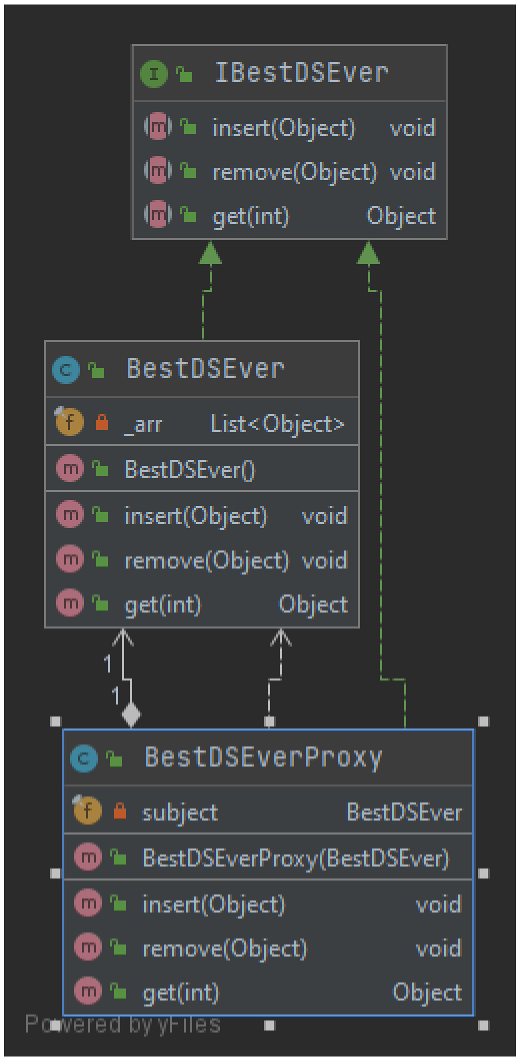
Object Oriented Analysis and Design

Homework-3 Report

Muhammed Özkan 151044084

Q1

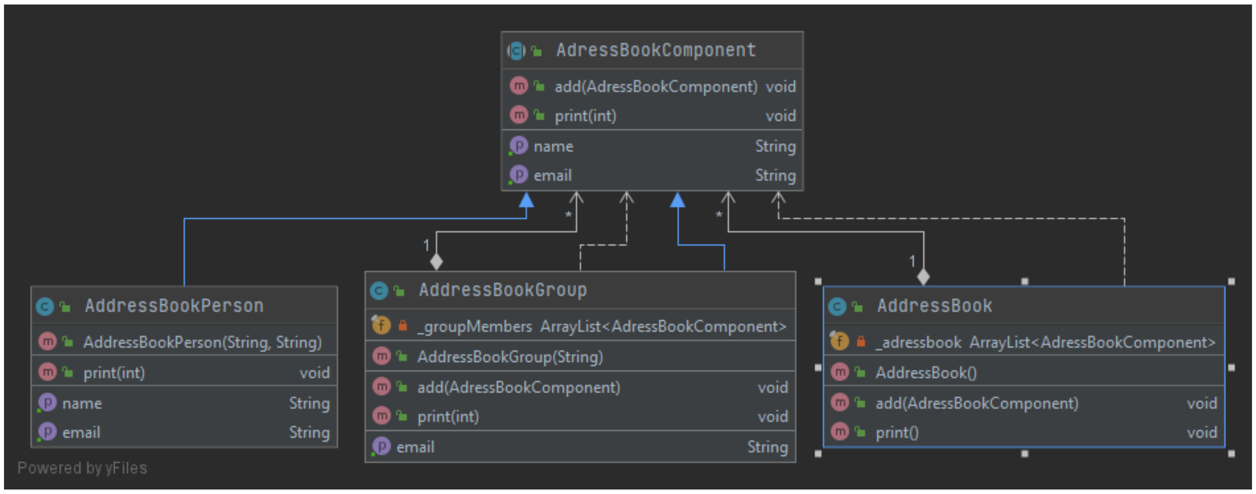
In this problem, I decided to use Proxy Pattern to make BestDSEver class thread safe. Because by applying this pattern, I can thread-safe the thread unsafe insertion, deletion and fetching functions with the **synchronized** keyword. I created a BestDSEverProxy class that implements the IBestDSEver interface. In the functions inside, I called the related functions using the synchronized keyword. The BestDSEverProxy class takes a BestDSEver object named subject as a parameter in its constructor method. I don't have an output or test code for this question.

