­­­Documentatie Laborator 1

**Cerinta pes curt:**

Sa se implementezein Java un program care sa adune sis a inmulteasca 2 matrici atat secvential cat si paralele. Datale de intrare se citest din fisier si sunt generate random. Dim. Matricii si numarul de threaduri pot fi dati ca parametrii.

Datele trebuie sa fie impartite echilibrat celor n threaduri.

In final trebuie afisate timburile globale celor2 executii.

**Detalii de implementare:**

\*Diagrama de clase atasata

Am folosit o clasa wrapper deasura unui array bidimensional **Matrix<T>,** clasa ofera metode auxiliare pe langa accesarea datelor (ex: clear(), toString(), getRow() etc…) pentru a facilita implementare.

Pentru a facilita implementare am creat diferete clase auxiliare: **Utils, Constants, FileManager.**

Clasele **BinaryOperatorThread** si **MultiplicationThread** extend **AbstractMatrixThread,** care la randul sau extinde clasa **Thread**. Fiecare clasa suprascrie metoda run(). BinaryOperatorThread poate primi orice operator binary, MultiplicationThread se ocupa doar de inmultire.

Clasele **AbstracManager, GenericOperationManager si MultiplicationManager** combina restul claselor si se ocupa cu executia efectiva a programelor.

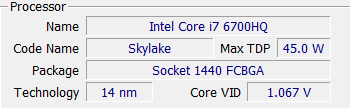
OBS: majoritatea claselor sunt facute cu Generics astel in fac ofera posibilitatea de a lucre sip e alte tipuri de date. DOAR inmultirea este limitata la numere intregi.

De asemenea numarul de coloane, randuri si threaduri sunt citite de la tastatura.

**Un flow al programului:**

1. FileManager creeaza fisierul
2. Se citesc de la tastatura parametrii
3. Se genereaza cele 2 matrici
4. Se executa prin intermediul managerului executia operatiei secventiale
5. Se printeaza rezultatele in fisier
6. Se creeaza threadurile si se executa paralel operatia
7. Se printeaza rezultatele in fisier

**Detalii masina:**

****

**Cazuri de testare adunare:**

|  |  |  |
| --- | --- | --- |
| Dimensiune Matrice | Nr. Thread-uri | Timp (ms) |
| 10x10 | 1 | 1 |
| 10x10 | 2 | 16 |
| 10x10 | 4 | 18 |
| 10x10 | 8 | 21 |
| 10x10 | 256 | 38 |
| 10x10 | 512 | 62 |
| 10x10 | 1024 | 101 |
|  |  |  |
| 500x500 | 1 | 19 |
| 500x500 | 2 | 23 |
| 500x500 | 4 | 33 |
| 500x500 | 8 | 35 |
| 500x500 | 256 | 45 |
| 500x500 | 512 | 65 |
| 500x500 | 1024 | 109 |
|  |  |  |
| 10000x10000 | 1 | ~4500 |
| 10000x10000 | 2 | 540 |
| 10000x10000 | 4 | 379 |
| 10000x10000 | 8 | 292 |
| 10000x10000 | 256 | 336 |
| 10000x10000 | 512 | 396 |
| 10000x10000 | 1024 | 2686 |

Cazuri de testare inmultire:

|  |  |  |
| --- | --- | --- |
| Dimensiune Matrice | Nr. Thread-uri | Timp (ms) |
| 10x10 | 1 | 1 |
| 10x10 | 2 | 17 |
| 10x10 | 4 | 19 |
| 10x10 | 8 | 24 |
| 10x10 | 256 | 43 |
| 10x10 | 512 | 68 |
| 10x10 | 1024 | 117 |
|  |  |  |
| 500x500 | 1 | 1463 |
| 500x500 | 2 | 851 |
| 500x500 | 4 | 525 |
| 500x500 | 8 | 458 |
| 500x500 | 256 | 617 |
| 500x500 | 512 | 1632 |
| 500x500 | 1024 | 1733 |
|  |  |  |
| 1000x1000 | 1 | 10583 |
| 1000x1000 | 2 | 5493 |
| 1000x1000 | 4 | 4407 |
| 1000x1000 | 8 | 3408 |
| 1000x1000 | 256 | 3567 |
| 1000x1000 | 512 | 3690 |
| 1000x1000 | 1024 | 3780 |