Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет систем управления и робототехники

Отчет по лабораторной работе №5

«Основные концепции ООП»

по дисциплине «Программирование»

Выполнил: студент гр. R3135, Дупак А. А.

Преподаватель: Письмак А. Е.

Санкт-Петербург
2019

Задание (вариант №3001)

Реализовать на базе программы из лабораторной работы №4 консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме.

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.LinkedHashMap.
- При запуске приложения коллекция должна автоматически заполняться значениями из файла.
- Имя файла должно передаваться программе с помощью аргумента командной строки.
- 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.util.Scanner.
- Запись данных в файл необходимо реализовать с помощью класса java.io.FileOutputStream.
- Все реализованные команды должны быть задокументированы в формате javadoc.
- Формат задания объектов в командах — json.

В интерактивном режиме программа должна поддерживать выполнение следующих команд:

- insert {String key} {element}: добавть новый элемент с заданным ключом.
- show: вывести в стандартный поток вывода все элементы коллекции в строковом представлении.
- Import {String path}: добавить в коллекцию все данные из файла.
- save: сохранить коллекцию в файл.
- Info: вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов).
- remove {String key}: удалить элемент из коллекции по его ключу.
- remove_greater_key {String key}: удалить из коллекции все элементы, ключ которых превышает заданный.

Материалы работы

Главный класс истории

```
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import java.lang.reflect.Type;
import java.util.Map;
import java.util.Scanner;
```

```
      public class Story {
          public static void main(String[] args) {
              Musican Guslya = new Musican("Guslya", "playing the
flute.");
              Painter Tubick = new Painter("Tubick", "");
              Town myTown = new Town();
              Child Pugovka = new Child("Pugovka", "");
              anonymous.Sound("Once.");
              try {
                  myTown.Start();
                  Guslya.Play();
                  Tubick.Sound("What a " +
Types.answer.Wonderful.toString() + " musician!");
                  Pugovka.Sound("I hear such musical instrument for the
first time.");
              }
              catch(NoLinkException e15) {
                  throw new NoLinkException();
              }
              double a = Math.random()*2-1;
              if (a > 0)
              {
                  Pugovka.Sound("I, like many, do not know such
instrument.");
              }
              else
              {
                  Pugovka.Sound("I, unlike many, know what a flute is.");
              }
              if (a < 0)
              {
                  Tubick.Sound("I also know how to play the flute.");
              }
              else
              {
                  Tubick.Sound("And I can't play the flute.");
              }
              Tubick.Sound("I and " + Guslya.name + "live together " +
myTown.Town_Name() + " " + myTown.Square());
              anonymous.Sound("In the house of baby " + Pugovka.name);
              try {
                  a = Math.random() * 2 - 1;
                  if (a > 0) {
                      Tubick.Sound("The room in which we settled is " +
Types.answer.Spacious.toString() + "!");
```

```
                } else if (a < 0) {
                    Tubick.Sound("The room in which we settled is " +
Types.answer.Bright.toString() + "!");
                } else throw new
WordException(Types.answer.Boring.toString());
            }
            catch(WordException e59) {
                Tubick.Sound("The room in which we settled is " +
Types.answer.Boring.toString() + "...");
            }
            myTown.End();

            System.out.println();
            Start enter = new Start();
            System.out.println("Data import...");
            String file_name = "data.json";
            enter.load(file_name);
            System.out.println("Interactive mode. Enter 'help' to see
all supported commands.");
            boolean exit = false;
            Gson gson = new Gson();
            Type type = new TypeToken<Map<String, String>>()
{}.getType();
            do {
                System.out.print("->");
                Scanner scan = new Scanner(System.in);
                String command = scan.next();
                String str_arguments = scan.nextLine();
                String[] arguments = str_arguments.split("}");
                Map<String, String> arg1, arg2;
                String key = null, path = null, name = null, skill =
null;
                try {
                    arg1 = gson.fromJson((arguments[0] + "}"), type);
                    key = arg1.get("key");
                    path = arg1.get("path");
                    arg2 = gson.fromJson((arguments[1] + "}"), type);
                    name = arg2.get("name");
                    skill = arg2.get("skill");
                }
                catch (Exception e) {}
                switch (command) {
                    case "help": enter.help();
                        break;
                    case "insert":
```

```
                        if (key != null && name != null && skill !=
null) {
                            enter.insert(key, new Child(name, skill));
                        }
                        else {
                            enter.arguments_error();
                        }
                        break;
                    case "remove":
                        if (key != null) {
                            enter.remove(key);
                        }
                        else {
                            enter.arguments_error();
                        }
                        break;
                    case "remove_greater_key":
                        if (key != null) {
                            enter.remove_greater_key(key);
                        }
                        else {
                            enter.arguments_error();
                        }
                        break;
                    case "show": enter.show();
                        break;
                    case "info":
                        enter.info();
                        break;
                    case "import":
                        if (path != null) {
                            enter.load(path);
                        }
                        else {
                            enter.arguments_error();
                        }
                        break;
                    case "save":
                        enter.save(file_name);
                        break;
                    case "exit":
                        exit = true;
                        break;
                    default:
                        enter.command_error();
```

```
            }
        } while (!exit);
        enter.save(file_name);
    }
    static Sounds anonymous = new Sounds() {
        public void Sound(String message) {
            System.out.println(message);
            Wait.delay();
        }
    };
}
```

Абстрактный класс персонажа

```
public abstract class AbsPerson implements Sounds {
    protected String name, skill;
    public AbsPerson(String name, String skill){
        this.name = name;
        this.skill = skill;
    }
    public void Sound(String msg) {
        System.out.println(this.name + " sound: " + msg);
    }
}
```

Класс ребенка

```
public class Child extends AbsPerson {
    public Child(String name, String skill){
        super(name, skill);
    }
    @Override
    public void Sound(String msg) {
        System.out.println(this.name + " sound: " + msg);
        Wait.delay();
    }
    String get_name() {
        return name;
    }
}
```

Класс музыканта

```
public class Musican extends AbsPerson implements Skills {
    public Musican(String name, String skill){
        super(name, skill);
    }
    @Override
```

```
    public void Sound(String msg) {
        System.out.println(this.name + " sound: " + msg);
        Wait.delay();
    }
    @Override
    public void Play() {
        System.out.println(this.name + " " + skill);
        Wait.delay();
    }
}
```

Класс художника

```
public class Painter extends AbsPerson {
    public Painter(String name, String skill){
        super(name, skill);
    }
    @Override
    public void Sound(String msg) {
        System.out.println(name + " sound: " + msg);
        Wait.delay();
    }
}
```

Класс города

```
public class Town {
     class Finishing {
     public void F() {
           System.out.println("The story ends here.");
           }
     }
     public void End()
     {
           Finishing end = new Finishing();
           end.F();
     }
    private String town_name = "Green";
    private String sqr_name = "Apple";
    public void Start() {
        System.out.println("In the town <" + this.town_name + ">");
        Wait.delay();
    }
    public String Town_Name() {
        return ("In the town <" + this.town_name + ">");
    }
    public String Square() {
```

```
        return ("On the square <" + this.sqr_name + ">");
    }
    @Override
    public boolean equals(Object obj)
    {
        if (obj == null)
        {
            return false;
        }
        if (getClass() != obj.getClass())
        {
            return false;
        }
        final Town other = (Town) obj;
        if ((this.town_name == null) ? (other.town_name !=
null) : !this.town_name.equals(other.town_name))
        {
            return false;
        }
        if ((this.sqr_name == null) ? (other.sqr_name != null) : !
this.sqr_name.equals(other.sqr_name))
        {
            return false;
        }
        return true;
    }
    @Override
    public int hashCode()
    {
        int hash = 3;
        hash = 19*hash + (this.town_name != null ?
this.town_name.hashCode() : 0);
        hash = 19*hash + (this.sqr_name != null ?
this.sqr_name.hashCode() : 0);
        return hash;
    }
    @Override
    public String toString()
    {
        return "Town{" + "Name=" + this.town_name + " Square=" +
this.sqr_name + "}";
    }
}
```

Интерфейс для звуков

```
public interface Sounds {
    void Sound(String msg);
}
```

Интерфейс для навыков

```
public interface Skills {
    void Play();
}
```

Класс прилагательных для описания

```
public class Types {
    enum answer {Wonderful, Cool, Special, Unusual, Amazing,
Spacious, Bright, Boring}
}
```

Класс реализующий задержку

```
import java.util.concurrent.TimeUnit;

public class Wait {
    public static void delay() {
        class Inf {
            public void get_inf() {
                System.out.println("Error n21");
            }
        }
        try {
            TimeUnit.SECONDS.sleep(1);
        } catch (InterruptedException e) {
            Inf myInf = new Inf();
            myInf.get_inf();
        }
    }
}
```

Собственный класс непроверяемого исключения

```
public class NoLinkException extends RuntimeException {
    public String toString() {
        return "Error n15";
    }
}
```

Собственный класс проверяемого исключения

```
public class WordException extends Exception {
    public WordException(String word) {
        System.out.print("Error n59: (" + word + ") ");
    }
}
```

Класс для работы с коллекцией объектов

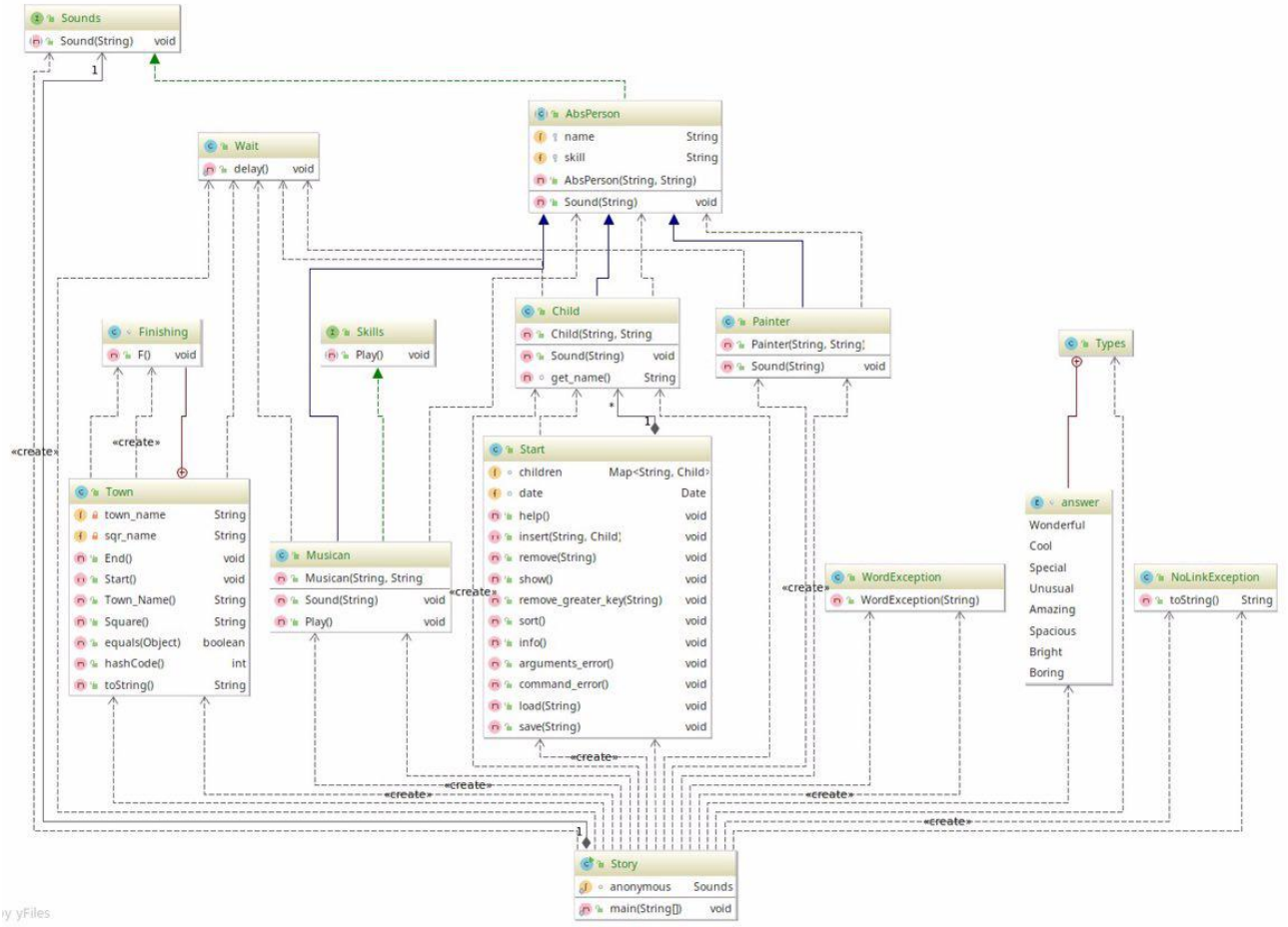
```
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.lang.reflect.Type;
import java.util.*;

public class Start {
    Map<String, Child> children = new LinkedHashMap<>();
    Date date = new Date();

    public void help() {
        System.out.println("insert {String key} {element}: add a
new element with a given key.");
        System.out.println("show: show all the elements of the
collection.");
        System.out.println("import {String path}: add to the
collection all the data from the file.");
        System.out.println("save: save current collection to
file.");
        System.out.println("info: show information about the
collection.");
        System.out.println("remove {String key}: remove element
from the collection by its key.");
        System.out.println("remove_greater_key {String key}: remove
from the collection all elements whose key exceeds the specified.");
        System.out.println("exit: exit interactive mode.");
        System.out.println("Example: ->insert {\"key\":\"a1\"}
{\"name\":\"Alex\",\"skill\":\"learn\"}");
    }

    public void insert(String key, Child element) {
        children.put(key, element);
        System.out.printf("Inserted. %d elements in the collection
now. %n", children.size());
```

```
        }

        public void remove(String key) {
            Child dead = children.remove(key);
            if (dead == null) {
                System.out.println("No such key in the collection.");
            } else {
                String form = "{\"name\": \"%s\", \"skill\": \"%s\"}
has been removed. %n";
                System.out.printf(form, dead.name, dead.skill);
            }
        }

        public void show() {
            sort();
            if (children.isEmpty()) {
                System.out.println("Collection is empty.");
            }
            else {
                String form = "{\"key\": \"%s\", \"element\":
{\"name\": \"%s\", \"skill\": \"%s\"}} %n";
                children.forEach((k, v) -> System.out.printf(form, k,
children.get(k).name, children.get(k).skill));
            }
        }

        public void remove_greater_key(String desired_key) {
            int start_size = children.size();
            Set<String> keys = children.keySet();
            List<String> trash = new ArrayList<>();
