Факультет программной инженерии и компьютерной техники

Программирование

Лабораторная работа №5

Вариант 311210

Преподаватель: Николаев Владимир Вячеславович

Выполнил: Кульбако Артемий Юрьевич

Р3112

Санкт-Петербург

2019

# Текст задания.

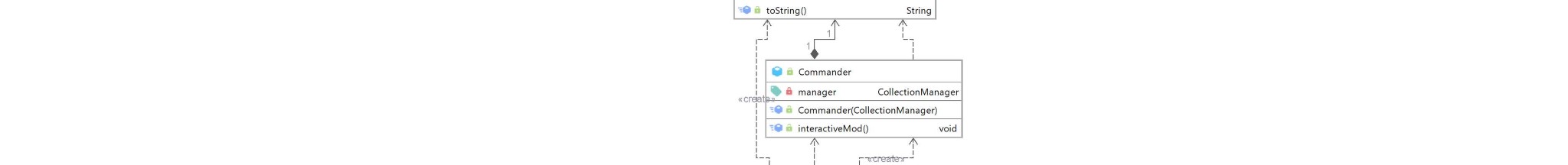
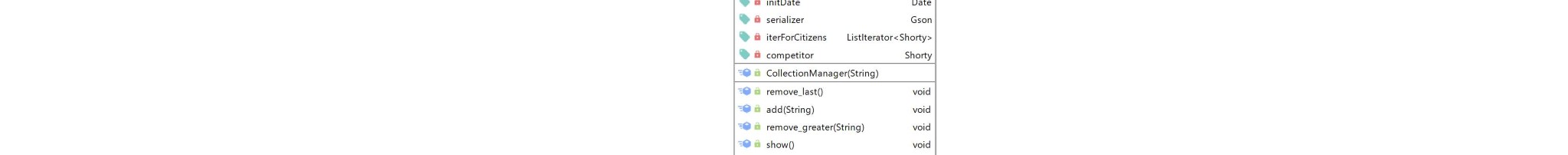
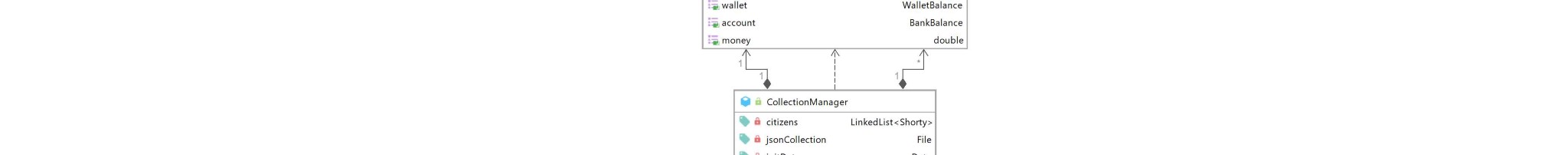
