CSE 443
OBJECT ORIENTED ANALYSIS AND DESIGN

HOMEWORK 03

REPORT

ÖMER ÇEVİK
161044004

1. <u>Part 1</u>

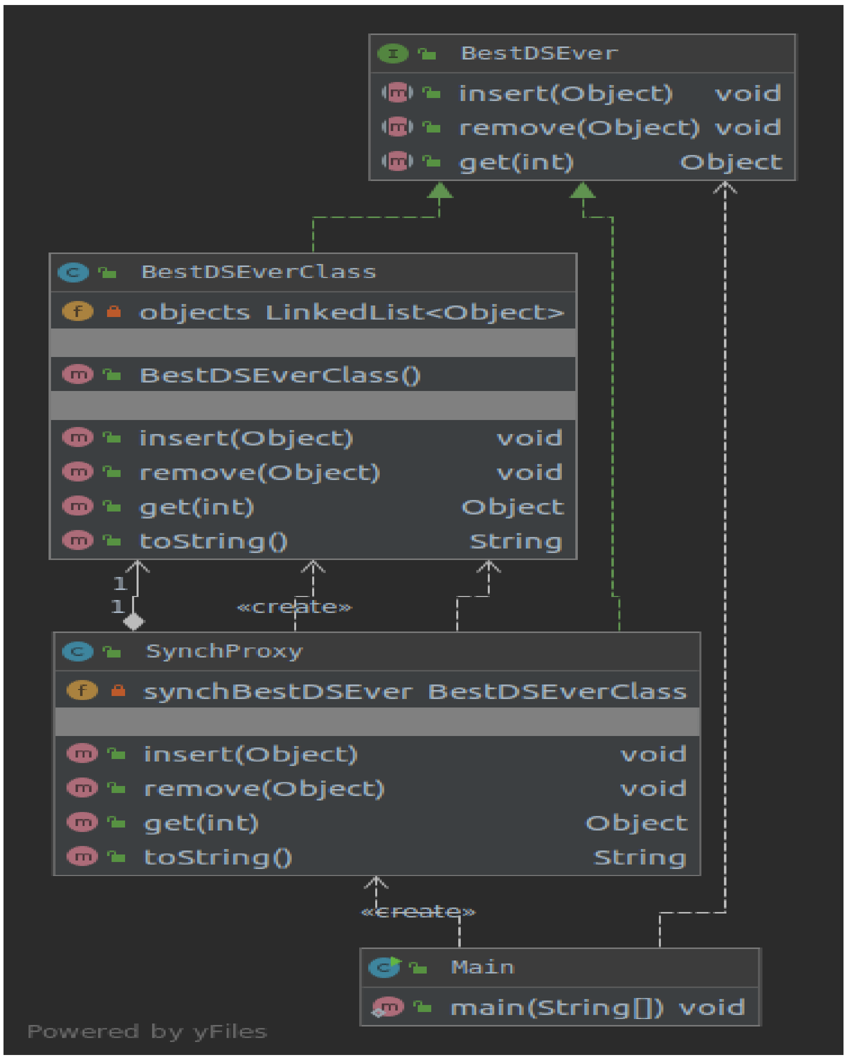
In that part, created the *BestDSEver* interface assumed declaring only *insert(), remove()* and *get()* methods. The "***Proxy***" design pattern is suits to design. So this three methods must be *synchronized* for threads. To make this synchronize work, created a new concrete class which is called *SynchProxy.* This class keeps an asynchronous class object which is called *BestDSEverClass* and *SynchProxy* concrete class implements the *BestDSEver* interface and overrides its methods.

Each overrided methods of *BestDSEver* interface are implemented to call asynchronous class's methods but using *synchronize* keyword. Each class are synchronized for threads and called the necessary methods of objects.

