Университет ИТМО
Кафедра вычислительной техники

Лабораторная работа №5
по дисциплине «Программирование»
Вариант: 666666666.

Выполнил: Юсюмбели Владислав Иванович
Проверил: Письмак Алексей Евгеньевич

Санкт—Петербург – 2017г.

1) Задание.
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.

2) Исходный код программы.

Класс lab5

```
import GUI.Storage;
import GUI.GUI;
/**
 * Created by slavik on 30.10.16.
 */

public class lab5 {
    public static void main(String[] args) {
        Runtime.getRuntime().addShutdownHook(new Thread() {
            @Override
            public void run() {
                try {
                    Storage.getInstanceOf().saveStorage();
                } catch (NullPointerException ex) {
                    System.out.println("Коллекция пуста, нечего сохранять в файл.");
                }
            }
        });

        new Thread(new GUI(args.length==0?null:args[0])).start();
    }
}
```

Класс GUI

```
package GUI;

import static commands.RemoveObject.*;
import static commands.AddObjects.*;
import java.io.*;
import java.util.*;

/**
 * Created by slavik on 19.02.17.
 *
 * @version 2.0
 * @since 1.0
 */
public class GUI implements Runnable {
    private String command, object;
    private Storage storage = new Storage();
    private boolean exit = false;
    private String file = null;
    private PrintWriter printWriter = new PrintWriter(System.out, true);
```

```
    public GUI(String file) {
        this.file = file;
    }

    @Override
    public void run() {
        if (file != null) {
            if (ParseCSV.getInstanceOf().readFromFile(file)) {
                storage = Storage.getInstanceOf();
                commandExecuting();
            }else dialog();
        } else{
//            System.out.println("Файл не найден");
            dialog();
        }
    }

    private void dialog() {
        printWriter.println("Загрузить дефолтовые данные или из файла?");
        Scanner sc = new Scanner(System.in);
        while (!exit) {
            printWriter.println("1 - дефолтовые");
            printWriter.println("2 - из файла");
            try {
                switch (sc.nextLine()) {
                    case "1": {
                        storage.loadDefaultObjects();
                        Storage.setInstanceOf(storage);
                        commandExecuting();
                        break;
                    }
                    case "2": {
                        printWriter.println("Введите путь к файлу");
                        if (ParseCSV.getInstanceOf().readFromFile(sc.nextLine())) {
                            storage = Storage.getInstanceOf();
                            commandExecuting();
                        }
                        break;
                    }
                    default:
                        printWriter.println("Введите корректные данные.");
                }
            } catch (NullPointerException ex) {
                printWriter.println("Приложение остановлено. Ожидалась команда .");
            } catch (NoSuchElementException | IllegalStateException e) {
                printWriter.println("Приложение остановлено. Ожидалась команда.");
                break;
            }
        }
    }


    //  Выборка команд
```

```
private void commandExecuting() {
    Date currentDate = new Date();
    System.out.println("Команда -help для получения справки");
    System.out.println("Ожидание команды");
    Scanner sc = new Scanner(System.in);
    command = sc.nextLine();

    while (command!=null) {

        parseCommand(command);

        switch (command) {
            case "-help": {
                helpForUser();
                break;
            }
            case "remove_greater_key": {
                removeGreaterKey(object);
                break;
            }
            case "add_if_max": {
                addIfMax(object);
                break;
            }
            case "save": {
                Storage.getInstanceOf().saveStorage();
                break;
            }
            case "remove_lower": {
                removeLower(object);
                break;
            }
            case "insert": {
                insetNewObject(object);
                break;
            }
            case "remove_greater": {
                removeGreater(object);
                break;
            }
            case "load": {
                Storage.getInstanceOf().loadFromFile();
                break;
            }
            case "info": {
                writeInfo(currentDate);
                break;
            }
            case "remove_all": {
                removeAll(object);
                break;
            }
            case "remove": {
                removeWithKey(object);
```

```
                break;
            }
            case "import": {
                importAllFromFile(object);
                break;
            }
            case "clear": {
                clear();
                break;
            }
            case "add_if_min": {
                addIfMin(object);
                break;
            }
            case "show_state": {
                printWriter.println(Storage.getInstanceOf().getFamily());
                break;
            }
            case "exit": {
                exit = true;
                break;
            }
            default:
                printWriter.println("Нет такой команды");
        }
        if (exit) {
            break;
        } else {
            printWriter.println("Ожидание команды");
            command = sc.nextLine();
        }
    }
}


//Parse входных данных
private void parseCommand(String command) {
    command = command.trim();
    try {
        this.command = command.substring(0, command.indexOf(' '));
        this.object = command.substring(command.indexOf(' '), command.length()).trim();
    } catch (IndexOutOfBoundsException ex) {
        this.command = command;
        this.object = null;
    }
}

/**
 * Команда: -help.
