Лабораторна робота №6.

Мета:

- Ознайомлення з моделлю потоків Java .
- Організація паралельного виконання декількох частин програми.
- Вимірювання часу паралельних та послідовних обчислень.
- Демонстрація ефективності паралельної обробки.

1. 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.

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

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

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

8. 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.

1.1 Розробник

Левицький Богдан, КН-108

1.2 Задача

Продемонструвати паралельну обробку контейнера

2 Опис програми

Програма працює з багатопотоковістю, перевіряє та порівнює роботи з багатопоточністю та без неї за допомогою простих алгоритмів, які не змінюють початкову колекцію.

2.1 Засоби ООП

Були використані різні класи та методи, структури даних та модифікатори доступу також бібліотека Thread

2.2 Ієрархія та структура класів

1.Клас `Main`, який викликає всі методи та класи створенні користувачем

2.Клас `ListArray` який зберігає та обробляє стрічки

3.Класи з простими алгоритмами

3.1. Клас `Words` підраховує кількість n-ої букв у слові

3.2. Клас `Min` знаходить найменше слово у контейнері

3.3. Клас `MaxTest` знаходить найбільше слово у контейнері

2.3 Важливі фрагменти програми.

Words

```java
@Override
public void run()
{
    time = System.currentTimeMillis();
    for(int i = 0;i < SIZE;i++)
    {
        if(System.currentTimeMillis() - time > timeEnd)
            thread.interrupt();
        try
        {
            thread.sleep(500);
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
        if(container.at(i).startsWith(Character.toString(letter)))
        {
            numOfWords++;
        }
    }
    System.out.println("Number of words starts with " + letter + " " + numOfWords);
    time = System.currentTimeMillis() - time;
}
```

MaxTest

```java
public void run()
{
    time = System.currentTimeMillis();
    for(int i = 0;i < SIZE;i++)
    {
        if(System.currentTimeMillis() - time > timeEnd)
            thread.interrupt();
        try
        {
            thread.sleep(500);
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }

        if(max < container.at(i).length())
        {
            max = container.at(i).length();
        }
    }
    System.out.println("MAX - " + max);
    time = System.currentTimeMillis() - time;
}
```

Min

```java
@Override
public void run() {
    time = System.currentTimeMillis();
    for(int i = 0;i < SIZE;i++)
    {
        if(System.currentTimeMillis() - time > timeEnd)
            thread.interrupt();

        try
        {
            thread.sleep(500);
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }

        if(min > container.at(i).length())
        {
            min = container.at(i).length();
        }

    }
    System.out.println("MIN - " + min);
    time = System.currentTimeMillis() - time;
}
```

2.4 Варіанти використання

Можна використовувати для показу різниці роботи з багатопоточністю й без

ВИСНОВКИ

Ми ознайомились з моделлю потоків в Java, організували паралельне виконання декількох частин програми, виміряли час паралельних та послідовних обчислень та продемонстрували ефективність паралельної обробки