

Звіт

Автор: Васильченко С., 1.KIT1186

Дата: 10.02.2020

Лабораторна робота №14

ПАРАЛЕЛЬНЕ ВИКОНАННЯ. БАГАТОПОТОЧНІСТЬ

Мета:

- Вимірювання часу паралельних та послідовних обчислень.
- Демонстрація ефективності паралельної обробки.

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.

ПРИКЛАДНА ЗАДАЧА:

Кадрове агентство. Сортуння за назвою фірми, за назвою запропонованої спеціальності, за вказаною освітою.

ОПИС ПРОГРАМИ

2.1 Опис змінних:

LinkedContainer<Recruitment> stringLinked = new LinkedContainer<>();//
об'єкт параметризованого контейнера

```
Recruitment rec1 = new Recruitment(); // об'єкт класа кадрового агентства  
Scanner scan = new Scanner(System.in); // змінна для активування  
зчитування з консолі
```

2.2 Ієрархія та структура класів.

final class Lab14 – головний клас. Містить метод main(точку входу у програму) та методи по роботі з програмою для реалізації індивідуального завдання.

interface Linked - інтерфейс контейнеру

class Recruitment - клас прикладної задачі кадрового агентства

class Threads - клас задачі для створення потоків

class LinkedContainer - параметризований клас-контейнер, котрий зберігає інформацію агентства

class DescendingIterator – клас, який реалізує зворотний ітератор для переміщення по списку

class Util - клас зберігаючий утиліти для обробки контейнера

class Demo – клас для демонстрації виконання програми в різних її варіаціях

class Exe – клас, який має два варіанта виконання, один з яких – меню з варіантами дії.

ТЕКСТ ПРОГРАМИ

File Lab14.java:

```
LinkedContainer<Recruitment> linkedContainer = new LinkedContainer<>();  
    Recruitment rec1 = new Recruitment();  
    Recruitment rec2 = new Recruitment();  
    Recruitment rec3 = new Recruitment();  
  
    rec1.setPayment(2600);  
    rec2.setPayment(3700);  
    for (int i = 0; i < 10000; i++) {  
        linkedContainer.add(rec1);  
        linkedContainer.add(rec2);  
    }  
    rec3.setPayment(600);  
    for (int i = 0; i < 50000; i++) {  
        linkedContainer.add(rec3);  
    }  
    rec2.setPayment(2500);  
    linkedContainer.add(rec2);  
    rec2.setPayment(500);  
    linkedContainer.add(rec2);  
    Treads.MyTread1 myTread1 = new Treads.MyTread1();  
    myTread1.set(linkedContainer);  
    Treads.MyTread2 myTread2 = new Treads.MyTread2();  
    myTread2.set(linkedContainer);
```

```

Treads.MyTread3 myTread3 = new Treads.MyTread3();
myTread3.set(linkedContainer);
System.out.println("++++++Another threads+++++++");
long start2 = System.currentTimeMillis();
ExecutorService executorService = Executors.newFixedThreadPool(3);
Future<Boolean> future = executorService.submit(myTread1);
Future<Boolean> future1 = executorService.submit(myTread2);
Future<Boolean> future2 = executorService.submit(myTread3);
    future.get();
    future1.get();
    future2.get();
future.cancel(true);
future1.cancel(true);
future2.cancel(true);
executorService.shutdown();
long finish2 = System.currentTimeMillis();
long result2 = finish2 - start2;
System.out.println("++++++Main thread+++++++");
long start = System.currentTimeMillis();
myTread1.call();
myTread2.call();
myTread3.call();
long finish = System.currentTimeMillis();
long result = finish - start;
System.out.println("Main time: " + result / 1000.0 + " seconds.");
System.out.println("Another threads time: " + result2 / 1000.0 + " seconds.");
}

