COMP 3411

Assignment 3

Martin Atanacio - T00684924

Thread Class

```
public class MultiplyMatrices extends Thread {

     // Variables
     private int[][] matrixA, matrixB, matrixResult;
     private int row, col;

     // Constructor
     // Accepts 3 matrices (1 being the result matrix), and the # of rows and cols
     public MultiplyMatrices(int[][] matrixA, int[][] matrixB, int[][] matrixResult, int
row, int col) {

          this.matrixA = matrixA;
          this.matrixB = matrixB;
          this.matrixResult = matrixResult;
          this.row = row;
          this.col = col;

     }

     @Override
     public void run() {

          int sum = 0;

          // finds the sum of the multiplications
          for (int i = 0; i < matrixA[0].length; i++) {
               sum += matrixA[row][i] * matrixB[i][col];
          }

          // store sum in its corresponding position in the matrix
          matrixResult[row][col] = sum;

     }
}
```

Driver Class

```
public class MatrixDriver {

    public static void main(String[] args) {

        // Matrix A set to [3 x 3] size
        int[][] matrixA = {
                { 1, 2, 3 },
                { 4, 5, 6 },
                { 7, 8, 9 }
        };

        // Matrix B set to [3 x 2] size
        int[][] matrixB = {
                { 10, 20 },
                { 30, 40 },
                { 50, 60 }
        };

        int rowsA = matrixA.length; // gets # of rows
        int colsA = matrixA[0].length; // gets # of columns
        int colsB = matrixB[0].length;

        // check if matrixA's cols == matrixB's rows, otherwise can't continue operation
        if (colsA != matrixB.length) {
            System.out.println("Matrix dimensions not compatible.");
            return;
        }

        // calculate size of result matrix [rowsA x colsB]
        int[][] resultMatrix = new int[rowsA][colsB];

        // create a Thread for each entry in result matrix (same size as result matrix)
        Thread[][] threads = new Thread[rowsA][colsB];

        // creates and starts threads to perform the matrix multiplication
        for (int i = 0; i < rowsA; i++) {
            for (int j = 0; j < colsB; j++) {
```

```
     // new thread instance responsible for calculating one entry
                    threads[i][j] = new MultiplyMatrices(matrixA, matrixB, resultMatrix, i,
j);
                    threads[i][j].start(); // start the thread
               }
          }

          // waits for all threads to finish their execution before printing
          try {
               for (int i = 0; i < rowsA; i++) {
                    for (int j = 0; j < colsB; j++) {

                         /*
                          * makes the current thread (main thread) wait for the specified
                          * thread (the thread representing the current element of the
                          * result matrix) to finish its execution
                          */
                         threads[i][j].join();

                    }
               }
          } catch (InterruptedException e) {
               e.printStackTrace();
          }

          // Print the result matrix
          for (int i = 0; i < rowsA; i++) {
               for (int j = 0; j < colsB; j++) {
                    System.out.print(resultMatrix[i][j] + " ");
               }
               System.out.println();
          }
     }
}
```

Output →

```
220 280
490 640
760 1000
```