Documentatie Laborator 5

**Analiza cerintelor:**

Aceeasi problema ca si la Laborator 4 dar cu urmatoarele modificari:

- Dintre cele p threaduri p1 sunt cititoare si p2 sunt de tip ‘worker’ (preiau din coada si adauga in lista rezultat) – p=p1+p2

- Coada in care se adauga monoamele citite are o capacitate maxima = MAX

- Implementarea pentru producator consumator (writer/reader) trebuie sa foloseasca mecanisme de tip wait notify (nu busy waiting)

- Sincronizarea operatiile cu lista inlantuita se fac la nivel de nod nu pentru intreaga lista!

o Sincronizarile se vor face folosind variabile de tip Lock asociate cu fiecare nod.

o Daca rezulta un nod cu coeficient nul acest nod se va sterge din lista!

Observatie: Este obligatoriu sa se foloseasca sablonul ‘producator-consumator’ pentru legatura dintre threadurile ‘reader’ si cele de tip ‘worker’ --- wait|notify.

Limbaj: la alegere intre Java si C++

Analiza timpului de executie pentru urmatoarele cazuri:

1) Rezolvare secventiala

2) 10 polinoame fiecare cu gradul maxim 1000 si cu maxim 50 monoame si MAX=20

a. p = 4, 6, 8 ; si p1=2 , p1=3

b. secvential ; si p1=2 , p1=3

3) 5 polinoame fiecare cu gradul maxim 10000 si cu maxim 100 monoame si MAX=30

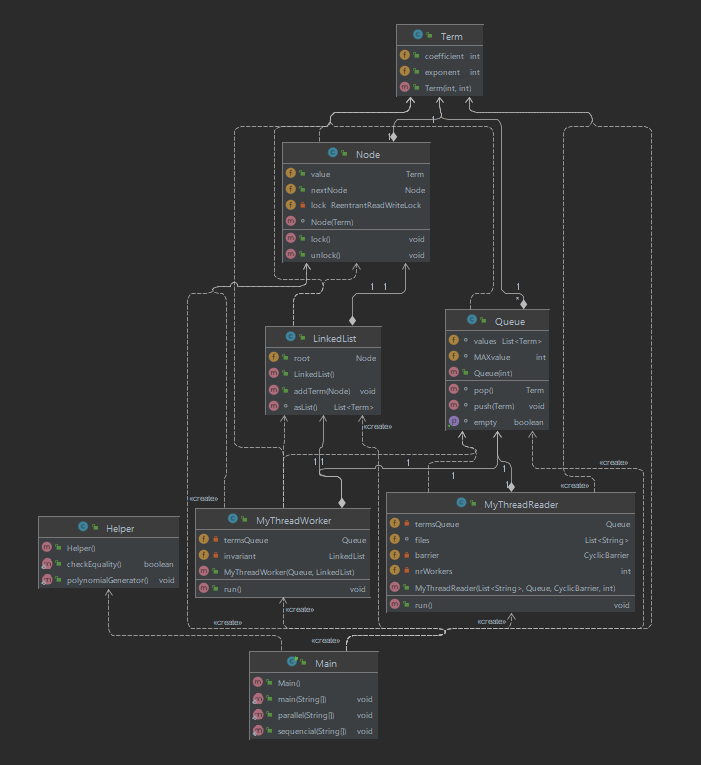
a. p = 4, 6, 8 ; si p1=2 , p1=3

b. secvential ; si p1=2 , p1=3

Analiza comparativa a performantei –Lab 4) versus Lab 5)

**Proiectare:**

**Java:**



**main(String[])-** functia principala de intrare in program

**sequencial(String[])** – functia de calculare secventiala a adunarii polinoamelor

**parallel(String[])** – functia de calculare paralela a adunarii polinoamelor

**Helper –** o clasa ce contine functii ajutatoare

**polynomialGenerator()** – o functie pentru generarea aleatoare de polinoame

**checkEquality()** – o functie pentru verificarea egalitatii fisierelor calculului secvential si paralel

**MyThreadReader** – clasa de readers ce extinde Thread

**MyThreadWorker** – clasa de workers ce extinde Thread

**run()** – functie suprascrisa din Thread in care se ia un monom din coada si se adauga la rezultat (lista inlantuita)

**Queue –** clasa pentru coada

**push(Term)** – adauga un monom in coada

**pop() –** ia un monom din coada

**LinkedList** – o clasa pentru lista inlantuita de monoame, ordonata dupa exponenti

**addTerm(Node)** – adaugarea unui monom la lista daca nu exista termen cu gradul exponentului sau adunarea la coeficientul corespunzator exponentului si stergerea acestuia daca coeficientul devine 0

**Node** – clasa ce contine monomul si referinta spre nodul urmator

**lock()** – functie pentru blocarea modificarii nodului

**unlock()** - functie pentru deblocarea modificarii nodului

**Term** – monom ce contine exponentul si coeficientul

**Detalii de implementare:**

Se citesc monoamele din fisier si se adauga in coada.Coada are un numar MAX de dimensiune la un moment dat. Atunci cand se ajunge la acest max, se asteapta pana ce se elibereaza din coada, cu ajutorul wait() si notifyAll(). La varianta secventiala, se iau pe rand fisierele si se adauga monoamele in lista inlantuita. Daca nu exista deja termen cu acel exponent sau se aduna la coeficient daca exista deja. Apoi se face scrierea rezultatului in fisierul “outputSeq.txt”.

La varianta paralela, p1 threaduri sunt de tip readers si p2 threaduri sunt de tip workers. Threadurile de tip reader citesc pe rand cate fisiere le sunt alocate si adauga monoamele in coada. Threadurile de tip worker iau din coada cate un monom si il adauga in lista inlantuita. Acesti pasi se repeta pana cand un mai exista monoame in coada,adica workerii dau de valoarea “null”. La final, se scrie rezultatul in fisierul “outputPar.txt”.

**Cazuri de testare:**

| **Input** | **Tip** | **p threads** | **p1** | **Timp** |
| --- | --- | --- | --- | --- |
| polinoame=10, max exponent=1000, monoame=50,  MAX=20 | secvential | 1 |  | 42.26 |  |
|  |
| paralel | 4 | 2 | 53.42 |  |
| 3 | 52.1 |  |
| 6 | 2 | 48.82 |  |
| 3 | 46.51 |  |
| 8 | 2 | 52.34 |  |
| 3 | 49.7 |  |
| polinoame=5, max exponent=10000, monoame=100,  MAX=30 | secvential | 1 |  | 47.55 |
| paralel | 4 | 2 | 53.71 |
| 3 | 50.33 |
| 6 | 2 | 53.25 |
| 3 | 50.77 |
| 8 | 2 | 51.69 |
| 3 | 52.70 |

**Analiza rezultatelor:**

**Lab4 vs Lab5 –**

Secvential vs Paralel –

Atat in cazul paralel cat si in cel secvential, un se observa imbunatatiri majore.In cazul celor 10 polinoame, rezultatul obtinut in cadrul laboratorului 5, pare sa fie putin mai bun, cu cat creste numarul de threaduri. In cazul celor 5 polinoame, rezultatul laboratorului 4 este mult mai bun decat cel obtinut in cadrul laboratorului 5.