            for (String key : keys) {
                if (key.compareTo(desired_key) > 0) {
                    trash.add(key);
                }
            }
            for (String key : trash) {
                children.remove(key);
            }
            if (children.size() < start_size) {
                System.out.printf("%d collection items have been
removed. %n", (start_size - children.size()));
            }
            else {
                System.out.printf("There are no any keys in the
collection greater than \"%s\". %n", desired_key);
```

```
            }
        }

        public void sort() {
            List<Map.Entry<String, Child>> list = new
ArrayList<>(children.entrySet());
            Collections.sort(list, new Comparator<Map.Entry<String,
Child>>() {
                @Override
                public int compare(Map.Entry<String, Child>
stringChildEntry, Map.Entry<String, Child> t1) {
                    return
stringChildEntry.getKey().compareTo(t1.getKey());
                }
            });
            Map<String, Child> sortedMap = new LinkedHashMap<>();
            for (Map.Entry<String, Child> entry : list) {
                sortedMap.put(entry.getKey(), entry.getValue());
            }
            children = sortedMap;
        }

        public void info() {
            System.out.printf("collection type: %s%n",
children.getClass());
            System.out.printf("initialisation date: %s%n",
date.toString());
            System.out.printf("number of elements: %s%n",
children.size());
        }

        public void arguments_error() {
            System.out.println("Input error. See 'help'");
        }

        public void command_error() {
            System.out.println("No such command. See 'help'");
        }

        public void load(String path) {
            Gson gson = new Gson();
            StringBuilder jsonStr = new StringBuilder();
            File file = new File(path);
            try {
                Scanner scan = new Scanner(file);
```

```
                while (scan.hasNext()) {
                    jsonStr.append(scan.next());
                }
                try {
                    Type type = new TypeToken<Map<String, Child>>()
{}.getType();
                    if (jsonStr.toString().length()>0) {
                        Map<String, Child> imported =
gson.fromJson(jsonStr.toString(), type);
                        children.putAll(imported);
                        System.out.println("The file was imported
successfully.");
                    }
                    else {
                        System.out.println("The file is empty.");
                    }
                }
                catch (Exception e) {
                    System.out.println("Incorrect file or format.");
                }
            }
            catch (FileNotFoundException e) {
                System.out.println("No such file or directory.");
            }
        }

        public void save(String file_name) {
            Gson gson = new Gson();
            sort();
            try {
                FileOutputStream output = new
FileOutputStream(file_name);
                String packed = gson.toJson(children);
                byte[] buffer = packed.getBytes();
                output.write(buffer);
                System.out.println("The collection has been saved.");
            }
            catch (IOException e) {
                System.out.println("Output error.");
            }
        }
    }
```

