Минобрнауки России | РГРТУ | Кафедра ВПМ

Курс «ПРОГРАММИРОВАНИЕ НА JAVA»

**Отчет о практической работе №**7

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.verbickaya0705.basic;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

import ru.rsreu.verbickaya0705.householdappliances.basic.HouseholdApplianceType;

public class Runner {

private Runner() {

}

private static Resourcer resourcer = ProjectResourcer.getInstance();

public static void main(String[] args) {

StringBuilder resultString = new StringBuilder();

// initializing the list and sorting by two ways

Initializer.initialize();

resultString.append(resourcer.getString("list.origin") + "\n");

resultString.append(Initializer.list.toStringTable());

resultString.append("\n" + resourcer.getString("list.sorted.default") + "\n");

resultString.append(Initializer.list.sortedByDefault().toStringTable());

resultString.append("\n" + resourcer.getString("list.sorted.model.type") + "\n");

resultString.append(Initializer.list.sortedByModelType().toStringTable());

// getting values of the second parameter without repeats

resultString.append("\n" + resourcer.getString("list.unique.model.values") + "\n");

resultString.append(Initializer.list.getUniqueModels().toString());

// deleting elements with some value of the third parameter

HouseholdApplianceType type = HouseholdApplianceType.iron;

resultString.append("\n\n" + resourcer.getString("list.deleted.type") + ": " + type.toString() + "\n");

resultString.append(Initializer.list.deletedByType(type).toStringTable());

// searching elements by value of the first parameter

String brand1 = "REDMOND";

String brand2 = "SomeBrand";

resultString.append("\n" + resourcer.getString("list.element.of.brand") + ": " + brand1 + "\n");

resultString.append(Initializer.list.searchByBrand(brand1.trim()).toString());

resultString.append("\n\n" + resourcer.getString("list.element.of.brand") + ": " + brand2 + "\n");

resultString.append(Initializer.list.searchByBrand(brand2.trim()).toString());

// output

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

}

}

## Класс NullHouseholdAppliance

package ru.rsreu.verbickaya0705.householdappliances.other;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

public class NullHouseholdAppliance implements Appliance {

private static Resourcer resourcer = ProjectResourcer.getInstance();

@Override

public String toString() {

return NullHouseholdAppliance.resourcer.getString("appliance.value.of.max.is.empty");

}

}

## Класс HouseholdAppliance

package ru.rsreu.verbickaya0705.householdappliances.basic;

import ru.rsreu.verbickaya0705.householdappliances.other.Appliance;

import ru.rsreu.verbickaya0705.householdappliances.other.ComparableApplianceType;

public class HouseholdAppliance implements Comparable<HouseholdAppliance>, Appliance {

private String brand;

private String model;

private ComparableApplianceType 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 ComparableApplianceType 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() + "\"";

return s;

}

@Override

public int compareTo(HouseholdAppliance o) {

if (o == this) {

return 0;

} else {

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

return this.compareByModelAndType(o);

} else {

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

}

}

}

public int compareByModelAndType(HouseholdAppliance o) {

if (o == this) {

return 0;

} else {

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

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

} else {

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

}

}

}

}

## Класс HouseholdAppliancesContainer

package ru.rsreu.verbickaya0705.householdappliances.basic;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.HashMap;

import java.util.HashSet;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Set;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

import ru.rsreu.verbickaya0705.householdappliances.comparators.HouseholdApplianceDefaultComparator;

import ru.rsreu.verbickaya0705.householdappliances.comparators.HouseholdApplianceModelTypeComparator;

import ru.rsreu.verbickaya0705.householdappliances.other.Appliance;

import ru.rsreu.verbickaya0705.householdappliances.other.NullHouseholdAppliance;

import ru.rsreu.verbickaya0705.utilities.StringOfTableFormer;

public class HouseholdAppliancesContainer {

private static Resourcer resourcer = ProjectResourcer.getInstance();

private List<HouseholdAppliance> items;

public HouseholdAppliancesContainer(HouseholdAppliance... args) {

this.items = new ArrayList<HouseholdAppliance>();

this.addItems(args);

}

public HouseholdAppliancesContainer(List<HouseholdAppliance> args) {

this.items = new ArrayList<HouseholdAppliance>();

this.addItems(args);

}

public void addItems(HouseholdAppliance... args) {

for (HouseholdAppliance item : args) {

if (!this.isInItemsBrands(item)) {

this.items.add(item);