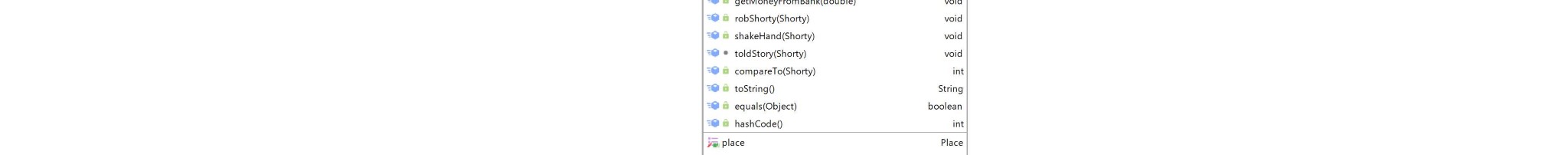
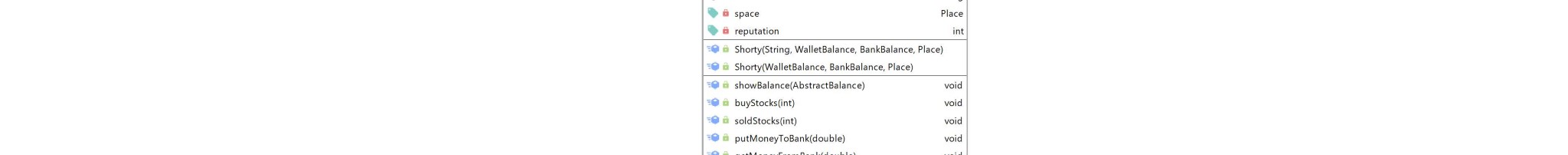
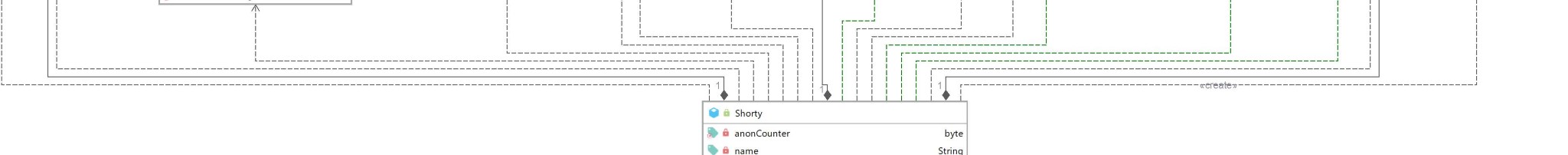
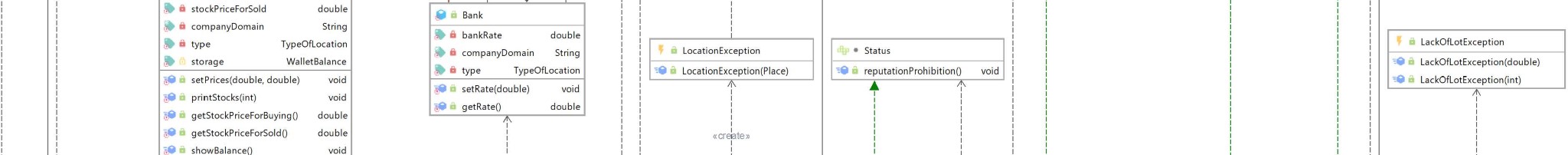
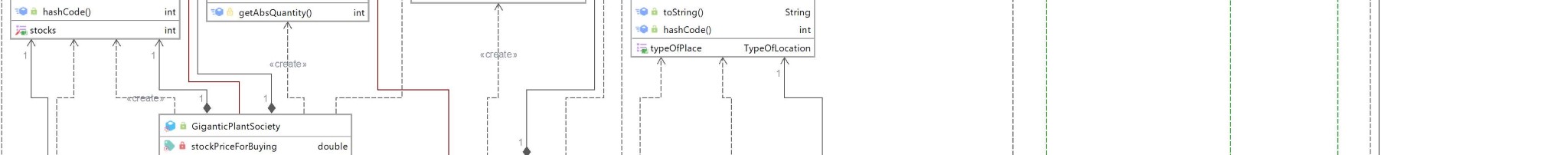
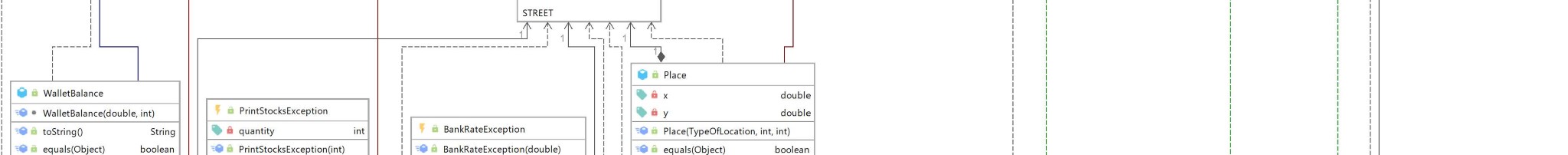
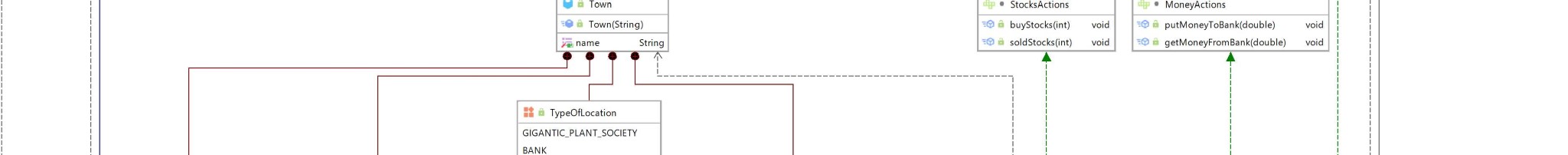
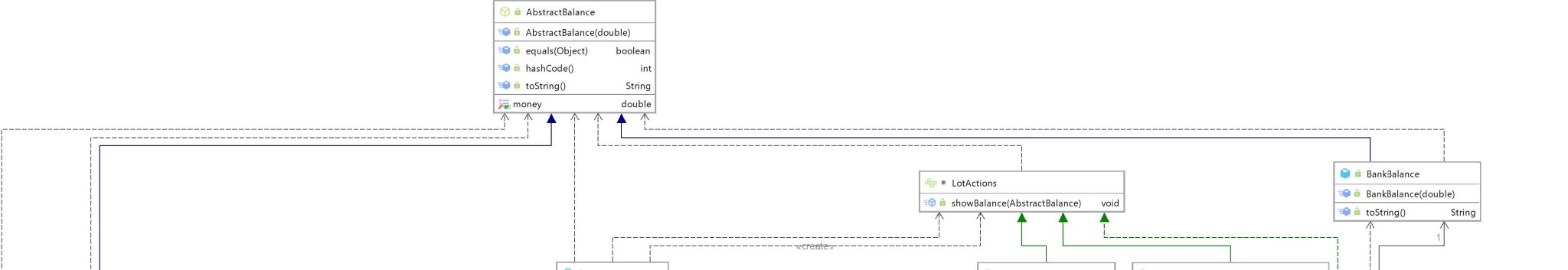
**Разработанная программа должна удовлетворять следующим требованиям:**

