Санкт-Петербургский Национальный Исследовательский Университет ИТМО
Факультет Программной Инженерии и Компьютерной Техники

Вариант №53947
Лабораторная работа №5
По дисциплине
Программирование

Выполнил студент группы P3132:
Юксель Хамза

Преподаватель:
Лазеев Сергей

Санкт-Петербург 2025 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.LinkedHashMap`
- 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.StreamReader`
- 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, 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`
- `insert null {element} : добавить новый элемент с заданным ключом`
- `update id {element} : обновить значение элемента коллекции, id которого равен заданному`
- `remove_key null : удалить элемент из коллекции по его ключу`
- `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)`
- `remove_lower {element} : удалить из коллекции все элементы, меньшие, чем заданный`
- `history : вывести последние 5 команд (без их аргументов)`
- `replace_if_lowe null {element} : заменить значение по ключу, если новое значение меньше старого`
- `count_greater_than_discount discount : вывести количество элементов, значение поля discount которых больше заданного`
- `filter_starts_with_name name : вывести элементы, значение поля name которых начинается с заданной подстроки`
- `print_descending : вывести элементы коллекции 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 Ticket {
    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 Coordinates coordinates; //Поле не может быть null
    private java.time.LocalDate creationDate; //Поле не может быть null, Значение этого поля должно ге-
нерироваться автоматически
    private int price; //Значение поля должно быть больше 0
    private long discount; //Значение поля должно быть больше 0, Максимальное значение поля: 100
    private String comment; //Длина строки не должна быть больше 631, Поле может быть null
    private TicketType type; //Поле не может быть null
    private Event event; //Поле может быть null
}
public class Coordinates {
    private float x; //Значение поля должно быть больше -661
    private Float y; //Значение поля должно быть больше -493, Поле не может быть null
}
public class Event {
    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 java.time.ZonedDateTime date; //Поле может быть null
    private EventType eventType; //Поле не может быть null
}
public enum TicketType {
    VIP,
    USUAL,
    BUDGETARY,
    CHEAP;
}
public enum EventType {
    CONCERT,
    E_SPORTS,
    BASKETBALL;
}
```

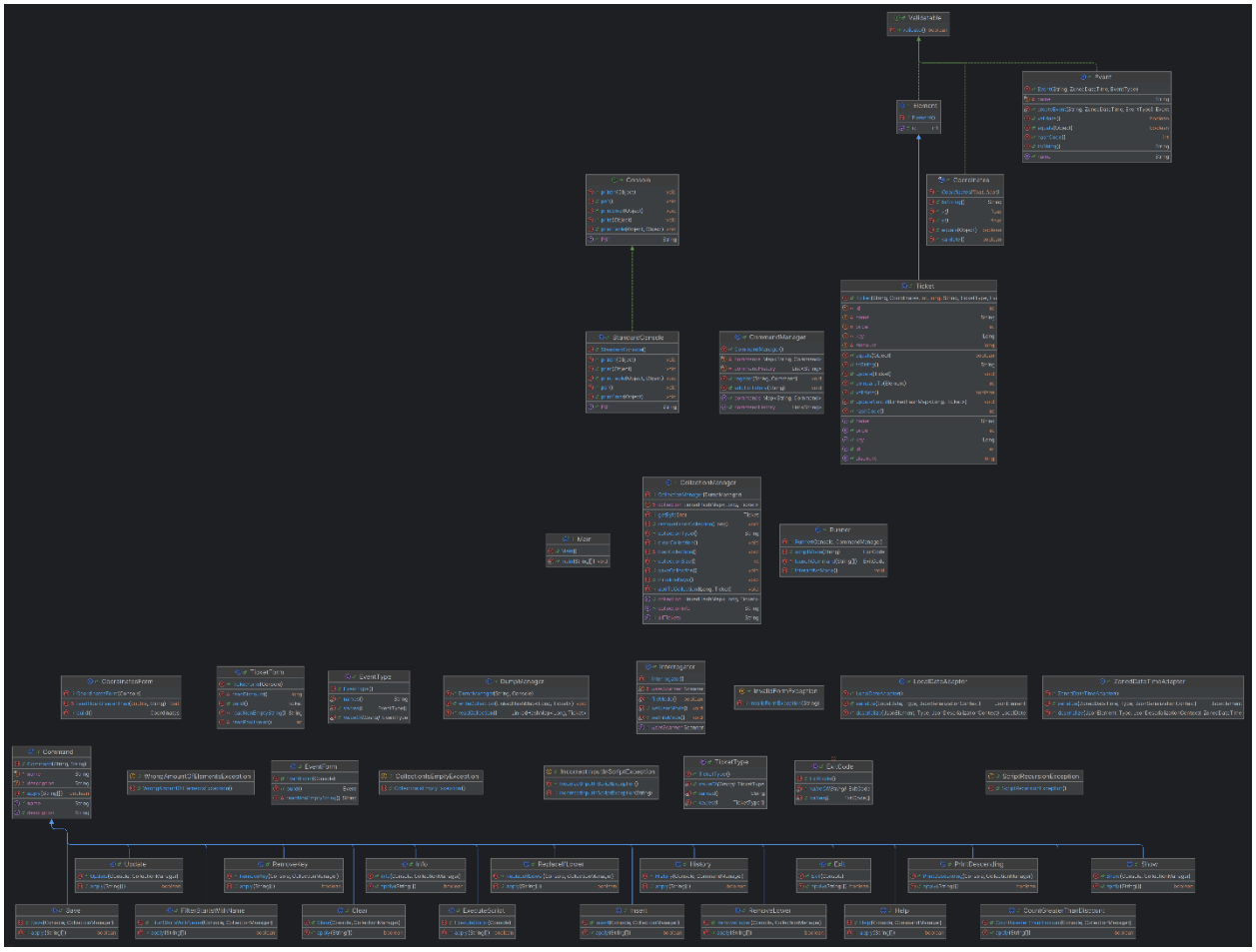
2. Исходный код программы.

Репозиторий:
https://github.com/yukselh20/ticket-console-app.git

3. Диаграмма классов реализованной объектной модели.

https://github.com/yukselh20/ticket-console-app/blob/main/diagram/uml-with-depend.png

https://github.com/yukselh20/ticket-console-app/blob/main/diagram/uml-no-depend.png



4. Вывод

Во время выполнения данной лабораторной работы я научился работать с различными структурами данных в Java и файлами, а также углубил свои знания о ООП в Java, изучил параметризованные типы, wildcard-параметры и утилиту javadoc.