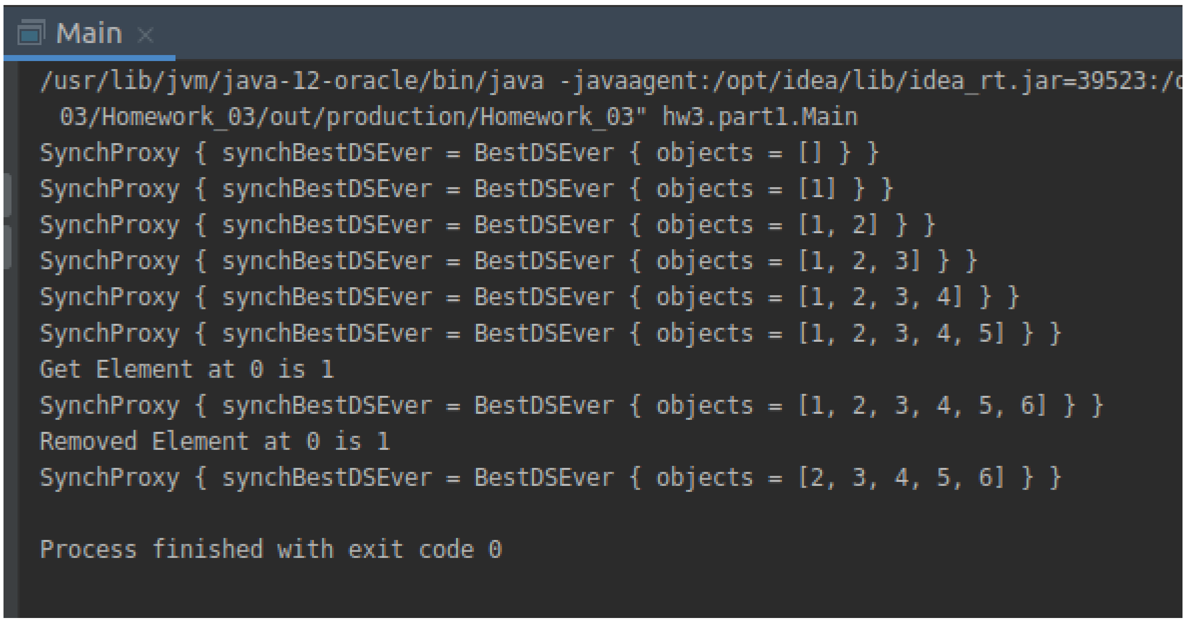
To show the results, also the *toString()* method of *Object* class is overrided.

In main method of programme, used 8 threads and main thread to call methods of class.

- **<u>Class Diagram For Part 1</u>**



- **Output Results For Part 1**



```
/usr/lib/jvm/java-12-oracle/bin/java -javaagent:/opt/idea/lib/idea_rt.jar=39523:/c
  03/Homework_03/out/production/Homework_03" hw3.part1.Main
SynchProxy { synchBestDSEver = BestDSEver { objects = [] } }
SynchProxy { synchBestDSEver = BestDSEver { objects = [1] } }
SynchProxy { synchBestDSEver = BestDSEver { objects = [1, 2] } }
SynchProxy { synchBestDSEver = BestDSEver { objects = [1, 2, 3] } }
SynchProxy { synchBestDSEver = BestDSEver { objects = [1, 2, 3, 4] } }
SynchProxy { synchBestDSEver = BestDSEver { objects = [1, 2, 3, 4, 5] } }
Get Element at 0 is 1
SynchProxy { synchBestDSEver = BestDSEver { objects = [1, 2, 3, 4, 5, 6] } }
Removed Element at 0 is 1
SynchProxy { synchBestDSEver = BestDSEver { objects = [2, 3, 4, 5, 6] } }

Process finished with exit code 0
```

2. Part 2

In that part, user enters the NxN 2D dimensional array size to calculate discrete fourier transform. Firstly the NxN size of array enters and sets it. Then selects the thread size to evaluate using multiple thread and starts the work.

