Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Программирование

Лабораторная работа №5

Работу выполнил:

Смирнова Ольга

Группа: Р3114

Вариант 311740

Санкт-Петербург

2020 г.

1. Текст задания.

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса Ticket, 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.HashSet
* 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.BufferedOutputStream
* 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 5 команд (без их аргументов)
* group\_counting\_by\_price : сгруппировать элементы коллекции по значению поля price, вывести количество элементов в каждой группе
* count\_by\_type type : вывести количество элементов, значение поля type которых равно заданному
* filter\_greater\_than\_type type : вывести элементы, значение поля type которых больше заданного

**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 Ticket {

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.LocalDateTime creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически

private long price; //Значение поля должно быть больше 0

private TicketType type; //Поле может быть null

private Venue venue; //Поле не может быть null

}

public class Coordinates {

private Integer x; //Значение поля должно быть больше -27, Поле не может быть null

private double y; //Значение поля должно быть больше -279

}

public class Venue {

private Integer 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 Integer capacity; //Поле не может быть null, Значение поля должно быть больше 0

}

public enum TicketType {

VIP,

USUAL,

BUDGETARY,

CHEAP;

}

**Отчёт по работе должен содержать:**

1. Текст задания.
2. Диаграмма классов разработанной программы.
3. Исходный код программы.
4. Выводы по работе.

2, 3. Исходный код и диаграмма классов разработанной программы:

<https://github.com/OLyaSHaaa/Prog-2>

4. Выводы по работе.

Во время выполнения лабораторной работы научилась работать с коллекциями в Java, применяла не только коллекцию HashSet, реализация работы с которой была необходима по заданию, но и такие коллекции, как HashMap – для хранения соответствий между объектом команды и строкой ее вызова, HashSet – для хранения всех id элементов и хранения списка запущенных скриптов, чтобы исключить случаи конфликта id элементов и предотвращения вызова рекурсии с помощью команды execute\_script, Stack – для хранения истории команд. Понял, что паттерны – это очень круто, их использование облегчает написание кода. Также получил много практики в обработке исключений. Много времени было потрачено на поиск багов в программе. Рейтинг вылезших исключений при изменении чего-либо возглавил NullPointerException, который я поймала не менее 20 раз.