**Министерство образования и науки Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ”**

**Лабораторная работа №4.**

ФИО студента: Бекмухаметов Владислав Робертович

Направление подготовки(специальность): 09.03.04

Группа: P3119

ФИО преподавателя: Письмак Алексей Евгеньевич

**Санкт-Петербург, 2021**

Оглавление

[Текст задания 3](#_Toc98162919)

[Диаграмма классов 6](#_Toc98162920)

[Исходный код программы. 7](#_Toc98162921)

[Вывод по работе 7](#_Toc98162922)

# 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.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, 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\_greater {element} : удалить из коллекции все элементы, превышающие заданный
* remove\_lower {element} : удалить из коллекции все элементы, меньшие, чем заданный
* remove\_any\_by\_type type : удалить из коллекции один элемент, значение поля type которого эквивалентно заданному
* group\_counting\_by\_price : сгруппировать элементы коллекции по значению поля price, вывести количество элементов в каждой группе
* print\_field\_descending\_comment : вывести значения поля comment всех элементов 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 long 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.LocalDateTime creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически

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

private String comment; //Длина строки не должна быть больше 356, Поле может быть null

private TicketType type; //Поле не может быть null

private Person person; //Поле не может быть null

}

public class Coordinates {

private Integer x; //Значение поля должно быть больше -960, Поле не может быть null

private Float y; //Поле не может быть null

}

public class Person {

private java.time.LocalDateTime birthday; //Поле не может быть null

private Float height; //Поле не может быть null, Значение поля должно быть больше 0

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

private String passportID; //Поле не может быть null

}

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.

**Вопросы к защите лабораторной работы:**

1. Коллекции. Сортировка элементов коллекции. Интерфейсы java.util.Comparable и java.util.Comparator.
2. Категории коллекций - списки, множества. Интерфейс java.util.Map и его реализации.
3. Параметризованные типы. Создание параметризуемых классов. Wildcard-параметры.
4. Классы-оболочки. Назначение, область применения, преимущества и недостатки. Автоупаковка и автораспаковка.
5. Потоки ввода-вывода в Java. Байтовые и символьные потоки. "Цепочки" потоков (Stream Chains).
6. Работа с файлами в Java. Класс java.io.File.
7. Пакет java.nio - назначение, основные классы и интерфейсы.
8. Утилита javadoc. Особенности автоматического документирования кода в Java.

# Изображение выглядит как текст, монитор, снимок экрана, черный Автоматически созданное описаниеДиаграмма классов

# Исходный код программы.

<https://github.com/YuikoSempai/Lab5>

# Вывод по работе

Данная работа помогла мне лучше понять, что такое коллекции, итераторы, компараторы. Немного разобрался в параметризованных типах данных. Изучил новые шаблоны программирования, один из которых использовал в данной работе. Изучил правила написания Javadoc и попробовал написать свой.