Звіт

Автор: Водолазський Микола Анатолійович

КІТ-118а

Лабораторна робота №10

ОБРОБКА ПАРАМЕТРИЗОВАНИХ КОНТЕЙНЕРІВ

***Мета:***

* Розширення функціональності параметризованих класів.

**Вимоги:**

Використовуючи програму рішення завдання лабораторної роботи №9:

1. Розробити параметризовані методи (Generic Methods) для обробки колекцій об’єктів

згідно прикладної задачі.

2. Продемонструвати розроблену функціональність (створення, управління та обробку

власних контейнерів) в діалоговому та автоматичному режимах.

Автоматичний режим виконання програми задається параметром командного

рядка -auto. Наприклад, java ClassName -auto .

В автоматичному режимі діалог з користувачем відсутній, необхідні данні

генеруються, або зчитуються з файлу.

3. Забороняється використання алгоритмів з Java Collections Framework

ПРИКЛАДНА ЗАДАЧА:

Кадрове агентство. Сортування за назвою фірми, за назвою запропонованої

спеціальності, за вказаною освітою.

ОПИС ПРОГРАМИ

***2.1 Опис змінних:***

LinkedContainer<SecondCreate> stringLinked = new LinkedContainer<>();// об’єкт параметризованого контейнера

Scanner scan = new Scanner(System.in); // змінна для активування зчитування з консолі

***2.2 Ієрархія та структура класів.***

Main class – головний клас. Містить метод main(точку входу у програму) та методи по роботі з програмою для реалізації індивідуального завдання.

interface iLinked - інтерфейс контенеру

class SecondCreate - клас прикладної задачі кадрового агенства

class linkedContainer - параметризований клас-контейнер, котрий зберігає інформацію агенства

**ТЕКСТ ПРОГРАМИ**

File Main.java:

import javax.xml.parsers.ParserConfigurationException;  
import javax.xml.transform.TransformerException;  
import java.io.\*;  
import java.util.Arrays;  
import java.util.Comparator;  
import java.util.LinkedList;  
import java.util.Scanner;  
import java.io.\*;  
import java.util.concurrent.\*;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
  
public class Main implements Serializable {  
  
 public static void fileRead() throws IOException, ParserConfigurationException, TransformerException, InterruptedException, ExecutionException, TimeoutException {  
 linkedContainer<SecondCreate> linkedContainer = new linkedContainer<SecondCreate>();  
  
  
 File file = new File("file.txt");  
  
 Scanner scanner = new Scanner(file);  
  
 String line = scanner.nextLine();  
 String[] array = line.split(" ");  
  
 String company = null;  
 String specialisation = null;  
 String workingConditions = null;  
 int payment = 0;  
 int workingExperience = 0;  
 String education = null;  
 String English = null;  
 String Licence = null;  
  
  
 for (int i = 0; i < array.length; i++) {  
 company = array[0].toString();  
 specialisation = array[1].toString();  
 workingConditions = array[2].toString();  
 payment = Integer.*parseInt*(array[3]);  
 workingExperience = Integer.*parseInt*(array[4]);  
 education = array[5].toString();  
 Licence = array[6].toString();  
 English = array[7].toString();  
 }  
  
  
 *regCheck*(company,specialisation,workingConditions,payment,workingExperience,education,Licence, English);  
  
  
  
 SecondCreate firstWorker = new SecondCreate(company, specialisation, workingConditions, payment, workingExperience, education,Licence,English);  
 SecondCreate secondCreate = new SecondCreate("epam", "teacher", "good", 100, 1, "none","no","no");  
 SecondCreate thirdWorker = new SecondCreate("globalLogic","teacher","10.00-19.00",300,11,"magistry","yes","yes");  
  
 SecondCreate[] arr = {firstWorker, secondCreate,thirdWorker};  
  
  
 System.*out*.println("SORT BY COMPANY NAME");  
 Arrays.*sort*(arr);  
  
 for (SecondCreate tmp : arr) {  
 System.*out*.println(tmp);  
 }  
 comparatorC comparatorC = new comparatorC();  
  
  
 System.*out*.println("SORT BY Specialisation");  
 Arrays.*sort*(arr,comparatorC);  
  
 for(SecondCreate tmpss : arr)  
 {  
 System.*out*.println(tmpss);  
 }  
  
  
 System.*out*.println("SORT BY EDUCATION");  
  
 secondComparator secondComparator = new secondComparator();  
 Arrays.*sort*(arr,secondComparator);  
  
 for(SecondCreate tmpp : arr)  
 {  
 System.*out*.println(tmpp);  
 }  
  
 linkedContainer.addFirst(firstWorker);  
 // linkedContainer.addLast(secondCreate);  
 // linkedContainer.addLast(thirdWorker);  
 System.*out*.println("Container size");  
 System.*out*.println(linkedContainer.size());  
  
  
 *serialize*(linkedContainer);  
  
 /\*System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");  
  
  
 for(int i=0;i<1000;i++) {  
 linkedContainer.addLast(secondCreate);  
 }  
  
  
  
  
 \*//\*Threads threads = new Threads(linkedContainer);  
 ExecutorService executorService = Executors.newFixedThreadPool(1);  
 executorService.submit(threads);  
 executorService.shutdown();\*//\*  
  
  
 // Начало потока с огранечением по времени  
  
 long begin = System.currentTimeMillis();  
  
 Threads.MyTread1 myTread1 = new Threads.MyTread1();  
 myTread1.set(linkedContainer);  
 Threads.MyThread2 myThread2 = new Threads.MyThread2();  
 myThread2.set(linkedContainer);  
 Threads.MyThread3 myThread3 = new Threads.MyThread3();  
 myThread3.set(linkedContainer);  
  
  
  
 ExecutorService executorService = Executors.newFixedThreadPool(3);  
 Future<Boolean> future = executorService.submit(myTread1);  
 Future<Boolean> future1 = executorService.submit(myThread2);  
 Future<Boolean> future2 = executorService.submit(myThread3);  
 future.get();  
 future1.get();  
 future2.get();  
 future.cancel(true);  
 future1.cancel(true);  
 future2.cancel(true);  
 executorService.shutdown();  
 try {  
// В следующей строке первым параметром идет кол-во секунд сколько будет выполняться поток  
 future.get(20, TimeUnit.SECONDS);  
 } catch (TimeoutException ex) {  
 future.cancel(true);  
 }  
 executorService.shutdown();  
  
 long finnish = System.currentTimeMillis();  
  
 long result = finnish - begin;  
 System.out.println("Time parralell threads was working " + result + " Milliseconds");  
  
  
 // Конец потока с ограничением по времени  
  
 // Два потока без ограничения по времени  
  
 long start = System.currentTimeMillis();  
  
 FirstThread threadFirst = new FirstThread(linkedContainer);  
 ExecutorService executorServiceFirst = Executors.newFixedThreadPool(1);  
 executorServiceFirst.submit(threadFirst);  
 executorServiceFirst.shutdown();  
  
 SecondThead threadSecond = new SecondThead(linkedContainer);  
 ExecutorService executorServiceSecond = Executors.newFixedThreadPool(1);  
 executorServiceSecond.submit(threadSecond);  
 executorServiceSecond.shutdown();  
  
  
 ThirdThread threadThird = new ThirdThread(linkedContainer);  
 ExecutorService executorServiceThird = Executors.newFixedThreadPool(1);  
 executorServiceThird.submit(threadThird);  
 executorServiceThird.shutdown();  
  
 long stop = System.currentTimeMillis();  
 long res = stop - start;  
  
 System.out.println("Time consecutive threads was working = " + res + " milliseconds");  
  
  
 linkedContainer.addLast(secondCreate);  
 linkedContainer.addLast(thirdWorker);\*/  
  
 // конец потоков без ограничения по времени  
  
 // Поиск элемента соответствующего заданым критериям  
 *textsort*(linkedContainer);  
  
 }  
  
 public static void textsort(linkedContainer<SecondCreate> linkedContainer)  
 {  
 for (SecondCreate t : linkedContainer)  
 {  
 Pattern p1 = Pattern.*compile*("teacher", Pattern.*CASE\_INSENSITIVE*);  
 Matcher m1 = p1.matcher(t.getSpecialisation());  
 if (m1.find()) {  
 if (t.getWorkingExperience() >= 10) {  
 Pattern p2 = Pattern.*compile*("yes", Pattern.*CASE\_INSENSITIVE*);  
 Matcher m2 = p2.matcher(t.getEnglish());  
 if (m2.find()) {  
 Pattern p3 = Pattern.*compile*("yes", Pattern.*CASE\_INSENSITIVE*);  
 Matcher m3 = p3.matcher(t.getLicence());  
 if (m3.find()) {  
 System.*out*.println(t);  
 }  
 }  
 }  
 }  
 }  
  
