

**Университет ИТМО**  
**кафедра ВТ**

Лабораторная работа №5 по программированию  
вариант 38186

выполнил Кадыров Амирджон Халимджонович (Р3110)

2019

Реализовать на базе программы из лабораторной работы №4 консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. Номенклатуру сохраняемых в коллекции объектов необходимо заранее согласовать с преподавателем.

**Разработанная программа должна удовлетворять следующим требованиям:**

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

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

1. **insert {element}** : Добавить новый элемент с заданным ключом.
2. **show**: Вывести в стандартный поток вывода все элементы коллекции в строковом представлении.
3. **remove\_lower {element}** : Удалить из коллекции все элементы, меньшие, чем заданный.
4. **remove {String key}** : Удалить элемент из коллекции по его ключу.
5. **info** : Вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.).
6. **import {String path}** : Добавить в коллекцию все данные из файла.
7. **show\_karlson** : Вывести на стандартный паток вывода изображения Карлсон.
8. **help**: Информация о командах.



# Исходный код программы

## Класс Main

```
import java.io.File;

public class Main {
    public static void main(String[] args) {
        CommandReaderAndExecutor readerAndExecutor = new
        CommandReaderAndExecutor(new
        KarlsonCollectionManager(getFileFromEnvironmentVariable()));
        readerAndExecutor.govern();
    }
    private static File getFileFromEnvironmentVariable() {
        String collectionPath = System.getenv("Karlson_PATH");
        if (collectionPath == null) {
            System.out.println("Путь должен передаваться через переменную
окружения Karlson_PATH");
            return null;
        }else{
            return new File(collectionPath);
        }
    }
}
```

## Класс KarlsonCollectionManager

```
import java.io.*;
import com.google.gson.Gson;
import java.text.SimpleDateFormat;
import java.util.*;

public class KarlsonCollectionManager {
    private HashMap<String,Karlson> karlsonHashMap;
    private Date initDate;
    private File fileForIO;
    public KarlsonCollectionManager(File collectionFile) {
        fileForIO = null;
        this.initDate = new Date();
        karlsonHashMap = new HashMap<String,Karlson>();
        if(collectionFile != null) {
            doImport(collectionFile);
            fileForIO = collectionFile;
        }
    }
    public KarlsonCollectionManager() {
        fileForIO = null;
        this.initDate = new Date();
        karlsonHashMap = new HashMap<String,Karlson>();
    }
    /**
     * Метод выводит на экран информацию о коллекции.
     */
    public void info() {
        System.out.println("Коллекция типа HashMap и содержит из объектов класса
Кarlson." + "\n" + "Keys : Karlson's name" + "\n" + "Values : Karlson's
Objects");
        System.out.println("Дата инициализации" + initDate);
        System.out.println("В коллекции " + karlsonHashMap.size() + "
элементов.");
    }
    /**
     * Метод удаляет элемент из коллекции по его ключу

```

```

    * @param karl : (String) - Name of Karlson Object
    */
    public void remove(String karl) {
        if (karlsonHashMap.remove(karl) != karlsonHashMap.get(karl)) {
            System.out.println("Элемент удалён");
        } else System.out.println("Коллекция не содержит данный элемент");
    }
    /**
     * Метод удаляет из коллекции все элементы, меньшие, чем заданный.
     * @param karlsonForCompare (Karlson) - Объект класса Карлсон.
     */
    public void removeLower(Karlson karlsonForCompare) {
        HashSet<Karlson> hsk = new HashSet<>(karlsonHashMap.values());
        Iterator<Karlson> iterator = hsk.iterator();

        int count = 0;
        while (iterator.hasNext()) {
            Karlson anotherKarlson = iterator.next();
            if (karlsonForCompare.getAge().compareTo(anotherKarlson.getAge()) >
0) {
                karlsonHashMap.remove(anotherKarlson.getName());
                count++;
            }
        }
        System.out.println("Удалено " + count + " элементов");
    }
    /**
     * Метод добавляет объект Карлсона в Коллекцию.
     * @param karlson : (Karlson) - Объект класса Карлсон.
     */
    public void put(Karlson karlson) {
        if (karlsonHashMap.put(karlson.name, karlson) ==
karlsonHashMap.get(karlson)) {
            System.out.println("Элемент добавлен");
        } else System.out.println("Коллекция уже содержит данный элемент");
    }
    /**
     * Метод добавит в коллекцию все данные из файла.
     * @param importFile: (java.io.File) - файл для чтения.
     */
    public void doImport(File importFile) {
        try{
            if(!(importFile.isFile())) throw new FileNotFoundException("Ошибка.
Указанный путь не ведёт к файлу");
            if (!(importFile.exists()))
                throw new FileNotFoundException("Файл коллекцией не найден.
Добавьте элементы вручную или импортируйте из другого файла");
            if (!importFile.canRead()) throw new SecurityException("Доступ
запрещён. Файл защищен от чтения");
            String jsonString = readJsonFromFile(importFile);
            if (!(Karlson.jsonToKarlsonHashMap(jsonString).isEmpty())){
                karlsonHashMap.putAll(Karlson.jsonToKarlsonHashMap(jsonString));
                System.out.println("Добавлены только полностью
инициализированные элементы");
            }else System.out.println("Ничего не добавлено, возможно
импортируемая коллекция пуста, или элементы заданы неверно");
        } catch (FileNotFoundException | SecurityException ex){
            System.out.println(ex.getMessage());
        } catch (IOException ex){
            System.out.println("Непредвиденная ошибка ввода: " + ex);
        } catch (JsonSyntaxException ex){
            System.out.println("Содержимое файла имеет неверный формат,
проверьте синтаксис он должен удовлетворять формату json, а после используйте
команду import {Path} для повторной ошибки");
        }
    }

