Минобрнауки России | РГРТУ | Кафедра ВПМ

Курс «ПРОГРАММИРОВАНИЕ НА JAVA»

**Отчет о практической работе № 5**

Выполнил:

Вербицкая Ирина Сергеевна

студент группы 143

электронная почта oora.frt@gmail.com

Проверил:

Пруцков Александр Викторович

д-р техн. наук, профессор кафедры ВПМ

Рязань 2023

# Задание

Звукозапись. Определить иерархию музыкальных композиций. Записать на диск сборку. Подсчитать продолжительность. Провести перестановку композиций диска на основе принадлежности к стилю. Найти композицию.

# Основные классы, реализующие задание

## Класс Runner

package ru.rsreu.verbickaya0505.classes.basic;

import ru.rsreu.verbickaya0505.classes.other.InvalidParamsException;

import ru.rsreu.verbickaya0505.classes.other.MusicDisk;

public class Runner {

private Runner() {

}

public static void main(String[] args) {

StringBuilder resultString = new StringBuilder();

try {

// ИНИЦИАЛИЗАЦИЯ КОМПОЗИЦИЙ

Initializer.initialize();

// ЗАПИСЬ ДИСКА НА СБОРКУ

MusicDisk myDisk = new MusicDisk(Initializer.Popular1, Initializer.Sacred1, Initializer.Academic1,

Initializer.Folk1);

myDisk.addCompositions(Initializer.Academic2, Initializer.Popular2, Initializer.Folk2, Initializer.Sacred2);

myDisk.addCompositions(Initializer.Folk3, Initializer.Popular3, Initializer.Academic3);

myDisk.addCompositions(Initializer.Popular4, Initializer.Academic4);

myDisk.addCompositions(Initializer.Popular5);

// ВЫВОД СОДЕРЖИМОГО ДИСКА + ПОДСЧЕТ ПРОДОЛЖИТЕЛЬНОСТИ

resultString.append(myDisk.toString());

// СОРТИРОВКА СОДЕРЖИМОГО ДИСКА ПО СТИЛЮ И ПОВТОРНЫЙ ВЫВОД

myDisk.sortByStyle();

resultString.append(myDisk.toString());

// ПОИСК КОМПОЗИЦИЙ

resultString.append(myDisk.stringOfSearchedByName("Итальянская полька"));

resultString.append(myDisk.stringOfSearchedByName("Ещё один день"));

resultString.append(myDisk.stringOfSearchedByName("Ещё одна ночь"));

} catch (InvalidParamsException e) {

resultString.append(e.toString());

} finally {

System.out.println(resultString);

}

}

}

## Класс AbstractComposition

package ru.rsreu.verbickaya0505.classes.compositions;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

import ru.rsreu.verbickaya0505.classes.other.InvalidParamsException;

import ru.rsreu.verbickaya0505.classes.other.TimeInSeconds;

import ru.rsreu.verbickaya0505.enums.ancestors.SomeEnum;

import ru.rsreu.verbickaya0505.enums.characteristics.CompositionTypes;

public abstract class AbstractComposition implements Comparable<AbstractComposition> {

private static Resourcer resourcer = ProjectResourcer.getInstance();

private String name;

private TimeInSeconds duration;

private SomeEnum style;

private String author;

private CompositionTypes type;

public AbstractComposition(String name, int durationInSeconds, SomeEnum style, String author)

throws InvalidParamsException {

if (("".equals(name)) || (name == null)) {

throw new InvalidParamsException(resourcer.getString("composition.with.no.name"));

}

if (durationInSeconds <= 0) {

throw new InvalidParamsException(resourcer.getString("composition.with.wrong.duration"));

}

this.setName(name);

this.setDuration(durationInSeconds);

this.setStyle(style);

this.setAuthor(author);

this.setType();

}

@Override

public String toString() {

String s = "";

s += resourcer.getString("composition.name") + " " + this.getName() + "\n";

s += resourcer.getString("composition.type") + " " + this.getType().toString() + "\n";

s += resourcer.getString("composition.duration") + " " + this.getDuration().toString() + "\n";

s += resourcer.getString("composition.style") + " " + this.getStyle().toString() + "\n";

if (!(this.getAuthor().equals(""))) {

s += resourcer.getString("composition.author") + " " + this.getAuthor() + "\n";

}

return s;

}

@Override

public int compareTo(AbstractComposition c) {

if (c == this) {

return 0;

} else {

if (c.getClass() != this.getClass()) {

return this.getType().compareTo(c.getType());

} else {

return this.getStyle().compareTo(c.getStyle());

}

}

}

public String getName() {

return this.name;

}

public TimeInSeconds getDuration() {

return this.duration;

}

public SomeEnum getStyle() {

return this.style;

}

public String getAuthor() {

return this.author;

}

public CompositionTypes getType() {

return this.type;

}

public void setName(String name) {

this.name = name;

}

public void setDuration(int durationInSeconds) {

this.duration = new TimeInSeconds(durationInSeconds);

}

public void setStyle(SomeEnum style) {

this.style = style;

}

public void setAuthor(String author) {

this.author = author;

}

public abstract void setType();

protected void setType(CompositionTypes type) {

this.type = type;

}

}

## Класс AcademicComposition

package ru.rsreu.verbickaya0505.classes.compositions;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

import ru.rsreu.verbickaya0505.classes.other.InvalidParamsException;

import ru.rsreu.verbickaya0505.enums.characteristics.AcademicMusicMoods;

import ru.rsreu.verbickaya0505.enums.characteristics.CompositionTypes;

import ru.rsreu.verbickaya0505.enums.styles.AcademicMusicStyles;

public class AcademicComposition extends AbstractComposition {

private static Resourcer resourcer = ProjectResourcer.getInstance();

private AcademicMusicMoods mood;

private String orchestra;

public AcademicComposition(String name, int durationInSeconds, AcademicMusicStyles style, String author,

AcademicMusicMoods mood, String orchestra) throws InvalidParamsException {

super(name, durationInSeconds, style, author);

this.setType();

this.setMood(mood);

this.setOrchestra(orchestra);

}

@Override

public String toString() {

String s = super.toString();

if (!(this.getMood().toString().equals(""))) {

s += resourcer.getString("academic.composition.mood") + " " + this.getMood().toString() + "\n";

}

if (!(this.getOrchestra().equals(""))) {

s += resourcer.getString("academic.composition.orchestra") + " " + this.getOrchestra() + "\n";

}

return s;

}

@Override

public void setType() {

super.setType(CompositionTypes.academicComposition);

}

public AcademicMusicMoods getMood() {

return this.mood;

}

public String getOrchestra() {

return this.orchestra;

}

public void setMood(AcademicMusicMoods mood) {

this.mood = mood;

}

public void setOrchestra(String orchestra) {

this.orchestra = orchestra;

}

}

## Класс MusicDisk

package ru.rsreu.verbickaya0505.classes.other;

import java.util.Arrays;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

import ru.rsreu.verbickaya0505.classes.compositions.AbstractComposition;

public class MusicDisk {

private static final String COMPOSITION\_SEPARATOR = "~~~~~~~~~~~~~~~~\n";

private static final String DISK\_SEPARATOR = "^^^^^^^^^^^^^^^^\n";

private static Resourcer resourcer = ProjectResourcer.getInstance();

private AbstractComposition[] compositions;

private TimeInSeconds duration;

public MusicDisk(AbstractComposition... args) {

this.compositions = new AbstractComposition[0];

this.addCompositions(args);

}

public void addCompositions(AbstractComposition... args) {

int oldLength = this.compositions.length;

int addingLength = args.length;

int newLength = oldLength + addingLength;

AbstractComposition[] newArray = new AbstractComposition[newLength];

System.arraycopy(this.compositions, 0, newArray, 0, oldLength);

System.arraycopy(args, 0, newArray, oldLength, addingLength);

this.compositions = new AbstractComposition[newLength];

System.arraycopy(newArray, 0, this.compositions, 0, newLength);

this.stateDuration();

}

public TimeInSeconds getDuration() {

return this.duration;

}

public void sortByStyle() {

Arrays.sort(this.compositions);

}

public AbstractComposition[] searchByName(String name) {

int length = this.countEntranceOfName(name);

AbstractComposition[] array = new AbstractComposition[length];

int i = 0;

for (AbstractComposition element : this.compositions) {

if (element.getName().equals(name)) {

array[i] = element;

i++;

}

}

return array;

}

public String stringOfSearchedByName(String name) {

String s = "\n" + resourcer.getString("the.result.of.search.by.name") + " \"" + name + "\":\n";

AbstractComposition[] array = this.searchByName(name);

if (array.length == 0) {

s += resourcer.getString("search.returned.no.results");

} else {

for (AbstractComposition item : array) {

s += MusicDisk.COMPOSITION\_SEPARATOR;

s += item.toString();

}

}

return s;

}

@Override

public String toString() {

String s = "\n";

s += MusicDisk.DISK\_SEPARATOR + resourcer.getString("here.is.disk.of.compositions") + "\n";

s += resourcer.getString("general.duration.of.compositions") + " " + this.getDuration().toString() + "\n";

for (int i = 0; i < this.compositions.length; i++) {

s += MusicDisk.COMPOSITION\_SEPARATOR;

s += resourcer.getString("here.is.composition");

s += String.format(" № %d ", i + 1) + "\n";

s += this.compositions[i].toString();

}

return s;

}

private void stateDuration() {

int generalDuration = 0;

for (AbstractComposition element : this.compositions) {

generalDuration += element.getDuration().getSecTime();

}

this.duration = new TimeInSeconds(generalDuration);

}

private int countEntranceOfName(String name) {

int count = 0;

for (AbstractComposition element : this.compositions) {

if (element.getName().equals(name)) {

count++;

}

}

return count;

}

}

## Класс TimeInSeconds

package ru.rsreu.verbickaya0505.classes.other;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

public class TimeInSeconds {

private static final int MINUT\_COEFF = 60;

private static final int HOUR\_COEFF = MINUT\_COEFF \* 60;

private static final int DAY\_COEFF = HOUR\_COEFF \* 24;

private static final int SEC\_ARRAY\_NUMBER = 0;

private static final int MINUT\_ARRAY\_NUMBER = 1;

private static final int HOUR\_ARRAY\_NUMBER = 2;

private static final int DAY\_ARRAY\_NUMBER = 3;

private static final int ARRAY\_NUMBER = 4;

private static Resourcer resourcer = ProjectResourcer.getInstance();

private int secTime;

private int[] formattedTime;

public TimeInSeconds(int time) {

this.setTime(time);

this.stateFormattedTime();

}

public void setTime(int time) {

this.secTime = time;

this.stateFormattedTime();

}

public int getSecTime() {

return this.secTime;

}

@Override

public String toString() {

String s = "";

int day = this.getFormattedTime()[DAY\_ARRAY\_NUMBER];

int hour = this.getFormattedTime()[HOUR\_ARRAY\_NUMBER];

int min = this.getFormattedTime()[MINUT\_ARRAY\_NUMBER];

int sec = this.getFormattedTime()[SEC\_ARRAY\_NUMBER];

if (day > 0) {

s += String.format("%d ", day) + resourcer.getString("time.days") + " ";

}

if (hour > 0) {

s += String.format("%d ", hour) + resourcer.getString("time.hours") + " ";

}

if (min > 0) {

s += String.format("%d ", min) + resourcer.getString("time.minuts") + " ";

}

if (sec > 0) {

s += String.format("%d ", sec) + resourcer.getString("time.seconds");

}

return s;

}

private void stateFormattedTime() {

int time = this.getSecTime();

this.formattedTime = new int[ARRAY\_NUMBER];

this.formattedTime[DAY\_ARRAY\_NUMBER] = time / TimeInSeconds.DAY\_COEFF;

time %= TimeInSeconds.DAY\_COEFF;

this.formattedTime[HOUR\_ARRAY\_NUMBER] = time / TimeInSeconds.HOUR\_COEFF;

time %= TimeInSeconds.HOUR\_COEFF;

this.formattedTime[MINUT\_ARRAY\_NUMBER] = time / TimeInSeconds.MINUT\_COEFF;

time %= TimeInSeconds.MINUT\_COEFF;

this.formattedTime[SEC\_ARRAY\_NUMBER] = time;

}

private int[] getFormattedTime() {

return this.formattedTime;

}

}