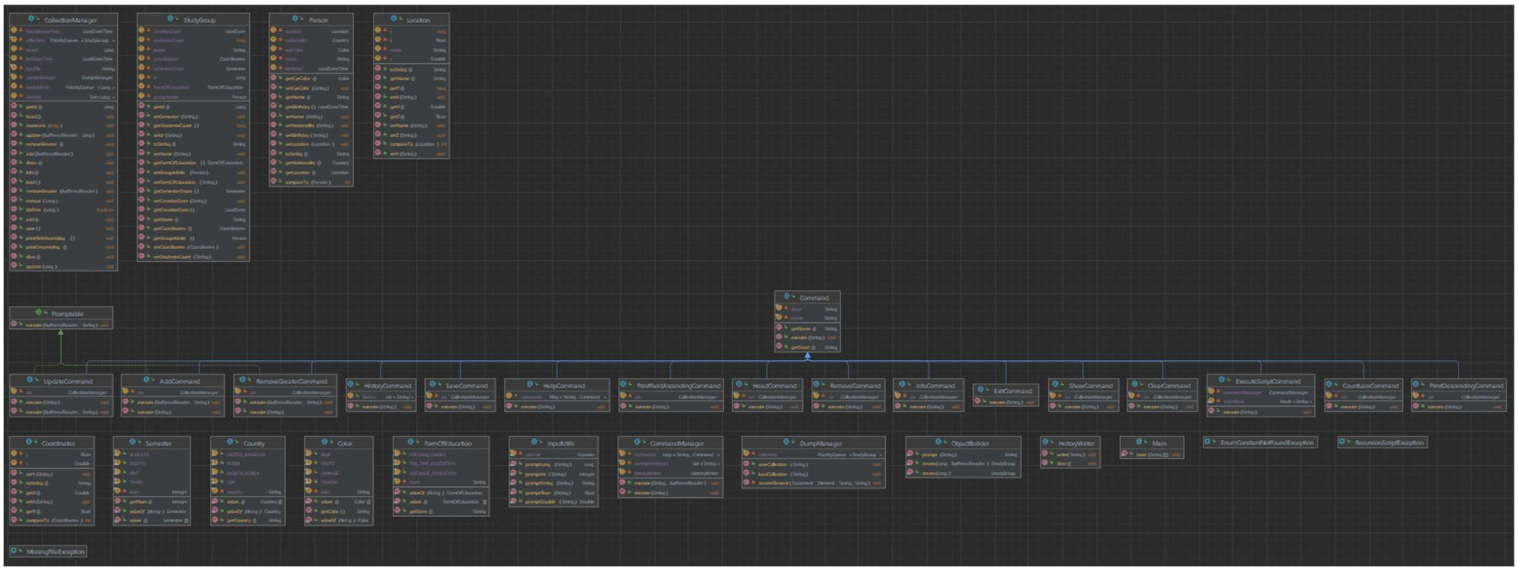
- Все аргументы команды, являющиеся стандартными типами данных (примитивные типы, классы-оболочки, String, классы для хранения дат), должны вводиться в той же строке, что и имя команды.
- Все составные типы данных (объекты классов, хранящиеся в коллекции) должны вводиться по одному полю в строку.
- При 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 StudyGroup {
    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 long studentsCount; //Значение поля должно быть больше 0
    private FormOfEducation formOfEducation; //Поле не может быть null
    private Semester semesterEnum; //Поле может быть null
    private Person groupAdmin; //Поле не может быть null
}
public class Coordinates {
    private Double x; //Значение поля должно быть больше -190, Поле не может быть
null
    private Float y; //Поле не может быть null
}
public class Person {
    private String name; //Поле не может быть null, Строка не может быть пустой
    private java.time.LocalDateTime birthday; //Поле не может быть null
    private Color eyeColor; //Поле может быть null
    private Country nationality; //Поле не может быть null
    private Location location; //Поле не может быть null
}
public class Location {
    private Double x; //Поле не может быть null
    private long y;
    private Float z; //Поле не может быть null
    private String name; //Поле не может быть null
}
public enum FormOfEducation {
    DISTANCE_EDUCATION,
```

```
    FULL_TIME_EDUCATION,
    EVENING_CLASSES;
}
public enum Semester {
    FIRST,
    THIRD,
    SEVENTH,
    EIGHTH;
}
public enum Color {
    BLUE,
    YELLOW,
    ORANGE,
    WHITE;
}
public enum Country {
    RUSSIA,
    UNITED_KINGDOM,
    USA,
    NORTH_KOREA;
}
```

Диаграмма классов реализованной объектной модели



Решение

Исходный код программы

Main.java

```
import managers.CollectionManager;
import managers.CommandManager;
import models.StudyGroup;

import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        Set<String> availableFields = new
LinkedHashSet<>(Arrays.asList("id", "название", "координаты", "дата
создания", "число студентов", "форма обучения", "семестр", "админ
группы"));
        System.out.println("Введите поля для сортировки коллекции
через запятую. Доступны: " + String.join(", ", availableFields) +
".");
        String[] fields = sc.nextLine().split(",");
        Set<String> sortFields = new HashSet<>();
        for (String field : fields) {
            if (availableFields.contains(field.trim().toLowerCase()))
{
                sortFields.add(field.trim().toLowerCase());
            } else {
                System.err.println("Неизвестное поле для сортировки: "
+ field);
            }
        }

        if (sortFields.isEmpty()) {
            System.out.println("Коллекция сохраняет порядок добавления
элементов, без дополнительной сортировки.");
        } else {
            System.out.println("Коллекция будет сортироваться по
полю(-ям): " + String.join(", ", sortFields));
        }

        PriorityQueue<StudyGroup> collection = new
PriorityQueue<>((group1, group2) -> {
            int result = 0;
            for (String field : fields) {
                switch (field.trim().toLowerCase()) {
                    case "id" -> result +=
group1.getId().compareTo(group2.getId());
                    case "название" -> result +=
group1.getName().compareTo(group2.getName());
                    case "координаты" -> result +=
group1.getCoordinates().compareTo(group2.getCoordinates());
                    case "дата создания" -> result +=
group1.getCreationDate().compareTo(group2.getCreationDate());
```

```
                    case "число студентов" -> result +=
Long.compare(group1.getStudentsCount(), group2.getStudentsCount());
                    case "форма обучения" -> result +=
group1.getFormOfEducation().compareTo(group2.getFormOfEducation());
                    case "семестр" -> result +=
group1.getSemesterEnum().compareTo(group2.getSemesterEnum());
                    case "админ группы" -> result +=
group1.getGroupAdmin().compareTo(group2.getGroupAdmin());
                }
            }
            return Math.max(Math.min(result, 1), -1);
        });

        String loadFile = null;
        if (args.length > 0) loadFile = args[0];
        CollectionManager collectionManager = new
CollectionManager(collection, loadFile);
        CommandManager commandManager = new
CommandManager(collectionManager);

        while (sc.hasNext()) {
            String line = sc.nextLine().trim();
            if (!line.isEmpty()) commandManager.execute(line);
        }
    }
}
```

Заключение

В результате выполнения лабораторной работы я научилась работать с коллекциями в Java, классами-оболочками, реализовывать сортировку через Comparable и Comparator и парсинг XML-файлов.