Cairo University
Faculty of Engineering
Advanced Programming Techniques

Lab - Threading

Requirement (1):
We will use for this requirement class **Matrix** we developed in the first lab with some modifications. All requirements preceded with (*) are already implemented in Lab 1.

Part 1: Class Matrix:

1. **MultiplicationException class (Inner class defined within Matrix class):**
   - **a)** This class should be **static** and **final**.
   - **b)** This class inherits from the built-in class Exception.
   - **c)** It has only public data member **errorMessage.**
   - **d)** The constructor takes as a parameter a string variable, and sets *errorMessage* with this string parameter.
   - **e)** It has only public method **message()** that prints *errorMessage.*

2. **Data Members(*):**
   - **a)** **rows:** A public integer that defines the rows of a matrix.
   - **b)** **cols:** A public integer that defines the columns of a matrix.
   - **c)** **numbers:** A public 2D dimensional array of integers (int[ ][ ]). Do not use lists.

3. **Constructor(*):**
   - **a)** The constructor takes as arguments two parameters M and N, sets the *rows* and *cols* variables of the matrix with M and N respectively.
   - **b)** The constructor initializes the *numbers* array.

4. **print() (*):**
   - **a)** This function prints the matrix row by row, where each row is separated by a new line and each column in separated by a space.

5. **setNumbers() (*):**
   - a) This function takes as a parameter a 1D array of numbers (use built-in arrays and not lists) ONLY.
   - b) If the length of this array is less than the size of the matrix, return.
   - c) Otherwise, set the matrix elements with the passed numbers rows by row.

6. **multiply():**
   - a) This function takes as a **parameter** matrix B, and **returns** the result of multiplication, and throws an exception of type **MultiplicationException.** What should be the prototype of the function?
   - b) Do the necessary checks before the multiplication process. If the multiplication is invalid, then throw an exception of type **MultiplicationException.** Pass to the object an informative error message that you think should be displayed to the user. (i.e. an informative error message should be *"Exception occured while trying to multiply two matrices of dimensions (A,B) and (C,D)"*)

c) Otherwise, implement the matrix multiplication algorithm and return the result. Do not call use any library or package.

7. **Main Function():**
   a) Create three Matrix objects **m1** (3x4), **m2** (4x2), **m3**(2x5)
   b) Create a one-dimensional array **arr**, and initialize it with numbers from 1 to 12.
   c) Set the numbers of m1, m2 and m3 with arr.
   d) Multiply m1 with m2 and print the result. Place the multiplication code in try-catch block.
   e) Multiply m1 with m3 and print the result. Place the multiplication code in try-catch block.
   f) Calculate the time needed for multiplying m1 and m2. [Hint: Use System.currentTimeMillis() or System.nanoTime()]
   g) Create two Matrix objects **m4 (500x500)** and **m5 (500x500).**
   h) Create a one-dimensional array **arr2** of size 250000
   i) Initialize the two matrices with random numbers. [Hint:
      Random rd = new Random() //creates a new random object.
      Loop i on the array **arr2**
      arr2[i] = rd.nextInt()
   J) Repeat e and f and report the time elapsed.

Part 2: Class MultiplicationThread.
This class should implement the Runnable interface.

1. **Data Members:**
   a) **A:** A private Matrix object that represents the first operand in multiplication.
   b) **B:** A private Matrix object that represents the second operand in multiplication.
   c) **result:** A private Matrix object that represents the result of multiplication.

2. **Constructor:**
   a) The constructor takes as arguments two parameters *firstMatrix* and *secondMatrix*, sets the *A* and *B* matrix objects with *firstMatrix* and *secondMatrix* respectively.
   b) The constructor should create the *result* matrix object.

3. **Run():**
   a) This function should check for the name of the thread running it. [Hint: Use Integer.parseInt()]
   b) If the name of the thread is 1, it should multiply the first half of rows of A with matrix B, and place the partial result of multiplication in C in the correct position.
   c) If the name of the thread is 2, it should multiply the second half of rows of A with matrix B, and place the partial result of multiplication in C in the correct position.
   d) Here, you can cheat your implementation in 1.6 with the necessary modifications.
   e) We do not require exception handling here. We assume the inputs are correct.

   Example: If A (4x5) is multiplied by B(5x6) for example
   Thread 1 should multiply A[0:2,:] with B[:,:] and Thread 2 should multiply A[2:4,:] with B[:,:]

If A has odd numbers of rows, you should break the symmetry.

4. **Main Function():**
   a) Create two Matrix objects **m1** (3x4), **m2** (4x2)
   b) Create a one-dimensional array **arr**, and initialize it with numbers from 1 to 12.
   c) Set the numbers of m1 and m2 with arr.
   d) Create two threads of type **MultiplicationThreads** and set their names 1 and 2 respectively. Both threads are initialized with the same m1 and m2.
   e) Start the two threads. Determine if we need to join the threads or not and why?
   f) Calculate the time needed for multiplying m1 and m2 using the two threads. [Hint: Use System.currentTimeMillis() or System.nanoTime()]
   g) Create two Matrix objects **m4 (500x500)** and **m5 (500x500).**
   h) Create a one-dimensional array **arr2** of size 250000
   i) Initialize the two matrices with random numbers.
   j) Repeat e and f and report the time elapsed.
   Hint for random number generation and initialization:
   Random rd = new Random() //creates a new random object.
   Loop i on the array **arr2**
   arr2[i] = rd.nextInt()

Requirement (2):

   a) Check requirement_student.java
   b) There is a bookstore that has different branches, each of them sell a number of books, these books are supplied by one supplier.
   c) A bookstore shouldn't sell a book when the number of books is 0, it should block (wait) and notify the supplier to provide more books.
   d) A supplier shouldn't provide a book when the max count of books is reached, When it provides a book, it should tell(notifies) the bookstores that there are more books available.
   e) Modify the requirement_student.java code to reflect this behaviour.
   f) All threads should execute in parallel, you cannot allow a thread to stop another thread (should guarantee progress).
   g) Follow the 8 TODOs in the code.