# Lab 6: Parallelizing techniques

## Methods:

multiplyPolynomialsSequentialSimple – fill both polynomials and compute the result by iterating over the arrays (n^2) and multiplying terms

multiplyPolynomialsSequentialThreading – fill both polynomials and for each term of the first array run a simpleMultiplication worker

multiplyPolynomialsSequentialKaratsuba – fill both polynomials and call karatsuba

multiplyPolynomialsThreadingKaratsuba

Karatsuba – divide et impera algorithm that has 4 recursive calls that all multiply polynomial terms and then put together the result

fillPolynomial – fill a polynomial with random data

SimpleMultiplicationWorker – runnable class (java) that computes a polynomial multiplied by 1 term (and stores the result in a staticvariable)

KaratsubaWorker – runnable class (java) that computes a Karatsuba recursive call

## Synchronization:

ExecutorService (Java)

Static variable to be stored in

## Performance measurement: