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

Вариант №31391
Лабораторная работа №5
По дисциплине
Программирование

Выполнил студент группы P3115:
Кохан Всеволод Андреевич

Преподаватель:
Вербовой Александр Александрович

Санкт-Петербург 2024 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.TreeSet`
- 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 `csv`
- Чтение данных из файла необходимо реализовать с помощью класса `java.io.InputStreamReader`
- Запись данных в файл необходимо реализовать с помощью класса `java.io.PrintWriter`
- 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)
- `head` : вывести первый элемент коллекции
- `add_if_max {element}` : добавить новый элемент в коллекцию, если его значение превышает значение наибольшего элемента этой коллекции
- `remove_lower` : удалить из коллекции все элементы, меньшие, чем заданный
- `history`: вывести последние 11 команд(без их аргументов)
- `filter_contains_name name` : вывести элементы, значение поля `name` которых содержит данную подстроку
- `print_ascending` :вывести элементы коллекции в порядке возрастания
- `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:**

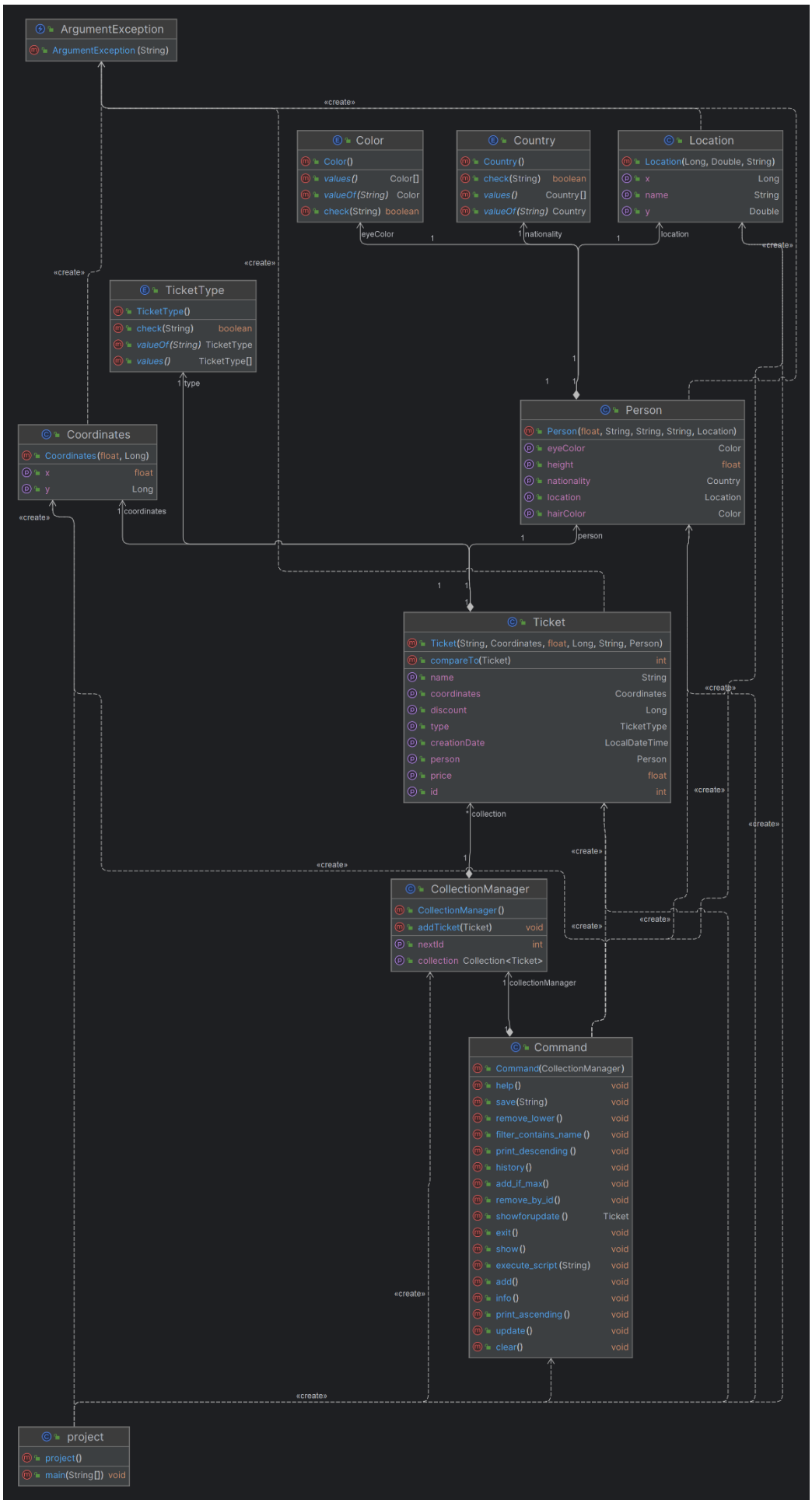
```
public class Ticket {
    private Integer id = 0;//Поле не может быть null, Значение поля должно быть больше 0, Значение
этого поля должно быть уникальным, Значение этого поля должно генерироваться автоматически
    private String name;//Поле не может быть null, Строка не может быть пустой
    private Coordinates coordinates;//Поле не может быть null
    private LocalDateTime creationDate;//Поле не может быть null, Значение этого поля должно
генерироваться автоматически
    private float price;//Значение поля должно быть больше 0
    private Long discount;//Поле может быть null, Значение поля должно быть больше 0, Максимальное
значение поля: 100
    private TicketType type;//Поле не может быть null
    private Person person;//Поле может быть null }

public class Coordinates {
    private float x;//Значение поля должно быть больше -948
    private Long y;//Поле не может быть null }

public class Person {


    private float height;//Значение поля должно быть больше 0
    private Color eyeColor;//Поле может быть null
    private Color hairColor;//Поле не может быть null
    private Country nationality;//Поле может быть null
    private Location location;//Поле не может быть null }

Location {
    private final Long x;//Поле не может быть null
    private Double y;//Поле не может быть null
    private String name;//Длина строки не должна быть больше 370, Поле может быть nul
}
public enum TicketType {
    VIP,
    USUAL,
    BUDGETARY,
    CHEAP; }

public enum Color {
    GREEN,
    WHITE,
    BROWN,
    ORANGE; }
public enum Country {
    UNITED_KINGDOM,
    USA,
    SPAIN,
    THAILAND;}
```

