Тема роботи

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

ВИМОГИ

Розробник:

Подоба Арсен Мирославович
КН-108

Загальне завдання:

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.

ОПИС ПРОГРАМИ

Засоби ООП
Декомпозиція для розділення завдання між класами.

Важливі фрагменти програми

```
public class
ParelelThread
                        {
                            RusWords vocabluary;
                            int mLimit;
                            static final double DEVIDER = 1000000;


                            ParelelThread(){
                                vocabluary = new RusWords();
                                vocabluary.fillContainer();
                            }
```

```
   public void setmLimit(int mLimit) { this.mLimit =
mLimit; }


   public void startThread(){
       Thread longestWord = new Thread(new
FinderLongestWord(vocabluary.getRusWords()));
       Thread wordsC = new Thread(new
FinderNumberOfWordsWhichStartC(vocabluary.getRusWords())
);
       Thread uniq = new Thread(new
FinderUniqWord(vocabluary.getRusWords()));


       long start = System.nanoTime();
       int count = 0;


       wordsC.start();
       longestWord.start();
       uniq.start();


       if(wordsC.isAlive() || longestWord.isAlive() ||
uniq.isAlive()){
           try {
               if(mLimit > 0)
               {
                   if((double)(System.nanoTime() -
start)/DEVIDER > mLimit)
                   {
                       count++;
                       wordsC.stop();
                   }
                   else
                       wordsC.join();
```

```
                }
                else
                    wordsC.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }


            try {
                if(mLimit > 0)
                {
                    if((double)(System.nanoTime() -
start)/DEVIDER > mLimit)
                    {
                        count++;
                        uniq.stop();
                    }
                    else
```

ВИСНОВКИ

В ході лабораторної роботи, я навчився розробляти алгоритми для паралельної обробки контейнерів і засікати час виконання.