Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Рязанский государственный радиотехнический университет»  
(ФГБОУ ВО «РГРТУ», РГРТУ)

Кафедра «Вычислительная и прикладная математика» (ВПМ)

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 6

по дисциплине

**«ПРОГРАММИРОВАНИЕ НА JAVA»**

Выполнил:

студент группы 643

Паршина Анна Романовна

Проверил:

Пруцков Александр Викторович,

д-р техн. наук, профессор кафедры ВПМ

Рязань 2019

# Задание

Транспортная компания: номер груза, отправитель, тип транспортного средства, перевозящего груз.

# Основные классы, реализующие задание

## Класс Runner

package ru.rsreu.parshina0613;

import java.util.Scanner;

import ru.rsreu.parshina0613.file.FileStructureCreator;

import ru.rsreu.parshina0613.file.FolderStructureCreator;

import ru.rsreu.parshina0613.transportcompany.TransportCompanyInitializer;

public class Runner {

private Runner() {

}

public static void main(String[] args) {

StringBuilder result = new StringBuilder();

FolderStructureCreator folderStructure = new FolderStructureCreator();

result.append(Resourcer.getString("message.directory")).append("\n").append(folderStructure.create())

.append("\n");

FileStructureCreator fileStructure = new FileStructureCreator();

String resultCreateFileData = fileStructure.createFillFileData();

result.append(String.format(Resourcer.getString("message.file"), Resourcer.getString("files.folder.data.name")))

.append("\n").append(resultCreateFileData).append("\n");

if (resultCreateFileData.contains(Resourcer.getString("message.fileWrite"))) {

String resultCopy = fileStructure.copyFileData();

result.append(

String.format(Resourcer.getString("message.copy"), Resourcer.getString("files.folder.data.name"),

Resourcer.getString("files.folder.backup.extension")))

.append("\n").append(resultCopy).append("\n");

if (resultCopy.contains(Resourcer.getString("message.fileCopy"))) {

TransportCompanyInitializer source = new TransportCompanyInitializer();

source.fillCompanies();

result.append(Resourcer.getString("message.arraySource")).append("\n")

.append(TableCreator.getTable(source));

TransportCompanyInitializer copy = new TransportCompanyInitializer();

String resultCopyArray = copy.fillCompaniesFromFile(

System.getProperty("user.dir") + Resourcer.getString("files.folder.backup.extension"));

result.append(String.format(Resourcer.getString("message.fillingArray"),

Resourcer.getString("message.arrayCopy"), Resourcer.getString("files.folder.backup.extension")))

.append("\n").append(resultCopyArray).append("\n");

if (resultCopyArray == Resourcer.getString("message.fillArray")) {

result.append(Resourcer.getString("message.arrayCopy")).append("\n")

.append(TableCreator.getTable(copy));

}

Scanner in = new Scanner(System.in);

System.out.print(result);

result = new StringBuilder();

System.out.print(String.format(Resourcer.getString("messege.question.move"),

Resourcer.getString("files.folder.data.name")));

String resultInput = in.next();

resultInput = resultInput.toUpperCase();

in.nextLine();

in.close();

if (resultInput.contains("Y")) {

String resultMove = fileStructure.moveFileData();

result.append(resultMove).append("\n");

if (resultMove.contains(Resourcer.getString("message.fileMove"))) {

TransportCompanyInitializer move = new TransportCompanyInitializer();

String resultMoveArray = move.fillCompaniesFromFile(

System.getProperty("user.dir") + Resourcer.getString("files.folder.move.data"));

result.append(String.format(Resourcer.getString("message.fillingArray"),

Resourcer.getString("message.arrayMove"),

Resourcer.getString("files.folder.move.data"))).append("\n").append(resultMoveArray)

.append("\n");

if (resultMoveArray == Resourcer.getString("message.fillArray")) {

result.append(Resourcer.getString("message.arrayMove")).append("\n")

.append(TableCreator.getTable(move));

if (move.compereTwoCompany(source)) {

result.append(String.format(Resourcer.getString("message.compereEqual"),

Resourcer.getString("message.arrayMove"),

Resourcer.getString("message.arraySource"))).append("\n");

} else {

result.append(String.format(Resourcer.getString("message.compereNotEqual"),

Resourcer.getString("message.arrayMove"),

Resourcer.getString("message.arraySource"))).append("\n");

}

if (move.compereTwoCompany(copy)) {

result.append(String.format(Resourcer.getString("message.compereEqual"),

Resourcer.getString("message.arrayMove"),

Resourcer.getString("message.arrayCopy"))).append("\n");

} else {

result.append(String.format(Resourcer.getString("message.compereNotEqual"),

Resourcer.getString("message.arrayMove"),

Resourcer.getString("message.arrayCopy"))).append("\n");

}

}

}

}

if (source.compereTwoCompany(copy)) {

result.append(String.format(Resourcer.getString("message.compereEqual"),

Resourcer.getString("message.arraySource"), Resourcer.getString("message.arrayCopy")));

} else {

result.append(String.format(Resourcer.getString("message.compereNotEqual"),

Resourcer.getString("message.arraySource"), Resourcer.getString("message.arrayCopy")));

}

}

}

System.out.print(result);

}

}

## Класс FileStructureCreator

package ru.rsreu.parshina0613.file;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import ru.rsreu.parshina0613.Resourcer;

import ru.rsreu.parshina0613.transportcompany.TransportCompanyInitializer;

public class FileStructureCreator {

private File fileData;

public FileStructureCreator() {

}

private String createFileData() throws FileException, IOException {

String result = "";

fileData = new File(System.getProperty("user.dir") + Resourcer.getString("files.folder.data.name"));

if (fileData.exists()) {

result = String.format(Resourcer.getString("message.fileFormatExists"), fileData.getAbsolutePath());

} else {

if (fileData.createNewFile()) {

result = String.format(Resourcer.getString("message.fileFormatCreated"), fileData.getAbsolutePath());

} else {

throw new FileException(

String.format(Resourcer.getString("message.fileFormatNotCreated"), fileData.getAbsolutePath()));

}

}

return result;

}

private String fillFileData(String data) throws IOException {

String result = Resourcer.getString("message.fileWrite");

FileWriter writer = new FileWriter(fileData);

writer.write(data);

writer.flush();

writer.close();

return result;

}

public String createFillFileData() {

StringBuilder result = new StringBuilder();

try {

result.append(this.createFileData()).append("\n");

TransportCompanyInitializer intializer = new TransportCompanyInitializer();

intializer.fillCompanies();

result.append(this.fillFileData(intializer.toString())).append("\n");

} catch (IOException exception) {

result.append(Resourcer.getString("message.fileWriteException"));

} catch (FileException exception) {

result.append(exception.getMessage());

}

return result.toString();

}

private String copyFileUsingStream(File source, File dest) throws IOException {

String result = Resourcer.getString("message.fileCopy");

InputStream is = null;

OutputStream os = null;

is = new FileInputStream(source);

os = new FileOutputStream(dest);

byte[] buffer = new byte[1024];

int length;

while ((length = is.read(buffer)) > 0) {

os.write(buffer, 0, length);

}

is.close();

os.close();

return result;

}

public String copyFileData() {

StringBuilder result = new StringBuilder();

File copyFile = new File(System.getProperty("user.dir") + Resourcer.getString("files.folder.backup.extension"));

if (fileData.exists()) {

try {

if (!copyFile.createNewFile()) {

copyFile.delete();

copyFile.createNewFile();

}

result.append(

String.format(Resourcer.getString("message.fileFormatCreated"), copyFile.getAbsolutePath()))

.append("\n");

} catch (IOException exception) {

result.append(

String.format(Resourcer.getString("message.fileFormatNotCreated"), copyFile.getAbsolutePath()))

.append("\n");

}

try {

result.append(copyFileUsingStream(fileData, copyFile)).append("\n");

} catch (IOException exception) {

result.append(Resourcer.getString("message.fileCopyException")).append("\n");

}

} else {

result.append(

String.format(Resourcer.getString("message.fileFormatNotCreated"), fileData.getAbsolutePath()))

.append("\n");

}

return result.toString();

}

public String moveFileData() {

StringBuilder result = new StringBuilder();

File moveFile = new File(System.getProperty("user.dir") + Resourcer.getString("files.folder.move.data"));

if (fileData.exists()) {

if (!fileData.renameTo(moveFile)) {

moveFile.delete();

fileData.renameTo(moveFile);

}

result.append(String.format(Resourcer.getString("message.fileFormatCreated"), moveFile.getAbsolutePath()))

.append("\n").append(Resourcer.getString("message.fileMove")).append("\n");

} else {

result.append(

String.format(Resourcer.getString("message.fileFormatNotCreated"), fileData.getAbsolutePath()))

.append("\n");

}

return result.toString();

}

}

## Класс FolderStructureCreator

package ru.rsreu.parshina0613.file;

import java.io.File;

import ru.rsreu.parshina0613.Resourcer;

public class FolderStructureCreator {

public FolderStructureCreator() {

}

private String createDirectories(String path) throws FileException {

String result = "";

final File directories = new File(System.getProperty("user.dir") + path);

if (!directories.exists()) {

if (directories.mkdir()) {

result = String.format(Resourcer.getString("message.directoriesFormatCreated"),

directories.getAbsolutePath());

} else {

throw new FileException(String.format(Resourcer.getString("message.directoriesFormatNotCreated"),

directories.getAbsolutePath()));

}

} else {

result = String.format(Resourcer.getString("message.directoriesFormatExists"),

directories.getAbsolutePath());

}

return result;

}

private String createFolderStructure(String path) {

String result = "";

try {

result = this.createDirectories(path);

} catch (FileException exception) {

result = exception.getMessage();

}

return result;

}

public String create() {

StringBuilder result = new StringBuilder();

result.append(this.createFolderStructure(Resourcer.getString("files.folder.source.name"))).append("\n")

.append(this.createFolderStructure(Resourcer.getString("files.folder.copy.name"))).append("\n")

.append(this.createFolderStructure(Resourcer.getString("files.folder.move.name"))).append("\n");

return result.toString();

}

}

## Класс TransportCompany

package ru.rsreu.parshina0613.transportcompany;

import ru.rsreu.parshina0613.Resourcer;

public class TransportCompany implements Comparable<TransportCompany> {

private int numberCargo;

private String sender;

private VehicleClass vehicle;

public TransportCompany(int namberCargo, String sender, VehicleClass vehicle) {

this.setNumberCargo(namberCargo);

this.setSender(sender);

this.setVehicle(vehicle);

}

public final int getNumberCargo() {

return this.numberCargo;

}

public final void setNumberCargo(int namberCargo) {

this.numberCargo = namberCargo;

}

public final String getSender() {

return this.sender;

}

public final void setSender(String sender) {

this.sender = sender;

}

public final VehicleClass getVehicle() {

return this.vehicle;

}

public final void setVehicle(VehicleClass vehicle) {

this.vehicle = vehicle;

}

@Override

public int compareTo(TransportCompany company) {

return -(this.hashCode() - company.hashCode());

}

@Override

public int hashCode() {

return this.getNumberCargo() + this.vehicle.getCargoCapacity();

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

TransportCompany other = (TransportCompany) obj;

if (this.numberCargo != other.getNumberCargo()) {

return false;

}

if (this.sender != other.getSender()) {

return false;

}

if (this.vehicle != other.getVehicle()) {

return false;

}

return true;

}

@Override

public String toString() {

return String.format(Resourcer.getString("message.formatToFile"), this.numberCargo, this.sender,

this.vehicle.getName(), this.vehicle.getCargoCapacity());