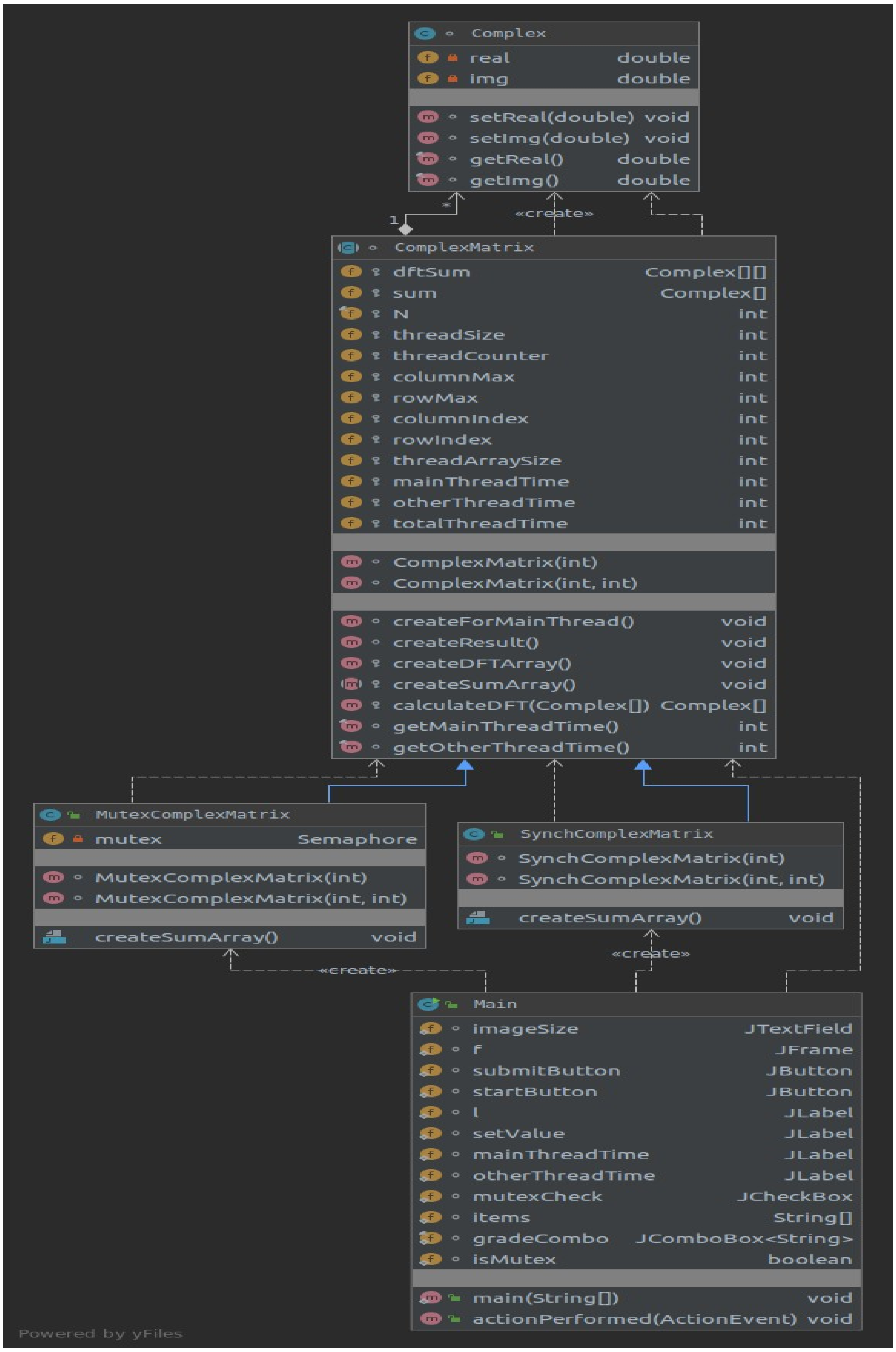
To create that programme, "*Compound*" design pattern is used. The "*Template*" and "*Command*" design patterns are used together. *Template* is for mathematical calculations and *Command* for to create GUI and its buttons, checkbox and text field parts.

The complex numbers are saved in *Complex* concrete class and used Complex array to keep results in *ComplexMatrix* abstract class. To synchronize the threads there are two distinct concrete classes which extends *ComplexMatrix* abstract class: *SynchComplexMatrix* and *MutexComplexMatrix*. *SynchComplexMatrix* concrete class overrides *createSumArray()* with ***synchronized*** keyword. *MutexComplexMatrix* concrete class overrides *createSumArray()* using ***mutex***.

