Лабораторна робота №13

Паралельне виконання. Багатопоточність

Мета: Ознайомлення з моделлю потоків Java. Організація паралельного виконання декількох частин програми.

1 ВИМОГИ

1. Використовуючи програми рішень попередніх задач, продемонструвати можливість паралельної обробки елементів контейнера: створити не менше трьох додаткових потоків, на яких викликати відповідні методи обробки контейнера.

2. Забезпечити можливість встановлення користувачем максимального часу виконання (таймаута) при закінченні якого обробка повинна припинятися незалежно від того знайдений кінцевий результат чи ні.

3. Для паралельної обробки використовувати алгоритми, що не змінюють початкову колекцію.

4. Кількість елементів контейнера повинна бути досить велика, складність алгоритмів обробки колекції повинна бути зіставна, а час виконання приблизно однаковий, наприклад:

1) пошук мінімуму або максимуму;
2) обчислення середнього значення або суми;
3) підрахунок елементів, що задовольняють деякій умові;
4) відбір за заданим критерієм;
5) власний варіант, що відповідає обраній прикладної області.

Розробник

- П.І.Б: Абдулаєв І. З.
- Группа: КІТ-119в
- Варіант: 1

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП:

Scanner inInt, inStr = new Scanner(System.in) – для введення обраних опцій користувачем з клавіатури;

XMLEncoder encoder = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("filename"));

encoder.writeObject(recuitingAgency); – нестандартна серіалізація;

XMLDecoder decoder = new XMLDecoder(new BufferedInputStream(new FileInputStream("filename")));

container = (MyContainer<Challanger>) decoder.readObject(); – нестандартна десеріалізація;

oos.writeObject(container);

container = (MyContainer<Challanger>) ois.readObject(); – стандартна десеріалізація;

Pattern pattern = Pattern.compile() – компілює регулярний вираз у шаблон;

Matcher matcher = pattern.matcher(information); – створює matcher, який буде відповідати даному вводу для цього шаблону.

2.2 Ієрархія та структура класів

Було створено класи Main (головний клас програми), Challanger (клас, що містить всі поля та методи прикладної області «Кадрове агенство»), MyConatainer (клас-контейнер), Node (клас-покажчик на елемент) та 3 класи, що реалізують інтерфейс Comparator для сортування за певними критеріями. Клас MyThread (реалізує інтерфейс Runnable для роботи з потоками).