* Класс, коллекцией экземпляров которого управляет программа, должен реализовывать сортировку по умолчанию.
* Для хранения необходимо использовать коллекцию типа java.util.LinkedList.
* При запуске приложения коллекция должна автоматически заполняться значениями из файла.
* Имя файла должно передаваться программе с помощью переменной окружения.
* Данные должны храниться в файле в формате json.
* При остановке приложения текущее состояние коллекции должно автоматически сохраняться в файл.
* Чтение данных из файла необходимо реализовать с помощью класса java.io.InputStreamReader.
* Запись данных в файл необходимо реализовать с помощью класса java.io.OutputStreamWriter.
* Все реализованные команды (см. ниже) должны быть задокументированы в формате javadoc.
* Формат задания объектов в командах - json.

**В интерактивном режиме программа должна поддерживать выполнение следующих команд:**

* remove\_last: удалить последний элемент из коллекции
* add {element}: добавить новый элемент в коллекцию
* remove\_greater {element}: удалить из коллекции все элементы, превышающие заданный
* show: вывести в стандартный поток вывода все элементы коллекции в строковом представлении
* clear: очистить коллекцию
* info: вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.)
* remove\_all {element}: удалить из коллекции все элементы, эквивалентные заданному
* load: перечитать коллекцию из файла
* remove {element}: удалить элемент из коллекции по его значению
* add\_if\_min {element}: добавить новый элемент в коллекцию, если его значение меньше, чем у наименьшего элемента этой коллекции

# Диаграмма классов разработанной программы.



# Исходный код программы\*.

\*Приведены только классы, необходимые для работы с коллекцией.