 * Помощь юзеру. Вывод реализованных функций с кратким описанием.
 * При вызове метода происходит считывание инфорамации из базы данных.
 *
 * @since 1.0
 */
```

```
    private void helpForUser() {
        try (FileReader fileReader = new FileReader("help")) {
            int c = fileReader.read();
            while (c != -1) {
                System.out.print((char) c);
                c = fileReader.read();
            }
        } catch (FileNotFoundException e) {
            System.out.println("Файл не найден");
        } catch (IOException e) {
            System.out.println("Проверьте корректность файла");
        }


//            ResultSet resultSet = new ConnectDB().readDataFromDB();
//            while (resultSet.next()) {
//                System.out.print(resultSet.getString(1));
//                for (int i = resultSet.getString(1).length(); i < 15; i++)
//                    System.out.print(" ");
//                System.out.println(resultSet.getString(2));
//            }
//        } catch (SQLException e) {
//            System.out.println("Соединение с базой разорвано.");
//            Thread.interrupted();
//        }


    }

    /**
     * Команда: info.
     * Выводит основную информацию о классе Storage.
     *
     * @param currentDate На вход ожидается дата заполнения коллекции {@link
Storage#family}
     * @since 1.0
     */
    private void writeInfo(Date currentDate) {
        Class cl = Storage.getInstanceOf().getFamily().getClass();
        printWriter.println("Имя коллекции - " + cl.getCanonicalName());
        printWriter.println("Дата инициализации - " + currentDate);
        printWriter.println("Количество элемнтов - " + Storage.getInstanceOf().getFamily().size());
        printWriter.println("Пакет - " + cl.getPackage());
        printWriter.println("Имя родительсокго класса - " + cl.getSuperclass().getName());
        printWriter.println("Интерфейсы: ");
        Class[] interfaces = cl.getInterfaces();
        for (Class c : interfaces) {
            printWriter.println(c.getName());
        }
    }
}
```

Класс ParseCSV

```
package GUI;

import deprecated.People;

import java.io.*;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

/**
 * Created by slavik on 25.02.17.
 */
public class ParseCSV {

    private final static ParseCSV instanceOf = new ParseCSV();
    private PrintWriter printWriter = new PrintWriter(System.out, true);
    private static final Storage storage;

    static {
        storage = new Storage();
    }


    public static ParseCSV getInstanceOf() {
        return instanceOf;
    }

    public boolean readFromFile(String path) {
        int count = 1;
        int numberOfReadLine = 0;
        try (FileReader fileReader = new FileReader(path)) {
            int c;
            String object = "";
            while ((c = fileReader.read()) != -1) {
                if ((char) c == '\n') {
                    if (fromCSV(object, count)) {
                        numberOfReadLine++;
                    }
                    count++;
                    object = "";
                } else {
                    object = object + (char) c;
                }
            }

            printWriter.printf("Было считано %d строк из файла %s\n", numberOfReadLine,path);
            Storage.setInstanceOf(storage);
            return true;
        } catch (FileNotFoundException e) {
            printWriter.println("Файл не найден.");
            return false;
        }catch (IOException e) {
            printWriter.println("При чтении файла произошла ошибка. Проверьте содержимое
файла.");
            return false;
```

```
        }
    }


    /**
     * Парсит CSV.
     * @param csv ожидается строка формата CSV.
     *          Пример входных данных: Имя;Возраст;[Miss/Bother/Chat;String[]
     *@param count Номер строки из файла.
     *@return boolean Сигнал об успешном парсинге строки.
