**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий, механики и оптики**

**Факультет «ПИиКТ»**

**Лабораторная работа №5**

**по дисциплине «Программирование»**

Выполнил студент группы Р3113

Ефимов Н.А.

Вариант:\_\_\_\_\_\_\_7003\_\_\_\_\_\_\_\_\_\_

Преподаватель: Письмак А.Е.\_\_\_

Санкт-Петербург

2019 г.

# Задание

Изображение выглядит как снимок экрана

Автоматически созданное описание

# Диаграмма классов

Изображение выглядит как текст

Автоматически созданное описание

# Исходный код

Main.java

import java.io.\*;

import sun.misc.Signal;

import sun.misc.SignalHandler;

#define println(args) System.out.println(args)

#define print(args) System.out.print(args)

public class Main {

protected static String filename;

protected static CmdWorker worker;

static {

initSignalHandlers();

}

public static void initSignalHandlers() {

SignalHandler signalHandler = new SignalHandler() {

@Override

public void handle(Signal sig) {

release();

}

};

DiagnosticSignalHandler.install("TERM", signalHandler);

DiagnosticSignalHandler.install("INT", signalHandler);

DiagnosticSignalHandler.install("ABRT", signalHandler);

DiagnosticSignalHandler.install("TSTP", signalHandler);

}

public static void release() {

worker.printToFile(filename);

System.exit(0);

}

public static void load() {

worker.readFromFile(filename);

}

public static void handling() throws IOException {

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

while (true) {

print("-> ");

String cmd = reader.readLine();

worker.doCmd(cmd);

}

}

public static void main(String ... args) {

worker = new CmdWorker();

if (args.length > 0) {

filename = args[0];

load();

} else

filename = "storage.xml";

try {

handling();

} catch (Exception e) {

e.printStackTrace();

}

}

}

DiagnosticSignalHandler.java

import sun.misc.Signal;

import sun.misc.SignalHandler;

#define println(args) System.out.println(args)

public class DiagnosticSignalHandler implements SignalHandler {

protected SignalHandler oldHandler;

protected SignalHandler handler;

// Static method to install the signal handler

public static void install(String signalName, SignalHandler handler) {

Signal signal = new Signal(signalName);

DiagnosticSignalHandler diagnosticSignalHandler = new DiagnosticSignalHandler();

SignalHandler oldHandler = Signal.handle(signal, diagnosticSignalHandler);

diagnosticSignalHandler.setHandler(handler);

diagnosticSignalHandler.setOldHandler(oldHandler);

}

protected DiagnosticSignalHandler() {}

protected void setOldHandler(SignalHandler oldHandler) {

this.oldHandler = oldHandler;

}

protected void setHandler(SignalHandler handler) {

this.handler = handler;

}

// Signal handler method

@Override

public void handle(Signal sig) {

// Go to new string after SIG print

println();

// println("Signal handler called for signal " + sig);

try {

handler.handle(sig);

// Chain back to previous handler, if one exists

if (oldHandler != SIG\_DFL && oldHandler != SIG\_IGN)

oldHandler.handle(sig);

} catch (Exception e) {

println("Signal handler failed, reason " + e);

}

}

}

CmdWorker.java

import java.util.PriorityQueue;

import java.util.Map;

import java.util.HashMap;

import java.util.function.Function;

import java.util.Date;

import java.util.Iterator;

import java.util.ArrayList;

import java.util.List;

import java.util.Arrays;

import org.json.JSONObject;

import org.json.JSONException;

import java.io.IOException;

import java.io.FileWriter;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.File;

import java.io.StringReader;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import javax.xml.transform.OutputKeys;

import org.w3c.dom.Attr;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.InputSource;

#define println(args) System.out.println(args)

#define printf(format, args) System.out.printf(format, args)

#define COLLECTION\_TYPE PriorityQueue

#define PRIOR\_TYPE City

final class Key {

public Function<String, Void> func;

public String description;

public Key(Function<String, Void> func, String description) {

this.func = func;

this.description = description;

