**GTU Department of Computer Engineering**

**CSE443 Object Oriented Analysis and Design**

**Fall 2021 - Homework 3 Report**

**Akif KARTAL**

**171044098**

**Question 1**

**Question 2 – Composite and Iterator**

**Solution**

**2.1** **Understanding the problem**

The problem is like the following picture;

Diagram

Description automatically generated

Here, emails contains both address and name of its owner, groups contains an arbitrary number of personal or group addresses also **groups are composite, emails are leaf**.

**2.2 Class Diagram**

**2.2.1 Classes in my solution**

Graphical user interface, application

Description automatically generated

**2.2.2 Composite Pattern Class Diagram**

Graphical user interface, text

Description automatically generated

Here, EmailComponent class is abstract class, Email is leaf class and CompositeEmail is a composite class with add, remove and get methods.

**2.2.3 Full Class Diagram**

Here, we will see full diagram note that I used iterator design pattern to traverse all tree easily as expected in homework pdf file.

Graphical user interface

Description automatically generated

Here, Composite iterator implements java.util iterator interface to use in composite email class. NullIterator class is used in Email leaf class. Main class is for test purpose.

**2.3 Test Results**

Check Main.java class and run to see results. A part of result is following;

**A screenshot of a computer

Description automatically generated with medium confidence**

**Question 3 – Concurrency patterns**

**Solution**

**3.1 Understanding the** **synchronization barrier problem**

In order to solve this problem, we will apply following solution in 2 different ways with java;

Text

Description automatically generated

**3.2 Using mutex(es) and monitor(s)**

I will explain my solution with simple example. You can check the source code.

**3.2.1 Classes in my solution**

Graphical user interface

Description automatically generated with low confidence

These are general classes in my solution.

**3.2.2 Shared Data between Threads**

**Text

Description automatically generated**

\*Atomic integer is much better between threads in java.

**3.2.3 Example Thread Class**

**Text

Description automatically generated**

\*Here, we are injecting shared data and overriding run method from runnable interface. See the comments of critical codes. This is java version of synchronization barrier problem solution.

**3.2.4 Creating Threads and Testing**

**Text

Description automatically generated**

**3.2.5 Output**

**Text

Description automatically generated**