2.3 Важливі фрагменти програми

Class Main

```
package ua.khpi.oop.abdulaev13;

import java.beans.XMLDecoder;
import java.beans.XMLEncoder;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```
import ua.khpi.oop.abdulaev07.Challanger;
import ua.khpi.oop.abdulaev07.DemandsToWork;
import ua.khpi.oop.abdulaev07.WorkExperience;
import ua.khpi.oop.abdulaev10.MyContainer;

public class Main {

    public static void main(String[] args) {
        MyContainer<Challanger> recruitingAgency = new
MyContainer<Challanger>();

        for (String str : args) {
            if(str.equals("-a") || str.equals("-auto")) {
                recruitingAgency = auto(recruitingAgency);
                return;
            }
        }
        recruitingAgency = menu(recruitingAgency);
    }

    private static MyContainer<Challanger> auto(MyContainer<Challanger>
recruitingAgency) {
        System.out.println("Adding elements...");

        File file = new File("recruitingAgency11.txt");

        try {
            String education;
            int day;
            int month;
            int year;
            String specializationPrevious;
            int experience;
            String specializationNext;
            int minSalary;
            String conditions;
            Scanner reader = new Scanner(file);
            while(reader.hasNextLine()) {
                String data = reader.nextLine();
                Pattern pattern = Pattern.compile("((\\w+(|\\s))*,\\s([1-
9]|[12]\\d|3[01])\\.([1-9]|1[012])\\.((19|20)\\d{2}),\\s" +
                        "(\\w+.)+,\\s([0-9]|[1-6][0-9]),\\s(\\w+.)+,\\s([1-
9]\\d{3,}),\\s(\\w+(\\.|\\s)(\\s|))+)");
                Matcher matcher = pattern.matcher(data);
                if(matcher.matches()) {
                    String[] information = data.split(",\\s");
                    education = information[0];
                    specializationPrevious = information[2];
                    experience = Integer.parseInt(information[3]);
                    specializationNext = information[4];
                    minSalary = Integer.parseInt(information[5]);
                    conditions = information[6];
                    String[] date = information[1].split("\\.");
                    day = Integer.parseInt(date[0]);
                    month = Integer.parseInt(date[1]);
                    year = Integer.parseInt(date[2]);

                    int id = recruitingAgency.getSize();
```

```
                    WorkExperience workExperienceAdd = new
WorkExperience(specializationPrevious, experience);
                    DemandsToWork demandsToWorkAdd = new
DemandsToWork(specializationNext,minSalary,conditions);
                    Challanger challangerAdd = new
Challanger(id++,education,day,month,year,workExperienceAdd,demandsToWorkAdd);
                    recruitingAgency.add(challangerAdd);
                }
            }
            reader.close();
            System.out.println("Adding was end.\n");
        } catch (FileNotFoundException e){
            e.printStackTrace();
        }

        System.out.println("List in Recruiting Agency:\n");
        if(recruitingAgency.getSize() > 0) {
            for(var element : recruitingAgency) {
                element.print();
            }
        }
        else {
            System.out.println("The recruiting agency is empty!\n");
        }

        task(recruitingAgency);

        int orderSort = 1;

        recruitingAgency.sort(new workExperienceComparator(), orderSort);
        System.out.println("Data sorted by work experience");

        System.out.println("List in Recruiting Agency:\n");
        if(recruitingAgency.getSize() > 0) {
            for(var element : recruitingAgency) {
                element.print();
            }
        }

        return recruitingAgency;
    }

    private static MyContainer<Challanger> menu(MyContainer<Challanger>
recruitingAgency) {
        boolean endprog = false;
        Scanner inInt = new Scanner(System.in);
        Scanner inStr = new Scanner(System.in);
        int menu;
        int menuSort;
        int orderSort;
        int menuSerialization;
        int menuDeserialization;

        while(!endprog)
        {
            System.out.println("1. Show all challanger");
            System.out.println("2. Add challanger");
            System.out.println("3. Delete chellanger");
```

```
            System.out.println("4. Clear list");
            System.out.println("5. Is empty recruiting agency?");
            System.out.println("6. Sort data");
            System.out.println("7. Serialize data");
            System.out.println("8. Deserialize data");
            System.out.println("9. Task");
            System.out.println("10. Thread task");
            System.out.println("0. Exit");
            System.out.print("Enter option: ");
            try
            {
                menu =  inInt.nextInt();
            }
            catch(java.util.InputMismatchException e)
            {
                System.out.println("Error! Ошибка ввода.");
                endprog = true;
                menu = 0;
            }
            System.out.println();
            switch(menu)
            {
                case 1:
                    if(recruitingAgency.getSize() > 0) {
                        for(var element : recruitingAgency) {
                            element.print();
                        }
                    }
                    else {
                        System.out.println("The recruiting agency is
empty!\n");
                    }
                    break;
                case 2:
                    String education;
                    int day;
                    int month;
                    int year;
                    String specializationPrevious;
                    int experience;
                    String specializationNext;
                    int minSalary;
                    String conditions;

                    Pattern patternEducation = Pattern.compile("(\\w+.)+");
                    Pattern patternDay = Pattern.compile("([1-
9]|[12]\\d|3[01])");
                    Pattern patternMonth = Pattern.compile("([1-9]|1[012])");
                    Pattern patternYear = Pattern.compile("(19|20)\\d{2}");
                    Pattern patternSpeñialization =
Pattern.compile("(\\w+.)+");
                    Pattern patternExperience = Pattern.compile("[0-9]|[1-
6][0-9]");
                    Pattern patternMinSalary = Pattern.compile("(^[1-
9]\\d{3,})");
                    Pattern patternConditions =
Pattern.compile("(\\w+(\\.|\\s)(\\s|))+");

                    System.out.println("Enter education of challanger: ");
```

```
                    try {
                        education = inStr.nextLine();
                        education = stringRegexCheck(education,
patternEducation);
                    }catch(java.util.InputMismatchException e) {
                        System.out.println("Error! Incorect input!");
                        break;
                    }

                    System.out.println("Enter day of dismissal: ");
                    try {
                        day = inInt.nextInt();
                        day = intRegexCheck(day, patternDay);
                    } catch(java.util.InputMismatchException e) {
                        System.out.println("Error! Incorect input!");
                        break;
                    }

                    System.out.println("Enter month of dismissal: ");
                    try {
                        month = inInt.nextInt();
                        month = intRegexCheck(month, patternMonth);
                    } catch(java.util.InputMismatchException e) {
                        System.out.println("Error! Incorect input!");
                        break;
                    }

                    System.out.println("Enter year of dismissal: ");
                    try {
                        year = inInt.nextInt();
                        year = intRegexCheck(year, patternYear);
                    } catch(java.util.InputMismatchException e) {
                        System.out.println("Error! Incorect input!");
                        break;
                    }

                    System.out.println("Enter pervious job: ");
                    try {
                        specializationPrevious = inStr.nextLine();
                        specializationPrevious =
stringRegexCheck(specializationPrevious, patternSpeñialization);
                    } catch(java.util.InputMismatchException e) {
                        System.out.println("Error! Incorect input!");
                        break;
                    }

                    System.out.println("Enter experience of working: ");
                    try {
                        experience = inInt.nextInt();
                        experience = intRegexCheck(experience,
patternExperience);
                    } catch(java.util.InputMismatchException e){
                        System.out.println("Error! Incorect input!");
                        break;
                    }

                    System.out.println("Enter next job: ");
                    try {
                        specializationNext = inStr.nextLine();
```

```
                        specializationNext =
stringRegexCheck(specializationNext, patternSpeñialization);
                    } catch(java.util.InputMismatchException e) {
                        System.out.println("Error! Incorect input!");
                        break;
                    }

                    System.out.println("Enter min salary: ");
                    try {
                        minSalary = inInt.nextInt();
                        minSalary = intRegexCheck(minSalary,
patternMinSalary);
                    }catch (java.util.InputMismatchException e) {
                        System.out.println("Error! Incorect input!");
                        break;
                    }

                    System.out.println("Enter whishes to the next job: ");
                    try {
                        conditions = inStr.nextLine();
                        conditions = stringRegexCheck(conditions,
patternConditions);
                    } catch(java.util.InputMismatchException e){
                        System.out.println("Error! Incorect input!");
                        break;
                    }
                    int id = recruitingAgency.getSize();

                    WorkExperience workExperienceAdd = new
WorkExperience(specializationPrevious, experience);
                    DemandsToWork demandsToWorkAdd = new
DemandsToWork(specializationNext,minSalary,conditions);
                    Challanger challangerAdd = new
Challanger(id++,education,day,month,year,workExperienceAdd,demandsToWorkAdd);
                    recruitingAgency.add(challangerAdd);
                    break;
                case 3:
                    System.out.println("Enter ID to delete: ");
                    int delete = inInt.nextInt();
                    boolean isExist = false;
                    if(recruitingAgency.getSize() > 0) {
                        for(var element : recruitingAgency) {
                            if(element.getRegistrationNum() == delete) {
                                isExist = true;
                            }
                        }
                        if(isExist) {
                            if(recruitingAgency.delete(delete))
                                System.out.println("Challanger was deleted
successfully.");
                            else
                                System.out.println("Error! Wrong ID.");
                        }
                        else
                            System.out.println("Error! Wrong ID.");
                    }
                    break;
                case 4:
                    recruitingAgency.clear();
```

```
                    System.out.println("RecruitingAgency is empty now.\n");
                    break;
                case 5:
                    if(recruitingAgency.isEmpty())
                        System.out.println("Recruiting agency is empty.\n");
                    else
                        System.out.println("Recruiting agency is not
empty.");
                    break;
                case 6:
                    System.out.println("1. Sort by Registration Number");
                    System.out.println("2. Sort by work experience");
                    System.out.println("3. Sort by demand to min salary");
                    System.out.println("4. Return to menu");
                    System.out.println("Enter option: ");
                    try
                    {
                        menuSort =  inInt.nextInt();
                    }
                    catch(java.util.InputMismatchException e)
                    {
                        System.out.println("Error! Ошибка ввода.");
                        break;
                    }
                    System.out.println();
                    System.out.println("How to sort data?");
                    System.out.println("1. Asc");
                    System.out.println("2. Desc");
                    System.out.println("Enter option: ");
                    try
                    {
                        orderSort =  inInt.nextInt();
                    }
                    catch(java.util.InputMismatchException e)
                    {
                        System.out.println("Error! Ошибка ввода.");
                        break;
                    }
                    switch(menuSort) {
                        case 1:
                            recruitingAgency.sort(new idComparator(),
orderSort);
                            System.out.println("Data sorted by Registration
Number\n");
                            break;
                        case 2:
                            recruitingAgency.sort(new
workExperienceComparator(), orderSort);
                            System.out.println("Data sorted by work
experience\n");
                            break;
                        case 3:
                            recruitingAgency.sort(new minSalazyComparator(),
orderSort);
                            System.out.println("Data sorted by demand to min
salary");
                            break;
                        case 4:
```

```
                            break;
                        default:
                            System.out.println("Error! Wrong num in Sort
menu.");
                            break;
                    }
                    break;
                case 7:
                    String filenameSerialization;
                    String filenameXML;

                    System.out.println("1. Serialization");
                    System.out.println("2. XML serialization");
                    System.out.println("0. Exit serialization");
                    try
                    {
                        menuSerialization =  inInt.nextInt();
                    }
                    catch(java.util.InputMismatchException e)
                    {
                        System.out.println("Error! Ошибка ввода.");
                        menuSerialization = 0;
                    }
                    switch(menuSerialization)
                    {
                        case 1:
                            System.out.println("\nEnter file name: ");
                            filenameSerialization = inStr.nextLine();
                            if (filenameSerialization.indexOf(".ser") == -1)
{
                                filenameSerialization += ".ser";
                            }
                            try(ObjectOutputStream oos = new
ObjectOutputStream(new BufferedOutputStream(new FileOutputStream
(filenameSerialization)))){
                                oos.writeObject(recruitingAgency);
                                System.out.println("Serialization
successful.");
                            } catch (Exception e){
                                System.out.println(e.getMessage());
                            }
                            break;
                        case 2:
                            System.out.print("Enter XML filename: ");
                            filenameXML = inStr.nextLine();
                            if (filenameXML.indexOf(".xml") == -1)
                                filenameXML += ".xml";
                            try(XMLEncoder encoder = new XMLEncoder(new
BufferedOutputStream(new FileOutputStream (filenameXML)))){
                                encoder.writeObject(recruitingAgency);
                                System.out.println("Serialization
successful.");
                            } catch (Exception e){
                                System.out.println(e.getMessage());
                            }
                            break;
                        case 0:
                            break;
                        default:
```

```
                            System.out.println("Error! Wrong num in menu.");
                            break;
                    }
                    break;
                case 8:
                    String filenameDeserialization;

                    System.out.println("1. Deserialization");
                    System.out.println("2. XML deserialization");
                    System.out.println("0. Exit deserialization");
                    try
                    {
                        menuDeserialization =  inInt.nextInt();
                    }
                    catch(java.util.InputMismatchException e)
                    {
                        System.out.println("Error! Ошибка ввода.");
                        menuDeserialization = 0;
                    }
                    switch(menuDeserialization)
                    {
                        case 1:
                            System.out.println("\nEnter file name: ");
                            filenameDeserialization = inStr.nextLine();
                            if (filenameDeserialization.indexOf(".ser") == -
1) {
                                filenameDeserialization += ".ser";
                            }
                            try(ObjectInputStream ois = new
ObjectInputStream(new BufferedInputStream(new FileInputStream
(filenameDeserialization)))){
                                recruitingAgency.clear();
                                recruitingAgency = (MyContainer<Challanger>)
ois.readObject();
                                System.out.println("Deserialization
successful.");
                            } catch (Exception e){
                                System.out.println(e.getMessage());
                            }
                            break;
                        case 2:
                            System.out.print("Enter XML filename: ");
                            filenameDeserialization = inStr.nextLine();
                            if (filenameDeserialization.indexOf(".xml") == -
1)
                                filenameDeserialization += ".xml";
                            try(XMLDecoder decoder = new XMLDecoder(new
BufferedInputStream(new FileInputStream (filenameDeserialization)))){
                                recruitingAgency.clear();
                                recruitingAgency = (MyContainer<Challanger>)
decoder.readObject();
                                System.out.println("Deserialization
successful.");
                            } catch (Exception e){
                                System.out.println(e.getMessage());
                            }
                            break;
                        case 0:
                            break;
```

```
                        default:
                            System.out.println("Error! Wrong num in menu.");
                            break;
                    }
                    break;
                case 9:
                    task(recruitingAgency);
                    break;
                case 10:
                    int time = -1;
                    int timeSet = 0;
                    MyThread[] thread = new MyThread[4];
                    System.out.println("Adding elements...");
                    MyContainer<Challanger> container = new
MyContainer<Challanger>();

                    File file = new File("recruitingAgency11.txt");
                    String education1 = null;
                    int day1 = 0;
                    int month1 = 0;
                    int year1 = 0;
                    String specializationPrevious1 = null;
                    int experience1 = 0;
                    String specializationNext1 = null;
                    int minSalary1 = 0;
                    String conditions1 = null;
                    int id1 = 0;
                    String education2 = null;
                    int day2 = 0;
                    int month2 = 0;
                    int year2 = 0;
                    String specializationPrevious2 = null;
                    int experience2 = 0;
                    String specializationNext2 = null;
                    int minSalary2 = 0;
                    String conditions2 = null;
                    int id2 =0;
                    try {
                        Scanner reader = new Scanner(file);
                        while(reader.hasNextLine()) {
                            String data = reader.nextLine();
                            String data1 = reader.nextLine();
                            Pattern pattern =
Pattern.compile("((\\w+(|\\s))*,\\s([1-9]|[12]\\d|3[01])\\.([1-
9]|1[012])\\.((19|20)\\d{2}),\\s" +
                                            "(\\w+.)+,\\s([0-9]|[1-6][0-
9]),\\s(\\w+.)+,\\s([1-9]\\d{3,}),\\s(\\w+(\\.|\\s)(\\s|))+)");
                            Matcher matcher = pattern.matcher(data);
                            if(matcher.matches()) {
                                String[] information = data.split(",\\s");
                                education1 = information[0];
                                specializationPrevious1 = information[2];
                                experience1 =
Integer.parseInt(information[3]);
                                specializationNext1 = information[4];
                                minSalary1 =
Integer.parseInt(information[5]);
                                conditions1 = information[6];
                                String[] date1 = information[1].split("\\.");
```

```
                                day1 = Integer.parseInt(date1[0]);
                                month1 = Integer.parseInt(date1[1]);
                                year1 = Integer.parseInt(date1[2]);
                            }
                            Matcher matcher1 = pattern.matcher(data1);
                            if(matcher1.matches()) {
                                String[] information1 = data1.split(",\\s");
                                education2 = information1[0];
                                specializationPrevious2 = information1[2];
                                experience2 =
Integer.parseInt(information1[3]);
                                specializationNext2 = information1[4];
                                minSalary2 =
Integer.parseInt(information1[5]);
                                conditions2 = information1[6];
                                String[] date2 =
information1[1].split("\\.");
                                day2 = Integer.parseInt(date2[0]);
                                month2 = Integer.parseInt(date2[1]);
                                year2 = Integer.parseInt(date2[2]);
                            }
                        }
                        reader.close();
                        System.out.println("Adding was end.\n");
                    } catch (FileNotFoundException e){
                        e.printStackTrace();
                    }
                    for(int i = 0; 20000 > i; i++) {
                        id1 = container.getSize();
                        WorkExperience workExperienceAdd1 = new
WorkExperience(specializationPrevious1, experience1);
                        DemandsToWork demandsToWorkAdd1 = new
DemandsToWork(specializationNext1,minSalary1,conditions1);
                        Challanger challangerAdd1 = new
Challanger(id1++,education1,day1,month1,year1,workExperienceAdd1,demandsToWor
kAdd1);
                        container.add(challangerAdd1);
                        id2 = container.getSize();
                        WorkExperience workExperienceAdd2 = new
WorkExperience(specializationPrevious2, experience2);
                        DemandsToWork demandsToWorkAdd2 = new
DemandsToWork(specializationNext2,minSalary2,conditions2);
                        Challanger challangerAdd2 = new
Challanger(id2++,education2,day2,month2,year2,workExperienceAdd2,demandsToWor
kAdd2);
                        container.add(challangerAdd2);
                    }
                    System.out.println("Adding was end.");

                    System.out.println("Want to set a maximum lead time?");
                    System.out.println("1. Yes");
                    System.out.println("2. No");
                    System.out.print("Enter option: ");
                    timeSet = inInt.nextInt();
                    if(timeSet == 1)
                    {
                        System.out.print("Enter the time in milliseconds: ");
                        time = inInt.nextInt();
                    }
```

```
                    try
                    {
                        for(int i = 0; 4 > i; i++)
                        {
                            thread[i] = new MyThread(container, "Thread " + i
);
                            thread[i].thread.start();
                        }
                        if(time > 0)
                        {
                            Thread.currentThread().sleep(time);
                            for(int i = 0; 4 > i; i++)
                                thread[i].disable();
                        }
                        for(int i = 0; 4 > i; i++)
                            thread[i].thread.join();
                    }
                    catch(InterruptedException ex)
                    {
                        System.out.println("Thread has been interrupted.");
                    }

                    container.clear();
                    time = -1;
                    System.out.println();
                    break;
                case 0:
                    endprog = true;
                    inInt.close();
                    inStr.close();
                    break;
                default:
                    System.out.println("Error! Wrong num in menu.");
                    break;
            }
        }
        return recruitingAgency;
    }
    public static int intRegexCheck(int value, Pattern pattern)
    {
        Matcher matcher;
        Scanner in = new Scanner(System.in);
        boolean ready = false;
        do
        {
            matcher = pattern.matcher(Integer.toString(value));
            if(!matcher.matches())
            {
                System.out.println("You've entered the wrong data. Try
again:");
                value = in.nextInt();
            }
            else
                ready = true;
        }
        while(!ready);
        return value;
    }
```

```
    public static String stringRegexCheck(String value, Pattern pattern)
    {
        Matcher matcher;
        Scanner in = new Scanner(System.in);
        boolean ready = false;
        do
        {
            matcher = pattern.matcher(value);
            if(!matcher.matches())
            {
                System.out.println("You've entered the wrong data. Try
again:");
                value = in.nextLine();
            }
            else
                ready = true;
        }
        while(!ready);
        return value;
    }

    public static void task(MyContainer<Challanger> recruitingAgency) {
        String conditions;
        String prevJob;
        Pattern patternManager = Pattern.compile(".*(M|m)anager.*");
        Pattern patternNot = Pattern.compile(".*(N|n)ot.*");
        Pattern patternBuisnessTrip = Pattern.compile(".*(B|b)uisness
trip.*");
        MyContainer<Challanger> task = new MyContainer<Challanger>();

        if(recruitingAgency.getSize() > 0) {
            for(var element : recruitingAgency) {
                conditions = element.getDemandsToWork().getConditions();
                prevJob = element.getWorkExperience().getSpecialization();
                Matcher matcher = patternManager.matcher(prevJob);
                if(matcher.matches()) {
                    Matcher matcherNot = patternNot.matcher(conditions);
                    if(matcherNot.matches()) {
                        Matcher matcherBuisnessTrip =
patternBuisnessTrip.matcher(conditions);
                        if(matcherBuisnessTrip.matches()) {
                            task.add(element);
                        }
                    }
                }
            }
        }
        if(task.getSize() > 0) {
            System.out.println("\nChallangers with wishes to dose not have a
buiness trip:\n");
            for(var challanger : task) {
                challanger.print();
            }
            System.out.println();
        }
        else {
            System.out.println("\nChallangers without wishes to dose not have
a buisness trip.\n");
        }
```

```
    }
}
```

Class MyContainer

```
package ua.khpi.oop.abdulaev13;

import java.io.Serializable;
import java.util.Comparator;
import java.util.Iterator;
import java.util.NoSuchElementException;

import ua.khpi.oop.abdulaev07.Challanger;
import ua.khpi.oop.abdulaev10.Node;


public class MyContainer<T> implements Iterable<T>, Serializable {
    private static final long serialVersionUID = 1487028470983100792L;

    public Node<T> head;
    private int size;

    public MyContainer() {
        super();
    }


    public int getSize() {
        return size;
    }
    public void setSize(int size) {
        this.size = size;
    }

    public T getElement(int id) {
        if(id < 0 || id > size) {
            System.out.println("Error! Wrong ID.");
            return null;
        }
        Node<T> temp = head;
        for(int i = 0; id > i; i++)    {
            temp = temp.next;
        }
        return temp.element;
    }

    public void add(T element) {
        Node<T> tmp = new Node<T>();

        if(head == null) {
            head = new Node<T>(element);
        }
        else {
            tmp = head;
            while(tmp.next != null) {
                tmp = tmp.next;
            }
            tmp.next = new Node<T>(element);
```

```
        }
        size++;
    }

    public boolean delete(int id) {
        Node<T> tmp = head;

        if(head != null) {
            if(id == 0) {
                head = head.next;
            }
            else {
                for(int i = 0; id-1 > i; i++) {
                    tmp= tmp.next;
                }
                if(tmp.next != null) {
                    tmp.next = tmp.next.next;
                }
                else
                    tmp.next = null;
                size--;
            }
            return true;
        }
        else {
            System.out.println("Container is empty!");
            return false;
        }
    }

    public void clear() {
        head = null;
        size = 0;
    }

    public Object[] toArray() {
        Object[] array = new Object[size];
        for(int i = 0; size > i; i++) {
            array[i] = getElement(i);
        }
        return array;
    }

    public String toString() {
        StringBuilder str = new StringBuilder();
        for(T element : this) {
            str.append(element + "\n");
        }
        return str.toString();
    }

    public boolean isEmpty() {
        if(size == 0)
            return true;
        else
            return false;
    }

    public Iterator<T> iterator() {
```

```
        return new Iterator<T>(){
            int index = 0;
            boolean check = false;

            @Override
            public boolean hasNext() {
                return size > index;
            }

            @Override
            public T next() {
                if(index != size) {
                    check = true;
                    return getElement(index++);
                }
                else
                    throw new NoSuchElementException();
            }

            @Override
            public void remove() {
                if(check) {
                    MyContainer.this.delete(index - 1);
                    check = false;
                }
            }

        };
    }

    public void sort (Comparator<T> comp, int order) {
        Object[] array = this.toArray();
        Object temp;
        boolean check;

        if (order == 1) {
            do {
                check = false;
                for(int i = 0; size - 1 > i; i++) {
                    if(comp.compare((T)array[i],(T)array[i+1]) == 1) {
                        temp = array[i];
                        array[i] = array[i + 1];
                        array[i + 1] = temp;
                        check = true;
                    }
                }
            } while (check == true);
        }
        else {
            do {
                check = false;
                for(int i = 0; size - 1 > i; i++) {
                    if(comp.compare((T)array[i],(T)array[i+1]) == -1) {
                        temp = array[i+1];
                        array[i+1] = array[i];
                        array[i] = temp;
                        check = true;
                    }
                }
```

```
            } while (check == true);
        }

        this.clear();
        for(Object obj : array) {
            this.add((T)obj);
        }
    }
}

class idComparator implements Comparator<Challanger>{
    @Override
    public int compare(Challanger o1, Challanger o2) {
        if(o1.getRegistrationNum() > o2.getRegistrationNum())
            return 1;
        else if (o1.getRegistrationNum() < o2.getRegistrationNum())
            return -1;
        else
            return 0;
    }
}

class workExperienceComparator implements Comparator<Challanger>{
    @Override
    public int compare(Challanger o1, Challanger o2) {
        if(o1.getWorkExperience().getExperience() >
o2.getWorkExperience().getExperience())
            return 1;
        else if (o1.getWorkExperience().getExperience() <
o2.getWorkExperience().getExperience())
            return -1;
        else
            return 0;
    }
}

class minSalazyComparator implements Comparator<Challanger>{
    @Override
    public int compare(Challanger o1, Challanger o2) {
        if(o1.getDemandsToWork().getMinSalary() >
o2.getDemandsToWork().getMinSalary())
            return 1;
        else if (o1.getDemandsToWork().getMinSalary() <
o2.getDemandsToWork().getMinSalary())
            return -1;
        else
            return 0;
    }

}
```

Class MyThread

```java
package ua.khpi.oop.abdulaev13;

import ua.khpi.oop.abdulaev07.Challanger;
import ua.khpi.oop.abdulaev10.MyContainer;

public class MyThread implements Runnable {
    private MyContainer<Challanger> container;
    private boolean isActive;
    Thread thread;

    public MyThread(MyContainer<Challanger> container2, String name){
        container = container2;
        isActive = true;
        thread = new Thread(this, name);
    }

    void disable() {
        isActive = false;
    }

    @Override
    public void run() {
        int count = count();

        System.out.println(Thread.currentThread().getName() + ": " + count);
        System.out.println(Thread.currentThread().getName() + " finished");
    }

    public int count() {
        int count = 0;
        int minSalary = 0;
        for(Challanger i : container) {
            if(isActive) {
                minSalary = i.getDemandsToWork().getMinSalary();
                if(minSalary > 10000) {
                    count++;
                }
            }
            else {
                break;
            }
        }
        return count;
    }
}
```

3 Результат роботи програми

```
Adding elements...
Adding was end.

List in Recruiting Agency:

ID: 0
Образование: School education
Дата увольнения: 31/5/2020
---Опыт работы---
Место предыдущей работы: HR-manager
Стаж: 4 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 5600
Желаемая будующая работа: Waiter
Желаемые условия будующей работы: Posibility to dose not have buisness trip. Paid vocations. Free dinner. Free cofie.
----------------------------------------
ID: 1
Образование: Higher education
Дата увольнения: 23/3/2021
---Опыт работы---
Место предыдущей работы: Waiter
Стаж: 3 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 17800
Желаемая будующая работа: Programmer
Желаемые условия будующей работы: Possibility to have buisness trip. Coffie machine in the office. Near home.
```