}

}

public class CmdWorker {

protected Map<String, Key> cmdMap;

protected COLLECTION\_TYPE<PRIOR\_TYPE> priorityQueue;

protected Date initDate;

protected Date lastChangeDate;

protected final String CMD\_NOT\_FOUND = "command not found";

protected final String VOID\_STR = "void";

{

initDate = new Date();

lastChangeDate = initDate;

cmdMap = new HashMap<>();

#define addCmd(name, func, desc) cmdMap.put(name, new Key(this::func, desc));

addCmd("add", add, "добавить новый элемент в коллекцию");

addCmd("remove\_first", removeFirst, "удалить первый элемент из коллекции");

addCmd("show", show, "вывести в стандартный поток вывода все элементы коллекции в строковом представлении");

addCmd("info", info, "вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.)");

addCmd("remove\_lower", removeLower, "удалить из коллекции все элементы, меньшие, чем заданный");

addCmd("remove", remove, "удалить элемент из коллекции по его значению");

addCmd("add\_if\_max", addIfMax, "добавить новый элемент в коллекцию, если его значение превышает значение наибольшего элемента этой коллекции");

addCmd("list", list, "просмотр списка команд");

addCmd("help", list, "просмотр списка команд");

addCmd("man", man, "описание команд");

addCmd("exit", exit, "выход");

#undef addCmd

}

public CmdWorker() {

priorityQueue = new COLLECTION\_TYPE<>();

}

public void readFromFile(final String filename) {

try {

FileReader reader = new FileReader(filename);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(new InputSource(reader));

doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("city");

for (int temp = 0; temp < nList.getLength(); temp++) {

Node nNode = nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT\_NODE) {

Element eElement = (Element)nNode;

priorityQueue.add(new City(eElement.getElementsByTagName("name").item(0).getTextContent(), Integer.parseInt(eElement.getElementsByTagName("size").item(0).getTextContent())));

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

public void printToFile(final String filename) {

try {

DocumentBuilderFactory documentFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder documentBuilder = documentFactory.newDocumentBuilder();

Document document = documentBuilder.newDocument();

Element cityCont = document.createElement("city-cont");

document.appendChild(cityCont);

Element city, name, size;

for (PRIOR\_TYPE elem : priorityQueue) {

city = document.createElement("city");

cityCont.appendChild(city);

name = document.createElement("name");

name.appendChild(document.createTextNode(elem.name));

city.appendChild(name);

size = document.createElement("size");

size.appendChild(document.createTextNode(elem.areaSize.toString()));

city.appendChild(size);

}

DOMSource domSource = new DOMSource(document);

StreamResult result = new StreamResult(new BufferedWriter(new FileWriter(filename)));

// create the xml file

//transform the DOM Object to an XML File

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

transformer.setOutputProperty(OutputKeys.OMIT\_XML\_DECLARATION, "no");

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

transformer.transform(domSource, result);

} catch (Exception e) {

e.printStackTrace();

}

}

protected PRIOR\_TYPE processInput(String jsonInput) {

PRIOR\_TYPE city = null;

String name;

Integer size;

try {

JSONObject obj = new JSONObject(jsonInput);

try {

name = obj.getString("name");

} catch (JSONException e) {

name = "Unnamed";

}

try {

size = obj.getInt("size");

} catch (JSONException e) {

size = 0;

}

city = new PRIOR\_TYPE(name, size);

} catch (JSONException e) {

println("Введите элемент!");

return city;

}

return city;

}

public void doCmd(String cmd) {

int indexOfSpace = cmd.indexOf(' ');

indexOfSpace = indexOfSpace == -1 ? cmd.length() : indexOfSpace;

String[] splittedCmd = {cmd.substring(0, indexOfSpace), indexOfSpace != cmd.length() ? cmd.substring(indexOfSpace + 1, cmd.length()) : VOID\_STR};

try {

cmdMap.get(splittedCmd[0]).func.apply(splittedCmd[1]);

