

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

НАПРАВЛЕНИЕ СИСТЕМНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО  
ОБЕСПЕЧЕНИЯ

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5**

**курса «Программирование»**

**по теме: «Принципы ООП»**

**Вариант № 236587**

Выполнил студент:

Тюрин Иван Николаевич

группа: Р3110

Преподаватель:

Письмак А. Е.,

Сорокин Р. Б.

Санкт-Петербург, 2022 г.

# Содержание

<b>Лабораторная работа № 5 Принципы ООП</b>	<b>2</b>
1. Задание варианта № 236587 . . . . .	2
2. Выполнение задания . . . . .	5
3. Исходный код программы . . . . .	5
4. Вывод . . . . .	5

# Лабораторная работа № 5

## Принципы ООП

### 1. Задание варианта № 236587

, , ,

**Описание предметной области, по которой должна быть построена объектная модель:**

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса `MusicBand`, 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.LinkedHashSet`
- 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 `json`
- 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)
- **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
- **history** : вывести последние 6 команд (без их аргументов)
- **average\_of\_number\_of\_participants** : вывести среднее значение поля `numberOfParticipants` для всех элементов коллекции
- **count\_less\_than\_albums\_count albumsCount** : вывести количество элементов, значение поля `albumsCount` которых меньше заданного
- **print\_ascending** : вывести элементы коллекции в порядке возрастания

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; 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:

```

1 public class MusicBand {
2     private Integer id; //Поле не может быть null, Значение поля должно
3                         //быть больше 0, Значение этого поля должно быть уникальным, Значение этого
4                         //поля должно генерироваться автоматически
5     private String name; //Поле не может быть null, Стока не может быть
6                         //пустой
7     private Coordinates coordinates; //Поле не может быть null
8     private java.time.LocalDate creationDate; //Поле не может быть null,
9                         //Значение этого поля должно генерироваться автоматически
10    private int numberOfParticipants; //Значение поля должно быть больше 0
11    private long albumsCount; //Значение поля должно быть больше 0
12    private java.util.Date establishmentDate; //Поле может быть null
13    private MusicGenre genre; //Поле не может быть null
14    private Label label; //Поле не может быть null
15 }
16 public class Coordinates {
17     private Double x; //Поле не может быть null
18     private Double y; //Поле не может быть null
19 }
20 public class Label {
21     private long bands;
22 }
23 public enum MusicGenre {
24     PROGRESSIVE_ROCK,
25     RAP,
26     BLUES,
27     PUNK_ROCK;
28 }
```

Листинг 1.2: Описание хранимых в коллекции классов

, , ,

## 2. Выполнение задания

В результате выполнения работы по приведенному техническому заданию была разработана объектная модель приложения, нарисована UML диаграмма классов (см. 1.2).

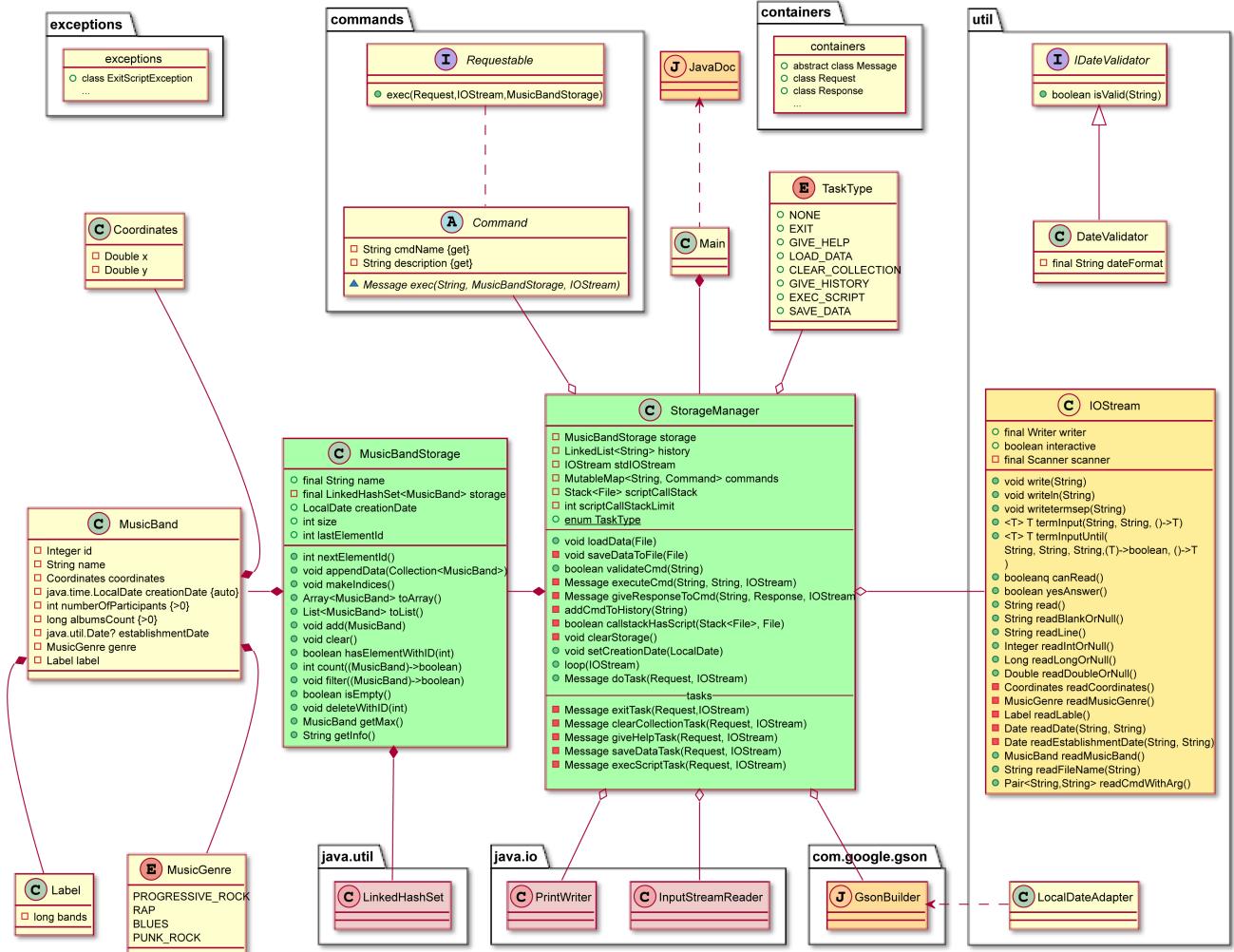


Рис. 1.2: UML диаграмма классов

## 3. Исходный код программы

Исходный код программы был размещен на удаленном сервере в личном репозитории. Код можно найти по ссылке: <https://github.com/e1turin/itmo-programming/lab-5-kotlin>.

## 4. Вывод

Укрепил знание принципов программирования SOLID и инъектирования зависимостей. Укрепил знания по работе с ЯП Java, изучил ЯП Kotlin, получил

больше знаний о парадигме ООП и ФП представленной в ЯП Kotlin. Получил еще один опыт выполнения технического задания. Научился основам конфигурирования сборщика проектов Gradle. Научился сдерживать ярость, несколько раз упсиховался, научился быстро писать отчет к работе, узнал на каких сайтах хороший код и откуда можно безопасно копипастить. Понял отличия между библиотеками по работе с JSON: Jackson и Gson (второй круче). Не понял зачем перед считыванием целой строки в жаве нужно предварительно считать пустую строку. Возможно научился основам упаковки проета в контейнеры Docker (если успею сделать до сдачи).