Middle East Technical University | Department of Computer Engineering

CENG 443

Introduction to Object-Oriented
Programming Languages and Systems

Spring 2022-2023

Lab 2 Preliminary - Scribe's Dilemma
(Ver 1.0)

1 Introduction

Keywords: *Concurrency*

In this assignment, you are asked to write a program to manage resources for a number of scribes.

2 Overview

A scribe's job is to record something to paper. What they write is not your concern, but how. To write, a scribe needs a pen and an ink bottle. These resources cannot be used by more than one scribe, so you should manage them using concurrency primitives.

When a scribe gets **both** a pen and an ink bottle, he will write a record and put both the pen and the bottle back until next time.

Number of scribes, pens and ink bottles will be given to your program as arguments:

```
java YourProgramName 5 2 4
```

will run your program with **5** scribes with **2** pens and **4** ink bottles.

3 Example Output

Due to indeterminism, your output will likely be different from the one below. What is important is:

- your output should not violate the specification e.g. if there are five pens and all of them taken, sixth scribe should not get a pen until another one put it back.
- Output should not be garbled like the output below:

```
This is the fiThis is the second sentence.st sentence.
```

- Number of scribes should start from one.

First few lines of an output of the program with 5 scribes 2 pens and 4 bottles can be seen below. Note that your app should run indefinitely until terminated:

```
Scribe 3 takes a bottle
Scribe 2 takes a bottle
Scribe 1 takes a bottle
Scribe 2 takes a pen
Scribe 4 takes a bottle
Scribe 2 writes a record
Scribe 3 takes a pen
Scribe 4 puts the bottle back
Scribe 4 takes a bottle
Scribe 2 puts the pen back
Scribe 2 puts the bottle back
Scribe 5 takes a bottle
Scribe 3 writes a record
Scribe 3 puts the pen back
Scribe 4 takes a pen
Scribe 5 takes a pen
Scribe 5 writes a record
Scribe 4 writes a record
Scribe 4 puts the pen back
Scribe 1 takes a pen
Scribe 5 puts the pen back
Scribe 5 puts the bottle back
....
```

In the example, you may notice that scribes sometimes put down items without writing. This is to prevent deadlocks, and you should handle these situations in your code also.

4 Some Remarks

Although your code will not be graded, and you have freedom in your design, your code should utilize concurrency nonetheless since the upcoming lab will be graded accordingly.

- Every Scribe Class should be a separate thread and act independent of each other.
- Resources (pen and ink bottle) should be managed using locks and not atomics.
- You should make sure that no deadlock, no busy-wait and no starvation (like some scribes not being able to manage to write in a reasonable time) will occur in runtime.
- Use Java 8 or a newer version