Gebze Technical University
Computer Engineering Department
CSE443 - Object Oriented Analysis and Design
Fall 2021-2022

Homework 3 - v1

Rule 1: no plagiarism. If detected, you will fail this course.

Rule 2: no late submissions. Learning to plan your schedule according to deadlines is part of your education and a valuable professional asset.

What to submit: a) the source code of your project *fully documented (with javadoc)*, b) a well-formatted pdf report of your design decision explanations, class diagrams & screenshots and c) an executable demo (in jar format) that fully illustrates your program’s capabilities.

****

Question 1 - ? (20 points): You are hired, because the previous developer quit to start her own company. She has coded a brilliant and obscure data structure: BestDSEver. This class supports linear complexity insertion/deletion/random access. It’s great, no one can understand it now that she left, nobody dares to modify it.

The problem is all of the supported methods:

```
void insert(Object o);
void remove(Object o);
Object get(int index);
```

are **thread unsafe**. As soon as multiple threads start to handle an instance of BestDSEver, all hell breaks loose. Your task? Simple; you need to render BestDSEver thread-safe without modifying the original class in any way whatsoever (use the appropriate design pattern to solve this problem); provide your design and code.

Question 2 – Composite and Iterator (20 points): You are to design and implement an email address book.
Requirements:
a) Every email address must be stored along with the name of its owner.
b) An email address can be personal, such as alivelioglu@xyz.com or it can belong to a group, such as cengstudents4th@gtu.edu.tr which contains an arbitrary number of personal or group addresses.
c) Your application must be able to print on screen all the addresses in the address book. Personal addresses must be printed as : “alivelioglu@xyz.com Ali Velioglu” while group addresses must be printed by printing on screen every email (personal or group) address that they contain.
d) Implement the email addresses through the Composite and Iterator design patterns.
e) Draw a detailed class diagram of your solution.

Question 3 – Concurrency patterns (60 points):
You now work for “Threads Incorporated”. Produce 2 square matrices A and B both of size 8192x8192 filled with complex numbers. We want to calculate the Discrete Fourier Transform of A+B. Bu wait, it’s not as simple as that. Luckily you have access to multiple cores, and your boss wants you to parallelize this process. How?

Well, matrix addition is fully parallelizable. You will have 4 threads, and each thread will work in parallel, and calculate a quarter of the sum A+B. For example thread0 will calculate A+B only for the first 4096 rows and only for the first 4096 columns (in other words the top left quarter of A+B); thread1 will calculate the next quarter and so on.

Then they will continue with the same logic to calculate the DFT of A+B. However you need to be careful. The DFT calculation stage must not start before the summation stage’s end. This is known as a “synchronization barrier problem”. Implement it using Java’s synchronized, wait, notify and notifyAll mechanisms (no mutexes or monitors) **(20 points)** and implement it once again, this time using mutex(es) and monitor(s) **(20 points)**.

Prepare a GUI for this process, where the user can set the underlying implementation (with monitors or not); and can see the total time gained by using multiple threads (with respect to using a single thread) at the end of the calculations (**20 points**). Be careful, the GUI must remain responsive during the calculations and the user should be able to cancel and restart if she wants to.

Good luck.