 }  
  
 public static void serialize(linkedContainer<SecondCreate> linkedContainer) throws IOException, ParserConfigurationException, TransformerException{  
 XmlRead xmlRead = new XmlRead();  
 XmlWrite xmlWrite = new XmlWrite();  
 xmlWrite.*write*(linkedContainer,"XML.xml");  
  
 linkedContainer<SecondCreate> newXml = XmlRead.*read*("XML.xml");  
  
 for(SecondCreate t : newXml )  
 {  
 System.*out*.println(t);  
 }  
  
 }  
  
  
 public static void regCheck(String company, String specialisation, String workingConditions, int payment, int workingExperience, String education, String License, String English)  
 {  
 if(company.matches("[a-zA-Z0-9]\*") == true)  
 {  
 System.*out*.println("OK");  
 }  
 else  
 {  
 System.*out*.println("Rename company");  
 }  
 if(specialisation.matches("[0-9]\*"))  
 {  
 System.*out*.println("OK");  
 }else{System.*out*.println("Rename specialisation");}  
 }  
  
  
  
 public static void manual() throws IOException, ClassNotFoundException, FileNotFoundException, TransformerException, ParserConfigurationException {  
 System.*out*.println("U have chosen manual mode");  
 int choose;  
  
 linkedContainer<SecondCreate> linkedContainer = new linkedContainer<>();  
 SecondCreate SecondCreate1 = null;  
  
 do{  
 System.*out*.println("Choose action ");  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("1. Create new element");  
 System.*out*.println("2. Add elem ");  
 System.*out*.println("3. Clear container ");  
 System.*out*.println("4. Convert to Array ");  
 System.*out*.println("5. Serialize ");  
 System.*out*.println("6. Deserialize ");  
 System.*out*.println("7. Xml serialize");  
 System.*out*.println("8. Xml deserialize");  
 choose = in.nextInt();  
 switch (choose) {  
 case 1:  
 Scanner din = new Scanner(System.*in*);  
 Scanner cin = new Scanner(System.*in*);  
 System.*out*.println("Enter company name");  
 String company = din.nextLine();  
 if(company.matches("[a-zA-Z0-9]\*")==true)  
 {  
 System.*out*.println("");  
 }else{System.*out*.println("NOT ok RENAME");company = din.nextLine();}  
 System.*out*.println("Enter specialisation");  
 String specialisation=din.nextLine();  
 if(specialisation.matches("[0-9]\*")==true)  
 {  
 System.*out*.println("ok");  
 }else{ System.*out*.println("NOT ok RENAME"); specialisation = din.nextLine();}  
 System.*out*.println("Enter working Conditions");  
 String workingConditions=din.nextLine();  
 System.*out*.println("Enter payment");  
 int payment=cin.nextInt();  
 System.*out*.println("Enter working Experience");  
 int workingExperience=cin.nextInt();  
 System.*out*.println("Enter education");  
 String education=din.nextLine();  
 System.*out*.println("Enter knowledge of English");  
 String English = cin.nextLine();  
 System.*out*.println("Enter driving licence");  
 String License = cin.nextLine();  
 SecondCreate1 = new SecondCreate(company,specialisation,workingConditions,payment,workingExperience,education,License,English);  
 break;  
  
 case 2:  
 System.*out*.println(linkedContainer.size());  
 linkedContainer.addLast(SecondCreate1);  
 System.*out*.println(linkedContainer.size());  
  
 for(SecondCreate tmp : linkedContainer)  
 {  
 System.*out*.println(tmp);  
 }  
  
 break;  
 case 3:  
 linkedContainer.clean();  
 System.*out*.println(linkedContainer.size());  
 break;  
 case 4:  
 Object []arr = linkedContainer.toArray().toArray();  
 for(int i=0; i<linkedContainer.size();i++)  
 {  
 System.*out*.println(arr[i]);  
 }  
 break;  
 case 5:  
 ObjectOutputStream objectOutputStream = new ObjectOutputStream(new FileOutputStream("store.txt"));  
 objectOutputStream.writeObject(linkedContainer);  
 objectOutputStream.close();  
  
