Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет Программной инженерии и компьютерной техники*

**Лабораторная работа**

Вариант №173388

**Студент**: Фролов Кирилл Дмитриевич

**Группа:** P3106

**Преподаватель**: Байрамова Хумай

г. Санкт-Петербург

2023г

# Оглавление

[Оглавление 2](#_Toc137480273)

[Задание 3](#_Toc137480274)

[Диаграмма классов 6](#_Toc137480275)

[Программа 6](#_Toc137480276)

[Выводы 7](#_Toc137480277)

[Список литературы 8](#_Toc137480278)

# Задание

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса City, 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.Hashtable
* 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 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
* 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\_greater {element} : удалить из коллекции все элементы, превышающие заданный
* history : вывести последние 14 команд (без их аргументов)
* replace\_if\_greater null {element} : заменить значение по ключу, если новое значение больше старого
* filter\_contains\_name name : вывести элементы, значение поля name которых содержит заданную подстроку
* filter\_starts\_with\_name name : вывести элементы, значение поля name которых начинается с заданной подстроки
* print\_unique\_meters\_above\_sea\_level : вывести уникальные значения поля metersAboveSeaLevel 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'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 City {

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 float area; //Значение поля должно быть больше 0

private Long population; //Значение поля должно быть больше 0, Поле не может быть null

private int metersAboveSeaLevel;

private Climate climate; //Поле не может быть null

private Government government; //Поле может быть null

private StandardOfLiving standardOfLiving; //Поле может быть null

private Human governor; //Поле может быть null

}

public class Coordinates {

private Integer x; //Поле не может быть null

private Integer y; //Значение поля должно быть больше -805, Поле не может быть null

}

public class Human {

private long height; //Значение поля должно быть больше 0

}

public enum Climate {

RAIN\_FOREST,

HUMIDSUBTROPICAL,

MEDITERRANIAN;

}

public enum Government {

DICTATORSHIP,

PUPPET\_STATE,

OLIGARCHY;

}

public enum StandardOfLiving {

ULTRA\_HIGH,

VERY\_HIGH,

HIGH,

NIGHTMARE;

