LABORATOR 1 - PPD

Analiza Cerintelor:

1. Cerinte:
   Implementati in limbajul JAVA urmatoarele probleme:
   A. adunarea a doua matrice de dimensiune N*M.
   B. Inmultirea a doua matrice de dimensiune (nxk), respectiv (k,m) folosind multithreading.

2. Constrangeri:
   A. Numărul de threaduri **p** trebuie să fie un parametru care poate fi citit înainte de începerea execuţiei.
   B. Datele de intrare corespunzătoare elementelor matricilor **se vor citi din fişiere** (care au fost ulterior create folosind generare aleatoare de numere)!
   C. La sârşitul programului se va afişa timpul global/total **T** de execuţie corespunzător operaţiei de adunare/înmulţire de matrice.
   D. Se va folosi o încărcare echilibrate de calcul pe fiecare thread.

Proiectare:

1. Clase:
   a. **StartApp** - reprezinta clasa de pornire a aplicatie in care se face citirea din fisier a datelor folosind metode specifice din clasa CitireMatrice, cat si impartirea matricilor in dimensiuni aproximativ egale pentru a se realiza adunarea/inmultirea folosind threaduri. Aceste metode sunt :
      i. `public static void inmultire(int threadsNum)`
      ii. `public static void adunare(int threadsNum)`
   b. Clasele **ThreadAdunare** si **ThreadInmultir**e sunt clasele ce extind clasa Thread java.lang si ele sunt responsabile de suprascrierea metodei run() specifica threadurilor astfel incat suma/inmultirea sa se faca pe respectivul thread intre anumite limite (in acest caz doar anumite linii ale matricilor vor fi incluse in calculul efectiv).
   c. Clasa **GenerareMatrice** este cea care se ocupa de generearea unor numere aleatorii (aici cu valori intre 0 si 5000) si apoi salvarea acestor valori sub forma de matrice in 2 fisiere - cate unul pentru fiecare matrice.
   d. Clasa **CitireMatrice** este clasa responsabila de citirea matricilor din fisierele specificate ale caror denumire se transmite ca si parametru metodei responsabile de citire din aceasta clasa.

2. Functii:

e. Cea mai importanta functie pentru operatia de adunare este:

```
int nrLiniiPerThread = LINII / threadsNum;
int restLinii = LINII % threadsNum;
int nrLiniiAux = nrLiniiPerThread;
int linieInceput = 0;
int linieSfarsit = 0;
matRezultatAdunare = new int[LINII][COLOANE];

double timpInceput = System.nanoTime();
for (int i=0; i<threadsNum; i++){
    if(restLinii != 0){
        nrLiniiPerThread = nrLiniiAux + 1;
        restLinii--;
    }
    else{
        nrLiniiPerThread = nrLiniiAux;
    }
    linieSfarsit += nrLiniiPerThread;
    new ThreadAdunare(mat1, mat2, matRezultatAdunare, linieInceput, linieSfarsit).run();
    linieInceput += nrLiniiPerThread;
}
```

f. Cea mai importanta functie pentru operatia de inmultire este:

```
if(LINII != COLOANE){
    System.out.println("Dimensiunile matricilor nu permit INMULTIREA");
    return;
}

ThreadInmultire[] threaduri = new ThreadInmultire[threadsNum];
int nrLiniiPerThread = LINII / threadsNum;
int restLinii = LINII % threadsNum;
int nrLiniiAux = nrLiniiPerThread;
int linieInceput = 0;
int linieSfarsit = 0;
int indexThread = 0;
matRezultatInmultire = new int[LINII][COLOANE];

double timpInceput = System.nanoTime();
for (int i=0; i<threadsNum; i++) {
    if (restLinii != 0) {
        nrLiniiPerThread = nrLiniiAux + 1;
        restLinii--;
    } else {
        nrLiniiPerThread = nrLiniiAux;
    }
    linieSfarsit += nrLiniiPerThread;
    threaduri[indexThread] = new ThreadInmultire(mat1, mat2, matRezultatInmultire, linieInceput, linieSfarsit)
    threaduri[indexThread].start();
    indexThread++;
    linieInceput += nrLiniiPerThread;
}
for (int i = 0; i < threadsNum; i++){
    try {
```

Relatii cu specificarea lor:

Cele doua functii in principal au rolul de a impartii matricea in parti mai mici pentru a putea fi transimsa fiecarui thread instantiat asfel incat niciunul dintre threaduri sa nu aiba nimic de lucru dar nici unul sa fie cu mult mai ocupat decat altul. De aceea incercam sa construim parti egale de calcul pentru numarul de threaduri disponibil/introdus de user.

Cazuri de testare:

Dimensiune Matrice	**Nr. Threaduri**	**Timp de Executie Adunare | Inmultire (nanosec)**		**Observatii Sistem**
5X5	5	4.2502319E7	3132133.0	• Intel Core i7-5500U CPU @ 2.40GHz • 8.00 GB RAM • 64bit SO • # of Cores 2 • # of Threads 4 • Processor Base Frequency 2.40 GHz • Max Turbo Frequency 3.00 GHz • Cache 4 MB • Bus Speed 5 GT/s DMI2 • TDP 15 W • Configurable TDP-down Frequency 600 MHz • Configurable TDP-down 7.5 W
5X5	2	3.8186015E7	2256727.0	
10X10	10	3.9501049E7	6116725.0	
10X10	5	4.4312147E7	4681948.0	
1000X1000	1000	1.16698021E8	2.727277393E9	
1000X1000	500	6.5556678E7	2.00441723E9	
1000X1000	200	8.3508707E7	1.942536232E9	