**Laborator 1**

**-documentație-**

Cerință: Implementati in Java si C++ (11 sau >11)

A) adunarea a doua ‘numere mari’

1) secvential

2) paralel - folosind multithreading

2.1 varianta simpla

2.2 varianta optimizata

**Algoritm secvențial**: se impart cele doua string-uri corespunzatorare numerelor in grupuri de cel mult

18 cifre si se creeaza doua array-uri cu elemente de tip long (cu cel mult 18 cifre fiecare). Se parcurg

array-urile in ordine inversa si se aduna elementele corespunzatoare aceleiasi pozitii si cu elementul

din vectorul rezultat(initializat cu 0) si se adauga restul impartirii la 1000000000000000000L in

vectorul rezultat, iar restul impartirii se insereaza in vectorul rezultat pe pozitia urmatoare(mai mica -

vectorul e parcurs in ordine inversa).

**Algoritm paralel varianta simpla**: se creeaza doi vectori ce contin cifrele celor doua numere.

Se calculeaza numarul de operatii pe care trebuie sa le faca un thread. Se pornesc o serie de

thread-uri care vor calcula de la o pozitie pana la o alta pozitie suma a cate 2 elemente din vector si

va insera pe pozitia corespunzatoare in vectorul rezultat restul impartirii, catul fiind inserat intr-un

vector de transporturi. Dupa terminarea tuturor thread-urilor se parcurge secvential vectorul de

transporturi si se adauga la vectorul rezultat.

**Algoritm paralel varianta optimizata**: se creeaza doi vectori ce contin cifrele celor doua numere.

Se calculeaza numarul de operatii pe care trebuie sa le faca un thread. Se pornesc o serie de

thread-uri care vor insera un vector C(vector care ne spune daca adunarea cifrelor de

pe acea pozitie corespunzatorare celor doua numere numere poate avea transport). Dupa terminarea

acestor thread-uri se pornesc altele care vor calcula vectorul de prefixe(vector care ne spune daca

acea pozitie va avea transport sau nu, in functie de vectorul C). La terminarea thread-urilor se

porneste o alta serie de thread-uri care va calcula vectorul de transporturi(vector care ne spune

valoarea transportului pentru acea pozitie). In final se pornesc o serie de thread-uri care aduna

transporturile elementelor corespunzatoare din vectorul rezultat.

