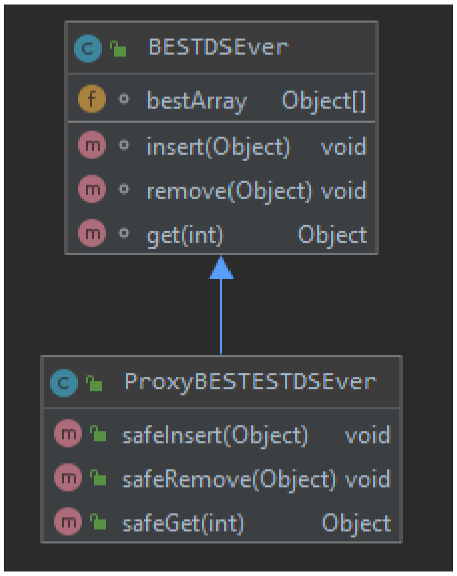
Part1)

Proxy design pattern was utilized instead of decorator or adapter as the intention here is controlling and managing the class. In order to make it thread-safe the class is wrapped with another class where method calls are synchronized.



Part2)

a) Synchronized

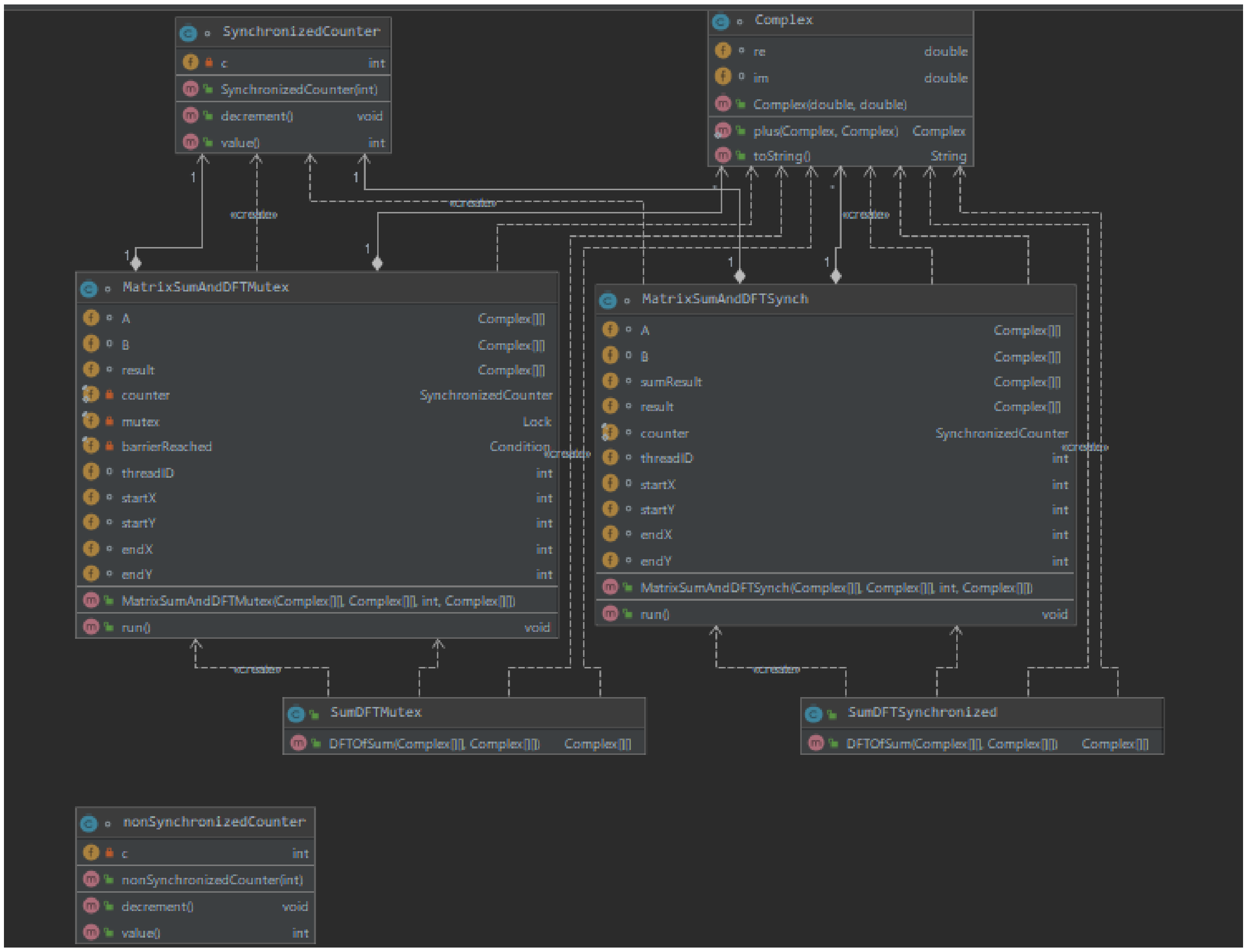
Synchronization barrier problem is solved using **SynchronizedCounter** class which has an integer counter which is initially equal to number of threads and each thread that finishes their sum part decreases this integer and notify other threads. They all wait until the counter is equal to 0. Also note that the DFT solution I found on the internet has time complexity of O(n^4) which is pretty slow. Although there are better ones, I had no time to make any improvements there.

b) Mutex

Synchronization barrier was solved in a similar manner with a counter but without the synchronized keyword, utilized;
**import java.util.concurrent.locks.ReentrantLock;** for mutex and
**import java.util.concurrent.locks.Condition;** for condition variable

to create critical secton and decrease and check the counter. Threads use signalAll(); method when they decrement counter.

Also the counter check for both implementations was done inside a while loop and both wait methods timeout every 100ms to avoid any deadlocks.


SynchronizedCounter
c int
SynchronizedCounter(int)
decrement() void
value() int
Complex
re double
im double
Complex(double, double)
plus(Complex, Complex) Complex
toString() String
«create»
«create»
«create»
1
1
1
1
1
1
MatrixSumAndDFTMutex
A Complex[][]
B Complex[][]
result Complex[][]
counter SynchronizedCounter
mutex Lock
barrierReached Condition
threadID int
startX int
startY int
endX int
endY int
MatrixSumAndDFTMutex(Complex[][], Complex[][], int, Complex[][])
run() void
MatrixSumAndDFTSynch
A Complex[][]
B Complex[][]
sumResult Complex[][]
result Complex[][]
counter SynchronizedCounter
threadID int
startX int
startY int
endX int
endY int
MatrixSumAndDFTSynch(Complex[][], Complex[][], int, Complex[][])
run() void
«create»
«create»
«create»
«create»
SumDFTMutex
DFTOfSum(Complex[][], Complex[][]) Complex[][]
SumDFTSynchronized
DFTOfSum(Complex[][], Complex[][]) Complex[][]
nonSynchronizedCounter
c int
nonSynchronizedCounter(int)
decrement() void
value() int