     */
    private boolean fromCSV(String csv, int count) {
        String[] object = csv.split(";");
        People people = new People();
        Method method = null;
        boolean state = false;

        try {
            people = new People(object[0]);
            people.setAge(Integer.parseInt(object[1]));
            state = true;
        } catch (IndexOutOfBoundsException ex) {
            printWriter.printf("Не удалось распознать объект в строке %d, у человека должно быть
хотябы имя и возраст \n", count);
        } catch (NumberFormatException ex) {
            printWriter.printf("Проверьте %d строку, возраст не может быть символом/строкой \n",
count);
        }

        if (state)
            for (int i = 2; i < object.length; ) {
                try {
                    Class cl = Class.forName(object[i] + "able");
                    method = people.getClass().getDeclaredMethod(("set" + object[i]), cl);
                    if (i + 1 < object.length) {
                        method.invoke(people, new People(object[i + 1]));
                    }
                    i += 2;
                } catch (ClassNotFoundException e) {
                    try {
                        method.invoke(people, new People(object[i]));
                    } catch (Exception ex) {
                        printWriter.printf("В строке %d ошибка %s, такой коллекции нет \n", count,
object[i]);
                    }
                    ++i;
                } catch (NoSuchMethodException e) {
                    printWriter.printf("В строке %d ошибка в объявлении метода %s \n", count,
object[i]);
                } catch (IllegalAccessException | InvocationTargetException e) {
                    printWriter.printf("В строке %d ошибка в вызове метода %s\n", count, object[i]);
                }
            }
        if (state)
```

```
            storage.getFamily().put(String.valueOf(storage.getFamily().size()), people);
        return state;
    }
}
```

Класс Storage

```
package GUI;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonSyntaxException;
import com.google.gson.reflect.TypeToken;
import deprecated.Food;
import deprecated.People;
import deprecated.Place;

import java.io.*;
import java.lang.reflect.Type;
import java.util.*;

/**
 * Created by slavik on 19.02.17.
 */
public class Storage {
    private Map<String, People> family = new LinkedHashMap<>();
    private final int allPlaces = 15;
    private Map<Integer, People> familyOfChild = new LinkedHashMap<>();
    private List<Place> places = new ArrayList<>();
    private List<People> atTable = new ArrayList<>();
    private static Storage instanceOf;

    public Storage() {
    }

    void loadFromFile() {
        GsonBuilder builder = new GsonBuilder();
        builder.registerTypeAdapter(Missable.class, new InterfaceAdapter<People>());
        builder.registerTypeAdapter(Chatable.class, new InterfaceAdapter<People>());
        builder.registerTypeAdapter(Botherable.class, new InterfaceAdapter<People>());
        Gson gson = builder.create();


        String string = "";
        try (FileReader reader = new FileReader("objects")) {
            int c;
            while ((c = reader.read()) != -1) {
                string = string + (char) c;
            }
            Type typeMap = new TypeToken<Map<String, People>>() {
            }.getType();
            Map<String,People> map = gson.fromJson(string,typeMap);
            if (map==null)
                throw new NullPointerException();
```

```
            family.clear();
            family.putAll(map);
            System.out.printf("%d объекта считано с файла 'objects'\n", family.size());
        } catch (JsonSyntaxException e) {
            System.out.println("Не удалось распознать объект, проверьте корректность данных");
            System.out.println(e.getCause());
        } catch(FileNotFoundException ex) {
            System.out.println("Файл не найден");
        }catch (IOException ex) {
            System.out.println("Произошла ошибка при чтении файла");
        }catch (NullPointerException ex){
            System.out.println("Произошла ошибка, возможно файл пуст");
        }
    }


    void loadDefaultObjects() {
        People x = new People();

        for (int i = 0; i < allPlaces; i++) {
            places.add(new Place());
        }

        x.setName("Малыш");
        x.setAge(7);
        places.get(0).setFull(x);
        family.put("0", x);

        x = new People();
        x.setName("мама");
        x.setAge(31);
        places.get(1).setFull(x);
        family.put("1", x);
        familyOfChild.put(0, x);

        x = new People();
        x.setName("Папа");
        x.setAge(32);
        places.get(2).setFull(x);
        family.put("2", x);
        familyOfChild.put(1, x);

        x = new People();
        x.setName("Босс");
        x.setAge(23);
        places.get(3).setFull(x);
        family.put("3", x);
        familyOfChild.put(2, x);

        x = new People();
        x.setName("Бетан");
        x.setAge(26);
        places.get(4).setFull(x);
        family.put("4", x);
```

```
    familyOfChild.put(3, x);

    x = new People();
    x.setName("Бок");
    x.setAge(38);
    places.get(5).setFull(x);
    family.put("5", x);

    x = new People();
    x.setName("Фрид");
    x.setAge(45);
    places.get(6).setFull(x);
    family.put("6", x);

    System.out.println("Данные загружены");
}


/**
 * Команда save.
 * Сохраняет весь объект типа {@link Storage} в файл.