}

}

## Класс TransportCompanyInitializer

package ru.rsreu.parshina0613.transportcompany;

import java.io.FileReader;

import java.io.IOException;

import ru.rsreu.parshina0613.Resourcer;

public class TransportCompanyInitializer {

public static final int COUNT\_COMPANY = 5;

public static final int CARGO1 = 1;

public static final int CARGO2 = 2;

public static final int CARGO3 = 3;

public static final int CARGO4 = 4;

public static final int CARGO5 = 5;

public static final String SENDER1 = "Вася";

public static final String SENDER2 = "Петя";

private TransportCompany[] companies;

public TransportCompanyInitializer() {

}

public TransportCompany[] getCompanies() {

return this.companies;

}

private boolean findCompanyForNumberCargo(int namberCargo) {

for (TransportCompany item : companies) {

if (item.getNumberCargo() == namberCargo) {

return true;

}

}

return false;

}

private int getNamberCargo(int index) {

int namberCargo = 0;

if (index == 0) {

namberCargo = CARGO1;

}

if (index == 1) {

namberCargo = CARGO2;

}

if (index == 2) {

namberCargo = CARGO3;

}

if (index == 3) {

namberCargo = CARGO4;

}

if (index == 4) {

namberCargo = CARGO5;

}

return namberCargo;

}

private String getSender(int index) {

String sender = "";

if (index >= 0 && index < 3) {

sender = SENDER1;

}

if (index >= 3) {

sender = SENDER2;

}

return sender;

}

private VehicleClass getVehicle(int index) {

VehicleClass vehicle = null;

if (index == 0 || index == 1) {

vehicle = VehicleClass.TRUCK;

}

if (index >= 2 || index == 3) {

vehicle = VehicleClass.CARGO\_SHIP;

}

if (index >= 4) {

vehicle = VehicleClass.CARGO\_AIRPLANE;

}

return vehicle;

}

private TransportCompany addCompany(int index) {

TransportCompany company;

int numberCargo = this.getNamberCargo(index);

String sender = this.getSender(index);

VehicleClass vehicle = this.getVehicle(index);

company = new TransportCompany(numberCargo, sender, vehicle);

return company;

}

public String fillCompanies() {

String result = "";

companies = new TransportCompany[COUNT\_COMPANY];

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

this.companies[i] = addCompany(i);

}

return result;

}

private String getCompanyFromFile(String path) throws IOException {

FileReader reader = new FileReader(path);

String buff = "";

int c;

while ((c = reader.read()) != -1) {

buff += (char) c;

}

reader.close();

return buff;

}

public String fillCompaniesFromFile(String path) {

String result = Resourcer.getString("message.fillArray");

try {

String arrayFromFile = this.getCompanyFromFile(path);

String[] company = arrayFromFile.split("\n");

companies = new TransportCompany[company.length];

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

String[] fields = company[i].split(" | ");

VehicleClass vehicle = null;

String vehiclefields = fields[4].trim();

if (i >= 2) {

vehiclefields += " " + fields[5].trim();

}

if (vehiclefields.contains(Resourcer.getString("message.truck"))) {

vehicle = VehicleClass.TRUCK;

}

if (vehiclefields.contains(Resourcer.getString("message.cargoShip"))) {

vehicle = VehicleClass.CARGO\_SHIP;

}

if (vehiclefields.contains(Resourcer.getString("message.cargoAirplane"))) {

vehicle = VehicleClass.CARGO\_AIRPLANE;

}

companies[i] = new TransportCompany(Integer.parseInt(fields[0].trim()), fields[2].trim(), vehicle);

}

} catch (IOException exception) {

return Resourcer.getString("message.fillArrayException");

} catch (NumberFormatException exception) {

return Resourcer.getString("message.fillArrayException");

}

return result;

}

public boolean compereTwoCompany(TransportCompanyInitializer other) {

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

if (this.companies[i].equals(other.getCompanies()[i])) {

return false;

}

}

return true;

}

@Override

public String toString() {

StringBuilder result = new StringBuilder();

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

result.append(this.companies[i]).append("\n");

}

return result.toString();

}

}