### Main.java – отвечает за старт программы.

*package LabFive;*

*import java.io.\*;*

*/\*\**

*\* @author Kulbako Artemy 265570*

*\* @version 3.0*

*\*/*

*public class Main {*

*public static void main(String[] args) throws IOException {*

*Commander commander = new Commander(new CollectionManager(System.getenv("Collman\_Path")));*

*commander.interactiveMod();*

*}*

*}*

### Shorty.java – представляет объекты, которыми манипулирует программа (приведён только необходимый код).

package LabFive;

import java.util.Objects;

/\*\*

\* Объекты класса {@code Shortly} имитируют коротышку - жителя цветочного города.

\*/

public class Shorty implements StocksActions, MoneyActions, Comparable<Shorty> {

private static byte anonCounter;

private String name;

private WalletBalance cash;

private BankBalance account;

transient private Town.Place space;

private int reputation;

{reputation = 0;}

static {anonCounter = 1;}

public Shorty(String name, WalletBalance cash, BankBalance account, Town.Place space) {

this.name = name;

this.cash = cash;

this.account = account;

this.space = space;

}

public Shorty(WalletBalance cash, BankBalance account, Town.Place space) {

this.name = "noname#" + anonCounter;

this.cash = cash;

this.account = account;

this.space = space;

anonCounter++;

}

public Town.Place getPlace() {

return space;

}

public WalletBalance getWallet(){

return cash;

}

public BankBalance getAccount(){

return account;

}

@Override

public int compareTo(Shorty p){

return (int) ((cash.getMoney() + account.getMoney() + cash.getStocks() \* Town.GiganticPlantSociety.getStockPriceForSold())

- (p.cash.getMoney() + p.account.getMoney() + p.cash.getStocks() \* Town.GiganticPlantSociety.getStockPriceForSold()));

}

public String getName() {

return name;

}

@Override

public String toString() {

return "Коротышка{" +

"имя = " + name +

", деньги в кошельке = " + cash +

", банковский счёт = " + account +

", space=" + space +

", репутация = " + reputation +

'}';

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Shorty)) return false;

Shorty shorty = (Shorty) o;

return reputation == shorty.reputation &&

Objects.equals(name, shorty.name) &&

Objects.equals(cash, shorty.cash) &&

Objects.equals(account, shorty.account) &&

Objects.equals(space, shorty.space);

}

@Override

public int hashCode() {

return Objects.hash(name, cash, account, space, reputation);

}

}

Commander.java – обработка пользовательских команд и интерактивный режим.

package LabFive;

import java.io.\*;

import java.util.\*;

public class Commander {

private CollectionManager manager;

private String userCommand;

private String[] finalUserCommand;

{

userCommand = "";

}

public Commander(CollectionManager manager) {

this.manager = manager;

}

public void interactiveMod() throws IOException {

try(Scanner commandReader = new Scanner(System.in)) {

while (!userCommand.equals("exit")) {

userCommand = commandReader.nextLine();

finalUserCommand = userCommand.trim().split(" ", 2);

try {

switch (finalUserCommand[0]) {

case "": break;

case "remove\_last":

manager.remove\_last();

break;

case "add":

manager.add(finalUserCommand[1]);

break;

case "remove\_greater":

manager.remove\_greater(finalUserCommand[1]);

break;

case "show":

manager.show();

break;

case "clear":

manager.clear();

break;

case "info":

System.out.println(manager.toString());

break;

case "remove\_all":

manager.remove\_all(finalUserCommand[1]);

break;

case "load":

manager.load();

break;

case "remove":

manager.remove(finalUserCommand[1]);

break;

case "add\_if\_min":

manager.add\_if\_min(finalUserCommand[1]);

break;

case "help":

manager.help();

break;

case "exit":

manager.save();

break;

case "sort":

manager.sort(finalUserCommand[1]);

break;

case "man":

manager.man(finalUserCommand[1]);

break;

default:

