Middle East Technical University | Department of Computer Engineering

CENG 443

Introduction to Object-Oriented Programming Languages and Systems

2024-2025 Fall

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 some 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 at the same time, 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.

3 Example Output

The command below will run your program with **5** scribes, **2** pens and **4** ink bottles:

```
java Scribes 5 2 4
```

The few lines of an output of the command above can be seen below. Your app should run **indefinitely** until terminated:

```
Scribe 3 takes a bottle (on Thread 11) (Action number: 3)
Scribe 2 takes a bottle (on Thread 10) (Action number: 4)
Scribe 1 takes a bottle (on Thread 9) (Action number: 1)
Scribe 4 takes a bottle (on Thread 12) (Action number: 2)
Scribe 2 takes a pen (on Thread 10) (Action number: 6)
Scribe 4 puts the bottle back (on Thread 12) (Action number: 8)
Scribe 3 takes a pen (on Thread 11) (Action number: 5)
Scribe 2 writes a record (on Thread 10) (Action number: 9)
Scribe 1 puts the bottle back (on Thread 9) (Action number: 7)
Scribe 3 writes a record (on Thread 11) (Action number: 11)
Scribe 5 takes a bottle (on Thread 13) (Action number: 10)
Scribe 2 puts the pen back (on Thread 10) (Action number: 12)
Scribe 4 takes a bottle (on Thread 12) (Action number: 13)
```

```
Scribe 3 puts the pen back (on Thread 11) (Action number: 14)
Scribe 5 puts the bottle back (on Thread 13) (Action number: 15)
Scribe 2 puts the bottle back (on Thread 10) (Action number: 16)
Scribe 4 takes a pen (on Thread 12) (Action number: 17)
Scribe 1 takes a bottle (on Thread 9) (Action number: 19)
Scribe 3 puts the bottle back (on Thread 11) (Action number: 18)
Scribe 5 takes a bottle (on Thread 13) (Action number: 20)
Scribe 1 takes a pen (on Thread 9) (Action number: 22)
Scribe 4 writes a record (on Thread 12) (Action number: 21)
Scribe 2 takes a bottle (on Thread 10) (Action number: 23)
Scribe 4 puts the pen back (on Thread 12) (Action number: 26)
Scribe 1 writes a record (on Thread 9) (Action number: 25)
Scribe 5 puts the bottle back (on Thread 13) (Action number: 24)
Scribe 2 puts the bottle back (on Thread 10) (Action number: 27)
Scribe 1 puts the pen back (on Thread 9) (Action number: 29)
Scribe 3 takes a bottle (on Thread 11) (Action number: 30)
Scribe 4 puts the bottle back (on Thread 12) (Action number: 28)
Scribe 1 puts the bottle back (on Thread 9) (Action number: 32)
Scribe 3 takes a pen (on Thread 11) (Action number: 33)
Scribe 5 takes a bottle (on Thread 13) (Action number: 31)
Scribe 2 takes a bottle (on Thread 10) (Action number: 34)
Scribe 4 takes a bottle (on Thread 12) (Action number: 35)
Scribe 3 writes a record (on Thread 11) (Action number: 36)
Scribe 2 puts the bottle back (on Thread 10) (Action number: 38)
Scribe 5 takes a pen (on Thread 13) (Action number: 37)
Scribe 3 puts the pen back (on Thread 11) (Action number: 40)
Scribe 1 takes a bottle (on Thread 9) (Action number: 41)
Scribe 4 puts the bottle back (on Thread 12) (Action number: 39)
Scribe 3 puts the bottle back (on Thread 11) (Action number: 43)
Scribe 5 writes a record (on Thread 13) (Action number: 42)
Scribe 2 takes a bottle (on Thread 10) (Action number: 45)
Scribe 1 takes a pen (on Thread 9) (Action number: 44)
Scribe 5 puts the pen back (on Thread 13) (Action number: 47)
Scribe 4 takes a bottle (on Thread 12) (Action number: 46)
Scribe 2 puts the bottle back (on Thread 10) (Action number: 48)
Scribe 1 writes a record (on Thread 9) (Action number: 49)
Scribe 5 puts the bottle back (on Thread 13) (Action number: 50)
Scribe 4 takes a pen (on Thread 12) (Action number: 51)
Scribe 3 takes a bottle (on Thread 11) (Action number: 52)
Scribe 1 puts the pen back (on Thread 9) (Action number: 53)
Scribe 4 writes a record (on Thread 12) (Action number: 54)
Scribe 2 takes a bottle (on Thread 10) (Action number: 55)
Scribe 1 puts the bottle back (on Thread 9) (Action number: 57)
Scribe 3 puts the bottle back (on Thread 11) (Action number: 56)
Scribe 4 puts the pen back (on Thread 12) (Action number: 58)
Scribe 2 takes a pen (on Thread 10) (Action number: 59)
Scribe 5 takes a bottle (on Thread 13) (Action number: 60)
Scribe 1 takes a bottle (on Thread 9) (Action number: 61)
Scribe 4 puts the bottle back (on Thread 12) (Action number: 62)
Scribe 2 writes a record (on Thread 10) (Action number: 63)
Scribe 5 takes a pen (on Thread 13) (Action number: 64)
Scribe 1 puts the bottle back (on Thread 9) (Action number: 65)

....
```

Also, in the example above, you can see that:

- The action numbers are not always in order. It is expected due to several reasons ranging from waiting in actions to garbage collection. Due to indeterminism, your output will likely be different not only in action order but order of scribes that take action. The important thing is that your output does not violate the specification: the number of pens currently used by scribes should not exceed the number of overall pens, etc.
- Scribes sometimes put down items without writing. This is to prevent deadlocks, and you should also handle these situations in your code.

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 exam will be graded accordingly.

- Your main class name should be **Scribes**.
- Numbering of scribes should start from 1.
- All scribes should run as a separate thread and act independent of each other.
- For every action (trying to take an ink bottle, dropping a pen, writing etc.), you should use the corresponding action method in **Actions** class.
- When ensuring coordination between scribes, you can use any synchronization primitive other than atomics.
- You should not make any assumptions based on timings of actions since different timing intervals may be used during the evaluation of the upcoming exam.
- You should make sure that no deadlock, no starvation, no busy-wait or no unnecessary wait will occur in runtime.
- All scribes should be able to write in a reasonable time-frame.
- To expand the remarks above, solution models below (but not limited to) are not valid solutions:

  Running all logic on a single thread

  Scribes waiting for each other to do their jobs, even when necessary tools are available

  Threads sleeping and waking up periodically to poll a variable

  Threads waking up more than necessary

  Threads causing other threads to block while sleeping

- Although unlikely, due to differences in Java implementations, print function may not always be atomic. While you certainly will not need it when solving the Lab exam question, you may need to synchronize print statements when solving a preliminary question on your own system.
- Use Java 8 or a newer version