Министерство образования Республики Беларусь

УО «Витебский государственный технологический университет»

Кафедра «Информационные системы и автоматизация производства»

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА

по дисциплине «Объектно-ориентированное программирование»

Вариант 4

Выполнила: студентка группы 2ИТ-8

Авласенко Анна Сергеевна

Проверил: старший преподаватель

Деркаченко Павел Григорьевич

Витебск, 2021

**ЗАДАНИЕ 1**

В решениях применять функциональные интерфейсы и лямбда-выражения. Там, где это возможно, использовать Stream API.

Напишите программу, которая считывает текстовый файл в список. Имя файла вводится в консоли. Программа должна напечатать слова из файла, отсортированные по длине. Если длина одинакова, использовать сортировку по алфавиту. Сортировка должна корректно поддерживать кириллицу (буква «ё» должна идти после «е» и перед «ж»).

**Код программы:**

package com;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.text.ParseException;

import java.text.RuleBasedCollator;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.Scanner;

import static java.util.ResourceBundle.getBundle;

public class Main {

public static void sorting(String name) {

try (BufferedReader br = new BufferedReader(new FileReader(name))) {

String str;

var arr = new ArrayList<String>();

while ((str = br.readLine()) != null) {

arr.add(str);

}

var rule = new RuleBasedCollator(getBundle("Delimiters").getString("ORDER")); arr.stream().sorted(rule::compare).sorted(Comparator.comparing(String::length)).forEach(System.out::println);

} catch (IOException | ParseException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

System.out.println("Введите имя файла:");

Scanner s = new Scanner(System.in);

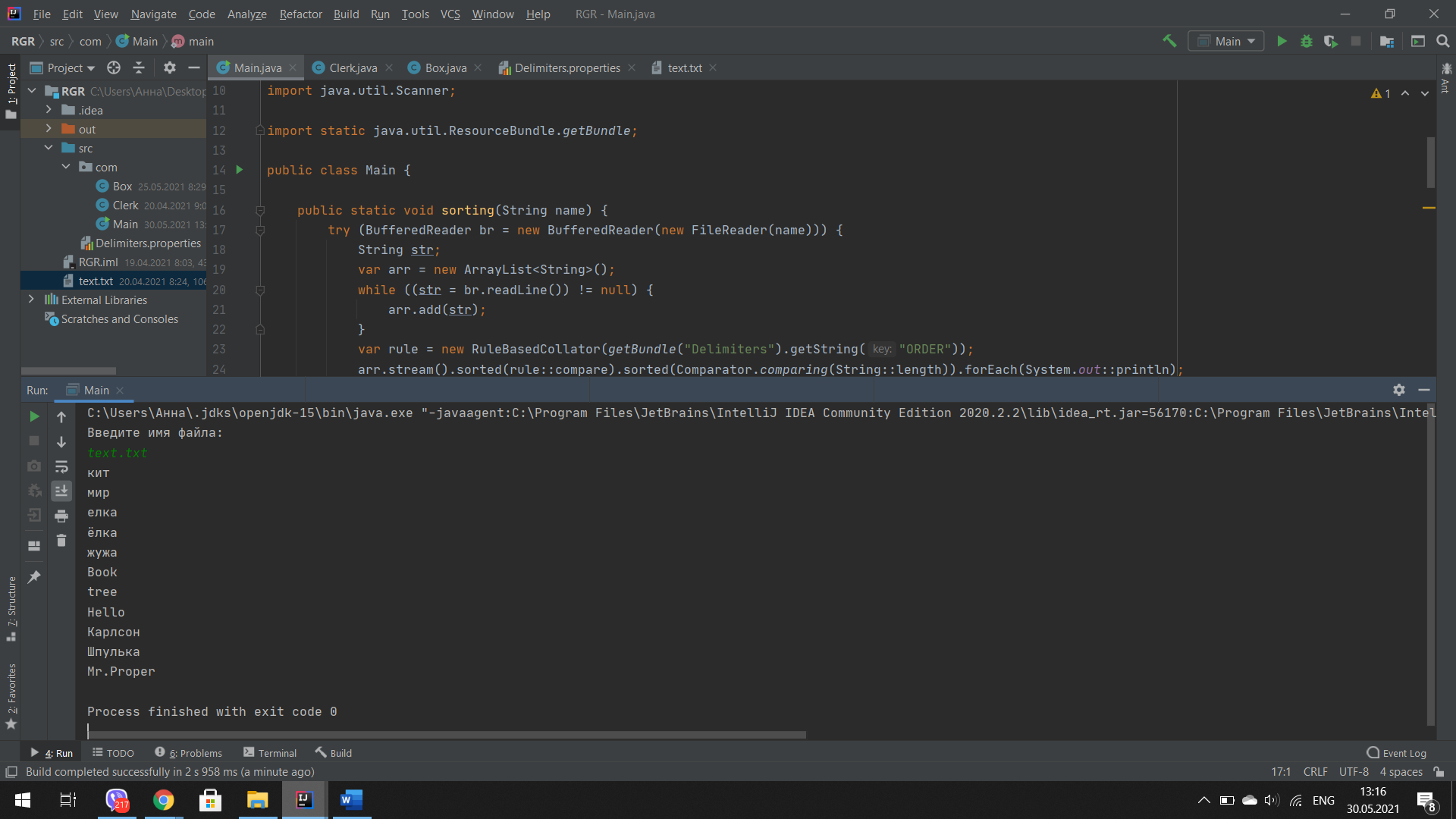
sorting(s.nextLine());