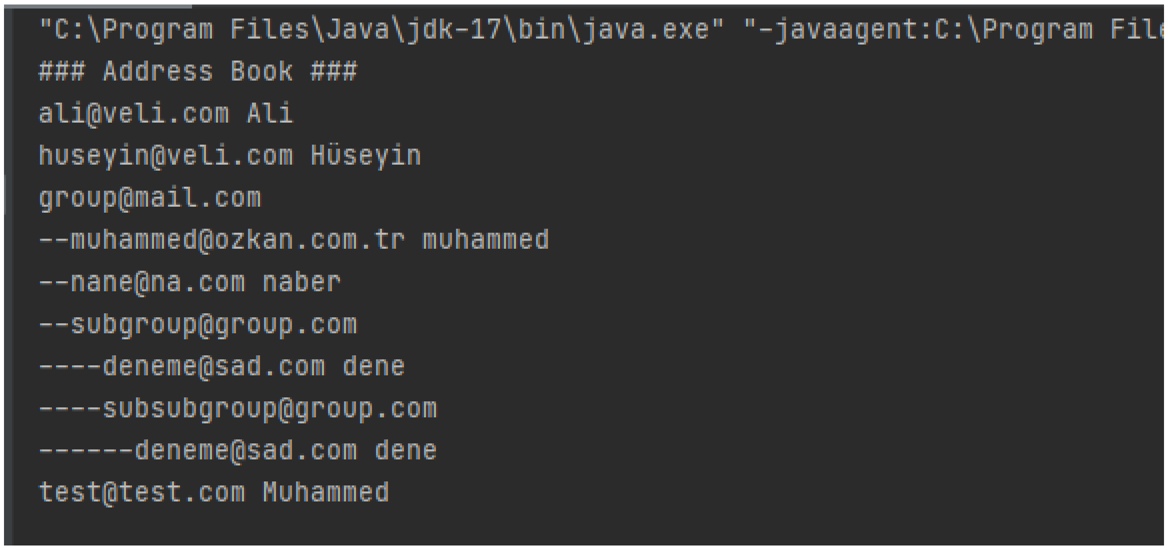


Q2

In order to apply the desired Composite and iterator pattern in this problem, I first created the AdressBookComponent abstract class. I then created the AdressBookPerson and AdressBookGroup classes that inherited this abstract class. Since the AdressBookGroup class can store both individuals and groups, it uses an AdressBookComponent in the array list structure. Finally, I created the AdressBook class where AdressBookComponents can be added to create an AdressBook class. All classes generate outputs using Iterator in the print function. If there are other iterators in them, it starts printing them first.

Sample output is available below. You can find the test codes of the output in “sourceCode\OOADHW3\Main.java” **Q2test()** function.



