**ФЕДЕРАЛЬОНЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗАВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**“НАЦИОНАЛЬНЫЙ ИСЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”**

Факультет Программной Инженерии и Компьютерной Техники

Дисциплина: «Программирование»

**ОТЧЁТ**

по лабораторной работе №5  
Вариант №407922

Выполнил:

Студент группы P3111

{name}

Проверил:

{name}

Санкт-Петербург

2024 г.

**Задание**

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса Vehicle, 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.LinkedHashMap
* 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, 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 5 команд (без их аргументов)

remove\_greater\_key null : удалить из коллекции все элементы, ключ которых превышает заданный

sum\_of\_engine\_power : вывести сумму значений поля enginePower для всех элементов коллекции

filter\_starts\_with\_name name : вывести элементы, значение поля name которых начинается с заданной подстроки

print\_unique\_type : вывести уникальные значения поля type 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 Vehicle {

private int 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.ZonedDateTime creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически

private int enginePower; //Значение поля должно быть больше 0

private float capacity; //Значение поля должно быть больше 0

private VehicleType type; //Поле не может быть null

private FuelType fuelType; //Поле может быть null

}

public class Coordinates {

private long x;

private Integer y; //Поле не может быть null

}

public enum VehicleType {

CAR,

BOAT,

SHIP,

MOTORCYCLE;

}

public enum FuelType {

ELECTRICITY,

DIESEL,

MANPOWER,

NUCLEAR,

ANTIMATTER;

}

**Код Программы**

**Диаграмма**

****

**Основные части кода**

Main.java

*/\*\*  
 \* Main class to start the program  
 \*  
 \* @see ConsoleRunner  
 \* @see Handler  
 \* @see ConsoleWorker  
 \*/*public class Main {  
 public static Path *ROOT\_FILE*;  
  
 */\*\*  
 \* main method to start the program  
 \* @param args - arguments from command line  
 \*/* public static void main(String[] args) {  
 *// validate arguments* if (args.length == 0) {  
 System.*out*.println("для использования программы необходимо ввести файл, с которым будет производиться работа.");  
 return;  
 }  
  
 *// set root file from arguments  
 ROOT\_FILE* = Paths.*get*(args[0]);  
  
 *// if file doesn't exist, create it* if (Files.*notExists*(*ROOT\_FILE*)) {  
 try {  
 Files.*createFile*(*ROOT\_FILE*);  
 System.*out*.printf("Файл %s успешно был создан%n", *ROOT\_FILE*.getFileName());  
 } catch (IOException e) {  
 System.*out*.printf("невозможно создать файл! %s%n", e.getMessage());  
 return;  
 }  
 }  
  
 *// create & start console runner* ConsoleRunner consoleRunner = *getConsoleRunner*();  
  
 new Thread(consoleRunner).start();  
 }  
  
 */\*\*  
 \* method to get {@link ConsoleRunner} with all necessary handlers  
 \* @return {@link ConsoleRunner}  
 \*/* private static ConsoleRunner getConsoleRunner() {  
 Handler handler = new Handler(List.*of*(  
 new Info(),  
 new Insert(),  
 new Show(),  
 new Update(),  
 new RemoveKey(),  
 new Clear(),  
 new Save(),  
 new ExecuteScript(),  
 new Exit(),  
 new RemoveGreater(),  
 new History(),  
 new RemoveGreaterKey(),  
 new SumOfEnginePower(),  
 new FilterStartsWithName(),  
 new PrintUniqueType()  
 ));  
  
 ConsoleWorker bufferedConsoleWorker = new BaseConsoleWorker();  
  
 return new ConsoleRunner(handler, bufferedConsoleWorker);  
 }  
}

Handler.java

*/\*\*  
 \* Class of handler  
  
 \* {@link Handler#commands} - list of commands  
 \* {@link Handler#requests} - deque of requests  
 \*/*public class Handler {  
 private final List<Command> commands;  
  
 public static volatile Deque<Request> *requests* = new ArrayDeque<>();  
  
 public Handler(List<Command> commands) {  
 this.commands = commands;  
 }  
  
 */\*\*  
 \* Handle request and return response  
 \*  
 \* @param request - request to handle  
 \* @return response to request or error message  
 \*/* public Response handle(Request request) {  
 *requests*.push(request);  
  