 break;  
 case 6:  
 /\* ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream("store.txt"));  
 linkedContainer<SecondCreate> newContainer = (linkedContainer<SecondCreate>)objectInputStream.readObject();  
  
 for (SecondCreate t : newContainer) {  
 System.out.println(t);  
 }\*/  
  
 break;  
 case 7:  
 /\*XmlWrite xxmlWrite = new XmlWrite();  
  
 xxmlWrite.write(linkedContainer,"XML.xml");\*/  
 break;  
 case 8:  
 /\*linkedContainer<SecondCreate> newXml = XmlRead.read("XML.xml");  
  
 for(SecondCreate t : newXml )  
 {  
 System.out.println(t);  
 }\*/  
  
 break;  
  
 default:  
 break;  
 }}while(choose!=9);  
  
 }  
  
  
  
 public static void main(String args[]) throws IOException, ParserConfigurationException, TransformerException, ClassNotFoundException, InterruptedException, ExecutionException, TimeoutException {  
  
  
 if(args[0].equals("-auto"))  
 {  
  
 System.*out*.println("U chose auto mode.");  
 System.*out*.println("1. Reading from file");  
 *fileRead*();  
  
 }else  
 {  
 *manual*();  
 }  
 }  
}

SecondCreate.java :

import java.io.Serializable;  
  
  
public class SecondCreate implements java.lang.Comparable<SecondCreate>, Serializable {  
  
 private String company;  
 private String specialisation;  
 private String workingConditions;  
 private int payment;  
 private int workingExperience;  
 private String education;  
 private String Licence;  
 private String English;  
  
 public SecondCreate(String company,String specialisation,String workingConditions,int payment,int workingExperience,String education,String Licence,String English)  
 {  
 this.company=company;  
 this.specialisation=specialisation;  
 this.workingConditions=workingConditions;  
 this.payment=payment;  
 this.workingExperience=workingExperience;  
 this.education=education;  
 this.Licence=Licence;  
 this.English=English;  
 }  
  
 public int getPayment()  
 {  
 return payment;  
 }  
  
 public String getSpecialisation()  
 {  
 return specialisation;  
 }  
  
 public String getEducation()  
 {  
 return education;  
 }  
  
 public String getCompany()  
 {  
 return company;  
 }  
  
 public String getWorkingConditions()  
 {  
 return workingConditions;  
 }  
  
 public int getWorkingExperience()  
 {  
 return workingExperience;  
 }  
  
 public String getLicence()  
 {  
 return Licence;  
 }  
  
 public String getEnglish()  
 {  
 return English;  
 }  
  
 @Override  
 public String toString() {  
 return "created object{" + "\n" +  
 "company name =" + company.toString() + "\n" +  
 "specialisation =" + specialisation + "\n" +  
 "workingConditions =" + workingConditions +"\n" +  
 "payment =" + payment+ "\n" +  
 "workingExperience =" + workingExperience + "\n"+  
 "education =" + education + "\n" +  
 "Licence = " + Licence + "\n" +  
 "English =" + English + "\n" +  
 '}' + "\n";  
 }  
  
 @Override  
 public int compareTo(SecondCreate o) {  
 SecondCreate entry = (SecondCreate) o;  
  
 int tmp = company.compareTo(entry.company);  
 // this.payment - ((SecondCreate)o).payment;  
 return tmp;  
 }  
  
}

iLinked.java:

package ua.khpi.oop.vasilchenko09.MyList;

import java.io.Serializable;

public interface Linked<T> extends DescendingIterator<T>, Serializable, Iterable<T> {

void addLast(T obj);

void addFirst(T obj);

int size();

T getElementByIndex(int index);

void saveAll();

void saveRec();

void add(T obj);

void clear();

boolean notEmpty();

void readRec();

void readAll();