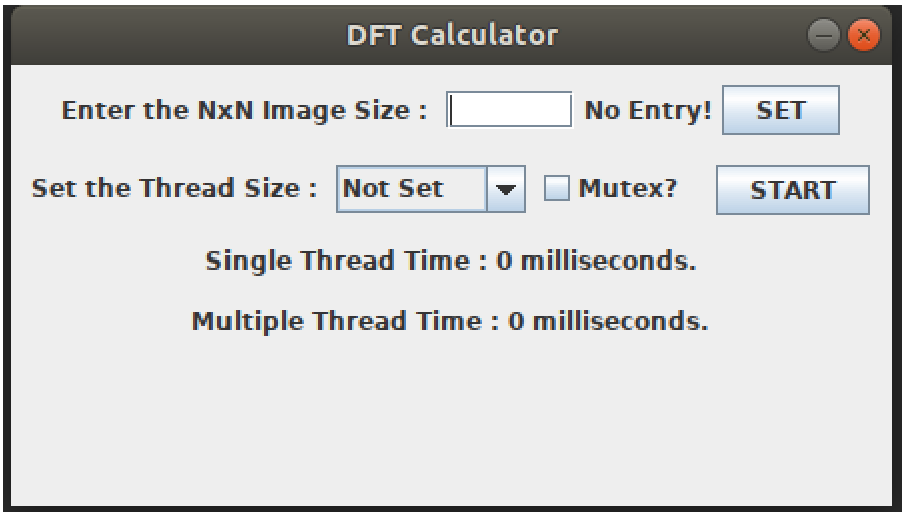
While working on multiple threads, each thread sum the A and B arrays and waits until all of them ends the sum operations then all of them calculates DFT together. End of the calculation programme shows up on the window the time of single thread's calculation time and multiple threads' calculation time.

Programme completely works fine and outputs well without stop and pause buttons.

- **<u>Class Diagram For Part 2</u>**



- <u>**Output Results For Part 2**</u>

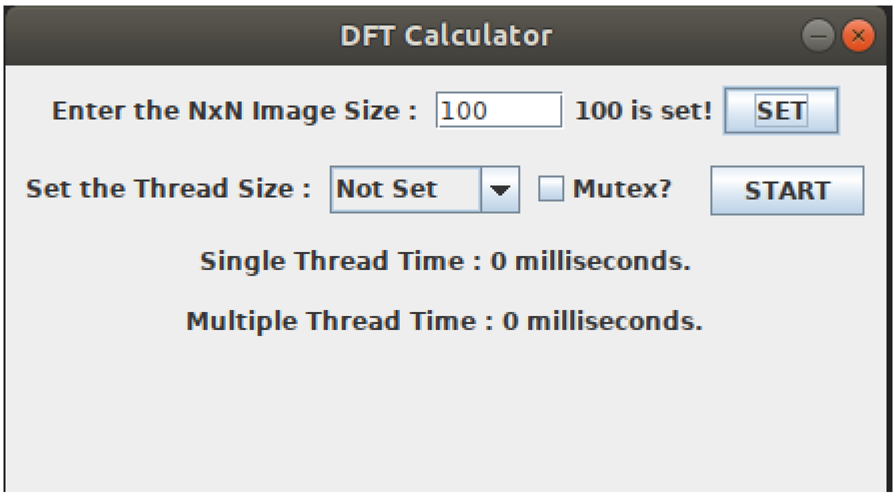

DFT Calculator

Enter the NxN Image Size : No Entry!

Set the Thread Size : Not Set

Single Thread Time : 0 milliseconds.

Multiple Thread Time : 0 milliseconds.

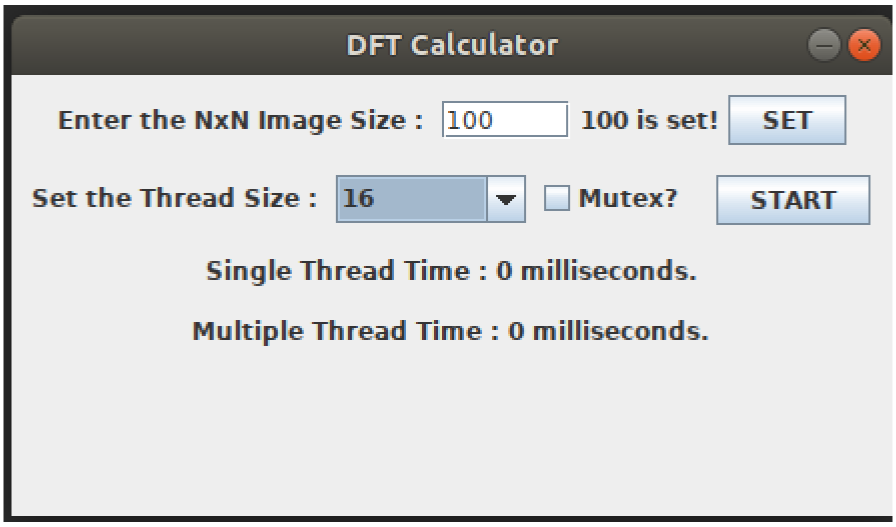

DFT Calculator

Enter the NxN Image Size : 100 100 is set!