```

```

    }
}
/**
 * @param fileForRead: (java.io.File) - файл для чтения.
 * @return string in format json (serialized object).
 * @throws IOException - throws Input-Output Exceptions.
 */
private String readJsonFromFile(File fileForRead) throws IOException {
    try(
        FileInputStream fileInpStream = new
FileInputStream(fileForRead);
        BufferedInputStream buffInpStream = new
BufferedInputStream(fileInpStream)) {
        LinkedList<Byte> collectionBytesList = new LinkedList<>();
        while (buffInpStream.available() > 0) {
            collectionBytesList.add((byte) buffInpStream.read());
        }
        char[] collectionChars = new char[collectionBytesList.size()];
        for (int i = 0; i < collectionChars.length; i++) {
            collectionChars[i] = (char) (byte) collectionBytesList.get(i);
        }
        return new String(collectionChars);
    }
}
/**
 * Завершает работу с коллекцией элементов, сохраняя ее в файл из которого
она была считана.
 * Если сохранение в исходный файл не удалось, то сохранение происходит в
файл с уникальным названием.
 */
public void finishWork() {
    File saveFile = (fileForIO != null) ? fileForIO : new File("");
    Gson gson = new Gson();
    try{
        String jsonstring = readJsonFromFile(saveFile);
        if(!Karlson.jsonToKarlsonHashMap(jsonstring).isEmpty()){
            try {
                BufferedWriter buffOutStr = new BufferedWriter(new
FileWriter(saveFile));
                buffOutStr.write(gson.toJson(karlsonHashMap));
                buffOutStr.flush();
                System.out.println("Коллекция сохранена в файл " +
saveFile.getAbsolutePath());
            } catch (IOException e){
                System.out.println("Сохранение коллекции не удалось");
            }
        } else
            System.out.println("Сохранение коллекции не удалось");

    } catch (IOException | NullPointerException e){
        Date d = new Date();
        SimpleDateFormat formatter = new
SimpleDateFormat("yyyy.MM.dd.hh.mm.ss");
        saveFile = new File("saveFile" + formatter.format(d) + ".txt");
        try(BufferedWriter buffOutStr = new BufferedWriter(new
FileWriter(saveFile))) {
            if (saveFile.createNewFile()) throw new IOException();
            buffOutStr.write(gson.toJson(karlsonHashMap));
            buffOutStr.flush();
            System.out.println("Коллекция сохранена в файл " +
saveFile.getAbsolutePath());
        } catch (IOException ex){
            System.out.println("Сохранение коллекции не удалось");
        }
    }
}

```

[illegible]

## Класс CommandReaderAndExecuter

```
import com.google.gson.Gson;
import com.google.gson.JsonSyntaxException;
import java.io.File;
import java.util.NoSuchElementException;
import java.util.Scanner;

class CommandReaderAndExecuter {
    private KarlsonCollectionManager collectionManager;
    private boolean needExit;
    CommandReaderAndExecuter(KarlsonCollectionManager collectionManager) {
        this.collectionManager = new KarlsonCollectionManager();
        if (collectionManager != null) this.collectionManager =
collectionManager;
        needExit = false;
    }
    public void govern() {
        while (!needExit) {
            String[] fullCommand = readAndParseCommand();
            Karlson forAction = null;
            if ((fullCommand[0].equals("import") ||
fullCommand[0].equals("insert") || fullCommand[0].equals("remove") ||
fullCommand[0].equals("remove_lower"))) {
                if (fullCommand.length == 1) {
                    System.out.println("Error, " + fullCommand[0] + " must
have argument.");
                    continue;
                }
                if ((fullCommand.length == 2) &&
!(fullCommand[0].equals("import")) && !(fullCommand[0].equals("remove"))) {
                    try {
                        Gson gson = new Gson();
                        forAction = gson.fromJson(fullCommand[1],
Karlson.class);
                        if ((forAction == null) || (forAction.getName() ==
null) || (forAction.getAge() == null) || (forAction.getPlace() == null)) {
                            System.out.println("Ошибка, элемент задан
неверно, возможно вы указали не все значения");
                            continue;
                        }
                    } catch (JsonSyntaxException ex) {
                        System.out.println("Ошибка, элемент задан неверно");
                        continue;
                    }
                }
            }
            switch (fullCommand[0]) {
                case "info":
                    collectionManager.info();
                    break;
                case "insert":
                    collectionManager.put(forAction);
                    break;
                case "remove":
                    collectionManager.remove(fullCommand[1]);
                    break;
                case "help":
                    collectionManager.help();
                    break;
                case "show_karlson":
                    collectionManager.show_karlson();
                    break;
                case "remove_lower":
```



```

        collectionManager.removeLower(forAction);
        break;
    case "import":
        collectionManager.doImport(new File(fullCommand[1]));
        break;
    case "exit":
        needExit = true;
        collectionManager.finishWork();
        break;
    case "show":
        collectionManager.show();
        break;
    default:
        System.out.println("Ошибка, Известная команда");
    }
}

private String[] readAndParseCommand() {
    Scanner consoleScanner = new Scanner(System.in);
    String command;
    String[] fullcomand;
    int count = 0;
    try {
        System.out.println("Введите команду");
        command = consoleScanner.nextLine();
        fullcomand = command.trim().split(" ", 2);
        if (fullcomand.length == 1) return fullcomand;
        else if (fullcomand[0].equals("insert") ||
fullcomand[0].equals("remove_lower")) {
            fullcomand[1] = fullcomand[1].trim();
            command = fullcomand[1];
            fullcomand[1] = "";
            while (!command.contains("{")) {
                fullcomand[1] += command;
                command = consoleScanner.nextLine().trim();
            }
            count += command.replace("{", "").length() -
command.replace("}", "").length();
            fullcomand[1] += command;
            while (count != 0) {
                command = consoleScanner.nextLine();
                fullcomand[1] += command;
                count += command.replace("{", "").length() -
command.replace("}", "").length();
            }
        } else return fullcomand;
    } catch (NoSuchElementException ex) {
        fullcomand = new String[1];
        fullcomand[0] = "exit";
    }
    return fullcomand;
}
}

```

## Класс Karlson

```

import com.google.gson.Gson;
import com.google.gson.JsonSyntaxException;
import java.util.HashMap;

public class Karlson implements Comparable<Karlson>{
    public String name;
    public Integer age;
    public Place place;
}

```

```

public Karlson(String name,Integer age, Place place){
    this.name = name;
    this.age = age;
    this.place = place;
}
public String getName(){
    return this.name;
}
public Integer getAge(){
    return this.age;
}
public Place getPlace() {
    return place;
}
@Override
public int compareTo(Karlson o) {
    return this.getAge().compareTo(o.getAge());
}
static HashMap<String,Karlson> jsonToKarlsonHashMap(String
jsonKarlsonHashMap) throws JsonSyntaxErrorException {
    try {
        Gson gson = new Gson();
        HashMap<String,Karlson> karlsonHashMap = new HashMap<>();
        int noInitializedCount = 0;
        if (jsonKarlsonHashMap.length() != 0) {
            Karlson[] thingsArray = gson.fromJson(jsonKarlsonHashMap,
Karlson[].class);
            for (Karlson i : thingsArray) {
                if ((i != null) && (i.getName() != null) && (i.getName()
!= null)&& (i.getPlace() != null)) {
                    karlsonHashMap.put(i.getName(),i);
                }else noInitializedCount++;
            }
            if (noInitializedCount > 0) System.out.println("Найдено " +
noInitializedCount + " не полностью инициализированных элементов");
            return karlsonHashMap;
        }catch (JsonSyntaxException ex){
            throw new JsonSyntaxErrorException();
        }
    }
}
}

```

## Класс Place

```

public class Place {
    private String place1;
    private String place2;
    public Place(String place1,String place2){
        this.place1=place1;
        this.place2=place2;
    }
}

```

## Вывод:

В ходе выполнение этой лабораторной работы я научился работать с коллекцией, научился реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме.