

Лабораторна робота №14

Паралельне виконання. Ефективність використання

Мета: Вимірювання часу паралельних та послідовних обчислень.
Демонстрація ефективності паралельної обробки.

1 ВИМОГИ

1. Забезпечити вимірювання часу паралельної обробки елементів контейнера за допомогою розроблених раніше методів.
2. Додати до алгоритмів штучну затримку виконання для кожної ітерації циклів поелементної обробки контейнерів, щоб загальний час обробки був декілька секунд.
3. Реалізувати послідовну обробку контейнера за допомогою методів, що використовувались для паралельної обробки та забезпечити вимірювання часу їх роботи.
4. Порівняти час паралельної і послідовної обробки та зробити висновки про ефективність розпаралелювання:
 - результати вимірювання часу звести в таблицю;
 - обчислити та продемонструвати у скільки разів паралельне виконання швидше послідовного.

1.1 Розробник

- П.І.Б: Абдулаєв І.З
- Група: 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<Challenger>) decoder.readObject();` –
нестандартна десеріалізація;
`oos.writeObject(container);`
`container = (MyContainer<Challenger>) 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), `Challenger` (клас, що містить всі поля та методи прикладної області «Кадрове агенство»), `MyConatainer` (клас-контейнер), `Node` (клас-показчик 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

Клас `Main`

```
package ua.khpi.oop.abdulaev14;  
  
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.Challenger;  
import ua.khpi.oop.abdulaev07.DemandsToWork;
```



```

        WorkExperience workExperienceAdd = new
WorkExperience(specializationPrevious, experience);
        DemandsToWork demandsToWorkAdd = new
DemandsToWork(specializationNext,minSalary,conditions);
        Challenger challengerAdd = new
Challenger(id++,education,day,month,year,workExperienceAdd,demandsToWorkAdd);
        recruitingAgency.add(challengerAdd);
    }
    }
    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<Challenger> menu(MyContainer<Challenger>
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 challenger");
System.out.println("2. Add challenger");
System.out.println("3. Delete challenger");
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 challenger: ");
        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);
    Challenger challengerAdd = new
Challenger(id++, education, day, month, year, workExperienceAdd, demandsToWorkAdd);
    recruitingAgency.add(challengerAdd);
    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 minSalaryComparator(),
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<Challenger>)
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:
    final int ARRAY_SIZE = 15000;
    final int NUM_OF_THREADS;
    final int NUM_OF_ITERATIONS;
    long time1, time2;
    int option = 0;

    System.out.println("Adding elements...");
    MyContainer<Challenger> container = new
MyContainer<Challenger>();
    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]|[1012])\\.((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; ARRAY_SIZE > i; i++) {
    id1 = container.getSize();
    WorkExperience workExperienceAdd1 = new

```

```

WorkExperience(specializationPrevious1, experience1);
        DemandsToWork demandsToWorkAdd1 = new
DemandsToWork(specializationNext1,minSalary1,conditions1);
        Challenger challengerAdd1 = new
Challenger(id1++,education1,day1,month1,year1,workExperienceAdd1,demandsToWor
kAdd1);

        container.add(challengerAdd1);
        id2 = container.getSize();
        WorkExperience workExperienceAdd2 = new
WorkExperience(specializationPrevious2, experience2);
        DemandsToWork demandsToWorkAdd2 = new
DemandsToWork(specializationNext2,minSalary2,conditions2);
        Challenger challengerAdd2 = new
Challenger(id2++,education2,day2,month2,year2,workExperienceAdd2,demandsToWor
kAdd2);

        container.add(challengerAdd2);
    }
    System.out.println("Adding was end.");

    System.out.println("How to do calculations?");
    System.out.println("1. Serial");
    System.out.println("2. Paralel");
    System.out.print("Enter option: ");
    option = inInt.nextInt();
    if(option == 1) {
        NUM_OF_THREADS = 1;
        NUM_OF_ITERATIONS = 3;
    }
    else if (option == 2){
        NUM_OF_THREADS = 3;
        NUM_OF_ITERATIONS = 1;
    }
    else {
        System.out.println("Wrong commannd");
        break;
    }
    try
    {
        MyThread[] thread = new MyThread[NUM_OF_THREADS];
        for(int i = 0; NUM_OF_THREADS > i; i++)
        {
            thread[i] = new MyThread(container, "Thread " +
i, NUM_OF_ITERATIONS);

            thread[i].thread.start();
        }
        time1 = System.currentTimeMillis();
        for(int i = 0; NUM_OF_THREADS > i; i++)
            thread[i].thread.join();
        time2 = System.currentTimeMillis();
        System.out.println("Time result: " + (double)(time2 -
time1)/1000 + " seconds");
    }
    catch (InterruptedException ex)
    {
        System.out.println("Thread has been interrupted.");
    }
}

```

```

        container.clear();
        System.out.println();
        break;
    case 0:
        endprog = true;
        recruitingAgency.clear();
        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
buisness trip:\n");
            for(var challenger : task) {
                challenger.print();
            }
            System.out.println();
        }
        else {
            System.out.println("\nChallangers without wishes to dose not have
a buisness trip.\n");
        }
    }
}

```

Клас MyContainer

```

package ua.khpi.oop.abdulaev14;

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;
    }
}

```

Клас MyThread

```

package ua.khpi.oop.abdulaev14;

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

public class MyThread implements Runnable {
    private MyContainer<Challanger> container;
    private boolean isActive;
    private int time;
}

```

```

Thread thread;

MyThread(MyContainer<Challenger> container2, String name, int time){
    container = container2;
    isActive = true;
    this.time = time;
    thread = new Thread(this, name);
}

void disable() {
    isActive = false;
}

private long count() throws InterruptedException {
    long begin = System.currentTimeMillis();
    Thread.currentThread().sleep(500);
    int minSalary = 0;
    int count = 0;
    for(Challenger i : container) {
        if(isActive) {
            minSalary = i.getDemandsToWork().getMinSalary();
            if(minSalary > 10000) {
                count++;
            }
        }
        else {
            System.out.println(Thread.currentThread().getName() + " was
stopped.");
            return -1;
        }
    }
    System.out.println(Thread.currentThread().getName() + ": " + count);
    System.out.println(Thread.currentThread().getName() + " finished");
    return (System.currentTimeMillis() - begin);
}

@Override
public void run() {
    long countTime = 0;
    long temp = 0;
    int count = 0;
    int minSalary = 0;

    for(int i = 0; time > i; i++) {
        try {
            temp = count();
        }
        catch(InterruptedException e) {
            e.printStackTrace();
        }
        countTime += temp;
    }
    System.out.println("Time spent: " + countTime + " milliseconds");
}
}

```

3 РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ

```
Enter option: 10

Adding elements...
Adding was end.

Adding was end.
How to do calculations?
1. Serial
2. Paralel
Enter option: 1
Thread 0: 15000
Thread 0 finished
Thread 0: 15000
Thread 0 finished
Thread 0: 15000
Thread 0 finished
Time spent: 6329 milliseconds
Time result: 6.329 seconds
```

Рисунок 14.1 – Результат роботи програми у середовищі IntelliJ IDEA

```
Enter option: 10

Adding elements...
Adding was end.

Adding was end.
How to do calculations?
1. Serial
2. Paralel
Enter option: 2
Thread 2: 15000
Thread 2 finished
Time spent: 2199 milliseconds
Thread 0: 15000
Thread 0 finished
Time spent: 2201 milliseconds
Thread 1: 15000
Thread 1 finished
Time spent: 2202 milliseconds
Time result: 2.202 seconds
```

Рисунок 14.1 – Результат роботи програми у середовищі IntelliJ IDEA

Висновок

Під час виконання лабораторної роботи було набуто навички роботи з паралельною обробкою та багатопоточністю і визначенням ефективності паралельної обробки даних в середовищі IntelliJ IDEA.