Set the Thread Size : Not Set

Single Thread Time : 0 milliseconds.

Multiple Thread Time : 0 milliseconds.

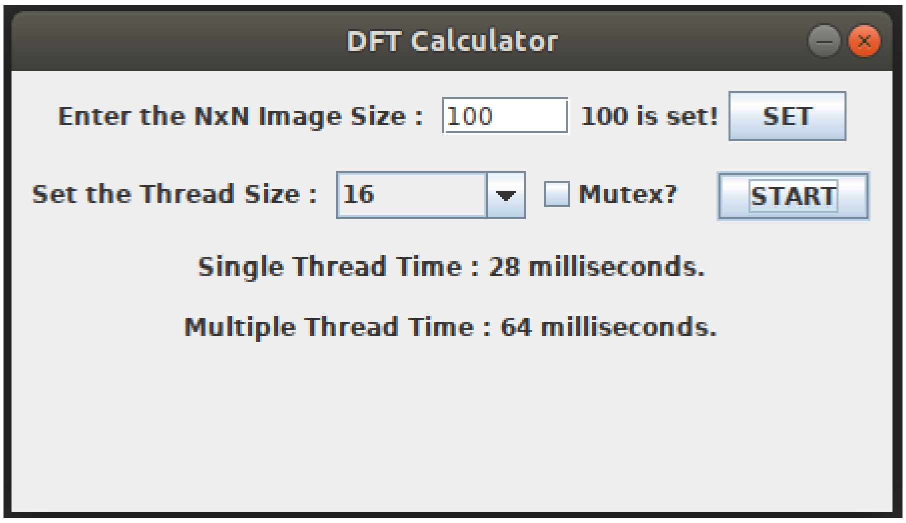

DFT Calculator

Enter the NxN Image Size : 100 100 is set!

Set the Thread Size : 16

Single Thread Time : 0 milliseconds.

Multiple Thread Time : 0 milliseconds.

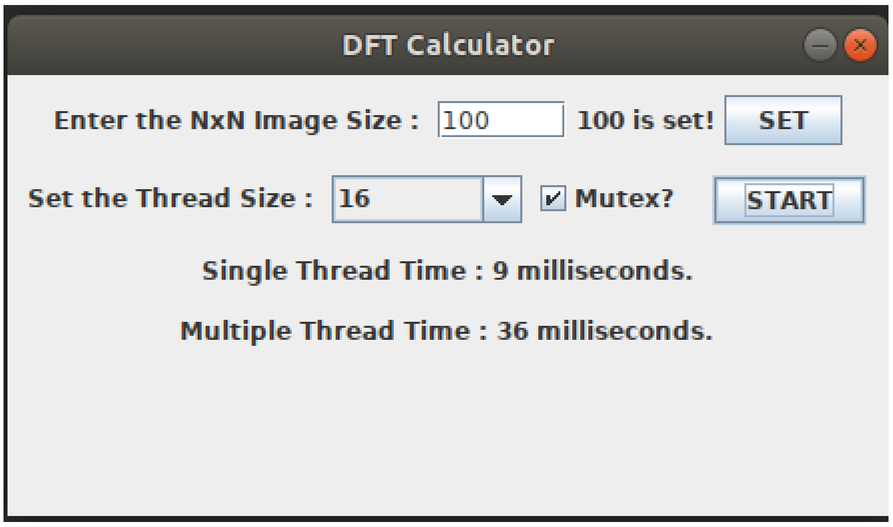


Enter the NxN Image Size : 100 100 is set!

Set the Thread Size : 16 ☐ Mutex?

Single Thread Time : 28 milliseconds.

Multiple Thread Time : 64 milliseconds.



Enter the NxN Image Size : 100 100 is set!

Set the Thread Size : 16 ☑ Mutex?

Single Thread Time : 9 milliseconds.

Multiple Thread Time : 36 milliseconds.