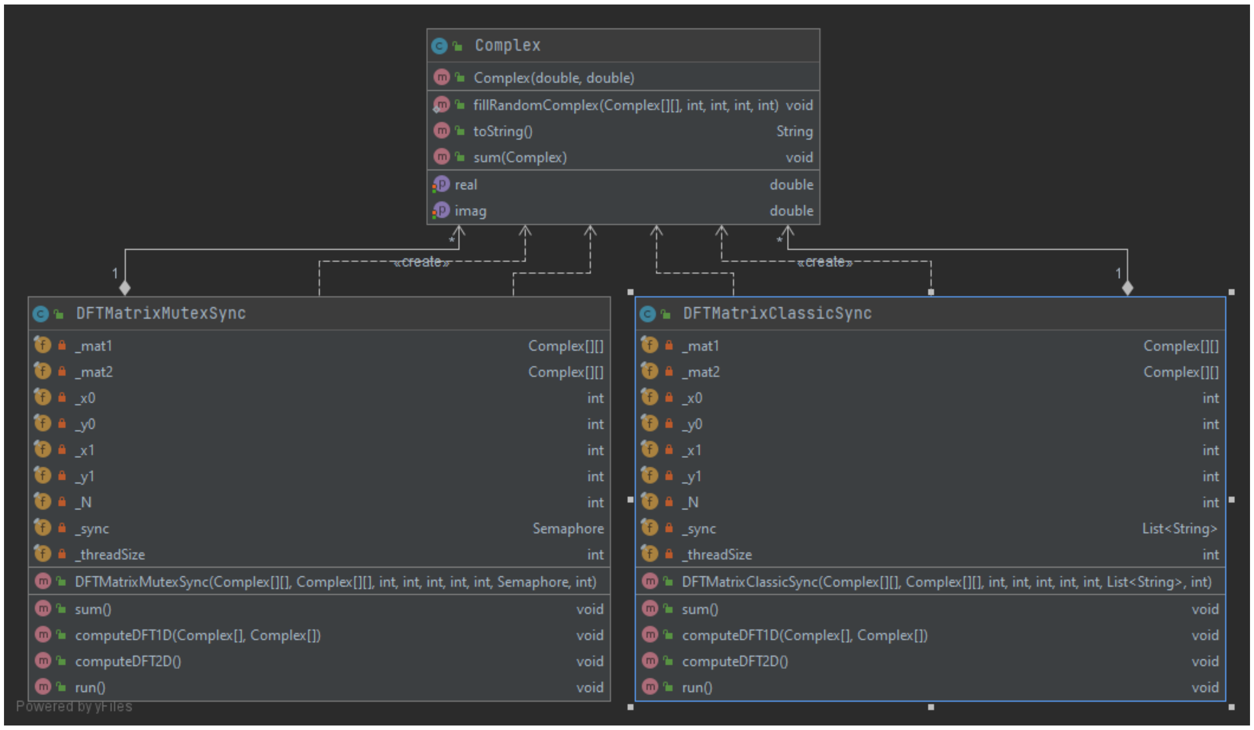


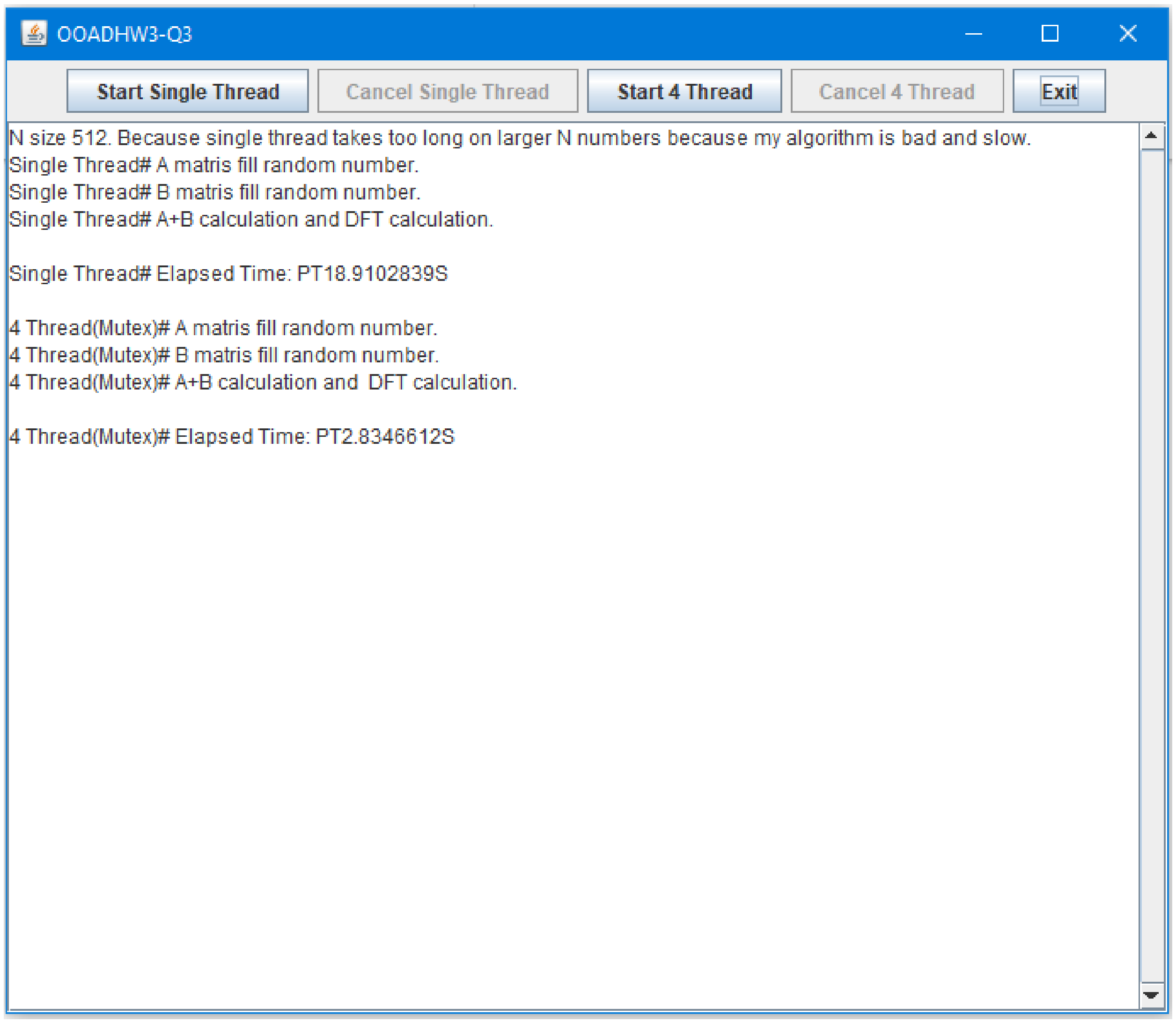
```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program File
### Address Book ###
ali@veli.com Ali
huseyin@veli.com Hüseyin
group@mail.com
--muhammed@ozkan.com.tr muhammed
--nane@na.com naber
--subgroup@group.com
----deneme@sad.com dene
----subsubgroup@group.com
------deneme@sad.com dene
test@test.com Muhammed
```

Q3

In this problem, two separate classes were created in order to collect 2 separate matrices and to perform DFT calculations as desired. While using the premium classic synchronization system. It performed this operation with a second-class mutex. The classic system uses a simple Arraylist. The thread performing the sum operation adds an element to the list. When the number of elements in the list is 4, notifyAll() is called for the second section. If the number of elements in the list is less than 4, it is expected with wait(). It is no different in the system with mutex. This time counting is done with a semaphore. When the number held by semaphore is 4, DFT part is started. When the calculations are finished, the elapsed time is calculated and printed on the screen. N=8192 could not be used. Because my algorithm is too slow. GUI is always responsive while operations are being performed. You can find an example of synchronization with the other classical method in the **Q3ClassicSync4ThreadTest()** function “sourceCode\OOADHW3\Main.java”. "sourceCode\OOADHW3\Q3\DFTMatrixClassicSync.java"

The following sites were used for DFT algorithm.
https://github.com/TTrojanovich/2D_Discrete_Fourier,
https://www.nayuki.io/page/how-to-implement-the-discrete-fourier-transform





```
N size 512. Because single thread takes too long on larger N numbers because my algorithm is bad and slow.
Single Thread# A matris fill random number.
Single Thread# B matris fill random number.
Single Thread# A+B calculation and DFT calculation.

Single Thread# Elapsed Time: PT18.9102839S

4 Thread(Mutex)# A matris fill random number.
4 Thread(Mutex)# B matris fill random number.
4 Thread(Mutex)# A+B calculation and  DFT calculation.

4 Thread(Mutex)# Elapsed Time: PT2.8346612S
```