System.out.println("Неопознанная команда. Наберите 'help' для справки.");

}

} catch (ArrayIndexOutOfBoundsException ex) {

System.out.println("Отсутствует аргумент.");

}

}

}

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Commander)) return false;

Commander commander = (Commander) o;

return Objects.equals(manager, commander.manager);

}

@Override

public int hashCode() {

int result = Objects.hash(manager, userCommand);

result = 31 \* result + Arrays.hashCode(finalUserCommand);

return result;

}

}

### CollectionManager.java – обеспечивает доступ к коллекции.

package LabFive;

import java.io.\*;

import java.util.\*;

import java.lang.reflect.\*;

import com.google.gson.\*;

import com.google.gson.reflect.\*;

public class CollectionManager {

private LinkedList<Shorty> citizens;

private File jsonCollection;

private Date initDate;

private Gson serializer;

private boolean wasStart;

protected static HashMap<String, String> manual;

{

wasStart = false;

serializer = new Gson();

citizens = new LinkedList<>();

manual = new HashMap<>();

manual.put("remove\_last", "Удалить последний элемент из коллекции");

manual.put("add", "Добавить новый элемент в коллекцию");

manual.put("remove\_greater", "Удалить из коллекции все элементы, превышающие заданный");

manual.put("show", "Вывести в стандартный поток вывода все элементы коллекции в строковом представлении");

manual.put("clear", "Очистить коллекцию");

manual.put("info", "Вывести в стандартный поток вывода информацию о коллекции.");

manual.put("remove\_all", "Удалить из коллекции все элементы, эквивалентные заданному");

manual.put("load", "Перечитать коллекцию из файла");

manual.put("remove", "Удалить элемент из коллекции по его значению");

manual.put("add\_if\_min", "Добавить новый элемент в коллекцию, если его значение меньше, чем у наименьшего элемента этой коллекции");

manual.put("exit", "Сохранить коллекцию в файл и завершить работу программы");

manual.put("sort", "Отсортировать элементы коллекции по указанному параметру. Аргументы: -n по имени, -m по деньгам, -h по хэш-коду");

}

public CollectionManager(String collectionPath) throws IOException {

try {

if (collectionPath == null) throw new FileNotFoundException();

} catch (FileNotFoundException ex) {

System.out.println("Путь до файла json нужно передать через переменную окружения Collman\_Path.");

System.exit(1);

}

File file = new File(collectionPath);

try {

if (file.exists()) this.jsonCollection = new File(collectionPath);

else throw new FileNotFoundException();

} catch (FileNotFoundException ex) {

System.out.println("Файл по указанному пути не существует.");

System.exit(1);

}

this.load();

this.initDate = new Date();

wasStart = true;

}

/\*\*

\* Удаляет последний элемент коллекции.

\*/

public void remove\_last() {

try {

citizens.removeLast();

System.out.println("Последний элемент коллекции удалён.");

save();

}

catch (NoSuchElementException ex) {

System.out.println("Нельзя удалить последний элемент коллекциию. Коллекция пуста.");

}

}

/\*\*

\* Добавляет в коллекцию элемент, согласно синтаксису.

\* @param person : Строка, которую необходимо десериализовать в объект класса Shorty, который будет добавлен в коллекцию.

\*/

public void add(String person) {

try {

if (citizens.add(serializer.fromJson(person, Shorty.class))) {

System.out.println("Элемент успешно добавлен.");

save();

}

} catch (JsonSyntaxException ex) {

System.out.println("Ошибка синтаксиса Json. Не удалось добавить элемент.");

}

}

/\*\*

\* Удаляет из коллекции все элементы, превыщающие элемент-параметр.

\* @param person : Строка, которую необходимо десериализовать в объект класса Shorty, с которым будут сравниваться элементы коллекции.

