Минобрнауки России | РГРТУ | Кафедра ВПМ

Курс «ПРОГРАММИРОВАНИЕ НА JAVA»

**Отчет о практической работе №**6

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

Бытовая техника: марка; модель; тип устройства.

# Основные классы, реализующие задание

## Класс Runner

package ru.rsreu.verbickaya0605.basic;

import java.util.Scanner;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

import ru.rsreu.verbickaya0605.exceptions.FixableException;

import ru.rsreu.verbickaya0605.exceptions.IrreparableException;

public class Runner {

private Runner() {

}

private static Resourcer resourcer = ProjectResourcer.getInstance();

public static void main(String[] args) {

try {

// 1 Creating folders structure

System.out.print(resourcer.getString("step.creating.folders") + "\n");

Initializer.initializeFolders();

// 2 Creating and initializing of objects` array

System.out.print(resourcer.getString("step.creating.array") + "\n");

try {

Initializer.initializeArray();

} catch (FixableException exception) {

System.out.print(exception.toString());

}

// 3 Creating of a file and filling it with array

System.out.print(resourcer.getString("step.creating.filling.file") + "\n");

Initializer.initializeFile();

// 4 Creating of a copy file

System.out.print(resourcer.getString("step.creating.copy.file") + "\n");

Initializer.initializeCopyFile();

// 5 Moving copy file

System.out.print(resourcer.getString("step.moving.file") + "\n");

System.out.print(resourcer.getString("message.press.f") + "\n");

Scanner in = new Scanner(System.in);

String key = in.next();

Initializer.initializeMoveFile(key.trim().toLowerCase().equals("f"));

in.close();

// 6 Reading source file and copy file to arrays

System.out.print(resourcer.getString("step.reading.files") + "\n");

try {

Initializer.initializeReadingToArrays();

} catch (FixableException exception) {

System.out.print(exception.toString());

}

// 7 Output arrays on a tables

System.out.print(resourcer.getString("step.output.arrays") + "\n");

System.out.print(Initializer.getArray1().toStringTable());

System.out.print(Initializer.getArray2().toStringTable());

// 8 Comparing arrays

System.out.print(resourcer.getString("step.compare.arrays") + "\n");

if (Initializer.getArray1().compareTo(Initializer.getArray2()) == 0) {

System.out.print(resourcer.getString("message.arrays.are.equal") + "\n");

} else {

System.out.print(resourcer.getString("message.arrays.not.equal") + "\n");

}

} catch (IrreparableException exception) {

System.out.print(exception.toString());

} catch (Throwable exception) {

System.out.print("\n" + resourcer.getString("error.unexpected"));

} finally {

System.out.print(resourcer.getString("message.the.end"));

}

}

}

## Класс Initializer

package ru.rsreu.verbickaya0605.basic;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

import ru.rsreu.verbickaya0605.exceptions.FixableException;

import ru.rsreu.verbickaya0605.exceptions.IrreparableException;

import ru.rsreu.verbickaya0605.householdappliances.HouseholdAppliance;

import ru.rsreu.verbickaya0605.householdappliances.HouseholdApplianceType;

import ru.rsreu.verbickaya0605.householdappliances.HouseholdAppliancesStorage;

public class Initializer {

private static Resourcer resourcer = ProjectResourcer.getInstance();

private Initializer() {

}

private static final String ROOT = System.getProperty("user.dir");

private static boolean moved;

private static HouseholdAppliance appliance1;

private static HouseholdAppliance appliance2;

private static HouseholdAppliance appliance3;

private static HouseholdAppliance appliance4;

private static HouseholdAppliance appliance5;

private static File folder1;

private static File folder2;

private static File myFile;

private static File myFileCopy;

private static HouseholdAppliancesStorage array;

private static HouseholdAppliancesStorage array1;

private static HouseholdAppliancesStorage array2;

public static HouseholdAppliancesStorage getArray() {

return Initializer.array;

}

public static HouseholdAppliancesStorage getArray1() {

return Initializer.array1;

}

public static HouseholdAppliancesStorage getArray2() {

return Initializer.array2;

}

public static void initializeFolders() throws IrreparableException {

Initializer.folder1 = new File(FilePathFormer.formPath(Initializer.ROOT,

resourcer.getString("files.folder.source.name"), resourcer.getString("files.folder.copy.name")));

Initializer.folder2 = new File(

FilePathFormer.formPath(Initializer.ROOT, resourcer.getString("files.folder.move.name")));

if ((!(Initializer.folder1.mkdirs())) && (!(Initializer.folder1.exists()))

|| (!(Initializer.folder2.mkdirs())) && (!(Initializer.folder2.exists()))) {

throw new IrreparableException(resourcer.getString("error.reason.wrong.folder.names"));

}

}

public static void initializeArray() throws FixableException {

Initializer.appliance1 = new HouseholdAppliance("Samsung", "RB37A5000WW", HouseholdApplianceType.refrigerator);

Initializer.appliance2 = new HouseholdAppliance("Philips", "DST3010/30", HouseholdApplianceType.iron);

Initializer.appliance3 = new HouseholdAppliance("Bosch", "SMS23BW01T", HouseholdApplianceType.dishwasher);

Initializer.appliance4 = new HouseholdAppliance("REDMOND", "RHB-2964", HouseholdApplianceType.blender);

Initializer.appliance5 = new HouseholdAppliance("Polaris", "PGS 2200VA", HouseholdApplianceType.steamer);

Initializer.array = new HouseholdAppliancesStorage(Initializer.appliance1, Initializer.appliance2,

Initializer.appliance3, Initializer.appliance4, Initializer.appliance5);

}

public static void initializeFile() throws IrreparableException {

String filePath = FilePathFormer.formPath(resourcer.getString("files.folder.source.name"),

resourcer.getString("files.file.data.name") + "." + resourcer.getString("files.file.text.extension"));

String fileFullPath = FilePathFormer.formPath(Initializer.ROOT, filePath);

Initializer.myFile = new File(filePath);

BufferedWriter writer = null;

try {

if (Initializer.myFile.exists()) {

Initializer.myFile.delete();

}

Initializer.myFile.createNewFile();

writer = new BufferedWriter(new FileWriter(filePath));

writer.write(Initializer.array.toString());

} catch (FileNotFoundException exception) {

throw new IrreparableException(resourcer.getString("error.reason.wrong.directory") + " " + fileFullPath);

} catch (IOException exception) {

throw new IrreparableException(resourcer.getString("error.reason.wrong.writing") + " " + fileFullPath);

} finally {

if (writer != null) {

try {

writer.close();

} catch (IOException exception) {

throw new IrreparableException(

resourcer.getString("error.reason.wrong.write.closing") + " " + fileFullPath);

}

}

}

}

public static void initializeCopyFile() throws IrreparableException {

String sourcePath = FilePathFormer.formPath(resourcer.getString("files.folder.source.name"),

resourcer.getString("files.file.data.name") + "." + resourcer.getString("files.file.text.extension"));

String destinationPath = FilePathFormer.formPath(resourcer.getString("files.folder.source.name"),

resourcer.getString("files.folder.copy.name"),

resourcer.getString("files.file.data.name") + "." + resourcer.getString("files.file.text.extension"));

String fullDestinationPath = FilePathFormer.formPath(Initializer.ROOT, destinationPath);

Initializer.myFileCopy = new File(destinationPath);

try {

if (Initializer.myFileCopy.exists()) {

Initializer.myFileCopy.delete();

}

Initializer.myFileCopy.createNewFile();

} catch (IOException exception) {

throw new IrreparableException(

resourcer.getString("error.reason.wrong.directory") + " " + fullDestinationPath);

}

String textFromFile = Initializer.readTextFromFile(sourcePath);

Initializer.writeTextToFile(textFromFile, destinationPath);

String newFileName = FilePathFormer.formPath(resourcer.getString("files.folder.source.name"),

resourcer.getString("files.folder.copy.name"),

resourcer.getString("files.file.data.name") + "." + resourcer.getString("files.file.backup.extension"));

File newBakFile = new File(newFileName);

if (newBakFile.exists()) {

newBakFile.delete();

}

if (!Initializer.myFileCopy.renameTo(newBakFile)) {

throw new IrreparableException(

resourcer.getString("error.reason.wrong.renaming") + " " + fullDestinationPath);

}

}

public static void initializeMoveFile(boolean key) throws IrreparableException {

String sourcePath = FilePathFormer.formPath(resourcer.getString("files.folder.source.name"),

resourcer.getString("files.file.data.name") + "." + resourcer.getString("files.file.text.extension"));

String destinationPath = FilePathFormer.formPath(resourcer.getString("files.folder.move.name"),

resourcer.getString("files.file.data.name") + "." + resourcer.getString("files.file.text.extension"));

String fullDestinationPath = FilePathFormer.formPath(Initializer.ROOT, destinationPath);

File myFileMove = new File(destinationPath);

Initializer.moved = key;

if (key) {

try {

if (myFileMove.exists()) {

myFileMove.delete();

}

myFileMove.createNewFile();

} catch (IOException exception) {

throw new IrreparableException(

resourcer.getString("error.reason.wrong.directory") + " " + fullDestinationPath);

}

String textFromFile = Initializer.readTextFromFile(sourcePath);

Initializer.writeTextToFile(textFromFile, destinationPath);

Initializer.myFile.delete();

Initializer.myFile = myFileMove;

} else {

if (myFileMove.exists()) {

myFileMove.delete();

}

}

}

public static void initializeReadingToArrays() throws FixableException, IrreparableException {

String path1;

if (Initializer.moved) {

path1 = FilePathFormer.formPath(resourcer.getString("files.folder.move.name"),

resourcer.getString("files.file.data.name") + "."

+ resourcer.getString("files.file.text.extension"));

} else {

path1 = FilePathFormer.formPath(resourcer.getString("files.folder.source.name"),

resourcer.getString("files.file.data.name") + "."

+ resourcer.getString("files.file.text.extension"));

}

String path2 = FilePathFormer.formPath(resourcer.getString("files.folder.source.name"),

resourcer.getString("files.folder.copy.name"),

resourcer.getString("files.file.data.name") + "." + resourcer.getString("files.file.backup.extension"));

Initializer.array1 = Initializer.formStorageFromFile(path1);

Initializer.array2 = Initializer.formStorageFromFile(path2);

}

private static String readTextFromFile(String sourcePath) throws IrreparableException {

String fullSourcePath = FilePathFormer.formPath(Initializer.ROOT, sourcePath);

BufferedReader reader = null;

StringBuilder textFromFile = new StringBuilder();

String string;

try {

reader = new BufferedReader(new FileReader(sourcePath));

while ((string = reader.readLine()) != null) {

textFromFile.append(string + "\n");

}

} catch (FileNotFoundException exception) {

throw new IrreparableException(resourcer.getString("error.reason.wrong.directory") + " " + fullSourcePath);

} catch (IOException exception) {

throw new IrreparableException(resourcer.getString("error.reason.wrong.reading") + " " + fullSourcePath);

} finally {

if (reader != null) {

try {

reader.close();

} catch (IOException exception) {

throw new IrreparableException(

resourcer.getString("error.reason.wrong.read.closing") + " " + fullSourcePath);

}

}

}

return textFromFile.toString();

}

private static void writeTextToFile(String someText, String destinationPath) throws IrreparableException {

String fullDestinationPath = FilePathFormer.formPath(Initializer.ROOT, destinationPath);

BufferedWriter writer = null;

try {

writer = new BufferedWriter(new FileWriter(destinationPath));

writer.write(someText);

} catch (FileNotFoundException exception) {

throw new IrreparableException(

resourcer.getString("error.reason.wrong.directory") + " " + fullDestinationPath);

} catch (IOException exception) {

throw new IrreparableException(

resourcer.getString("error.reason.wrong.writing") + " " + fullDestinationPath);

} finally {

if (writer != null) {

try {

writer.close();

} catch (IOException exception) {

throw new IrreparableException(

resourcer.getString("error.reason.wrong.write.closing") + " " + fullDestinationPath);

}

}

}

}

private static HouseholdAppliancesStorage formStorageFromFile(String sourcePath)

throws FixableException, IrreparableException {

String fileText = Initializer.readTextFromFile(sourcePath);

HouseholdAppliancesStorage resultStorage = new HouseholdAppliancesStorage();

HouseholdAppliance item;

StringBuilder brand = new StringBuilder();

StringBuilder model = new StringBuilder();

StringBuilder type = new StringBuilder();

int flag = 0;

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

if ((fileText.charAt(i) == '\"')) {

flag = 1;

}

if ((flag == 1) && (fileText.charAt(i) == ' ')) {

flag = 2;

}

if (fileText.charAt(i) == '\n') {

flag = 0;

if (HouseholdApplianceType.toType(type.toString().trim()) == null) {

throw new IrreparableException("error.reason.invalid.file.content" + " " + sourcePath);

} else {

item = new HouseholdAppliance(brand.toString().trim(), model.toString().trim(),

HouseholdApplianceType.toType(type.toString().trim()));

resultStorage.addItem(item);

}

type = new StringBuilder();

brand = new StringBuilder();

model = new StringBuilder();

}

if ((fileText.charAt(i) != '\n') && (fileText.charAt(i) != '\"')) {

if (flag == 0) {

type.append(fileText.charAt(i));

} else if (flag == 1) {

brand.append(fileText.charAt(i));

} else if (flag == 2) {

model.append(fileText.charAt(i));

}

}

}

return resultStorage;

}

}

## Класс HouseholdAppliance

package ru.rsreu.verbickaya0605.householdappliances;

public class HouseholdAppliance implements Comparable<HouseholdAppliance> {

private String brand;

private String model;

private HouseholdApplianceType type;

public HouseholdAppliance(String brand, String model, HouseholdApplianceType type) {

this.setBrand(brand.trim());

this.setModel(model.trim());

this.setType(type);

}

public String getBrand() {

return brand;

}

public void setBrand(String brand) {

this.brand = brand;

}

public String getModel() {

return model;

}

public void setModel(String model) {

this.model = model;

}

public HouseholdApplianceType getType() {

return type;

}

public void setType(HouseholdApplianceType type) {

this.type = type;

}

@Override

public int hashCode() {

int code = this.getBrand().length() \* this.getModel().length() + this.getType().toString().length();

return code;

}

@Override

public boolean equals(Object o) {

if (o == this) {

return true;

}

if (!(o instanceof HouseholdAppliance)) {

return false;

}

HouseholdAppliance obj = (HouseholdAppliance) o;

return (this.compareTo(obj) == 0);

}

@Override

public String toString() {

String s = "";

s += this.getType().toString() + " \"" + this.getBrand() + " " + this.getModel() + "\"\n";

return s;

}

@Override

public int compareTo(HouseholdAppliance o) {

if (o == this) {

return 0;

} else {

if (this.getType() == this.getType()) {

if (this.getBrand().equals(o.getBrand())) {

return this.getModel().compareTo(o.getModel());

} else {

return this.getBrand().compareTo(o.getBrand());

}

} else {

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

}

}

}

}

## Класс HouseholdAppliancesStorage

package ru.rsreu.verbickaya0605.householdappliances;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

import ru.rsreu.verbickaya0605.exceptions.FixableException;

public class HouseholdAppliancesStorage implements Comparable<HouseholdAppliancesStorage> {

private static Resourcer resourcer = ProjectResourcer.getInstance();

private HouseholdAppliance[] items;

public HouseholdAppliancesStorage(HouseholdAppliance... args) throws FixableException {

this.items = new HouseholdAppliance[0];

this.addItems(args);

}

public HouseholdAppliance[] getItems() {

return this.items;

}

public void addItems(HouseholdAppliance... args) throws FixableException {

boolean errorFlag = false;

for (HouseholdAppliance item : args) {

if (this.isInItemsBrands(item)) {

errorFlag = true;

} else {

this.addItem(item);

}

}

if (errorFlag) {

throw new FixableException(resourcer.getString("error.reason.adding.invalid.brand"),

resourcer.getString("error.act.adding.invalid.brand"));

}

}

public void addItem(HouseholdAppliance item) {

int newLength = this.getItems().length + 1;

HouseholdAppliance[] newArray = new HouseholdAppliance[newLength];

System.arraycopy(this.getItems(), 0, newArray, 0, this.getItems().length);

newArray[newLength - 1] = item;

this.items = new HouseholdAppliance[newLength];

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

}

public String toStringTable() {

StringBuilder resultString = new StringBuilder("\n");

String separator = "|";

final int len = 20;

for (HouseholdAppliance item : this.getItems()) {

resultString.append(item.getType().toString()

+ HouseholdAppliancesStorage.spasesBeforeLegth(len, item.getType().toString()) + separator);

resultString.append(

item.getBrand() + HouseholdAppliancesStorage.spasesBeforeLegth(len, item.getBrand()) + separator);

resultString.append(item.getModel() + HouseholdAppliancesStorage.spasesBeforeLegth(len, item.getModel())

+ separator + "\n");

}

return resultString.toString();

}

private static String spasesBeforeLegth(int length, String arg) {

StringBuilder resultString = new StringBuilder();

for (int i = 0; i < length - arg.length(); i++) {

resultString.append(" ");

}

return resultString.toString();

}

private boolean isInItemsBrands(HouseholdAppliance someItem) {

boolean result = false;

for (HouseholdAppliance item : this.getItems()) {

if (item.getBrand().equals(someItem.getBrand())) {

result = true;

break;

}

}

return result;

}

@Override

public String toString() {

StringBuilder resultString = new StringBuilder();

for (HouseholdAppliance item : this.getItems()) {

resultString.append(item.toString());

}

return resultString.toString();

}

@Override

public int compareTo(HouseholdAppliancesStorage o) {

if (o.getItems().length > this.getItems().length) {

return -1;

} else if (o.getItems().length < this.getItems().length) {

return 1;

} else {

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

if (o.getItems()[i].compareTo(this.getItems()[i]) != 0) {

return 1;

}

}

return 0;

}

}

}

## Класс MyException

package ru.rsreu.verbickaya0605.exceptions;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

public class MyException extends Exception {

private static final long serialVersionUID = 1L;

private static final Resourcer RESOURSER = ProjectResourcer.getInstance();

private final String errorReason;

public MyException(String errorReason) {

this.errorReason = errorReason;

}

public String getErrorReason() {

return this.errorReason;

}

@Override

public String toString() {

String resultString = RESOURSER.getString("error.label") + " " + this.getErrorReason() + "\n";

return resultString;

}

}