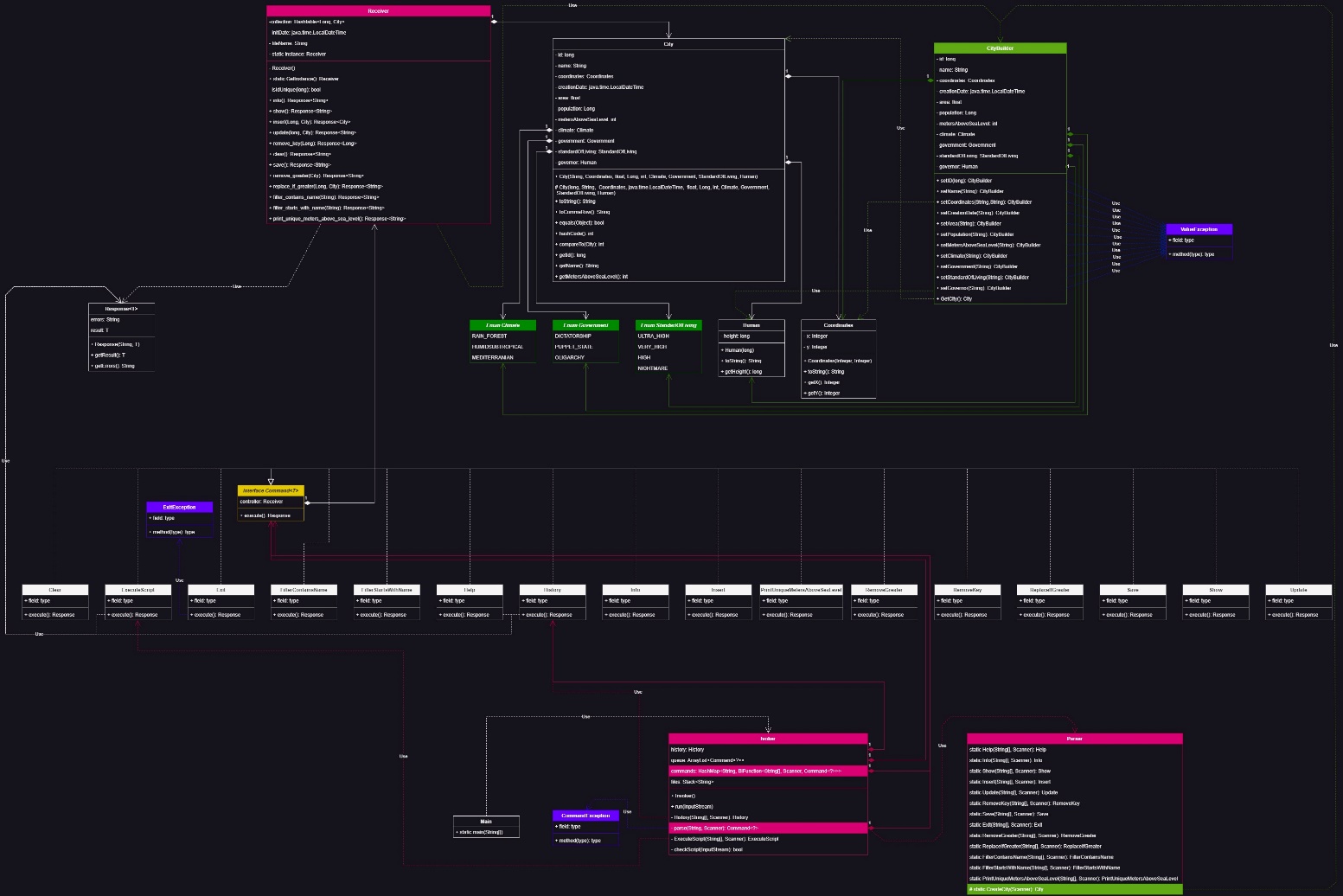
}

**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.

Диаграмма классов

*Рисунок 1*

(Диаграмма в более высоком разрешении лежит на GitHub вместе с программой)

Программа

<https://github.com/DiedCloud/ITMO/tree/main/Programming/Lab5/Project>

# Выводы

Разработано приложение для манипуляции одним из базовых видов представления данных – словарем. Программа расширяема – благодаря паттерну команда с легкостью можно добавить больше команд для более детального взаимодействия с хранимыми объектами. Взаимодействие с пользователем требует проверки правильности ввода – в программе для этого реализован отдельный класс – CityBuilder, и не только.

# Список литературы

1. Academy, V. (15 08 2017 г.). *Java 8 Ссылки на методы*. Получено из vertex: <https://vertex-academy.com/tutorials/ru/java-8-ssylki-na-metody/>
2. alexVengrovsk. (04 09 2014 г.). *Основные отличия Java IO и Java NIO*. Получено из Хабр: <https://habr.com/ru/articles/235585/>
3. Oracle. (б.д.). *Class BufferedWriter*. Получено из Документация Java 10: <https://docs.oracle.com/javase/10/docs/api/java/io/BufferedWriter.html>
4. Oracle. (б.д.). *Class Hashtable<K,V>*. Получено из Документация Java 8: <https://docs.oracle.com/javase/8/docs/api/java/util/Hashtable.html>
5. Oracle. (б.д.). *Class Scanner*. Получено из Документация Java 7: <https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html>
6. Oracle. (б.д.). *Package java.util.function*. Получено из Документация Java 8: <https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html>
7. *Буферизация символьных потоков. BufferedReader и BufferedWriter*. (26 04 2018 г.). Получено из METANIT.COM: <https://metanit.com/java/tutorial/6.9.php>
8. *Документирование javadoc*. (б.д.). Получено из Java-online.ru: <https://java-online.ru/java-javadoc.xhtml>
9. *Интерфейс Map и класс HashMap*. (25 04 2018 г.). Получено из METANIT.COM: <https://metanit.com/java/tutorial/5.8.php>
10. *Интерфейсы Comparable и Comparator. Сортировка*. (24 04 2018 г.). Получено из METANIT.COM: <https://metanit.com/java/tutorial/5.6.php>
11. Керри, Э. (03 10 2022 г.). *В чем разница между HashMap и Hashtable*. Получено из JAVARUSH: <https://javarush.com/groups/posts/3965-kofe-breyk-173-v-chem-raznica-mezhdu-hashmap-i-hashtable-kak-nayti-i-ispravitjh-iskljuchenie-nu>
12. *Команда*. (б.д.). Получено из Refactoring guru: <https://refactoring.guru/ru/design-patterns/command>
13. *Одиночка*. (б.д.). Получено из Refactoring guru: <https://refactoring.guru/ru/design-patterns/singleton>
14. *Потоки ввода-вывода. Работа с файлами*. (25 04 2018 г.). Получено из METANIT.COM: <https://metanit.com/java/tutorial/6.1.php>
15. Сбер. (19 07 2018 г.). *Пришел, увидел, обобщил: погружаемся в Java Generics*. Получено из Хабр: <https://habr.com/ru/companies/sberbank/articles/416413/>
16. *Цепочки потоков*. (б.д.). Получено из JAVARUSH: <https://javarush.com/quests/lectures/questsyntaxpro.level15.lecture02>