Рисунок 13.1 – Результат роботи програми у автоматичкому режимі

```
Challangers with wishes to dose not have a buiness trip:

ID: 0
Образование: School education
Дата увольнения: 31/5/2020
---Опыт работы---
Место предыдущей работы: HR-manager
Стаж: 4 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 5600
Желаемая будующая работа: Waiter
Желаемые условия будующей работы: Posibility to dose not have buisness trip. Paid vocations. Free dinner. Free cofie.
----------------------------------------

Data sorted by work experience
List in Recruiting Agency:

ID: 1
Образование: Higher education
Дата увольнения: 23/3/2021
---Опыт работы---
Место предыдущей работы: Waiter
Стаж: 3 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 17800
Желаемая будующая работа: Programmer
Желаемые условия будующей работы: Possibility to have buisness trip. Coffie machine in the office. Near home.
----------------------------------------
ID: 0
Образование: School education
Дата увольнения: 31/5/2020
---Опыт работы---
Место предыдущей работы: HR-manager
Стаж: 4 год(а)
---Желания по будующей работе---
Желаемая минимальная зарплата: 5600
Желаемая будующая работа: Waiter
Желаемые условия будующей работы: Posibility to dose not have buisness trip. Paid vocations. Free dinner. Free cofie.
----------------------------------------
```

