1.Тема роботи

Паралельне виконання. Багатопоточність. Ефективність використання.

Мета

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

Розробник: Бердник Д.І КН-108

1-Варіант

2.Загальне завдання

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.

Опис програми

Я створив 3 класи кожен Extends Thread під якусь лінійну функцію лінійної обробки великого обсягу даних такі як пошук кількості even/odd numbers і чисел менших за 50

Важливі фрагменти програми

```
public class Main{

    private static final double DIVIDER = 1_000_000;

    public static void main(String[] args) throws InterruptedExcepti
        Scanner in = new Scanner(System.in);
        Random rgen = new Random();
        int[] arr = new int[5000000];
        for(int i = 0; i<arr.length;i++) {
            int random = rgen.nextInt(100);
            arr[i] = random;
        }
        //count num of even numbers
        Task1 t1 = new Task1(arr);
        //count num of odd numbers
        Task2 t2 = new Task2(arr);
        //count num of numbers <= 50
        Task3 t3 = new Task3(arr);
        System.out.println("m - multithading \ng - gradualy");
        char ch = in.next().charAt(0);
        System.out.println("Type limit time in ms: ");
        float limit = in.nextFloat();
```

Використання програми

```
m - multithading
g - gradualy
m
Type limit time in ms:
100
Result task 1 :2499970
Result task 2 :2500030
Result task 3 :2549830
Multithread - 19.8094
```

```
m - multithading
g - gradualy
g
Type limit time in ms:
100
Result task 1 :2500046
Result task 2 :2499954
Result task 3 :2550113
Gradually - 34.9723
```