Диаграмма классов



Результат

Once.

In the town <Green>

Guslya playing the flute.

Tubick sound: What a Wonderful musician!

Pugovka sound: I hear such musical instrument for the first time.

Pugovka sound: I, unlike many, know what a flute is.

Tubick sound: I also know how to play the flute.

Tubick sound: I and Guslya live together In the town <Green> On the square <Apple>

In the house of baby Pugovka

Tubick sound: The room in which we settled is Spacious!

The story ends here.

Обработка исключения

Error n59: (Boring) Tubick sound: The room in which we settled is Boring…

Работа в интерактивном режиме

Data import...
The file was imported successfully.
Interactive mode. Enter 'help' to see all supported commands.
->show
{"key": "a1", "element": {"name": "Robert", "skill": "swim"}}
{"key": "f8", "element": {"name": "Mary", "skill": "fly"}}
{"key": "g4", "element": {"name": "Kate", "skill": "draw"}}
->insert {"key":"a54"} {"name":"Bob","skill":"run"}
Inserted. 4 elements in the collection now.
->remove {"key":"f8"}
{"name": "Mary", "skill": "fly"} has been removed.
->help
insert {String key} {element}: add a new element with a given key.
show: show all the elements of the collection.
import {String path}: add to the collection all the data from the file.
save: save current collection to file.
info: show information about the collection.
remove {String key}: remove element from the collection by its key.
remove_greater_key {String key}: remove from the collection all elements whose key exceeds the specified.
exit: exit interactive mode.
Example: ->insert {"key":"a1"} {"name":"Alex","skill":"learn"}
->exit
The collection has been saved.

Data import...
The file was imported successfully.
Interactive mode. Enter 'help' to see all supported commands.
->remove_greater_key {"key":"b"}
1 collection items have been removed.
->save
The collection has been saved.

Выводы

В результате проделанной работы на основе описания предметной области, было реализовано консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме.

Вместе с тем, были приобритены навыки работы с коллекцией объектов в программых.