```

Recruitment.java :

```

package ua.khpi.oop.vasilchenko09.First;

import java.util.Scanner;

public class Recruitment {
    private String firm;
    private String specialty;
    private String workingConditions;
    private int payment;

    private String needsSpeciality;
    private int experience;
    private String education;

    private boolean confirm ;

    public void setFirm(final String firm) {
        this.firm = firm;
    }

    public void setSpecialty(final String specialty) {
        this.specialty = specialty;
    }

    public void setWorkingConditions(final String workingConditions) {
        this.workingConditions = workingConditions;
    }

    public void setPayment(final int payment) {
        this.payment = payment;
    }

    public void setConfirm(final boolean confirm) {
        this.confirm = confirm;
    }

    public int getPayment() {
        return payment;
    }
}

```

```

    public String getWorkingConditions() {
        return workingConditions;
    }

    public String getSpecialty() {
        return specialty;
    }

    public String getFirm() {
        return firm;
    }

    public Recruitment() {
        firm = null;
        specialty = null;
        workingConditions = null;
        payment = 0;
        needsSpeciality = null;
        experience = 0;
        education = null;
        confirm = false;
    }

    public String show() {
        String show;
        show = "Фирма: " + firm + "\n" +
            "Специальность: " + specialty + "\n" +
            "Условия работы: " + workingConditions + "\n" +
            "Оплата: " + payment + "\n";
        if (confirm) {
            show += "Необходимая специальность: " + needsSpeciality + "\n";
            show += "Опыт: " + experience + "\n";
            show += "Образование: " + education + "\n";
        }
        return show;
    }

    public Recruitment(final Recruitment obj) {
        firm = obj.firm;
        specialty = obj.specialty;
        workingConditions = obj.workingConditions;
        payment = obj.payment;
        needsSpeciality = obj.needsSpeciality;
        experience = obj.experience;
        education = obj.education;
        confirm = obj.confirm;
    }

    public void generateVacancy() {
        Scanner scan = new Scanner(System.in);
        Scanner scan2 = new Scanner(System.in);
        int choose;
        System.out.print("\nВведите фирму: ");
        firm = scan.nextLine();
        System.out.print("\nВведите специальность: ");
        specialty = scan.nextLine();
        System.out.print("\nВведите условия работы: ");
        workingConditions = scan.nextLine();
        System.out.print("\nВведите оплату: ");
        payment = scan.nextInt();
        System.out.println("Желаете добавить дополнительные условия работы? 1 - Да. 0 - Нет:");

        choose = scan.nextInt();
        while (true) {
            if (choose == 1) {
                System.out.print("\nВведите необходимую специальность: ");
                needsSpeciality = scan2.nextLine();
            }

```

```

        System.out.print("\nНеобходимое образование: ");
        education = scan2.nextLine();
        System.out.print("\nнеобходимый опыт работы: ");
        experience = scan2.nextInt();
        confirm = true;
        break;
    } else if (choose == 0) {
        needsSpeciality = null;
        experience = 0;
        education = null;
        break;
    } else {
        System.out.println("Ошибка! Повторите ввод: ");
    }
}

public void setExperience(final int experience) {
    this.experience = experience;
}

public int getExperience() {
    return experience;
}

public void setNeedsSpeciality(final String needsSpeciality) {
    this.needsSpeciality = needsSpeciality;
}

public String getNeedsSpeciality() {
    return needsSpeciality;
}

public void setEducation(final String education) {
    this.education = education;
}

public String getEducation() {
    return education;
}

public boolean getConfirms() {
    return confirm;
}

@Override
public String toString() {
    return show();
}
}

```

Linked.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();
    }

```

DescendingIterator.java:

```

package ua.khpi.oop.vasilchenko09.MyList;

import java.util.Iterator;

public interface DescendingIterator<T> {
    Iterator<T> descendingIterator();
}

```

Util.java:

```

package ua.khpi.oop.vasilchenko10.MyList;

import ua.khpi.oop.vasilchenko10.First.Recruitment;

import java.util.Arrays;

public class Util {
    public static <T extends Recruitment> void sortFirm(LinkedContainer<T> obj) {
        Recruitment[] array = new Recruitment[obj.size()];
        for (int i = 0; i < obj.size(); i++) {
            array[i] = obj.getElementByIndex(i);
        }
        Arrays.sort(array, Recruitment.compareByFirm);
        obj.clear();
        for (int i = 0; i < array.length; i++) {
            obj.add((T) array[i]);
        }
    }

    public static <T extends Recruitment> void sortSpecialty(LinkedContainer<T> obj) {
        Recruitment[] array = new Recruitment[obj.size()];
        for (int i = 0; i < obj.size(); i++) {
            array[i] = obj.getElementByIndex(i);
        }
        Arrays.sort(array, Recruitment.compareBySpecialty);
        obj.clear();
        for (int i = 0; i < array.length; i++) {
            obj.add((T) array[i]);
        }
    }

    public static <T extends Recruitment> void sortEducation(LinkedContainer<T> obj) {
        Recruitment[] array = new Recruitment[obj.size()];
        int count = 0;
        for (int i = 0; i < obj.size(); i++) {
            array[i] = obj.getElementByIndex(i);
            if (array[i].getEducation() == null) {
                count++;
            }
        }
        Recruitment[] buff = new Recruitment[count];
        for (int i = 0, j = 0; i < array.length; i++) {
            if (array[i].getEducation() == null) {
                buff[j] = array[i];
                j++;
            }
        }
        Recruitment[] temp = new Recruitment[obj.size() - count];
        for (int i = 0, j = 0; i < array.length; i++) {
            if (array[i].getEducation() != null) {
                temp[j] = array[i];
                j++;
            }
        }
    }
}

```

```

    }
}
Arrays.sort(temp, Recruitment.compareByEducation());
obj.clear();
for (int i = 0; i < temp.length; i++) {
    obj.add((T) temp[i]);
}
for (int i = 0; i < buff.length; i++) {
    obj.add((T) buff[i]);
}
}

public static void chooseMenu() {
    System.out.println();
    System.out.println("1. Add vacancy.");
    System.out.println("2. Show all vacancies.");
    System.out.println("3. Clear container.");
    System.out.println("4. Check elements in container.");
    System.out.println("5. Size of container.");
    System.out.println("6. Get element by index.");
    System.out.println("7. Save data to file.");
    System.out.println("8. Read data from file.");
    System.out.println("9. Sorting data in container.");
    System.out.println("0. End of work.");
    System.out.print("Write your choose there: ");
}
}

```

Demo.java:

```

package ua.khpi.oop.vasilchenko13.Tests;

import ua.khpi.oop.vasilchenko13.First.Recruitment;
import ua.khpi.oop.vasilchenko13.MyList.LinkedContainer;
import ua.khpi.oop.vasilchenko13.MyList.Treads;

import java.io.IOException;
import java.util.InputMismatchException;
import java.util.concurrent.*;

import static ua.khpi.oop.vasilchenko13.MyList.Treads.MyTread1;
import static ua.khpi.oop.vasilchenko13.MyList.Treads.MyTread2;
import static ua.khpi.oop.vasilchenko13.MyList.Treads.MyTread3;

public class Demo {
    public static void main(String[] args) throws InputMismatchException, IOException,
        InterruptedException, TimeoutException, ExecutionException {
        //Lab13.main(new String[] {"-auto"});
        //Lab13.main(null);

        LinkedContainer<Recruitment> linkedContainer = new LinkedContainer<>();
        Recruitment rec1 = new Recruitment();
        Recruitment rec2 = new Recruitment();
        Recruitment rec3 = new Recruitment();

        rec1.setPayment(2600);
        rec2.setPayment(3700);
        for (int i = 0; i < 10000; i++) {
            linkedContainer.add(rec1);
            linkedContainer.add(rec2);
        }
        rec3.setPayment(600);
        for (int i = 0; i < 1200; i++) {
            linkedContainer.add(rec3);
        }
    }
}

```