}

}

}

public void addItems(List<HouseholdAppliance> args) {

Iterator<HouseholdAppliance> iterator = args.iterator();

while (iterator.hasNext()) {

HouseholdAppliance item = iterator.next();

if (!this.isInItemsBrands(item)) {

this.items.add(item);

}

}

}

public HouseholdAppliancesContainer sortedByDefault() {

Comparator<HouseholdAppliance> comparator = new HouseholdApplianceDefaultComparator();

List<HouseholdAppliance> list = new ArrayList<HouseholdAppliance>();

list = this.items;

Collections.sort(list, comparator);

return new HouseholdAppliancesContainer(list);

}

public HouseholdAppliancesContainer sortedByModelType() {

Comparator<HouseholdAppliance> comparator = new HouseholdApplianceModelTypeComparator();

List<HouseholdAppliance> list = new ArrayList<HouseholdAppliance>();

list = this.items;

Collections.sort(list, comparator);

return new HouseholdAppliancesContainer(list);

}

public Set<String> getUniqueModels() {

Iterator<HouseholdAppliance> iterator = this.items.iterator();

Set<String> models = new HashSet<String>();

while (iterator.hasNext()) {

HouseholdAppliance item = iterator.next();

models.add(item.getModel());

}

return models;

}

public HouseholdAppliancesContainer deletedByType(HouseholdApplianceType type) {

List<HouseholdAppliance> list = new ArrayList<HouseholdAppliance>();

list = this.items;

Iterator<HouseholdAppliance> iterator = list.iterator();

while (iterator.hasNext()) {

HouseholdAppliance item = iterator.next();

if (item.getType().compareTo(type) == 0) {

iterator.remove();

}

}

return new HouseholdAppliancesContainer(list);

}

public Appliance searchByBrand(String brand) {

Map<String, HouseholdAppliance> map = this.getMapOfBrandsAndAppliances();

if (map.containsKey(brand)) {

return map.get(brand);

} else {

return new NullHouseholdAppliance();

}

}

public String toStringTable() {

final int columnWidth = 22;

String header = StringOfTableFormer.formStringOfTable(columnWidth, resourcer.getString("appliance.type"),

resourcer.getString("appliance.brand"), resourcer.getString("appliance.model")) + "\n";

StringBuilder resultString = new StringBuilder(header);

Iterator<HouseholdAppliance> iterator = this.items.iterator();

while (iterator.hasNext()) {

HouseholdAppliance item = iterator.next();

resultString.append(StringOfTableFormer.formStringOfTable(columnWidth, item.getType().toString(),

item.getBrand(), item.getModel()) + "\n");

}

return resultString.toString();

}

@Override

public String toString() {

StringBuilder resultString = new StringBuilder();

Iterator<HouseholdAppliance> iterator = this.items.iterator();

while (iterator.hasNext()) {

HouseholdAppliance string = iterator.next();

resultString.append(string.toString() + "\n");

}

return resultString.toString();

}

private Map<String, HouseholdAppliance> getMapOfBrandsAndAppliances() {

Map<String, HouseholdAppliance> resultMap = new HashMap<String, HouseholdAppliance>();

Iterator<HouseholdAppliance> iterator = this.items.iterator();

while (iterator.hasNext()) {

HouseholdAppliance item = iterator.next();

resultMap.put(item.getBrand(), item);

}

return resultMap;

}

private boolean isInItemsBrands(HouseholdAppliance someItem) {

Iterator<HouseholdAppliance> iterator = this.items.iterator();

while (iterator.hasNext()) {

HouseholdAppliance item = iterator.next();

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

return true;

}

}

return false;

}

}

## Класс StringOfTableFormer

package ru.rsreu.verbickaya0705.utilities;

public class StringOfTableFormer {

private StringOfTableFormer() {

}

private static final String SEPARATOR = "|";

public static String formStringOfTable(int columnWidth, String... args) {

StringBuilder resultString = new StringBuilder(StringOfTableFormer.SEPARATOR);

for (String item : args) {

resultString.append(StringOfTableFormer.completeBySpaces(columnWidth, item));

resultString.append(StringOfTableFormer.SEPARATOR);

}

return resultString.toString();

}

private static String completeBySpaces(int spacesCount, String string) {

StringBuilder resultString = new StringBuilder(string);

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

resultString.append(" ");

}

return resultString.toString();

}

}