\*/

public void remove\_greater(String person) {

if (citizens.size() != 0) {

int beginSize = citizens.size();

try {

citizens.removeIf(p -> (p != null && p.compareTo(serializer.fromJson(person, Shorty.class)) > 0));

System.out.println("Из коллекции удалено " + (beginSize - citizens.size()) + " элементов.");

save();

} catch (JsonSyntaxException ex) {

System.out.println("Ошибка синтаксиса Json. Не удалось удалить элементы.");

}

}

else System.out.println("Элемент не с чем сравнивать. Коллекция пуста.");

}

/\*\*

\* Выводит все элементы коллекции.

\*/

public void show() {

if (citizens.size() != 0) citizens.forEach(p -> System.out.println(serializer.toJson(p) + " hashCode=" + p.hashCode()));

else System.out.println("Коллекция пуста.");

}

/\*\*

\* Удаляет все элементы коллекции.

\*/

public void clear() {

citizens.clear();

System.out.print("Коллекция очищена.");

save();

}

/\*\*

\* Удаляет все элементы из коллекции равные параметру.

\* @param person : Строка, которую необходимо десериализовать в объект класса Shorty, с которым будут сравниваться элементы коллекции.

\*/

public void remove\_all(String person) {

if (citizens.size() != 0) {

int beginSize = citizens.size();

try {

citizens.removeIf(p -> p.equals(serializer.fromJson(person, Shorty.class)));

System.out.println("Из коллекции удалено " + (beginSize - citizens.size()) + " элементов.");

save();

} catch (JsonSyntaxException ex) {

System.out.println("Ошибка синтаксиса Json. Не удалось удалить элемент.");

}

}

else System.out.println("Элемент не с чем сравнивать. Коллекция пуста.");

}

/\*\*

\* Десериализует коллекцию из файла json.

\* @throws IOException если файл пуст или защищён.

\*/

public void load() throws IOException {

int beginSize = citizens.size();

try {

if (!jsonCollection.exists()) throw new FileNotFoundException();

} catch (FileNotFoundException ex) {

System.out.println("Файла по указанному пути не существует.");

if (!wasStart) System.exit(1);

else return;

}

try {

if (!jsonCollection.canRead() || !jsonCollection.canWrite()) throw new SecurityException();

} catch (SecurityException ex) {

System.out.println("Файл защищён от чтения и/или записи. Для работы программы нужны оба разрешения.");

if (!wasStart) System.exit(1);

else return;

}

try {

if (jsonCollection.length() == 0) throw new JsonSyntaxException("");

} catch (JsonSyntaxException ex) {

System.out.println("Файл пуст.");

if (!wasStart) System.exit(1);

else return;

}

try (BufferedReader inputStreamReader = new BufferedReader(new InputStreamReader(new FileInputStream(jsonCollection)))) {

System.out.println("Идёт загрузка коллекции " + jsonCollection.getAbsolutePath());

String nextLine;

StringBuilder result = new StringBuilder();

while ((nextLine = inputStreamReader.readLine()) != null) {

result.append(nextLine);

}

Type collectionType = new TypeToken<LinkedList<Shorty>>() {

}.getType();

try {

LinkedList<Shorty> addedShorty = serializer.fromJson(result.toString(), collectionType);

for (Shorty s: addedShorty) {

if (!citizens.contains(s)) citizens.add(s);

}

} catch (JsonSyntaxException ex) {

System.out.println("Ошибка синтаксиса Json. Коллекция не может быть загружена.");

System.exit(1);

}

System.out.println("Коллекций успешно загружена. Добавлено " + (citizens.size() - beginSize) + " элементов.");

}

}

/\*\*

\* Удаляет из коллекции элемент по значения, согласно синтаксису.

\* @param person : Строка, которую необходимо десериализовать в объект класса Shorty, который будет удалён из коллекции.