2. Исходный код программы.

Репозиторий:
https://github.com/VsevKokhan/ITMOProg/tree/master/lab5

3. Диаграмма классов реализованной объектной модели.

https://github.com/VsevKokhan/ITMOProg/blob/master/lab5/lab5.png


ArgumentException
ArgumentException(String)
«create»
Color
Color()
values() Color[]
valueOf(String) Color
check(String) boolean
Country
Country()
check(String) boolean
values() Country[]
valueOf(String) Country
Location
Location(Long, Double, String)
x Long
name String
y Double
eyeColor
nationality
location
TicketType
TicketType()
check(String) boolean
valueOf(String) TicketType
values() TicketType[]
type
Coordinates
Coordinates(float, Long)
x float
y Long
coordinates
Person
Person(float, String, String, String, Location)
eyeColor Color
height float
nationality Country
location Location
hairColor Color
person
Ticket
Ticket(String, Coordinates, float, Long, String, Person)
compareTo(Ticket) int
name String
coordinates Coordinates
discount Long
type TicketType
creationDate LocalDateTime
person Person
price float
id int
collection
CollectionManager
CollectionManager()
addTicket(Ticket) void
nextId int
collection Collection<Ticket>
collectionManager
Command
Command(CollectionManager)
help() void
save(String) void
remove_lower() void
filter_contains_name() void
print_descending() void
history() void
add_if_max() void
remove_by_id() void
showforupdate() Ticket
exit() void
show() void
execute_script(String) void
add() void
info() void
print_ascending() void
update() void
clear() void
project
project()
main(String[]) void

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.