```

    }
    rec2.setPayment(2500);
    linkedContainer.add(rec2);
    rec2.setPayment(500);
    linkedContainer.add(rec2);
    MyTread1 myTread1 = new MyTread1();
    myTread1.set(linkedContainer);
    MyTread2 myTread2 = new MyTread2();
    myTread2.set(linkedContainer);
    MyTread3 myTread3 = new MyTread3();
    myTread3.set(linkedContainer);
    ExecutorService executorService = Executors.newFixedThreadPool(3);
    Future<Boolean> future = executorService.submit(myTread1);
    Future<Boolean> future1 = executorService.submit(myTread2);
    Future<Boolean> future2 = executorService.submit(myTread3);
    try {
        future.get(2, TimeUnit.SECONDS);
        future1.get(2, TimeUnit.SECONDS);
        future2.get(2, TimeUnit.SECONDS);
    } catch (TimeoutException ex) {
        future.cancel(true);
        future1.cancel(true);
        future2.cancel(true);
    }
    executorService.shutdown();
}
}

```

Exe.java

```

package ua.khpi.oop.vasilchenko10.Tests;

import ua.khpi.oop.vasilchenko10.First.Recruitment;
import ua.khpi.oop.vasilchenko10.MyList.LinkedContainer;
import ua.khpi.oop.vasilchenko10.MyList.Util;

import java.util.Scanner;

public class Exe {
    public static void auto() {
        System.out.println("It is auto mode");
        System.out.println("Creating object container:");
        LinkedContainer<Recruitment> linkedContainer = new LinkedContainer<>();
        System.out.println("Reading from file: ");
        linkedContainer.readRec();
        System.out.println("Print container:");
        for (Recruitment s : linkedContainer) {
            System.out.println(s);
        }
        System.out.println("Size: ");
        System.out.println(linkedContainer.size());
        System.out.println("Clear: ");
        linkedContainer.clear();
        System.out.println("Size: ");
        System.out.println(linkedContainer.size());
        System.out.println("Reading from file: ");
        linkedContainer.readRec();
        System.out.print("Not empty? : ");
        System.out.println(linkedContainer.notEmpty());
        System.out.println("Element with index 0");
        System.out.println(linkedContainer.getElementByIndex(0));
    }

    public static void menu() {
        System.out.println("It is menu mode");
        LinkedContainer<Recruitment> linkedContainer = new LinkedContainer<>();
        Recruitment rec1 = new Recruitment();
    }
}

```



```

Scanner scan = new Scanner(System.in);
boolean loop = true;
while (loop) {
    Util.chooseMenu();
    int choose = scan.nextInt();
    switch (choose) {
        case 1:
            rec1.generateVacancy();
            linkedContainer.add(rec1);
            System.out.println("Done!");
            break;
        case 2:
            if (linkedContainer.notEmpty()) {
                for (Recruitment s : linkedContainer) {
                    System.out.println(s);
                }
            } else {
                System.out.println("Ошибка! Список пустой!");
            }
            break;
        case 3:
            if (linkedContainer.notEmpty()) {
                linkedContainer.clear();
                System.out.println("Успешно!");
            } else {
                System.out.println("Ошибка! Массив пустой!");
            }
            break;
        case 4:
            if (linkedContainer.notEmpty()) {
                System.out.println("Your container have data.");
            } else {
                System.out.println("Your container doesn't have data.");
            }
            break;
        case 5:
            System.out.println("Size of container: " + linkedContainer.size());
            break;
        case 6:
            if (linkedContainer.notEmpty()) {
                System.out.print("Entered index from 0 to " + (linkedContainer.size()
- 1) + ": ");

                int choose1 = scan.nextInt();
                if (choose1 > (linkedContainer.size() - 1) || choose1 < 0) {
                    System.out.println("Repeat enter");
                } else {
                    System.out.println(linkedContainer.getElementByIndex(choose1));
                }
                break;
            }
        case 7:
            if (linkedContainer.notEmpty()) {
                linkedContainer.saveRec();
            } else {
                System.out.println("Your container is empty");
            }
            break;
        case 8:
            if (linkedContainer.notEmpty()) {
                linkedContainer.clear();
            }
            linkedContainer.readRec();
            break;
        case 9:
            if (linkedContainer.notEmpty()) {
                System.out.println("What field do you want to sort by?");
                System.out.println("1. Sort by firm");
                System.out.println("2. Sort by specialty");
                System.out.println("3. Sort by education");
            }
    }
}

```

