НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Об’єктно-орієнтоване програмування ч.2»

Звіт з лабораторної роботи №13
Тема: «Паралельне виконання. Багатопоточність»

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

Перевірив:
Пугачев Р.В.

Харків – 2020

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

1 ВИМОГИ

1.1 Розробник

Кліщов Б. Р., КИТ-118в

1.2 Загальне завдання

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

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

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

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

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Були розроблені класи для роботи з багатопоточністю.

2.2 Ієрархія та структура класів

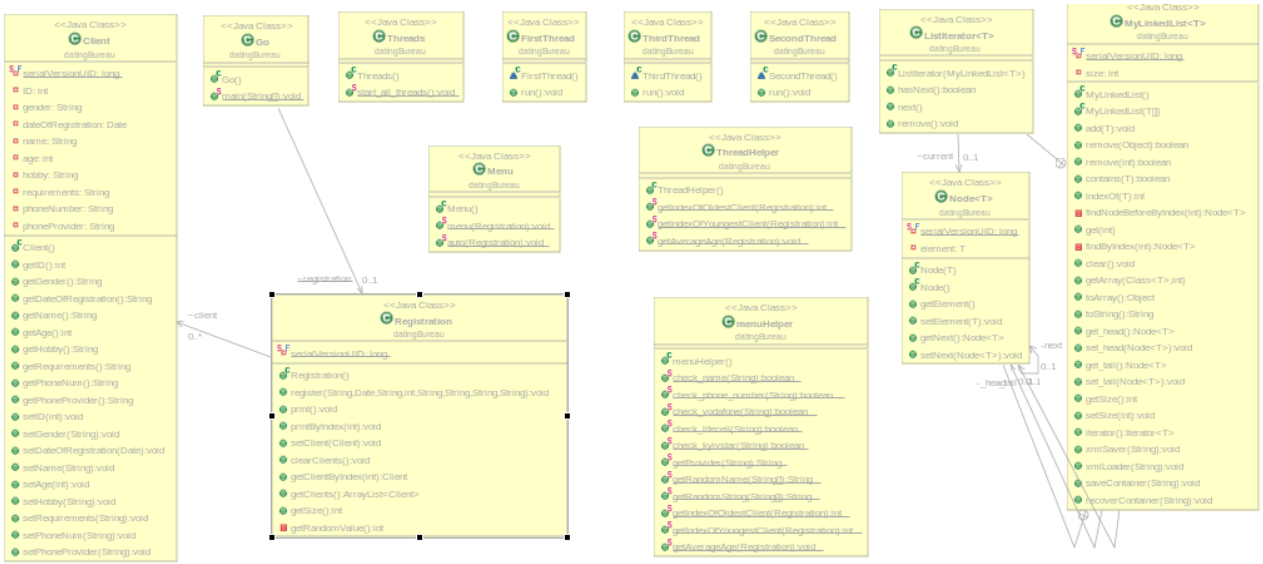

Рисунок 1. Діаграми класів

2.3 Важливі фрагменти програми
Час, відведений на виконання програми генерується автоматично.

```java
public class Threads {
    public static void start_all_threads() throws InterruptedException {
        System.out.println("Set the timer [0 - 100 000 ms]: ");
        Random random = new Random ();
        int timer_num = random.nextInt(100000);
        System.out.println("Starting all threads..." + "\n");

        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();
        Thread.sleep(timer_num);
        t1.interrupt();
        t2.interrupt();
        t3.interrupt();
        System.out.println("Finishing all threads...");
    }

}
```

Рисунок 2. Стартування усіх процесів

```java
public static void auto(Registration registration) throws InterruptedException {
    Random random = new Random ();
    String[] gender = {"male", "female"};
    String[] name = {"John", "Victor", "James", "Lana", "Nika", "Albert"};
    String[] hobby = {"fishing", "football", "chess", "cooking", "fighting", "games"};
    String[] requirements = {"honestly", "kindness", "sense of humor"};
    String[] phoneNums = {"+38(067)-106-46-38", "+38(066)-697-32-99", "+38(093)-782-39-50", "+38(097)-782-39-50",
                          "+38(099)-782-39-50", "+38(063)-782-39-50"};
    String phoneNumber;

    for(int i = 0; i < 100000; i++) {
        phoneNumber =  menuHelper.getRandomString(phoneNums);
        registration.register(
                menuHelper.getRandomString(gender),
                new Date(),
                menuHelper.getRandomName(name),
                random.nextInt(40) + 16,
                menuHelper.getRandomString(hobby),
                menuHelper.getRandomString(requirements),
                phoneNumber,
                menuHelper.getProvider(phoneNumber));
    }
    Threads.start_all_threads();
```

Рисунок 3. Регістрація 100000 користувачів

```java
class FirstThread implements Runnable {
    public void run() {

        System.out.println("First Thread: I`m started" + "\n");
        try {
            if (!Thread.currentThread().isInterrupted()) {
                ThreadHelper.getAverageAge(Go.registration);
                System.out.println("First thread: I`m finished!!!" + "\n");
            } else {
                throw new InterruptedException();
            }
        } catch (InterruptedException e) {
            System.out.println("First Thread is interrupted");
        }
    }
}
```

Рисунок 4. Перший процес

3 ВАРІАНТИ ВИКОРИСТАННЯ

```
Set the timer [0 - 100 000 ms]:
Starting all threads...

First Thread: I`m started

Second Thread: I`m started

Third Thread: I`m started

Finishing all threads...
First Thread is interrupted
First Thread is interrupted
First Thread is interrupted
```

Рисунок 5. У разі якщо процес не встигне

```
Set the timer [0 - 100 000 ms]:
Starting all threads...

First Thread: I`m started

Second Thread: I`m started

Third Thread: I`m started

Average age: 35

First thread: I`m finished!!!

Secound thread: I`m finished!!!

Third thread: I`m finished!!!
```

Рисунок 6. У разі якщо процес встигне

ВИСНОВКИ

Я ознайомився з моделлю потоків Java, організував паралельне виконання декількох частин програми.