**}**

**ВАРІАНТИ ВИКОРИСТАННЯ**

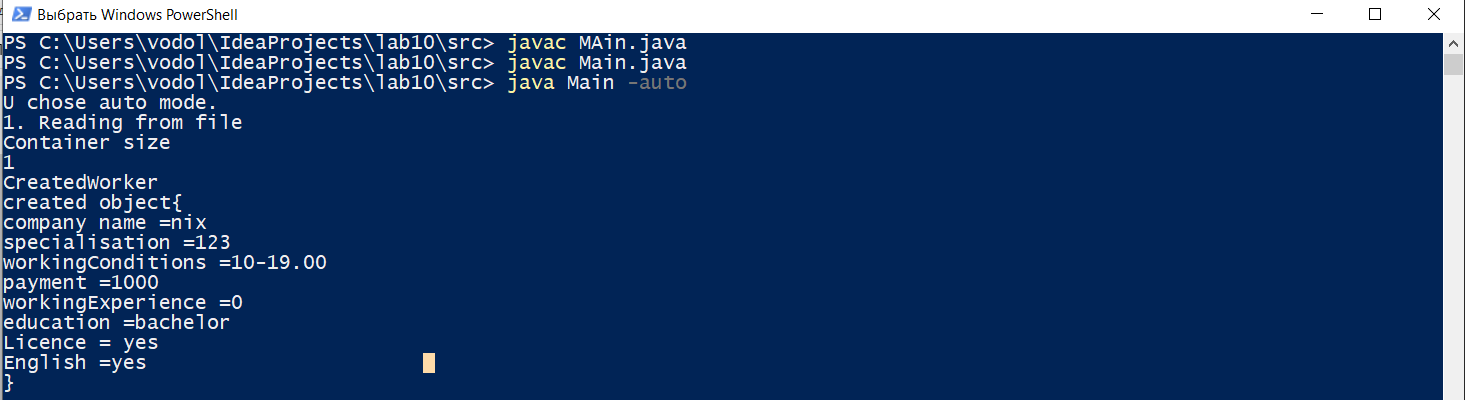
****

Рис. 10.1 – Результат роботи програми

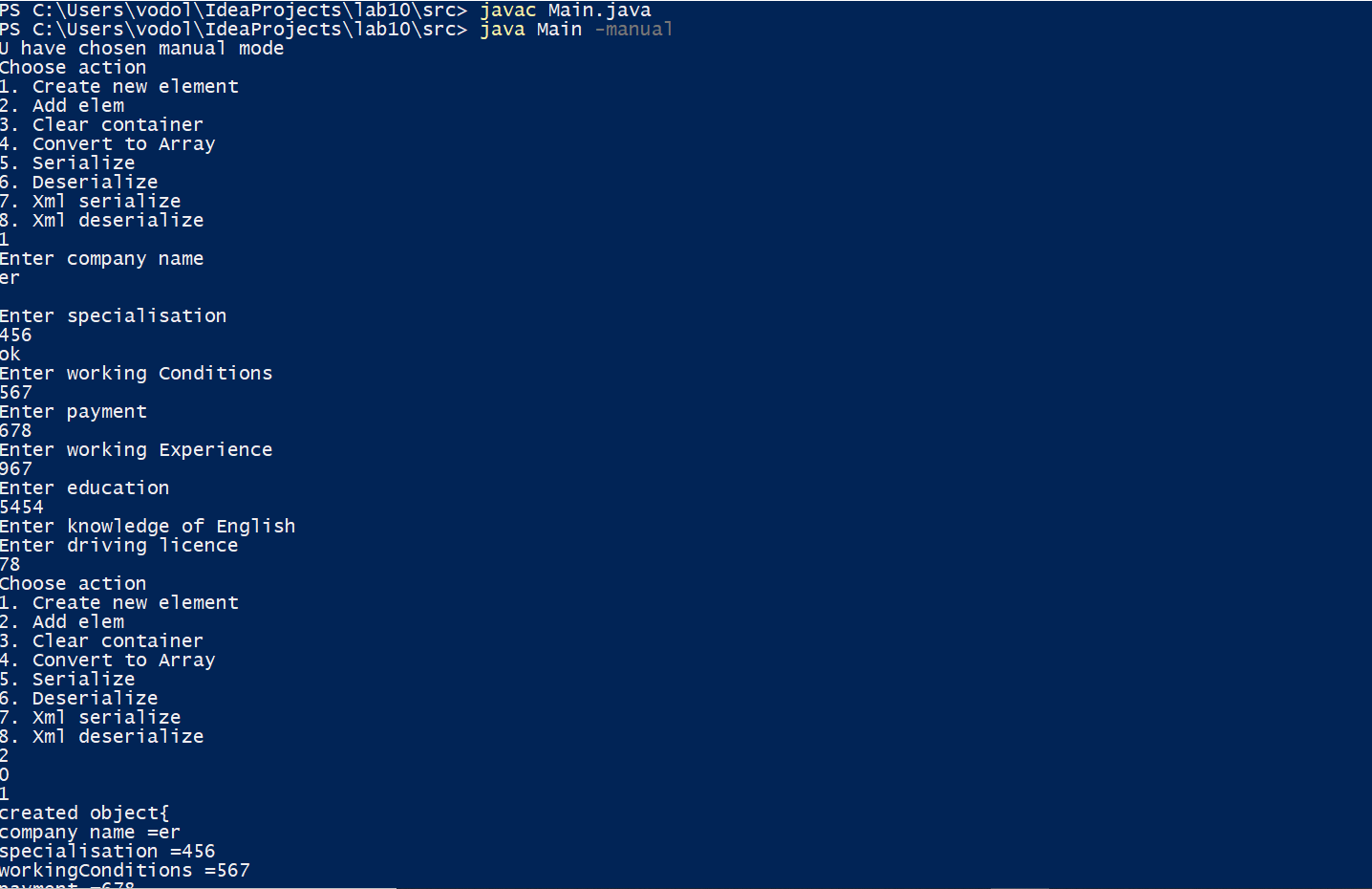


Рис. 10.2 – Результат роботи програми

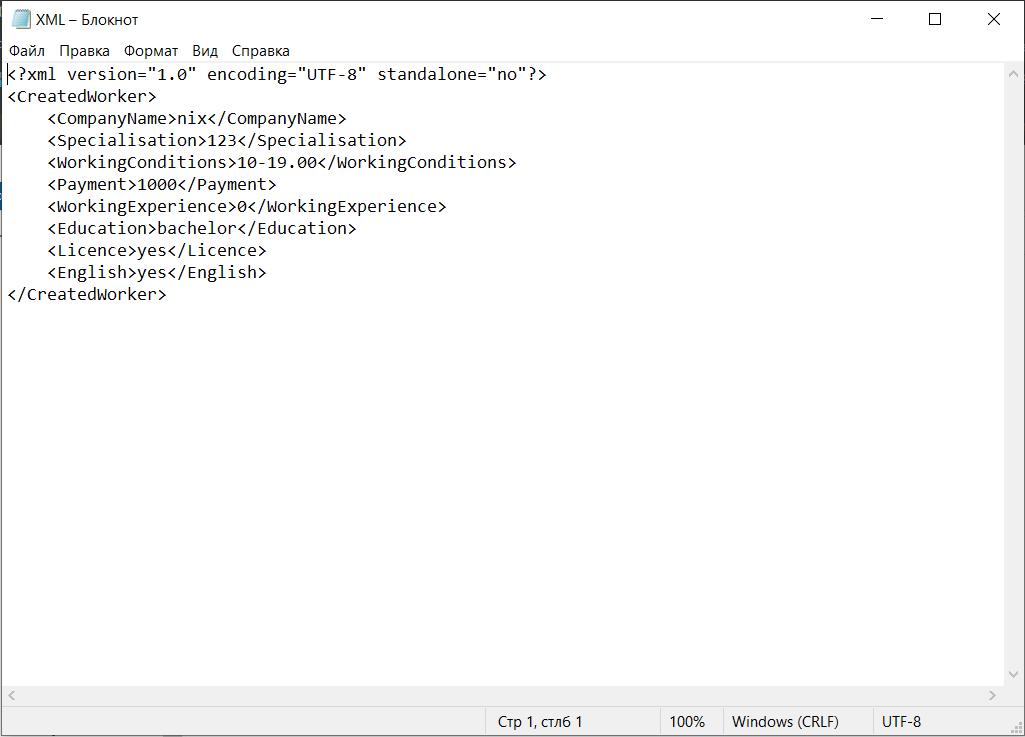


Рис. 10.3 – Результат роботи програми

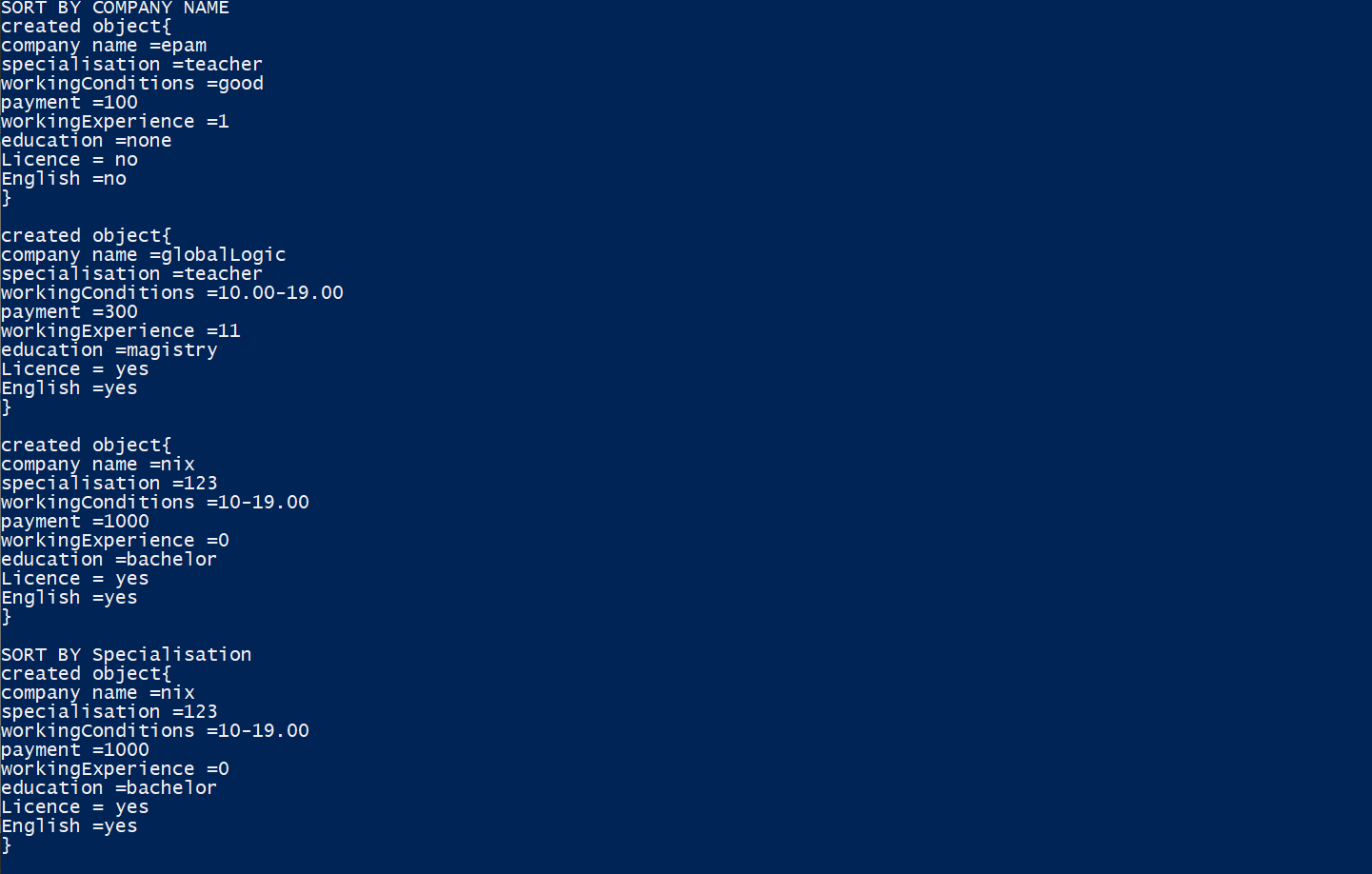


Рис. 10.4 – Сортування контейнеру за назвою компанії, назвою спеціальності та за вказаною освітою.

Програму можна використовувати задля створення бази даних. Завдяки параметризації зв’язного списка, базу даних можна використати для будь-яких типів даних. Переважно у нашому варіанті - кадрове агенство, в якому представляються різноманітні вакансії. Программа може виконуватися в двох режимах. Автоматичному і ручному режимах керування. До попередньої лабораторної роботи була додана можливість сортування контейнеру за різними полями.

**ВИСНОВКИ**

При виконанні лабораторної роботи набуто практичних навичок щодо розробки параметризованих класів. Завдяки цієї можливості в JAVA, можливо створювати колекції та інші класи на основі будь-яких типів. Також навчився обробляти параметризовані контейнери, навчився сортувати параметризовані контейнери за різними полями.