}

}

**Delimiters:**

ORDER=< а, А < б, Б < в, В < г, Г < д, Д < е, Е < ё, Ё < ж, Ж < з, З < и, И < й, Й < к, К < л, Л < м, М < н, Н < о, О < п, П < р, Р < с, С < т, Т < у, У < ф, Ф < х, Х < ц, Ц < ч, Ч < ш, Ш < щ, Щ < ъ, Ъ < ы, Ы < ь, Ь < э, Э < ю, Ю < я, Я



**ЗАДАНИЕ 2**

В решениях применять синхронизацию только там, где это оправдано. При необходимости использовать функционал пакета java.util.concurrent и его подпакетов.

Создать класс "Ящик", который вмещает 100 формуляров. Ящик одновременно наполняют 3 клерка, каждый формуляром своего вида. По заполнении ящика вывести на экран, сколько формуляров каждого вида находится в ящике.

package com;

public class Main {

public static void main(String[] args) throws InterruptedException {

Box box = new Box();

Clerk cl1 = new Clerk(box, " Clerk1");

Clerk cl2 = new Clerk(box, " Clerk2");

Clerk cl3 = new Clerk(box, " Clerk3");

cl1.start();

cl2.start();

cl3.start();

cl1.join();

cl2.join();

cl3.join();

System.out.println(cl1.getType() + "-" + cl1.getCount());

System.out.println(cl2.getType() + "-" + cl2.getCount());

System.out.println(cl3.getType() + "-" + cl3.getCount());

}

}

**Класс Clerk:**

package com;

import java.util.concurrent.ThreadLocalRandom;

public class Clerk extends Thread {

private Box box;

private String type;

private int count = 0;

public Clerk(Box box, String type) {

this.box = box;

this.type = type;

}

public String getType() {

return type;

}

public int getCount() {

return count;

}

@Override

public void run() {

int amt;

while (box.getCnt() < 100) {

amt = ThreadLocalRandom.current().nextInt(0, 3);

try {

box.put(amt);

count += amt;

} catch (IllegalAccessException e) {

System.err.println("Коробка заполнена!");

}

}

}

}

**Класс Box:**

package com;

public class Box {

private volatile int cnt = 0;

public synchronized void put(int amount) throws IllegalAccessException {

if (cnt + amount > 100) throw new IllegalAccessException();

else cnt += amount;

}

public int getCnt() {

return cnt;

}

}