 *
 * @version 1.0
 */
public void saveStorage() {
    GsonBuilder builder = new GsonBuilder();

    builder.registerTypeAdapter(Missable.class, new InterfaceAdapter<People>());
    builder.registerTypeAdapter(Chatable.class, new InterfaceAdapter<People>());
    builder.registerTypeAdapter(Botherable.class, new InterfaceAdapter<People>());

    Gson gson = builder.create();

    try (PrintWriter printWriter = new PrintWriter("objects")) {
        printWriter.println(gson.toJson(Storage.getInstanceOf().getFamily()));
        System.out.println("Коллекция 'family' была сохранена в файл 'objects'");
    }catch (FileNotFoundException ex ){
        System.out.println("Не хватает прав на запись в файл 'objects'");
    }catch (Exception e) {
        PrintWriter printWriter = new PrintWriter(System.out,true);
        printWriter.println("При сериализации произошла ошибка");
    }
}

public Map<String, People> getFamily() {
    return family;
}


public int getAllPlaces() {
    return allPlaces;
}

public static Storage getInstanceOf() {
```

```
        return instanceOf;
    }

    public  List<Place> getPlaces() {
        return places;
    }

    public static void setInstanceOf(Storage storage) {
        instanceOf = storage;
    }

}
```

Класс AddObjects

```
package commands;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonSyntaxException;

import java.io.*;
import java.lang.reflect.Type;
import java.util.Map;

import GUI.*;
import com.google.gson.reflect.TypeToken;
import deprecated.People;

/**
 * Created by slavik on 22.02.17.
 */

public class AddObjects {
    private static Gson gson = new GsonBuilder().create();
    private static PrintWriter printWriter = new PrintWriter(System.out, true);

    /**
     * Команда: add_if_max.
     * Добавляет новый элемент в коллекцию, если его значение превышает значение
наибольшего элемента этой коллекции.
     *
     * @param object Экземплят типа {@link People} для добавления в коллекцию.
     * @version 2.0
     */
    public static void addIfMax(String object) {
        try {
            People people = gson.fromJson(object, People.class);
            if (people == null) {
                throw new NullPointerException();
            }
            boolean flag = true;
            for (People peopleCollection : Storage.getInstanceOf().getFamily().values()) {
```

```
            if (peopleCollection.getAge() > people.getAge()) {
                flag = false;
                break;
            }
        }

        if (flag) {

Storage.getInstanceOf().getFamily().put(String.valueOf(Storage.getInstanceOf().getFamily().size()
), people);
            printWriter.println("Объект успешно добавлен");
        } else {
            System.out.println("Объект не добавлен");
        }
    } catch (JsonSyntaxException ex) {
        printWriter.println("Не удалось распознать объект, проверьте корректность данных");
        printWriter.println(ex.getCause());
    } catch (NullPointerException ex) {
        printWriter.println("Не ввели данные об объекте");
    }
}

/**
 * Команда add_if_min.
 * Добавляет новый элемент в коллекцию, если его значение меньше, чем у наименьшего
элемента этой коллекции.
 *
 * @param object Ожидается конкретный экземпляр класса {@link People}
 * @version 2.0
 */
public static void addIfMin(String object) {
    try {
        People people = gson.fromJson(object, People.class);
        if (people == null) {
            throw new NullPointerException();
        }
        boolean flag = true;
        for (People peopleCollection : Storage.getInstanceOf().getFamily().values()) {
            if (peopleCollection.getAge() < people.getAge()) {
                flag = false;
                break;
            }
        }

        if (flag) {

Storage.getInstanceOf().getFamily().put(String.valueOf(Storage.getInstanceOf().getFamily().size()
), people);
            printWriter.println("Объект успешно добавлен");
        } else {
            System.out.println("Объект не добавлен");
        }
    } catch (JsonSyntaxException ex) {
        printWriter.println("Не удалось распознать объект, проверьте корректность данных");
```

```
            printWriter.println(ex.getCause());
        } catch (NullPointerException ex) {
            printWriter.println("Не ввели данные об объекте");
        }
    }

    /**
     * Команда import.
     * добавляет в коллекцию все данные из файла.
