Laboratory 3

Testing:

Both application (the one in java and the one in c++) were tested with matrices with size { } on different number of threads i.e{ }. In the tables below I added the results.

Applications were tested on the following platform: **CPU:** 2.8 GHz quad-core core i7
Nr of threads: 8
OS: Windows 10 RS5
Architecture : 64 bit

An asynchronous call is a method invocation that will be executed in a separate thread (or core or processor); so, the caller of the method does not wait for the result of the execution and continue doing what is next; in this way, the compiler/processor/operating system can optimize the execution of the program and execute several routines at the same time. The standard library provides the mechanisms to perform those asynchronous calls and store the results until the caller will actually need them. It is triggered when the get method is called.

Thread pool provides reusable thread. Thread creation is very bulky process which increases overhead.
How thread pools works:
Store tasks to be executed on threads in a task queue, a) Each thread's loop function waits for a wake-up notification, b) Then removes the first task (if any) from the task queue and executes it. c) Executing a task with the pool enqueues it in the task queue, d) Then notifies one waiting thread (if any are available). e) When a thread completes its task, it checks for any tasks in the queue. If there are none, it goes back to waiting for a wake-up notification.

Results:

Java:

ThreadPool:

NrThreads/ Size of matrix	1		5		50		1000	
	Add	Mult	Add	Mult	Add	Mult	Add	Mult

Future:

NrThreads/ Size of matrix	1		5		50		1000	
	Add	Mult	Add	Mult	Add	Mult	Add	Mult

Cplusplus:

ThreadPool:

NrThreads/ Size of matrix	1		5		50		1000	
	Add	Mult	Add	Mult	Add	Mult	Add	Mult

Future:

NrThreads/ Size of matrix	1		5		50		1000	
	Add	Mult	Add	Mult	Add	Mult	Add	Mult