**Звіт**

**Лабораторна работа 13. Паралельне виконання. Багатопоточність**

**Мета роботи**:

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

## 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:
    - пошук мінімуму або максимуму;
    - обчислення середнього значення або суми;
    - підрахунок елементів, що задовольняють деякій умові;
    - відбір за заданим критерієм;
    - власний варіант, що відповідає обраній прикладної області.

**1.1. Розробник**: Момот Роман Євгенійович, КІТ119а, варіант №14.

## 2. ОПИС ПРОГРАМИ

**2.1. Засоби ООП**: клас, метод класу, поле класу.

**2.2. Ієрархія та структура класів:** один публічний клас **Main**, публічний клас **Event**, у полях якого є час початку події, тривалість, адреса події, імена людей, опис події, гетери, сетери, конструктор класу та метод виведення даних класу. Також є клас **Node**, який виконує роль покажчика на елемент і клас **MyContainer**, який містить покажчик на головний елемент та методи обробки масиву елементів. Клас **MyThread**, який виконує роль потока.

**2.3. Важливі фрагменти програми:**

```
public class MyThread implements Runnable{

	private boolean isActive;

	void disable() {
		isActive = false;
	}

	Thread thread;
	private MyContainer<Event> arr;

	MyThread(MyContainer<Event> arr, String name){
		this.arr = arr;
		isActive = true;
		thread = new Thread(this, name);
	}

	@Override
	public void run() {
		long count = 0;

		for(Event i : arr) {
			if(isActive) {
				count += i.getDuration();
			}
			else {
				break;
```

```
                }
            }

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

}

private static MyContainer<Event> menu(MyContainer<Event> arr) {
            Scanner scan = new Scanner(System.in);
            boolean stop = false;
            int choise, choise2;

            ArrayList<String> people = new ArrayList<String>();
            people.add("John");
            people.add("Bill");
            people.add("Івасик");
            Event evToCompare = new Event(new
GregorianCalendar(2002,3,28), 120, "ул. Революции",
                        people, "Pest party ever");
            arr.add(evToCompare);

            do {
                System.out.println("What to do?");
                System.out.println("1. Output data");
                System.out.println("2. Add element");
                System.out.println("3. Delete element");
                System.out.println("4. Is empty?");
```

```
                System.out.println("5. Serialization");
                System.out.println("6. Deserialization");
                System.out.println("7. Sort data");
                System.out.println("8. Find the number of people
(Multithreading)");
                System.out.println("9. Terminate program");
                System.out.println("====================");
                System.out.print("Your choise: ");
                choise = scan.nextInt();

                switch(choise) {
                case 1:
                        System.out.println("\nChoose the output method");
                        System.out.println("1. Using foreach");
                        System.out.println("2. Using toArray");
                        System.out.println("3. Find element by criteria");
                        System.out.println("4. Return");
                        System.out.println("================");
                        System.out.print("Your choise: ");
                        choise2 = scan.nextInt();
                        System.out.println( );

                        switch(choise2) {
                        case 1:
                                if(arr.getSize() > 0){
                                        for(var i : arr) {
                                                i.outputData();
                                                System.out.println( );
                                        }
```

```
                System.out.println( );
            }
            else {
                System.out.println("Array is empty.\n");
            }
            break;

        case 2:
            if(arr.getSize() > 0) {
                Object[] tempArr = arr.toArray();
                for (int i = 0; i < tempArr.length; i++) {
                    System.out.println(i + ")");
                    ((Event)tempArr[i]).outputData();
                }
            }
            else {
                System.out.println("Array is empty.");
            }
            break;
        case 3:
            if(arr.getSize() == 0) {
                System.out.println("Array is empty.\n");
                break;
            }

            Pattern pattYear;
            Pattern pattCity;
            Pattern pattDuration = Pattern.compile("^([2][5-
9]+)|([3-9][0-9]+)|([1-9][0-9]{2,})$");
```

```
                Matcher matcher1, matcher2, matcher3;

                String regex = "^(?)|(?)|(?)$";

                System.out.println("Task: Знайти всі конференції, що пройшли\n"
                        + "-за останні три роки "
                        + "\n-в Харкові та області "
                        + "\n-з тривалістю не менше доби.");
                System.out.println("Передбачити можливість незначної зміни умов пошуку.");

                System.out.print("\nEnter the year: ");
                int year = scan.nextInt();
                for (int i = 0; i < 3; i++) {
                    regex = regex.substring(0,regex.indexOf('?')) + Integer.toString(year - i) + regex.substring(regex.indexOf('?') + 1, regex.length());
                }
