НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Об’єктно-орієнтоване програмування ч.2»

*Звіт з лабораторної роботи №14*
*Тема: «Паралельне виконання. Ефективність використання»*

Виконав:
ст. гр. КИТ-118в
Кліщов Б. Р.

Перевірив:
Пугачев Р.В.

Харків – 2020

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

# 1 ВИМОГИ

## 1.1 Розробник

**Кліщов Б. Р., КИТ-118в**

## 1.2 Загальне завдання

Забезпечити вимірювання часу паралельної обробки елементів контейнера за допомогою розроблених раніше методів.

Додати до алгоритмів штучну затримку виконання для кожної ітерації циклів поелементної обробки контейнерів, щоб загальний час обробки був декілька секунд.

Реалізувати послідовну обробку контейнера за допомогою методів, що використовувались для паралельної обробки та забезпечити вимірювання часу їх роботи.

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.

# 2 ОПИС ПРОГРАМИ

## 2.1 Засоби ООП

## 2.2 Ієрархія та структура класів

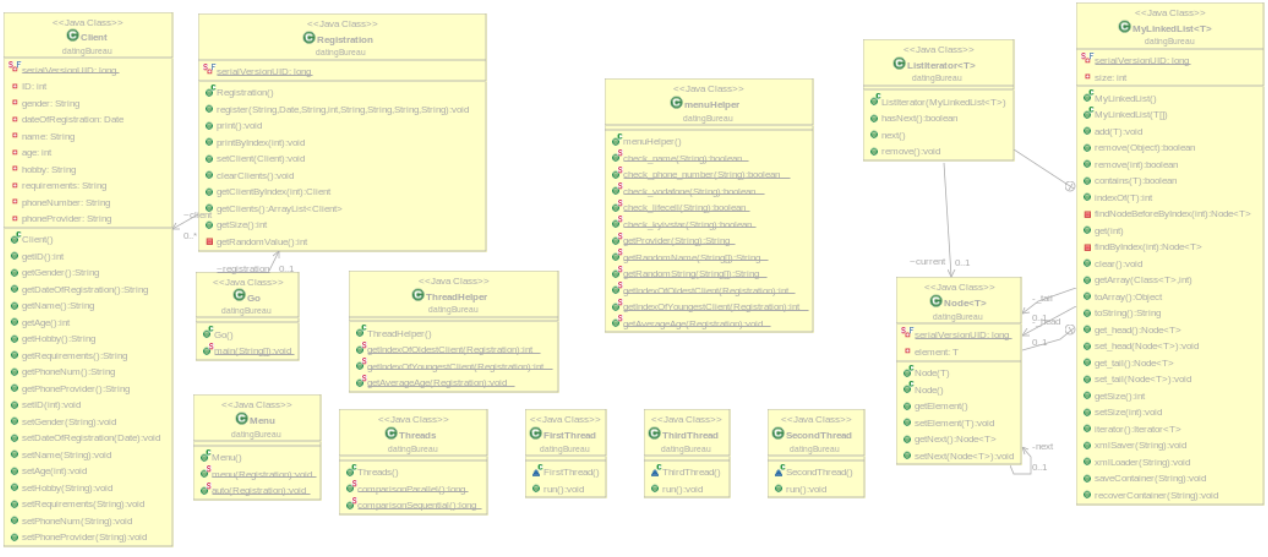


**Рисунок 1. Діаграма класів**

## 2.3 Важливі фрагменти програми

```java
public static int getIndexOfYoungestClient(Registration reg) throws InterruptedException {
    int index = 0;
    for(int i = 1; i < reg.getSize(); i++) {
        if (!Thread.currentThread().isInterrupted()) {
            if(reg.getClientByIndex(i).getAge() < reg.getClientByIndex(index).getAge()) {
                index = i;
            }
            Thread.sleep(100);
        } else {
            throw new InterruptedException();
        }
    }
    return index;
}
public static void getAverageAge(Registration reg) throws InterruptedException {
    int averageAge = 0;
    for(int i = 0; i < reg.getSize(); i++) {
        if (!Thread.currentThread().isInterrupted()) {
            averageAge = averageAge + reg.getClientByIndex(i).getAge();
            Thread.sleep(100);
        } else {
            throw new InterruptedException();
        }
    }
    averageAge = averageAge / reg.getSize();
    System.out.println("Average age: " + averageAge + "\n");
}
```

Рисунок 2. Алгоритми для розрахунку

```java
public static long comparisonParallel() {
    long time_start = System.currentTimeMillis();
    System.out.println("Starting all threads...");
    FirstThread first = new FirstThread();
    Thread t1 = new Thread(first,"FirstThread");

    SecondThread second = new SecondThread();
    Thread t2 = new Thread(second,"SecondThread");

    ThirdThread third = new ThirdThread();
    Thread t3 = new Thread(third,"ThirdThread");

    t1.start();
    t2.start();
    t3.start();

    try {
        t1.join();
        t2.join();
        t3.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("Finishing all threads...");
    return System.currentTimeMillis() - time_start;
}
```

Рисунок 3. Вимірювання паралельного виконання

```java
public static long comparisonSequential() {
    long time_start1 = System.currentTimeMillis();
    System.out.println("Starting sequence...");
    try {
        ThreadHelper.getAverageAge(Go.registration);
        ThreadHelper.getIndexOfOldestClient(Go.registration);
        ThreadHelper.getIndexOfYoungestClient(Go.registration);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("Finishing sequence...");
    return System.currentTimeMillis() - time_start1;
}
```

Рисунок 4. Вимірювання послідовного виконання

## 3 ВАРІАНТИ ВИКОРИСТАННЯ

```
Starting all threads...
First Thread: I`m started

Second Thread: I`m started

Third Thread: I`m started

Secound thread: I`m finished!!!

Third thread: I`m finished!!!

Average age: 37

First thread: I`m finished!!!

Finishing all threads...
Starting sequence...
Average age: 37

Finishing sequence...
--------------------------------------
|Time of paralel executing    |10109ms |
--------------------------------------
|Time of sequential executing |29850ms |
--------------------------------------
```

Рисунок 5. Результат

## ВИСНОВКИ

Я виміряв час паралельного та послідовного обчислень, продемонстрував ефективність паралельної обробки.