     *
     * @param path Ожидатеся имя файла или путь к файлу, содержащий коллекцию {@link
Storage#family}
     * @version 2.0
     */
    public static void importAllFromFile(String path) {
        GsonBuilder builder = new GsonBuilder();
        builder.registerTypeAdapter(Missable.class, new InterfaceAdapter<People>());
        builder.registerTypeAdapter(Chatable.class, new InterfaceAdapter<People>());
        builder.registerTypeAdapter(Botherable.class, new InterfaceAdapter<People>());
        Gson gson = builder.create();
        boolean flag = true;
        try {
            int first = path.indexOf('{');
            if (first < 0)
                throw new NullPointerException();
            int last = ++first;
            while (path.charAt(last) != '}') {
                last++;
            }
            path = path.substring(first, last);
        } catch (NullPointerException | IndexOutOfBoundsException e) {
            printWriter.println("Ошибка в имени файла");
            flag = false;
        }

        if (flag) {
            try (FileReader reader = new FileReader(path)) {
                int c;
                String string = "";
                while ((c = reader.read()) != -1) {
                    string = string + (char) c;
                }

                int size = Storage.getInstanceOf().getFamily().size();
                Type typeMap = new TypeToken<Map<String, People>>() {
                }.getType();
                Storage.getInstanceOf().getFamily().putAll(gson.fromJson(string, typeMap));
                printWriter.printf("%d объекта считано с файла 'objects'\n",
Storage.getInstanceOf().getFamily().size() - size);
            } catch (JsonSyntaxException e) {
                printWriter.println("Не удалось распознать объект, проверьте корректность данных");
                printWriter.println(e.getCause());
            } catch (FileNotFoundException ex) {
                System.out.println("Файл не найден");
```

```
            } catch (IOException | IndexOutOfBoundsException ex) {
                printWriter.println("Произошла ошибка при чтении файла");
            } catch (NullPointerException ex) {
                printWriter.println("Произошла ошибка, возможно файл пуст");
            }
        }
    }

    /**
     * Команда insert.
     * Добавляет новый элемент с заданным ключом.
     *
     * @param string Экземплят типа {@link People} для добавления в коллекцию.
     * @version 2.0
     */
    public static void insetNewObject(String string) {
        boolean flag = true;
        String object = null,key = null;
        try {
            key = string.substring(string.indexOf('{') + 1, string.indexOf('}'));
            int first = string.indexOf('}') + 2;
            int last = first;
            while (string.charAt(last) != '}') {
                last++;
            }
            object = string.substring(first-1, last+1);
            object.trim();
        }catch (IndexOutOfBoundsException ex){
            System.out.println("Не правильно введены данные об объекте");
            flag = false;
        }catch (NullPointerException ex){
            System.out.println("Введите данные об объекте");
            flag = false;
        }

        if(flag){
            try {
                People people = gson.fromJson(object, People.class);
                if (people == null) {
                    throw new NullPointerException();
                }
                Storage.getInstanceOf().getFamily().put(key, people);
                printWriter.println("Объект успешно добавлен");
            }catch (NullPointerException ex){
                printWriter.println("Не ввели данные об объекте");
            } catch (JsonSyntaxException ex) {
                printWriter.println("Не удалось распознать объект, проверьте корректность данных");
                printWriter.println(ex.getCause());
            }
        }

    }
}
```

Класс RemoveObjects

```
package commands;
import GUI.Storage;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonSyntaxException;
import deprecated.People;

import java.io.PrintWriter;

/**
 * Created by slavik on 21.02.17.
 */
public class RemoveObject {
    private static Gson gson = new GsonBuilder().create();
    private static PrintWriter printWriter = new PrintWriter(System.out, true);

    /**
     * Команда: remove_greater_key.
     * Удаляет из коллекции все элементы, ключ которых превышает заданный.
     *
     * @param key Ключ определенного объекта, который лежит в коллекции {@link
Storage#family}.
     *            Ожидается формат {String}
     * @version 2.0
     */
    public static void removeGreaterKey(String key) {
        try {
            int size = Storage.getInstanceOf().getFamily().size();

            int first = key.indexOf('{');
            if (first < 0)
                throw new IndexOutOfBoundsException();

            int last = ++first;
            while (key.charAt(last) != '}') {
                last++;
            }

            final String substring = key.substring(first, last);
            Storage.getInstanceOf().getFamily().entrySet().removeIf(entry ->
entry.getKey().compareTo(substring) > 0);
            printWriter.printf("Операция выполнена успешно. Удалено %d объекта\n", size -
Storage.getInstanceOf().getFamily().size());
        } catch (IndexOutOfBoundsException e) {
            printWriter.println("Укажите корректный ключ");
        } catch (NullPointerException ex) {
            printWriter.println("Ужите ключ");
        }
    }

    /**
     * Команда remove.
```

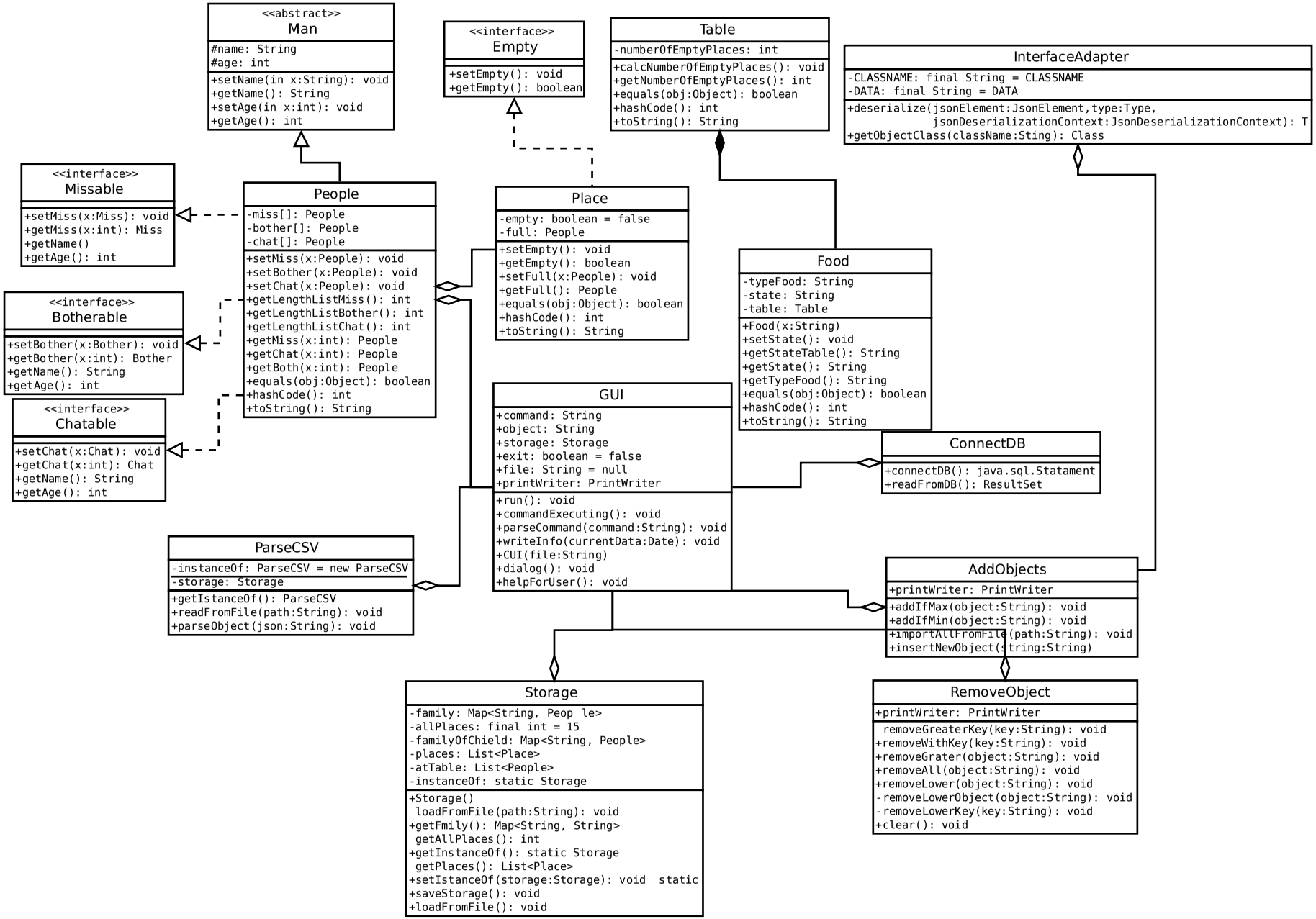
```
     * Удаляет элемент из коллекции по его ключу.
     *
     * @param key  Ключ - строковая переменная определенного объекта, который лежит в
коллекции {@link Storage#family}
     * @version 2.0
     */
    public static void removeWithKey(String key) {
        try {
            int size = Storage.getInstanceOf().getFamily().size();

            int first = key.indexOf('{');
            if (first < 0)
                throw new IndexOutOfBoundsException();

            int last = ++first;
            while (key.charAt(last) != '}') {
                last++;
            }

            final String substring = key.substring(first, last);
            Storage.getInstanceOf().getFamily().remove(substring);
            printWriter.printf("Операция выполнена успешно. Удалено %d объекта\n", size -
Storage.getInstanceOf().getFamily().size());
        } catch (IndexOutOfBoundsException e) {
            printWriter.println("Укажите корректный ключ");
        } catch (NullPointerException ex) {
            printWriter.println("Ужите ключ");
        }
    }

    /**
     * Команда remove_greater.
