Project 2 background: Matrix Multiplication using Threads

Given two matrices, A and B, where matrix A contains M rows and K columns and matrix B contains K rows and N columns, the matrix product of A and B is matrix C, where C contains M rows and N columns. The entry in matrix C for row i, column j ($C_{i,j}$) is the sum of the products of the elements for row i in matrix A and column j in matrix B. That is:

$$C_{i,j} = \sum_{n=0}^{K-1} A_{i,n} * B_{n,j}$$

Example: if A is a 3-by-2 matrix and B is a 2-by-3 matrix, element $C_{3,1}$ is the sum of $A_{3,1} * B_{1,1}$ and $A_{3,2} * B_{2,1}$.

For this project, you must calculate each element $C_{i,j}$ in a separate *worker thread*. This will involve creating M * N worker threads. The main – or parent – thread will create and initialize the matrices A and B and create matrix C (dimensions determined by A and B dimensions), which will hold the product of matrices A and B. These matrices will be created in the parent and passed to each worker thread through its constructor's parameter list.

Passing Parameters to each Thread

As stated above, the parent thread will create M * N worker threads, passing each worker the values it needs to use in calculating the matrix product. Our approach is for the main thread to create and initialize the matrices A and B and create matrix C. The main thread will then create the worker threads, passing the three matrices, along with row i and column j, to the constructor for each worker. Thus, the worker thread class (which implements `Runnable`) needs to have instance variables for all five of them. The constructor will simply assign each parameter to its corresponding instance variable. Then the `run` method will calculate the matrix product and store it into the given row and column of result matrix C. There is nothing to return (run is void).

Waiting for Threads to Complete

Once all worker threads have completed, the main thread will output the product contained in matrix C. This requires the main thread to wait for all worker threads to finish before it can output the matrix product. Example: If `worker` is a `Thread` object, then `worker.join()` will suspend the main thread until `worker` has finished. Since you will have an array of worker threads, the `join()` will have to be performed on all of them. Do this using a nested loop.