} catch (NullPointerException e) {

println(CMD\_NOT\_FOUND);

}

}

#define void Void

#define return return null

#define check() if (city == null) return;

/\*\*

\* <p>Удаляет из коллекции все элементы, меньшие, чем заданный.</p>

\*

\* @param jsonElem Строка где содержится json элемент

\* @return null, возвращает null в любом случае

\*/

public void removeLower(String jsonElem) {

PRIOR\_TYPE city = processInput(jsonElem);

check();

for (Iterator<PRIOR\_TYPE> iterator = priorityQueue.iterator(); iterator.hasNext(); ) {

PRIOR\_TYPE elem = iterator.next();

if (elem.compareTo(city) < 0) {

iterator.remove();

lastChangeDate = new Date();

return;

}

}

return;

}

/\*\*

\* <p>Добавить новый элемент в коллекцию, если его значение превышает значение наибольшего элемента этой коллекции.</p>

\*

\* @param jsonElem Строка где содержится json элемент

\* @return null, возвращает null в любом случае

\*/

public void addIfMax(String jsonElem) {

PRIOR\_TYPE city = processInput(jsonElem);

check();

for (PRIOR\_TYPE elem : priorityQueue)

if (elem.compareTo(city) >= 0)

return;

priorityQueue.add(city);

lastChangeDate = new Date();

return;

}

/\*\*

\* <p>Удалить элемент из коллекции по его значению.</p>

\*

\* @param jsonElem Строка где содержится json элемент

\* @return null, возвращает null в любом случае

\*/

public void remove(String jsonElem) {

PRIOR\_TYPE city = processInput(jsonElem);

check();

for (Iterator<PRIOR\_TYPE> iterator = priorityQueue.iterator(); iterator.hasNext(); ) {

PRIOR\_TYPE elem = iterator.next();

if (elem.equals(city)) {

iterator.remove();

lastChangeDate = new Date();

return;

}

}

return;

}

/\*\*

\* <p>Добавить новый элемент в коллекцию.</p>

\*

\* @param jsonElem Строка где содержится json элемент

\* @return null, возвращает null в любом случае

\*/

public void add(String jsonElem) {

PRIOR\_TYPE city = processInput(jsonElem);

check();

priorityQueue.add(city);

lastChangeDate = new Date();

return;

}

#undef check

/\*\*

\* <p>Вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.).</p>

\*

\* @param NULL строка где будет пустая строка или VOID\_STR

\* @return null, возвращает null в любом случае

\*/

public void info(String NULL) {

println("Класс коллекции: " + priorityQueue.getClass().getName());

try {

println("Класс элементов: " + priorityQueue.peek().getClass().getName());

} catch(Exception e) {}

println("Колиечество элементов: " + priorityQueue.size());

printf("Дата инициализации: %s\n", initDate.toLocaleString());

printf("Дата последнего изменения: %s\n", lastChangeDate.toLocaleString());

return;

}

/\*\*

\* <p>Вывести в стандартный поток вывода все элементы коллекции в строковом представлении.</p>

\*

\* @param NULL строка где будет пустая строка или VOID\_STR

\* @return null, возвращает null в любом случае

\*/

public void show(String NULL) {

for (PRIOR\_TYPE elem : priorityQueue)

printf("%s\n", elem);

return;

}

/\*\*

\* <p>Удалить первый элемент из коллекции.</p>

\*

\* @param NULL строка где будет пустая строка или VOID\_STR

\* @return null, возвращает null в любом случае

\*/

public void removeFirst(String NULL) {

priorityQueue.poll();

lastChangeDate = new Date();

return;

}

/\*\*

\* <p>Выход.</p>

\*

\* @param NULL строка где будет пустая строка или VOID\_STR

\* @return null, возвращает null в любом случае

\*/

public void exit(String NULL) {

Main.release();

return;

}

/\*\*

\* <p>Команда для просмотра описания команд.</p>

\*

\* @param cmd Строка где содержится команда для просмотра ее описания

\* @return null, возвращает null в любом случае

\*/

public void man(String cmd) {

try {

println(cmdMap.get(cmd).description);

} catch (NullPointerException e) {

println("right syntax is man <command>");

}

return;

}

/\*\*

\* <p>Просмотр списка команд.</p>

\*

\* @param NULL строка где будет пустая строка или VOID\_STR

\* @return null, возвращает null в любом случае

\*/

public void list(String NULL) {

for (String cmd : cmdMap.keySet())

println(cmd);

return;

}

#undef void

#undef return

public void sort() {

Arrays.sort(priorityQueue.toArray());

}

}

City.java

import java.util.Arrays;

class City extends Place implements Comparable

{

protected Integer areaSize;

public City(String name, Integer areaSize) {

super(name);

this.areaSize = areaSize;

}

public void setAreaSize(Integer areaSize) {

this.areaSize = areaSize;

}

public Integer getAreaSize() {

return areaSize;

}

@Override

public int compareTo(Object object) {

if (this == object)

return 0;

if (object == null)

throw new NullPointerException();

if (this.getClass() != object.getClass())

throw new ClassCastException();

City city = (City)object;

return this.areaSize.compareTo(city.areaSize) + this.name.compareTo(city.name);

}

@Override

public boolean equals(Object object) {

if (this == object)

return true;

if (object == null)

return false;

if (this.getClass() != object.getClass())

return false;

City city = (City)object;

if (!this.name.equals(city.name) || !this.areaSize.equals(city.areaSize))

return false;

return true;

}

@Override

public int hashCode() {

return Arrays.deepHashCode((Object[])new Object[]{name, areaSize});

}

@Override

public String toString() {

return "Имя города: " + name + "\n" + "Площадь: " + areaSize;

}

}

Place.java

import java.util.Arrays;

abstract class Place

{

protected String name = "Без имени";

public Place(String string) {

name = string;

}

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

@Override

public boolean equals(Object object) {

if (this == object)

return true;

if (object == null)

return false;

if (this.getClass() != object.getClass())

return false;

Place place = (Place)object;

if (this.name != place.name)

return false;

return true;

}

@Override

public int hashCode() {

return Arrays.deepHashCode((Object[])new Object[]{name});

}

}

Makefile

test\_args = storage.xml

compile\_options = -XDignore.symbol.file

src\_dir = src/

preprocessed\_dir = preprocessed/

binaries\_dir = bin/

lib\_path = lib/

class\_path = $(lib\_path)\*:.

startup\_string = java -cp $(class\_path):$(binaries\_dir) Main

# Startups

test: $(binaries\_dir)\*.class

######## START #########

$(startup\_string) $(test\_args)

run: $(binaries\_dir)\*.class

######## START #########

$(startup\_string)

# Java doc

docs: $(preprocessed\_dir)\*.java

mkdir docs

javadoc -cp $(lib\_path)\* $(preprocessed\_dir)\* -d docs

# Clear

clear:

rm -f $(binaries\_dir)\* $(preprocessed\_dir)\*

# Preprocessing

$(preprocessed\_dir)\*.java: $(src\_dir)\*.java

# Copy all

cp -rf $(src\_dir)\* $(preprocessed\_dir)

# Do smth by hands

cpp -P $(src\_dir)CmdWorker.java $(preprocessed\_dir)CmdWorker.java

cpp -P $(src\_dir)DiagnosticSignalHandler.java $(preprocessed\_dir)DiagnosticSignalHandler.java

cpp -P $(src\_dir)Main.java $(preprocessed\_dir)Main.java

# Compiling

$(binaries\_dir)\*.class: $(preprocessed\_dir)\*.java

javac -cp $(class\_path) $(preprocessed\_dir)\*.java -d $(binaries\_dir) $(compile\_options)

# Вывод

В ходе выполнения лабораторной работы я овладел навыками работы с препроцессором c++, просмотра документации java, работы с коллекциями и даже создание скромной, но все-таки своей документации с помощью Javadoc!!!