     * Удаляет из коллекции все элементы, превышающие заданный.
     *
     * @param object Ожидается строка формата json для преобразования в объект {@link
People}.
     * @version 2.0
     * @since 1.0
     */
    public static void removeGreater(String object) {

        try {
            int size = Storage.getInstanceOf().getFamily().size();
            People people = gson.fromJson(object, People.class);
            if (people == null)
                throw new NullPointerException();
            Storage.getInstanceOf().getFamily().entrySet().removeIf(entry -> entry.getValue().getAge()
> people.getAge());
            printWriter.printf("Операция выполнена успешно. Удалено %d объекта\n", size -
Storage.getInstanceOf().getFamily().size());
        } catch (JsonSyntaxException ex) {
            printWriter.println("Не удалось распознать объект, проверьте корректность данных");
            printWriter.println(ex.getCause());
        } catch (NullPointerException ex) {
```

```
            printWriter.println("Не ввели данные об объекте");
        }

    }

    /**
     * Команда remove_all.
     * Удалят из коллекции все элементы, эквивалентные заданному.
     *
     * @param object Ожидается строка формата json для преобразования в объект {@link
People}
     * @version 2.0
     * @since 1.0
     */
    public  static void removeAll(String object) {
        try {
            int size = Storage.getInstanceOf().getFamily().size();
            People people = gson.fromJson(object, People.class);
            if (people == null)
                throw new NullPointerException();
            Storage.getInstanceOf().getFamily().entrySet().removeIf(entry ->
entry.getValue().getAge()==people.getAge());
            printWriter.printf("Операция выполнена успешно. Удалено %d объекта\n", size -
Storage.getInstanceOf().getFamily().size());
        } catch (JsonSyntaxException ex) {
            printWriter.println("Не удалось распознать объект, проверьте корректность данных");
            printWriter.println(ex.getCause());
        } catch (NullPointerException ex) {
            printWriter.println("Не ввели данные об объекте");
        }
    }

    /**
     * Команда remove_lower.
     * Удаляет из коллекции все элементы, меньшие, чем заданный.
     *
     * @param object Строковая переменная.
     *               Можно передать строку в формате json для парсинга в конкретный экзеплят
{@link People}.
     *               Так же можно передать ключ для коллекции {@link Storage#family}.
     * @version 1.0
     * @see RemoveObject#removeLowerKey(String)
     * @see RemoveObject#removeLowerObject(String)
     */
    public static void removeLower(String object) {
        if (!removeLowerObject(object)) {
            removeLowerKey(object);
        }
    }


    /**
     * Команда remove_lower.
     * Удаляет из коллекции все элементы, меньшие, чем заданный.
```

```
     *
     * @param object Ожидается строка формата json для преобразования в конкретный
экземпляр класса {@link People}
     * @version 2.0
     * @since 1.0
     * @return boolean Сигнал об успешном распозновании объекта.
     */
    private static boolean removeLowerObject(String object) {
        try {
            People people = gson.fromJson(object, People.class);
            int size = Storage.getInstanceOf().getFamily().size();
            if (people == null)
                throw new NullPointerException();
            Storage.getInstanceOf().getFamily().entrySet().removeIf(entry -> entry.getValue().getAge()
< people.getAge());

            printWriter.printf("Удаление по объекту. Операция выполнена успешно. Удалено %d
объекта\n", size - Storage.getInstanceOf().getFamily().size());
            return true;
        } catch (Exception ex) {
            return false;
        }
    }


    /**
     * Команда remove_lower.
     * Удаляет из коллекции все элементы, ключ которых меньше, чем заданный.
     *
     * @param key Ожидается ключ для коллекции {@link Storage#family}.
     * @version 1.0
     */
    private static void removeLowerKey(String key) {
        try {
            int size = Storage.getInstanceOf().getFamily().size();

            int first = key.indexOf('{');
            if (first < 0)
                throw new IndexOutOfBoundsException();

            int last = ++first;
            while (key.charAt(last) != '}') {
                last++;
            }

            final String substring = key.substring(first, last);
            Storage.getInstanceOf().getFamily().entrySet().removeIf(entry ->
entry.getKey().compareTo(substring) < 0);
            printWriter.printf("Удаление по ключу. Операция выполнена успешно. Удалено %d
объекта\n", size - Storage.getInstanceOf().getFamily().size());
        } catch (IndexOutOfBoundsException e) {
            printWriter.println("Укажите корректный ключ");
        } catch (NullPointerException ex) {
            printWriter.println("Ужите ключ");
```

```
        }
    }

    /**
     * Команда clear.
     * Очищает коллекцию.
     *  @version 1.0
     *  @since 1.0
     */
    public static void clear() {
        int size = Storage.getInstanceOf().getFamily().size();
        Storage.getInstanceOf().getFamily().clear();
        System.out.printf("Коллекция очищена. Удалено %d объектов\n",size);
    }
}
```

3) Выод: В ходе выполнения лабораторной работы мною было реализовано на базе программы из лабораторной работы №4 консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме.


<<abstract>>
Man
#name: String
#age: int
+setName(in x:String): void
+getName(): String
+setAge(in x:int): void
+getAge(): int
<<interface>>
Empty
+setEmpty(): void
+getEmpty(): boolean
Table
-numberOfEmptyPlaces: int
+calcNumberOfEmptyPlaces(): void
+getNumberOfEmptyPlaces(): int
+equals(obj:Object): boolean
+hashCode(): int
+toString(): String
InterfaceAdapter
-CLASSNAME: final String = CLASSNAME
-DATA: final String = DATA
+deserialize(jsonElement:JsonElement,type:Type,
jsonDeserializationContext:JsonDeserializationContext): T
+getObjectClass(className:Sting): Class
<<interface>>
Missable
+setMiss(x:Miss): void
+getMiss(x:int): Miss
+getName()
+getAge(): int
People
-miss[]: People
-bother[]: People
-chat[]: People
+setMiss(x:People): void
+setBother(x:People): void
+setChat(x:People): void
+getLengthListMiss(): int
+getLengthListBother(): int
+getLengthListChat(): int
+getMiss(x:int): People
+getChat(x:int): People
+getBoth(x:int): People
+equals(obj:Object): boolean
+hashCode(): int
+toString(): String
Place
-empty: boolean = false
-full: People
+setEmpty(): void
+getEmpty(): boolean
+setFull(x:People): void
+getFull(): People
+equals(obj:Object): boolean
+hashCode(): int
+toString(): String
Food
-typeFood: String
-state: String
-table: Table
+Food(x:String)
+setState(): void
+getStateTable(): String
+getState(): String
+getTypeFood(): String
+equals(obj:Object): boolean
+hashCode(): int
+toString(): String
<<interface>>
Botherable
+setBother(x:Bother): void
+getBother(x:int): Bother
+getName(): String
+getAge(): int
<<interface>>
Chatable
+setChat(x:Chat): void
+getChat(x:int): Chat
+getName(): String
+getAge(): int
GUI
+command: String
+object: String
+storage: Storage
+exit: boolean = false
+file: String = null
+printWriter: PrintWriter
+run(): void
+commandExecuting(): void
+parseCommand(command:String): void
+writeInfo(currentData:Date): void
+CUI(file:String)
+dialog(): void
+helpForUser(): void
ConnectDB
+connectDB(): java.sql.Statament
+readFromDB(): ResultSet
ParseCSV
-instanceOf: ParseCSV = new ParseCSV
-storage: Storage
+getIstanceOf(): ParseCSV
+readFromFile(path:String): void
+parseObject(json:String): void
AddObjects
+printWriter: PrintWriter
+addIfMax(object:String): void
+addIfMin(object:String): void
+importAllFromFile(path:String): void
+insertNewObject(string:String)
Storage
-family: Map<String, Peop le>
-allPlaces: final int = 15
-familyOfChield: Map<String, People>
-places: List<Place>
-atTable: List<People>
-instanceOf: static Storage
+Storage()
loadFromFile(path:String): void
+getFmily(): Map<String, String>
getAllPlaces(): int
+getInstanceOf(): static Storage
getPlaces(): List<Place>
+setIstanceOf(storage:Storage): void static
+saveStorage(): void
+loadFromFile(): void
RemoveObject
+printWriter: PrintWriter
removeGreaterKey(key:String): void
+removeWithKey(key:String): void
+removeGrater(object:String): void
+removeAll(object:String): void
+removeLower(object:String): void
-removeLowerObject(object:String): void
-removeLowerKey(key:String): void
+clear(): void