**Analiza cerintelor**

Se cere adunarea a n polinoame folosind o implementare multithreading. Polinoamele trebuie reprezentate sub forma de lista inlantuita ordonata dupa exponenti. Sincronizarea operatiilor cu lista inlantuita se realizeaza la nivel de nod. De asemenea, trebuie folosit un numar p1 de threaduri care citesc polinoamele din fisier, astfel incat p1 (threadurile care citesc) + p2 (threadurile care aduna) = p (numarul total de threaduri). Fisierele in care se afla monoamele nu pot contine monoame cu coeficient 0.

**Proiectare**

Am implementat o structura de date de tip lista inlantuita in care metoda de inserare este sincronizata la nivel de nod, folosind o varibila de tipul Lock.

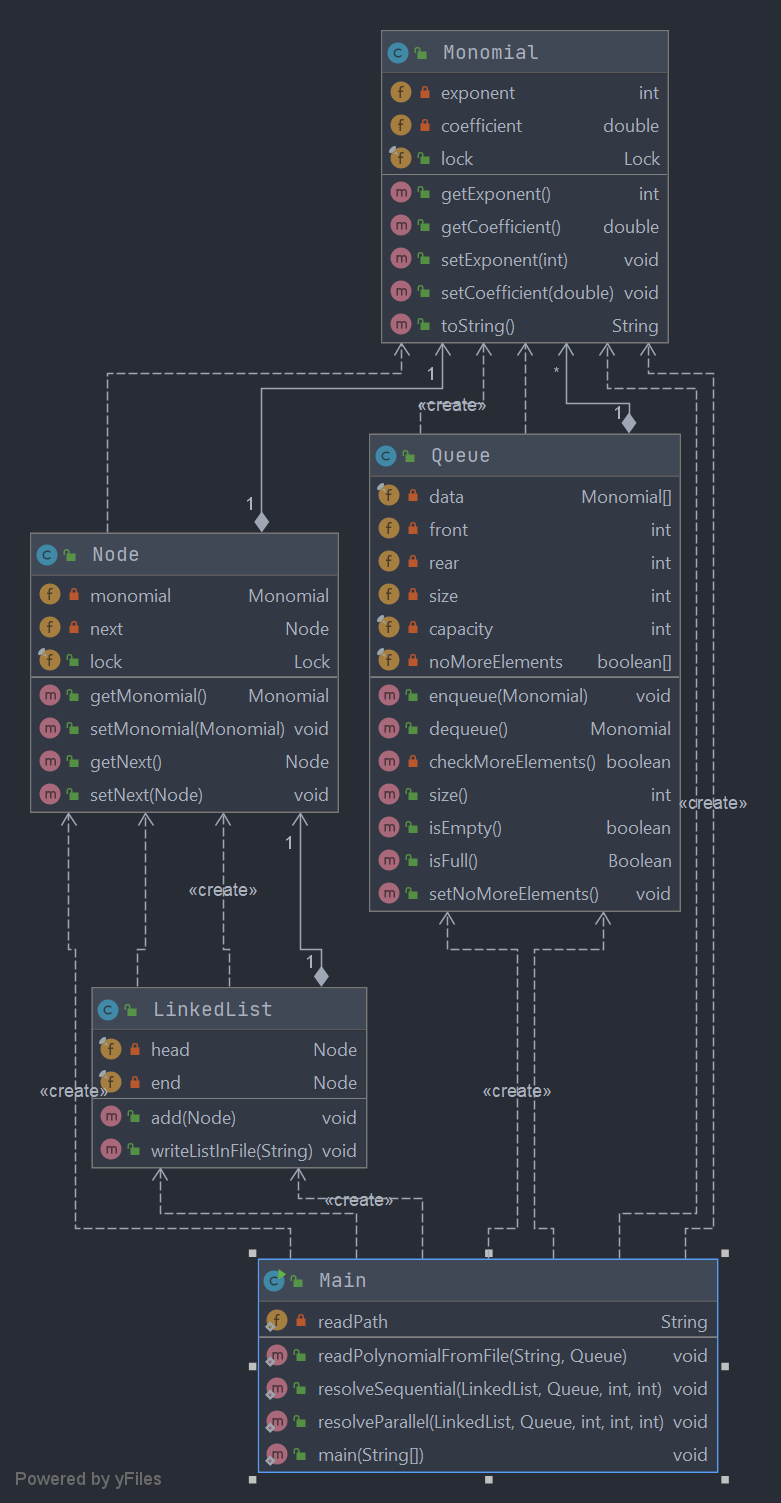
Am implementat o strucutra de date de tip coada in care metodele de adaugare si stergere(returneaza ultimul element si il sterge) sunt sincronizate folosind “synchronized”, dar si “wait” si “notify”.