Рисунок 13.2 – Результат роботи програми у автоматичкому режимі

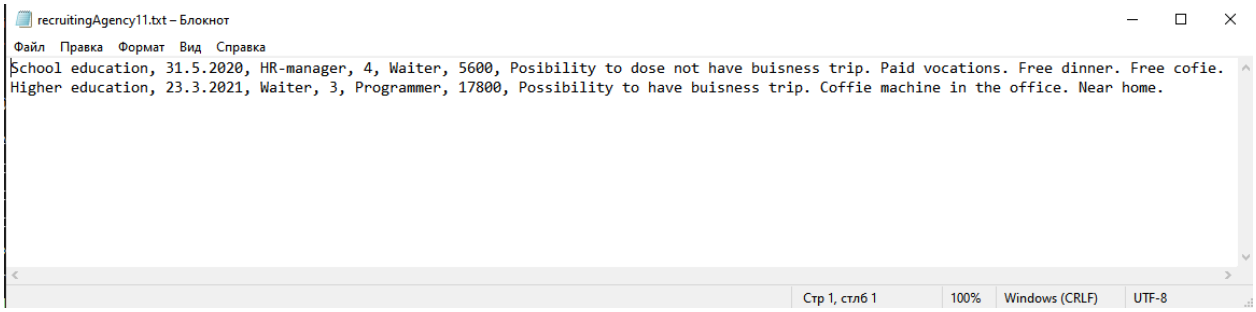

Рисунок 13.3 – Текстовий файл з претендентами

```
Enter option: 1

ID: 0
Образование: School education
Дата увольнения: 23/4/2019
---Опыт работы---
Место предыдущей работы: Selling manager
Стаж: 6 лет
---Желания по будующей работе---
Желаемая минимальная зарплата: 13400
Желаемая будующая работа: Selling manager
Желаемые условия будующей работы: Possibility to have buisness trip. Paid voactions.
----------------------------------------
1. Show all challanger
2. Add challanger
3. Delete chellanger
4. Clear list
5. Is empty recruiting agency?
6. Sort data
7. Serialize data
8. Deserialize data
9. Task
0. Exit
Enter option: 9

Challangers without wishes to dose not have a buisness trip.
```

Рисунок 13.4 – Тестування пошуку претендента, котрий не бажає їздити у відрядження

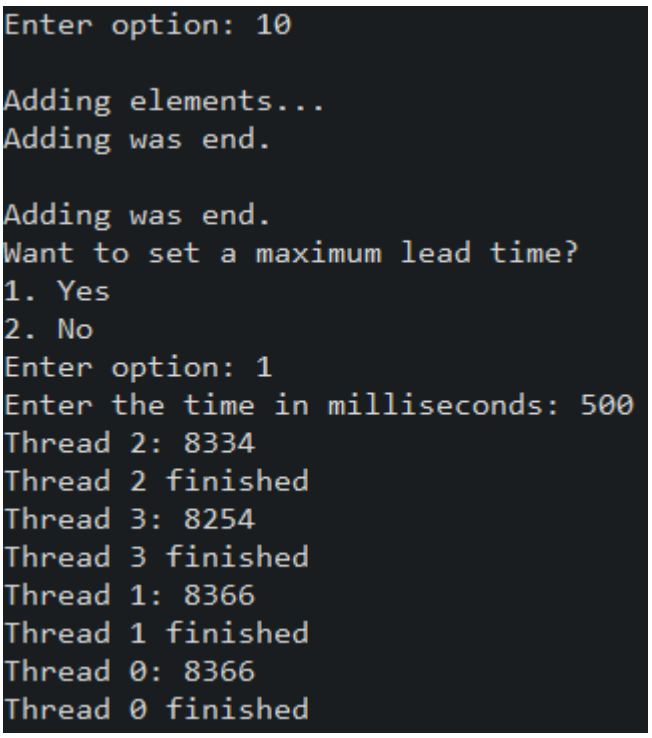

Рисунок 13.5 – Тестування потоків з перериванням

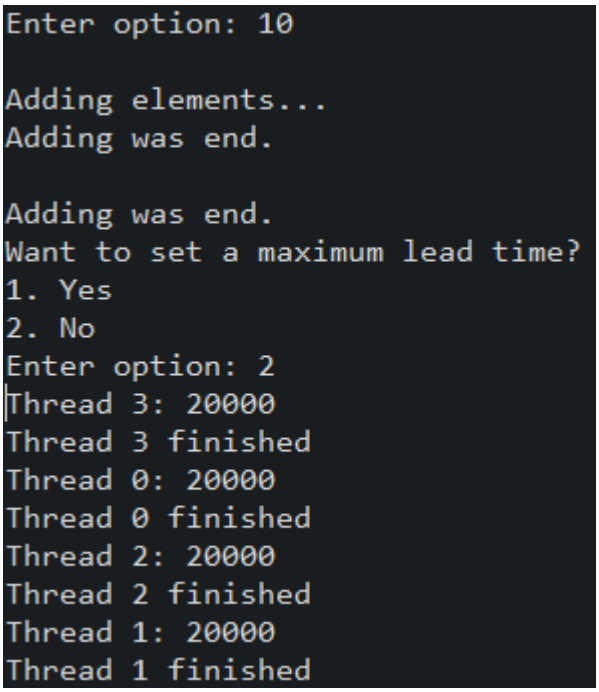

Рисунок 13.6 – Тестування потоків без переривання

Висновок

Під час виконання лабораторної роботи було набуто навички роботи з паралельною обробкою та багатопоточністю в середовищі IntelliJ IDEA.