```

        int choose2 = scan.nextInt();
        boolean loop2 = true;
        while (loop2) {
            switch (choose2) {
                case 1:
                    Util.sortFirm(linkedContainer);
                    loop2 = false;
                    break;
                case 2:
                    Util.sortSpecialty(linkedContainer);
                    loop2 = false;
                    break;
                case 3:
                    Util.sortEducation(linkedContainer);
                    loop2 = false;
                    break;
                default:
                    System.out.println("Ошибка. Неверное число!");
                    break;
            }
        }
    } else {
        System.out.println("Your container is empty");
    }
    break;
case 0:
    System.out.println("Thanks for working!");
    loop = false;
    break;
default:
    System.out.println("Ошибка. Неверное число!");
    break;
}
}
}
}
}

```

ВАРІАНТИ ВИКОРИСТАННЯ

```
C:\Program Files\Java\jdk-15.0.2\bin\java.exe -Xjavaawt:  
Number of vacancies with higher payment: 10000  
Sum payment = 63723000  
Avg payment = 3005  
Max payment = 3700  
Min payment = 500
```

Рис. 13.1 – Результат роботи програми

```
1. Add vacancy.  
2. Show all vacancies.  
3. Clear container.  
4. Check elements in container.  
5. Size of container.  
6. Get element by index.  
7. Save data to file.  
8. Read data from file.  
9. Sorting data in container.  
10. Create selection from container.  
0. End of work.  
Write your choose there: |
```

Рис. 13.2 – Результат роботи програми

```
10. Create selection from container.  
0. End of work.  
Write your choose there: 10  
#1  
Фирма: NIX  
Специальность: teacher  
Условия работы: full time, car, english  
Оплата: 1200  
Необходимая специальность: C++  
Опыт: 12  
Образование: Higher
```

Рис. 13.3 – Результат роботи програми

```
It is auto mode
Creating object container:
Reading from file:
Print container:
Фирма: Ерат
Специальность: Java
Условия работы: full time
Оплата: 2600
Необходимая специальность: Java Junior
Опыт: 15
Образование: Higher

Фирма: NIX
Специальность: teacher
Условия работы: full time, car, english
Оплата: 1200
Необходимая специальность: C++
Опыт: 12
Образование: Higher

Фирма: GlobalLogic
Специальность: C#
Условия работы: full time
Оплата: 1800
Необходимая специальность: C
Опыт: 1
Образование: Higher
```

Рис. 13.4 – Результат работы программы

```
Size:
3
Clear:
Size:
0
Reading from file:
Not empty? : true
Element with index 0
Фирма: Ерат
Специальность: Java
Условия работы: full time
Оплата: 2600
Необходимая специальность: Java Junior
Опыт: 15
Образование: Higher
```

Рис. 13.5 – Результат работы программы

Результуюча таблиця

№ п/п	Кол-во элементов	Время многопоточности, sec	Время main потока, sec	Дополнительная задержка, msec	В сколько раз быстрее	В среднем
1	152	2,721	7,928	10	2,914	2,460
2	152	0,509	1,286	1	2,527	
3	152	3,976	11,734	15	2,951	
4	13002	26,246	75,011	1	2,858	
5	13002	36,598	103,959	2	2,841	
6	13002	0,607	1,114	-	1,835	
7	22002	3,043	5,625	-	1,849	
8	50002	28,838	77,619	-	2,692	
9	95002	185,908	311,500	-	1,676	

Програму можна використовувати задля створення бази даних. Завдяки параметризації зв'язного списку, базу даних можна використати для будь-яких типів даних. Переважно у нашому варіанті - кадрове агенство, в якому представляються різноманітні вакансії. Також для вибору доступно багато інших можливостей. Реалізовано меню для поліпшення користування програмою. Реалізовано можливість створення виборки. Додано многопоточне використання програми.

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. Завдання виконане! Програма працює успішно!