14. Паралельне виконання. Ефективність використання

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

1 ВИМОГИ

1.1 Розробник

Інформація про розробника:

- Шарма Олександр Раджнішович
- НТУ “ХПІ” НТУ “ХПІ” КІТ-119б
- Варіант 2(25)

1.2 Загальне завдання

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.

1.3 Задача

Кадрове агентство. Знайти всі вакансії, де потрібні викладачі (педагоги, вчителі) зі стажем не менше 10 років, які знають англійську мову та володіють автомобілем.

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Композиція, інкапсуляція.

2.2 Ієрархія та структура даних

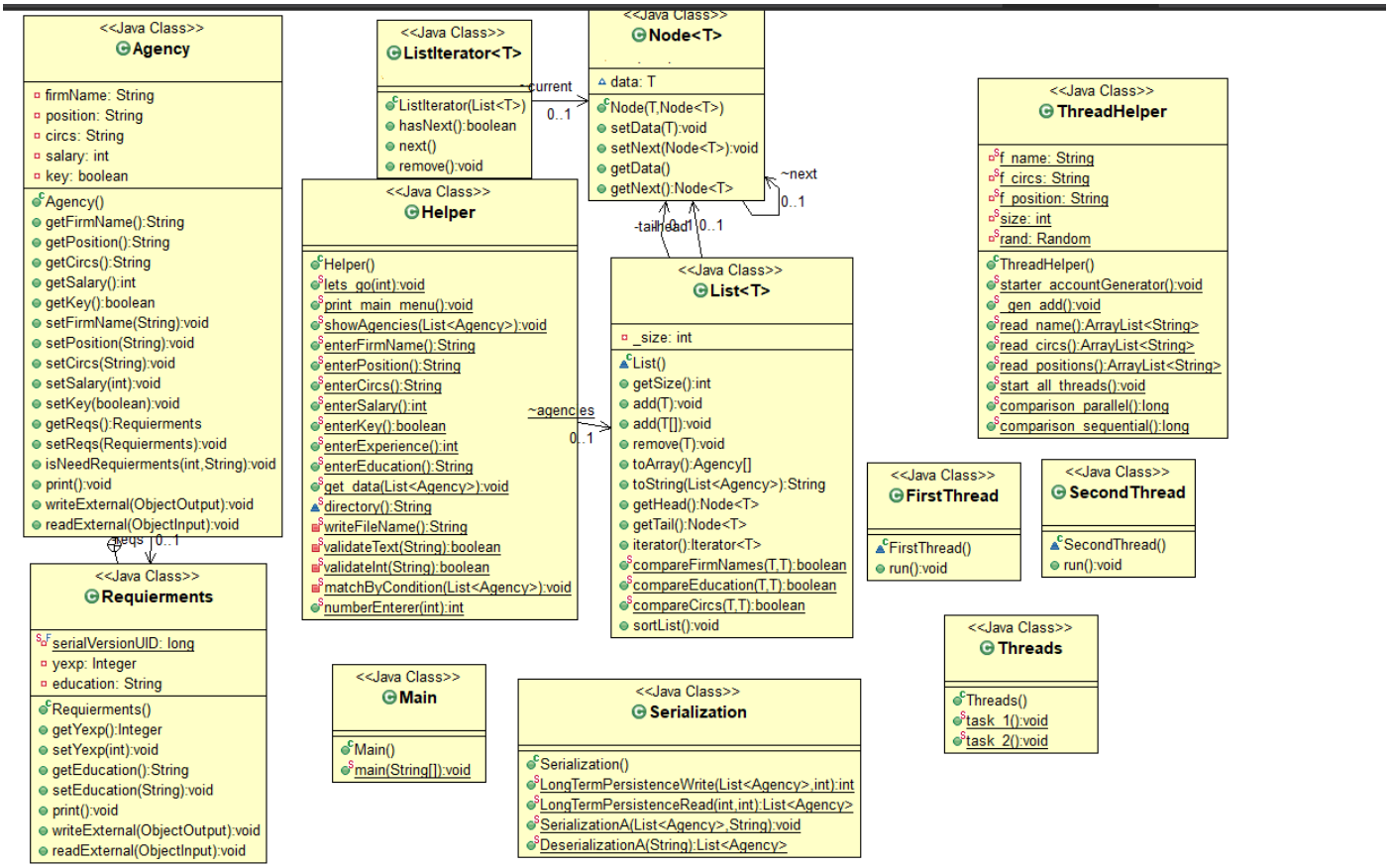

Рисунок 1 – діаграма класів

2.3 Важливі фрагменти програми

```java
public static long comparison_parallel() {
    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");

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

    try {
        t1.join();
        t2.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("Finishing all threads...");
    return System.currentTimeMillis() - time_start;
}
public static long comparison_sequential() {
    long time_start = System.currentTimeMillis();
    System.out.println("Starting sequence...");
    try {
        Threads.task_1();
        Threads.task_2();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("Finishing sequence...");
    return System.currentTimeMillis() - time_start;
```

Рисунок 2 – два методи для обробки послідовно та паралельно

3 ВАРІАНТИ ВИКОРИСТАННЯ

Програма створена для роботи з прикладною задачею. Для коректної роботи були реалізовані методи введення та отримання даних, також дані приховані від користувача, щоб не порушувати суттєвість об’єкту.

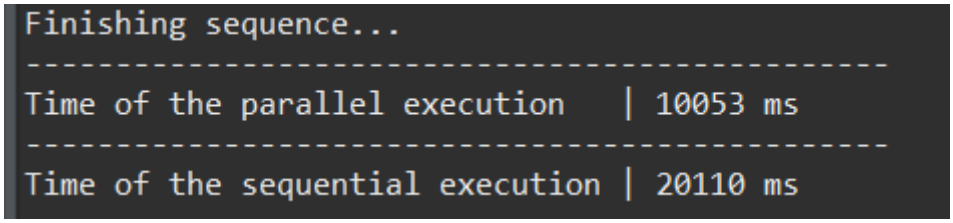

Рисунок 3 – Порівняння часу виконання обробки

ВИСНОВКИ

В даній лабораторній роботі було розроблено методи паралельної обробки даних. Набуто навичок об’єктно-орієнтованого підходу.