Multithreading polynomial addition documentation

Requirement:

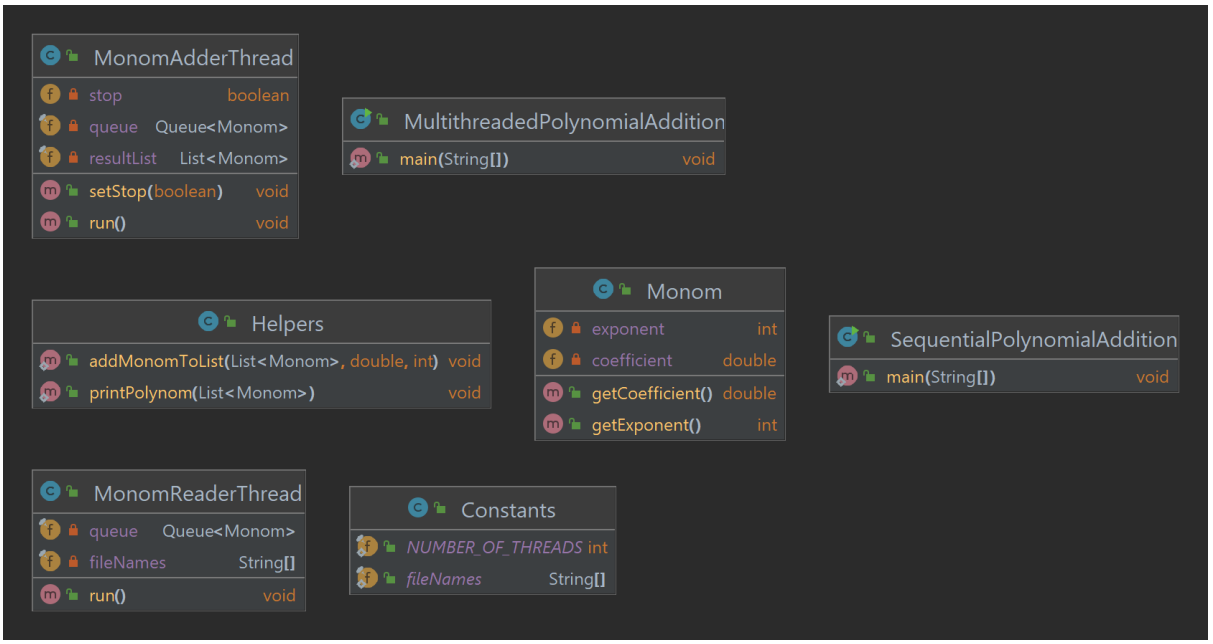
There are n polynomials represented by a list of monomials. The task is to add the polynomials using a multithreading implementation (p threads).

The polynomial is represented in memory by a linked list (one node = one monomial), ordered by the exponents of the monomials, with the following invariant (true predicate at any point in execution) of representation:

- monomials are ordered by exponents
- monomials with a coefficient of 0 are not stored in the list
- there are no two nodes (monomials) with the same exponent
- the polynomials are read from files - one file for each polynomial.
- a file contains information about the coefficient and exponent of each monomial in a polynomial
- the input files are created by generating random numbers

Both a sequential implementation and a multithreading implementation are required, using the producer-consumer pattern and synchronization at the list level.

Project structure



Testing results:

Test case	Number of threads	Execution time (ms)
10 polynoms maximum exponent: 1000 maximum monoms: 50	sequential	15.0156
	4	16.7017
	6	19.3439
	8	21.7994

5 polynoms maximum exponent: 10000 maximum monoms: 100	sequential	15.3746
	4	18.8026
	6	20.181
	8	23.8049

Analysis:

- The execution time is the smallest in the case of sequential implementation. The execution time increases with the number of threads.
- The two tests are similar in terms of execution time, but a small difference is observed, with the first test having a shorter execution time.