                pattYear = Pattern.compile(regex);

                System.out.print("Enter the city: ");
                scan.nextLine();
                String city = scan.nextLine();
                city = city.concat("(.*)");
                pattCity = Pattern.compile(city);

                for(var i : arr) {
                    matcher1 = pattYear.matcher(Integer.toString(i.getStartTime().get(Calendar.YEAR)));
```

```
                                        matcher2 = pattCity.matcher(i.getAddress());
                                        matcher3 =
pattDuration.matcher(Integer.toString(i.getDuration()));

                                        System.out.println( );
                                        if(matcher1.matches() &&
matcher2.matches() && matcher3.matches()) {
                                                i.outputData();
                                        }
                                }

                                break;

                        case 4:
                                System.out.println("\nReturning\n");
                                break;

                        default:
                                System.out.println("You've entered the wrong
number");
                                break;
                        }
                        break;

                case 2:
                        Event newEvent = inputNewEvent();
                        arr.add(newEvent);

                        break;
```

```
case 3:
        if(arr.getSize() > 0) {
                System.out.print("Enter the index of element: ");
                choise = scan.nextInt();

                arr.delete(choise);
        } else {
                System.out.println("Array is empty.");
        }
        break;

case 4:
        if(arr.isEmpty()) {
                System.out.println("Array is empty.");
        } else {
                System.out.println("Array isn't empty.");
        }
        break;

case 5:
        System.out.println("\nChoose the method");
        System.out.println("1. Standard serialization");
        System.out.println("2. XML serialization");
        System.out.println("3. Return");
        System.out.println("========================");
        System.out.print("Your choise: ");
        choise2 = scan.nextInt();

        switch(choise2) {
```

```
                    case 1:
                        scan.nextLine();
                        System.out.print("Enter the name of file: ");
                        String filename = scan.nextLine();

                        if (filename.indexOf(".ser") == -1) {
                            filename += ".ser";
                        }

                        try(ObjectOutputStream oos = new
ObjectOutputStream(new BufferedOutputStream(new
FileOutputStream(filename)))){
                            oos.writeObject(arr);
                            System.out.println("Serialization
successful.");
                        }catch(Exception ex){
                            System.out.println(ex.getMessage());
                            ex.printStackTrace();
                        }

                        break;

                    case 2:
                        scan.nextLine();
                        System.out.print("Enter the name of file: ");
                        filename = scan.nextLine();

                        if (filename.indexOf(".xml") == -1) {
                            filename += ".xml";
                        }
```

```
                    try(XMLEncoder encoder = new
XMLEncoder(new BufferedOutputStream(new FileOutputStream(filename)))){
                        encoder.writeObject(arr);
                        System.out.println("Serialization
successful.");
                    }
                    catch(Exception ex){
                        System.out.println(ex.getMessage());
                    }
                    break;

                case 3:
                    break;

                default:
                    System.out.println("You've entered the wrong
command.");
                    break;
                }

                break;

            case 6:
                System.out.println("\nChoose the method");
                System.out.println("1. Standard deserialization");
                System.out.println("2. XML deserialization");
                System.out.println("3. Return");
                System.out.println("========================");
                System.out.print("Your choise: ");
```

```
                choise2 = scan.nextInt();

                switch(choise2) {
                case 1:
                        scan.nextLine();
                        System.out.print("Enter the name of file: ");
                        String filename = scan.nextLine();

                        if (filename.indexOf(".ser") == -1) {
                                filename += ".ser";
                        }

                        try(ObjectInputStream oos = new
ObjectInputStream(new BufferedInputStream(new FileInputStream(filename)))){
                                arr.clear();
                                arr = (MyContainer<Event>)
oos.readObject();
                                System.out.println("\nSerialization
successful.");
                        }catch(Exception ex){
                                System.out.println(ex.getMessage());
                        }


                        break;

                case 2:
                        scan.nextLine();
                        System.out.print("Enter the name of file: ");
                        filename = scan.nextLine();
```

```
                    if (filename.indexOf(".xml") == -1) {
                        filename += ".xml";
                    }

                    try(XMLDecoder decoder = new
XMLDecoder(new BufferedInputStream(new FileInputStream(filename)))){
                        arr.clear();
                        arr = (MyContainer<Event>)
decoder.readObject();
                        System.out.println("Serialization
successful.\n");
                    }catch(IOException ex){
                        System.out.println( );
                    }
                    break;

                case 3:
                    break;

                default:
                    System.out.println("You've entered the wrong
command.");
                    break;
                }

                break;
            case 7:
                System.out.println("\nChoose sorting field:");
                System.out.println("1. Sort by event date");
```

```
                System.out.println("2. Sort by event length");
                System.out.println("3. Sort by number of people");
                System.out.println("4. Return");

System.out.println("============================");
                System.out.print("Your choise: ");
                choise2 = scan.nextInt();

                switch(choise2) {
                case 1:
                        arr.sort(new EventDateComparator());
                        System.out.println("\nData sorted\n");
                        break;

                case 2:
                        arr.sort(new EventLengthComparator());
                        System.out.println("\nData sorted\n");
                        break;

                case 3:
                        arr.sort(new EventPeopleNumberComparator());
                        System.out.println("\nData sorted\n");
                        break;

                case 4:
                        System.out.println("\nReturning\n");
                        break;

                default:
```

```
                        System.out.println("\nYou have entered the wrong
number.\n");
                        break;
                    }
                    break;
                case 8:
                    int time = -1;
                    int numberOfPeople;
                    boolean choise3;
                    ArrayList<String> newPeople = new
ArrayList<String>();

                    MyThread[] threads = new MyThread[3];

                    System.out.println("\nAdding new elements...");
                    for(int i = 0; i < 30000; i++) {
                        numberOfPeople = (int) (2 + Math.random() * 10);
                        for(int j = 0; j < numberOfPeople; j++) {
                            newPeople.add("j");
                        }
                        newEvent = new Event(new GregorianCalendar(),
i, Integer.toString(i), newPeople, Integer.toString(i));
                        arr.add(newEvent);
                        newPeople = new ArrayList<String>();
                    }


                    System.out.print("Want to set a maximum lead time? ");
                    if((choise3 = scan.nextBoolean()) == true) {
```

```
                System.out.print("Enter the time in milliseconds:
");
                time = scan.nextInt();
            }

            try {
                for(int i = 0; i < 3; i++) {
                    threads[i] = new MyThread(arr, "Thread " +
(i+1));
                    threads[i].thread.start();
                }

                if(time > 0) {
                    Thread.currentThread().sleep(time);

                    for(int i = 0; i < 3; i++) {
                        threads[i].disable();
                    }
                }

                for(int i = 0; i < 3; i++) {
                    threads[i].thread.join();
                }
            }
            catch(InterruptedException ex) {
                System.out.println("Thread has been interrupted.");
            }
            arr.clear();
```

```
                System.out.println( );
                break;

            case 9:
                System.out.println("Terminating the program.");
                stop = true;
                break;

            default:
                System.out.println("You have entered the wrong
number.");
                break;
            }
        }while(!stop);

        scan.close();
        return arr;
    }
```

## Результат роботи програми

```
What to do?
1. Output data
2. Add element
3. Delete element
4. Is empty?
5. Serialization
6. Deserialization
7. Sort data
8. Find the number of people (Multithreading)
9. Terminate program
====================
Your choise: 8

Adding new elements...
Want to set a maximum lead time? false
Thread 1: 449985120
Thread 1 finished
Thread 2: 449985120
Thread 2 finished
Thread 3: 449985120
Thread 3 finished

What to do?
1. Output data
2. Add element
3. Delete element
4. Is empty?
5. Serialization
6. Deserialization
7. Sort data
8. Find the number of people (Multithreading)
9. Terminate program
====================
Your choise: 8

Adding new elements...
Want to set a maximum lead time? true
Enter the time in milliseconds: 100
Thread 3: 11875501
Thread 1: 11618610
Thread 2: 11608971
Thread 2 finished
Thread 1 finished
Thread 3 finished
```

## Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з багатопоточністтю.

Програма протестована, виконується без помилок.