\*/

public void remove(String person) {

if (citizens.size() != 0) {

try {

if (citizens.remove(serializer.fromJson(person, Shorty.class))) {

System.out.println("Элемент успешно удалён.");

save();

} else System.out.println("Такого элемента нет в коллекции.");

} catch (JsonSyntaxException ex) {

System.out.println("Ошибка синтаксиса Json. Не удалось удалить элемент.");

}

}

else System.out.println("Элемент не с чем сравнивать. Коллекция пуста.");

}

/\*\*

\* Добавляет в коллекцию элемент переданный в качестве параметра, если он меньше минимального элемента коллекции.

\* @param person : Строка, которую необходимо десериализовать в объект класса Shorty, который будет добавлен, если удовлетворяет условию.

\*/

public void add\_if\_min(String person) {

if (citizens.size() != 0) {

Shorty mainCompetitor = serializer.fromJson(person, Shorty.class);

Shorty competitor = Collections.min(citizens);

if (competitor.compareTo(mainCompetitor) > 0)

if (citizens.add(mainCompetitor)) {

System.out.println("Элемент успешно добавлен.");

save();

} else System.out.println("Не удалось добавить элемент.");

}

else System.out.println("Элемент не с чем сравнивать. Коллекция пуста.");

}

/\*\*

\* Сортирует коллекция согласно указанному ключу.

\* @param arg -n по имени, -m по деньгам, -h по хэш-коду.

\*/

public void sort(String arg) {

switch (arg) {

case "-n":

citizens.sort((o1, o2) -> o1.getName().compareTo(o2.getName()));

save();

System.out.println("Коллекция отсортирована по именам (по возрастания).");

break;

case "-h":

citizens.sort((o1, o2) -> o1.hashCode() - o2.hashCode());

save();

System.out.println("Коллекция отсортирована по хэш-коду (по возрастания).");

break;

case "-m":

citizens.sort((o1, o2) -> (int)

((o1.getWallet().getMoney() + o1.getAccount().getMoney() + o1.getWallet().getStocks() \* Town.GiganticPlantSociety.getStockPriceForSold())

- (o2.getWallet().getMoney() + o2.getAccount().getMoney() + o2.getWallet().getStocks() \* Town.GiganticPlantSociety.getStockPriceForSold())));

save();

System.out.println("Коллекция отсортирована по суммарному капиталу (по возрастания).");

break;

default: System.out.println("Неверный аргумент. Синтаксис 'sort -{n/m/h}'.");

}

}

/\*\*

\* Выводит на экран список доступных пользователю команд.

\*/

public void help() {

System.out.println("Lab4 made by Artemy Kulbako, build 3\n" +

"Данные коллекции сохраняются автоматически после каждой успешной модификации.");

System.out.println("Команды: " + manual.keySet() + "\nman {команда} для справки.");

}

/\*\*

\* Выводит справку для конкретной команды.

\* @param arg : Имя команды.

\*/

public void man(String arg) {

System.out.println(manual.get(arg));

}

/\*\*

\* Сериализует коллекцию в файл json.

\*/

public void save() {

try (Writer writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream((jsonCollection))))) {

writer.write(serializer.toJson(citizens));

} catch (Exception ex) {

System.out.println("Возникла непредвиденная ошибка. Коллекция не может быть записана в файл. Мне жаль, что всё так вышло.");

}

}

/\*\*

\* Выводит информацию о коллекции.

\*/

@Override

public String toString() {

return "Тип коллекции: " + citizens.getClass() +

"\nДата инициализации: " + initDate +

"\nКоличество элементов: " + citizens.size();

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof CollectionManager)) return false;

CollectionManager manager = (CollectionManager) o;

return Objects.equals(citizens, manager.citizens) &&

Objects.equals(jsonCollection, manager.jsonCollection);

}

@Override

public int hashCode() {

return Objects.hash(citizens);

}

}

# Выводы по работе.

В процессе выполнения лабораторной работы я познакомился с пакетами языка: java.io, java.util.Collections. Эти знания пригодятся для создания программного обеспечения, которые манипулирует данными, получил навыки работы с исключениями в Java.