Pentru programul secvential, toate monoamele din toate fisierele sunt citite in coada. In urmatorul pas, monoamele sunt adunate rand pe rand la polinomul final, folosind lista inlantuita.

Pentru programul paralel, sunt pornite p1 threaduri care citeste monoamele din toate fisiere si le adauga in coada. Numarul de fisiere primit de fiecare thread este calculat astfel: numarul de fisiere este impartit la numarul de threaduri, iar daca apare rest anumite threaduri primesc un fisier in plus. In acelasi timp, restul threadurilor verifica daca exista elemente in coada, iar in cazul in care nu exista, acestea asteapta sa fie notificate de cate primul thread ca a mai fost adaugat inca un element. In cazul in care exista elemente in coada, threadurile aduna elementele in polinomul rezultat.

Pentru a aduna un monom la polinomul rezultat, acesta este salvat in lista inlantuita. Daca lista este vida, se face lock nodului “head” si se adauga nodul. Altfel, se porneste de la primul nod, se blocheaza primul nod si se cauta pozitia in ordine descrescatoare dupa exponent pe care trebuie adaugat noul monom. Dupa gasirea pozitiei, se verificat daca mai exista acest exponent, daca da, se aduna coeficientul, altfel, se insereaza un nou nod.

In final polinomul este scris in fisier.



|  |  |  |  |
| --- | --- | --- | --- |
| **Nr polinoame** | **Nr threads** | | **Timp executie** |
| 10 polinoame fiecare cu gradul maxim 1000 si cu maxim 100 monoame | secvential | p1 = 2 | 194.9 |
| p1 = 3 | 200.6 |
| 4 | p1 = 2 | 191.9 |
| p1 = 3 | 196.9 |
| 6 | p1 = 2 | 199.4 |
| p1 = 3 | 227.1 |
| 8 | p1 = 2 | 197.8 |
| p1 = 3 | 200.8 |
| 5 polinoame fiecare cu gradul maxim 10000 si cu maxim 500 monoame | secvential | p1 = 2 | 339.5 |
| p1 = 3 | 356 |
| 4 | p1 = 2 | 332 |
| p1 = 3 | 348.9 |
| 6 | p1 = 2 | 348.3 |
| p1 = 3 | 397.7 |
| 8 | p1 = 2 | 365.9 |
| p1 = 3 | 396.1 |

Timpi de executie Lab04

|  |  |  |
| --- | --- | --- |
| **Nr polinoame** | **Nr threads** | **Timp executie** |
| 10 polinoame fiecare cu gradul maxim 1000 si cu maxim 100 monoame | secvential | 141.2 |
| 4 | 142.8 |
| 6 | 152.9 |
| 8 | 141.4 |
| 5 polinoame fiecare cu gradul maxim 10000 si cu maxim 500 monoame | secvential | 192.8 |
| 4 | 186.8 |
| 6 | 189.9 |
| 8 | 196.9 |

**Analiza**

* Cei mai buni timpi de executie sunt obtinuti in cazul in care sunt folosite 4 threaduri iar doar 2 dintre acestea sunt reader
* Timpii de executie sunt mai mari in cazurile in care sunt folosite 3 threaduri reader
* Timpii de executie obtinuti in laborotrul 4 sunt mai buni decat cei din laboratorul 5