

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ «Санкт-Петербургский национальный исследовательский  
университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ  
ТЕХНИКИ

## **ЛАБОРАТОРНАЯ РАБОТА №5**

по дисциплине

«ПРОГРАММИРОВАНИЕ»

Вариант № 3116002

***Выполнил:***

Студент группы Р3116

Билошицкий Михаил Владимирович

***Преподаватель:***

Гаврилов Антон Валерьевич

Санкт-Петербург, 2023

# Содержание

Задание.....	3
Исходный код программы .....	5
UML диаграмма классов .....	6
Вывод.....	7

# Задание

Введите вариант: 3116C

## Внимание! У разных вариантов разный текст задания!

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса `Product`, 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.HashMap`
- 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.util.Scanner`
- 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, 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
- `replace_if_lower null {element}` : заменить значение по ключу, если новое значение меньше старого
- `remove_lower_key null` : удалить из коллекции все элементы, ключ которых меньше, чем заданный
- `min_by_name` : вывести любой объект из коллекции, значение поля name которого является минимальным
- `group_counting_by_price` : сгруппировать элементы коллекции по значению поля price, вывести количество элементов в каждой группе
- `print_unique_manufacture_cost` : вывести уникальные значения поля manufactureCost всех элементов в коллекции

**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'e; введена строка вместо числа; введенное 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 Product {
    private Integer id; //Поле не может быть null, Значение поля должно быть больше 0, Значение этого поля должно бы
    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 генерировать
    private Integer price; //Поле не может быть null, Значение поля должно быть больше 0
    private int manufactureCost;
    private UnitOfMeasure unitOfMeasure; //Поле может быть null
    private Person owner; //Поле не может быть null
}
public class Coordinates {
    private Double x; //Максимальное значение поля: 511, Поле не может быть null
    private double y;
}
public class Person {
    private String name; //Поле не может быть null, Строка не может быть пустой
    private int weight; //Значение поля должно быть больше 0
    private Color eyeColor; //Поле может быть null
    private Color hairColor; //Поле может быть null
    private Country nationality; //Поле может быть null
    private Location location; //Поле может быть null
}
public class Location {
    private Double x; //Поле не может быть null
    private Float y; //Поле не может быть null
    private String name; //Длина строки не должна быть больше 606, Поле может быть null
}
public enum UnitOfMeasure {
    CENTIMETERS,
    SQUARE_METERS,
    PCS,
    LITERS,
    GRAMS;
}
public enum Color {
    GREEN,
    RED,
    BLUE,
    WHITE,
    BROWN;
}
public enum Color {
    GREEN,
    YELLOW,
    ORANGE,
    WHITE,
    BROWN;
}
public enum Country {
    UNITED_KINGDOM,
    SPAIN,
    JAPAN;
}
```

#### 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. Поток 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.

# Исходный код программы

Ссылка на GitHub с исходным кодом:

[https://github.com/michael-bill/labs\\_programming\\_itmo/tree/main/Laba5](https://github.com/michael-bill/labs_programming_itmo/tree/main/Laba5)

# UML диаграмма классов



Чтобы ознакомиться с ней поближе, ссылка на .png файл также прикреплена в GitHub репозитории.

[https://github.com/michael-bill/labs\\_programming\\_itmo/blob/main/Laba5/class%20diagram.png](https://github.com/michael-bill/labs_programming_itmo/blob/main/Laba5/class%20diagram.png)

# Вывод

Во время выполнения лабораторной работы я ознакомился с основными коллекциями Java, интерфейсами `java.util.Comparable`, `java.util.Comparator` и `Map`. Научился использовать параметризованные типы данных Java. Научился работать с файлом, записывая и считывая данные в него используя байтовые и символьные потоки, а также класс `java.io.File`. Полученные знания мне пригодятся в будущем и в дальнейшем процессе обучения.