Diagrama de clase: Java

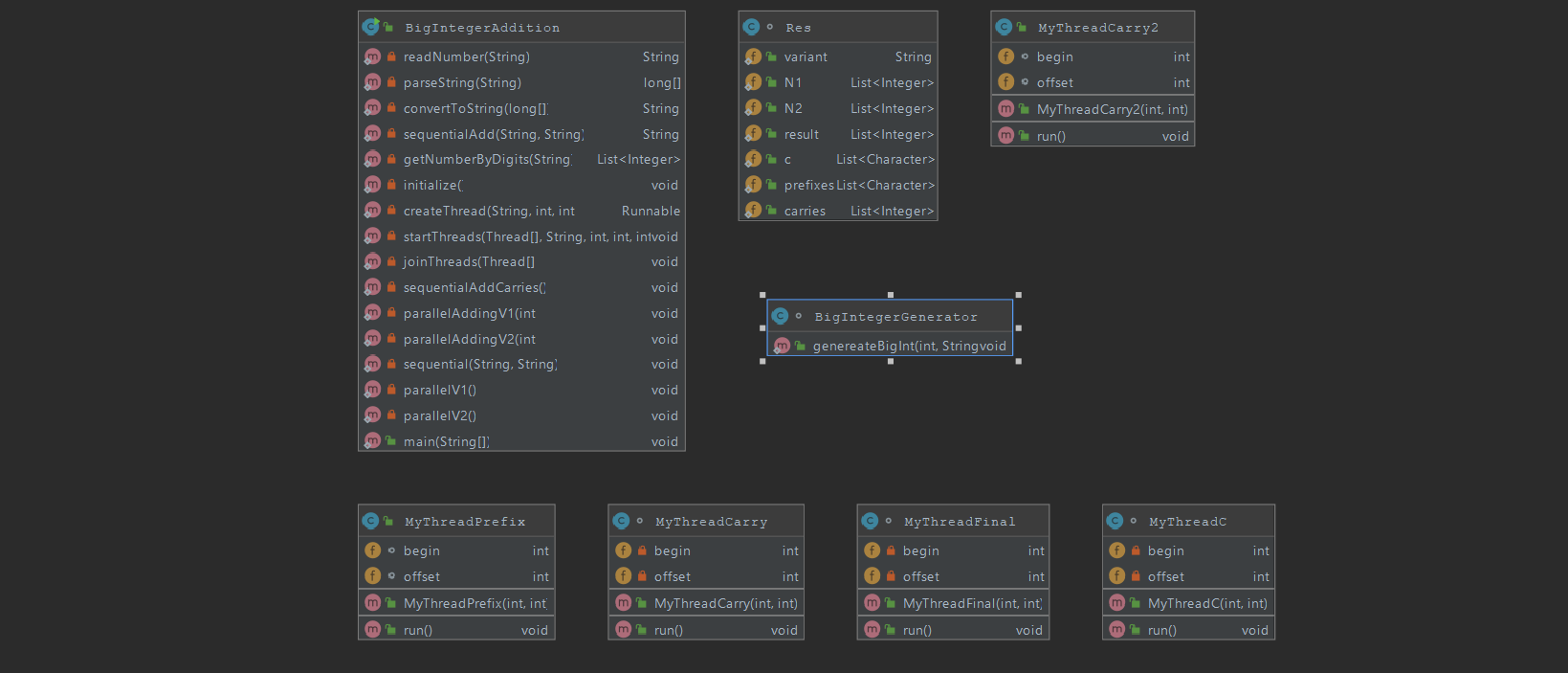
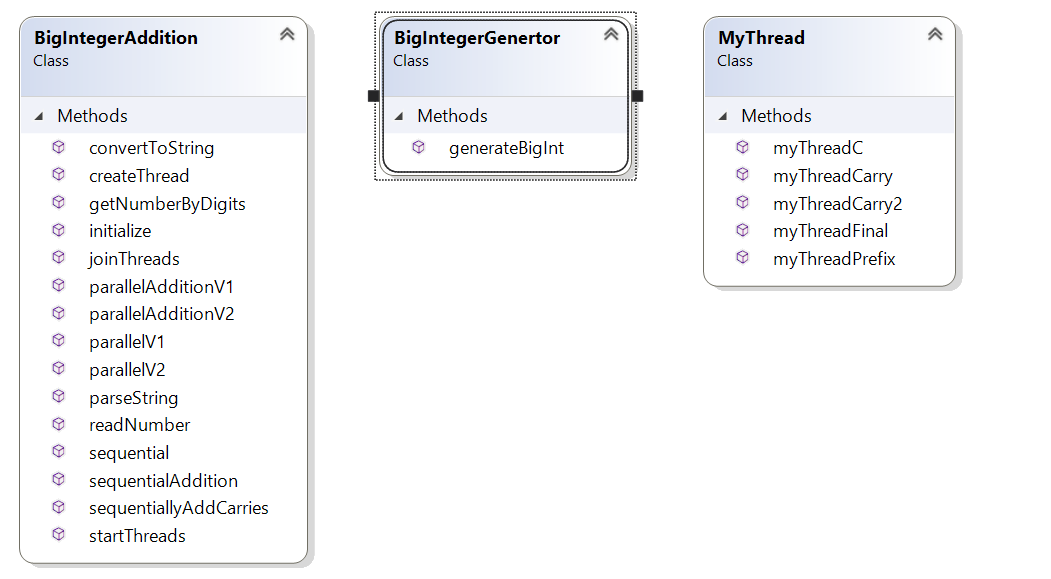


Diagrama de clase: C++



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr. cifre | Limbaj | Nr.thread-uri | Secvential | Paralel V1 | Paralel V2 |
| 5 | JAVA | 2 | 0.01 | 0.002 | 0.008 |
| 100 | JAVA | 10 | 0.019 | 0.004 | 0.016 |
| 1000 | JAVA | 15 | 0.015 | 0. 007 | 0.021 |
| 100000 | JAVA | 50 | 0.274 | 0.064 | 0.086 |
| 1000000 | JAVA | 100 | 13.997 | 0.232 | 0.356 |
| 5 | C++ | 2 | 0.0673224 | 0.0942579 | 0.066248 |
| 100 | C++ | 5 | 0.0114523 | 0.0077505 | 0.0138718 |
| 100 | C++ | 2 | ------------- | 0.0692002 | 0.043939 |
| 10000 | C++ | 3 | 0.0580401 | 0.0244115 | 0.0375611 |
| 10000 | C++ | 5 | ------------- | 0.0259125 | 0.0466295 |