COS40003 Concurrent Programming

Duration: 60 min

Problem: Multi-threaded Log Processing Problem

Background

A system continuously generates log messages (log entries).
Each log entry includes:

- A **system timestamp** (when it was generated)
- A **log category** (INFO, WARNING, or ERROR)
- A **message string** in the form of "Producer-M message N.", eg, "Producer-2 message 5." denoting the 5th message generated by Producer-2

You are required to implement a **multithreaded log processing program** that generates log messages, filters and sorts them, and finally outputs them in **chronological order** to the console.

Detailed Requirements

System Structure

The system consists of the following types of threads:

1. **Producer Threads (multiple: Producer-1 and Producer-2)**
   - Continuously generate log entries (**every 0.2 second and for a total of 5 seconds**).
   - Log category follows the following distribution: INFO 60%, WARNING 30%, ERROR 10%.
   - Producers write their logs into a **shared buffer**.
2. **Filter Thread (single)**
   - Reads logs from the shared buffer.
   - Discards all INFO-category logs.
   - Sends the remaining WARNING and ERROR logs to another buffer called **filtered buffer**.
3. **Sorter Thread (single)**
   - Reads logs from the filtered buffer.
   - Sorts them by timestamp.

  - **Every 1 second**, outputs the logs in chronological order, and discard the logs that have been outputted.

Synchronization and Thread Safety Requirements

- All the buffers must be **thread-safe**.
- You are **not allowed** to use existing thread-safe queue classes (e.g. Java’s BlockingQueue). You must implement synchronization manually (eg., using locks, condition variables, etc.).
- **Busy waiting** is not allowed.
- The system must be **deadlock-free.**

Marking (20 Marks)

- correct output and the code has correct concurrency logic (15 marks)
- code inspection: clean and clear (5 marks)

Code Submission

- Please submit your code to canvas before the submission deadline.

Hint:

The following code helps you output the current system date and time.

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");
Long timestamp = System.currentTimeMillis();
// this value can help you rank the logs
Date date = new Date(timestamp);
System.out.println("[" + sdf.format(date) + "]");
```

Example Output:

Please see next page

```
Starting LogProcessor demo with 2 producers for 5s...
----------------------------
[2025-10-31 16:05:19.413][WARNING] Producer-1 message 1.
[2025-10-31 16:05:19.840][WARNING] Producer-2 message 3.
[2025-10-31 16:05:20.047][WARNING] Producer-1 message 4.
[2025-10-31 16:05:20.259][WARNING] Producer-1 message 5.
----------------------------
[2025-10-31 16:05:20.459][WARNING] Producer-2 message 6.
[2025-10-31 16:05:21.301][WARNING] Producer-1 message 10.
[2025-10-31 16:05:21.301][WARNING] Producer-2 message 10.
----------------------------
[2025-10-31 16:05:21.508][WARNING] Producer-1 message 11.
[2025-10-31 16:05:22.133][WARNING] Producer-1 message 14.
[2025-10-31 16:05:22.335][ERROR] Producer-2 message 15.
[2025-10-31 16:05:22.335][WARNING] Producer-1 message 15.
----------------------------
[2025-10-31 16:05:22.755][ERROR] Producer-2 message 17.
[2025-10-31 16:05:22.755][WARNING] Producer-1 message 17.
[2025-10-31 16:05:22.957][ERROR] Producer-1 message 18.
[2025-10-31 16:05:22.957][WARNING] Producer-2 message 18.
[2025-10-31 16:05:23.357][WARNING] Producer-1 message 20.
[2025-10-31 16:05:23.560][ERROR] Producer-1 message 21.
[2025-10-31 16:05:23.560][WARNING] Producer-2 message 21.
----------------------------
[2025-10-31 16:05:23.761][WARNING] Producer-2 message 22.
[2025-10-31 16:05:24.163][WARNING] Producer-1 message 24.
[2025-10-31 16:05:24.364][ERROR] Producer-2 message 25.
----------------------------
Final statistics:
Processed INFO = 29
Processed WARNING = 16
Processed ERROR = 5
```