 *// validate query* if (request == null)  
 return new Response("ошибка! запрос не может быть пустым.");  
  
 if (request.getCommand() == null || request.getCommand().isBlank())  
 return new Response("ошибка! команда не может быть пустой.");  
  
 *// check if user wants help* if (request.getCommand().equals("help"))  
 return getHelp();  
  
 *// find suitable command* List<Command> suitableCommands = commands.stream()  
 .filter(command -> command.getName().equals(request.getCommand()))  
 .toList();  
  
 *// validate suitable commands* if (suitableCommands.isEmpty())  
 return new Response("ошибка! неизвестная команда: " + request.getCommand() + "\nвведите 'help' для справки.");  
 if (suitableCommands.size() > 1)  
 System.*out*.println("WARN! найдено несколько подходящих команд: " + suitableCommands);  
  
 *// execute command* return suitableCommands.get(0).execute(request);  
 }  
  
 */\*\*  
 \* Get help message for all commands  
 \* @return help message  
 \*/* public Response getHelp() {  
 final int maxCommandNameLength = commands.stream()  
 .map(Command::getName)  
 .mapToInt(String::length).max().orElse(0);  
  
 return new Response(commands.stream()  
 .map(command -> String.*format*("%n \* %" + maxCommandNameLength + "s - %s", command.getName(), command.getHelp()))  
 .collect(Collectors.*joining*()));  
 }  
}

ConsoleRunner.java

public class ConsoleRunner implements Runnable {  
 public static final int *MAX\_RECURSION\_DEPTH* = 10;  
  
 private final Handler handler;  
 private final ConsoleWorker consoleWorker;  
  
 private int currentRecursionDepth = 0;  
 private final Deque<String> inboundQueries = new ArrayDeque<>();  
  
 public ConsoleRunner(Handler handler, ConsoleWorker consoleWorker) {  
 this.handler = handler;  
 this.consoleWorker = consoleWorker;  
 }  
  
 @Override  
 public void run() {  
 *// try load collection* CollectionManager.*getInstance*().getCollection();  
  
 String input;  
  
 while ((input = consoleWorker.get("[%tl:%tM:%tS] ~ ")) != null) {  
 currentRecursionDepth = 0;  
 process(input);  
 }  
  
 consoleWorker.send("goodbye!");  
 }  
  
 private void process(String input) {  
 if (currentRecursionDepth > *MAX\_RECURSION\_DEPTH*) {  
 consoleWorker.send("Глубина рекурсии не может быть > " + *MAX\_RECURSION\_DEPTH*);  
 inboundQueries.clear();  
 return;  
 }  
  
 Request request = new Request();  
  
 while (input.contains("element")) {  
 input = input.replaceFirst("element", "");  
 Vehicle inputVehicle = inputVehicle();  
  
 System.*out*.println(CollectionManager.*getInstance*().getCollection().keySet()  
 .stream()  
 .max(Integer::*compare*).orElse(0) + 1);  
  
 request.getCollection().put(CollectionManager.*getInstance*().getCollection().keySet()  
 .stream()  
 .max(Integer::*compare*).orElse(0) + 1, inputVehicle);  
 }  
  
 String[] parts = input.trim().split(" ", 2);  
  
 request.setCommand(parts[0].trim());  
  
 if (parts.length == 2) request.setText(parts[1].trim());  
  
 Response response = handler.handle(request);  
  
 if (!inboundQueries.isEmpty()) currentRecursionDepth++;  
 inboundQueries.addAll(response.getInboundRequests());  
 while (!inboundQueries.isEmpty()) process(inboundQueries.pollLast());  
  
 if (response.getMessage() != null) consoleWorker.send(response.getMessage());  
  
 if (response.getVehicles() != null)  
 response.getVehicles().stream().map(Vehicle::toString).forEach(consoleWorker::send);  
  
 consoleWorker.skip();  
 }  
  
 private Vehicle inputVehicle() {  
 consoleWorker.skip();  
 consoleWorker.send("Новая транспортное средство:");  
 consoleWorker.skip();  
 consoleWorker.send("> Введите основную информацию: ");  
  
 Vehicle vehicle = new Vehicle();  
 while (!input("название", vehicle::setName, str -> str)) ;  
 while (!input("мощность", vehicle::setEnginePower, Integer::parseInt)) ;  
 while (!input("вместимость", vehicle::setCapacity, Float::parseFloat)) ;  
 while (!input("тип транспорта " + Arrays.toString(VehicleType.values()), vehicle::setType, VehicleType::valueOf))  
 ;  
 while (!input("тип топлива " + Arrays.toString(FuelType.values()), vehicle::setFuelType, FuelType::valueOf)) ;  
  
 consoleWorker.skip();  
 consoleWorker.send("> Введите информацию о координатах: ");  
  
 Coordinates coordinates = new Coordinates();  
  
 while (!input("x", coordinates::setX, Long::parseLong)) ;  
 while (!input("y", coordinates::setY, Integer::parseInt)) ;  
  
 vehicle.setCoordinates(coordinates);  
  
 return vehicle;  
 }  
  
 private <K> boolean input(String fieldName, Consumer<K> setter, Function<String, K> parser) {  
 try {  
 setter.accept(parser.apply(consoleWorker.get(" - " + fieldName)));  
 return true;  
 } catch (Exception ex) {  
 consoleWorker.send(ex.getMessage());  
 return false;  
 }  
 }  
  
}

**Результат работы**

[6:29:00] ~ : info

Коллекция:

Тип: se.ifmo.lab5.core.collection.UserCollection

Дата инициализации: Mon Mar 04 06:29:00 MSK 2024

Количество элементов: 0

[6:29:04] ~ : insert 2 element

Новая транспортное средство:

> Введите основную информацию:

- название: 911

- мощность: 222

- вместимость: 4

- тип транспорта [CAR, BOAT, SHIP, MOTORCYCLE]: CAR

- тип топлива [ELECTRICITY, DIESEL, MANPOWER, NUCLEAR, ANTIMATTER]: DIESEL

> Введите информацию о координатах:

- x: 23

- y: 43

1

элемент с id 2 успешно добавлен!

[6:29:34] ~ : show

элементы коллекции

# Транспортное средство '911' ID{2}:

> координаты: x: 23, y: 43

> дата создания: 2024-03-04T06:29:08.125991505+03:00[Europe/Moscow]

> мощность: 222

> объем: 4.0

> тип: CAR

> тип топлива: DIESEL

[6:29:58] ~ : help

\* info - получить информацию о коллекции (тип, дата инициализации, количество элементов)

\* insert - insert {id} {element} - добавить новый элемент с заданным ключом

\* show - вывести все элементы в коллекции (в строковом представлении)

\* update - update id {element} - обновить значение элемента коллекции по id

\* remove\_key - remove\_key {id} - удалить элемент из коллекции по его ключу

\* clear - очистить коллекцию

\* save - сохранить коллекцию в файл

\* execute\_script - execute\_script file\_name - считать и исполнить скрипт из указанного файла.

\* exit - завершить программу (без сохранения в файл)

\* remove\_greater - remove\_greater {element} - удалить из коллекции все элементы, превышающие заданный

\* history - вывести последние 5 команд (без их аргументов)

\* remove\_greater\_key - remove\_greater\_key {id} - удалить из коллекции все элементы, ключ которых превышает заданный

\* sum\_of\_engine\_power - вывести сумму значений поля enginePower для всех элементов коллекции

\* filter\_starts\_with\_name - filter\_starts\_with\_name name - вывести элементы, значение поля name которых начинается с заданной подстроки

\* print\_unique\_type - вывести уникальные значения поля type всех элементов в коллекции

**Вывод**

В результате работы я создал программу, которая помогла мне разобраться в работе с форматом .json посредством JAVA, так же я познакомился со Stream API и разными типами коллекций в JAVA. Приобрел неоценимый опыт в создании консольных приложений и смог построить легко масштабируемое приложение.