Operating Systems Project 2

Thread-Based Process Simulation and Synchronization  
Course: Operating Systems 4320  
Semester: Spring 2025  
Group Member:

1. Son Nguyen
2. Sheila Corona
3. Harry Cao

Date: 04/29/2025

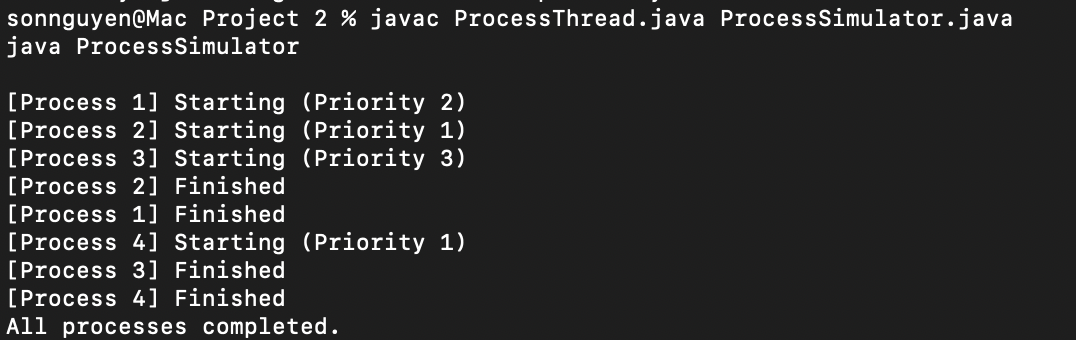
# **1. Introduction**

This project involved simulating real-time process execution using threads and solving a classic synchronization problem using mutexes and semaphores in Java. We reused the processes.txt input file to represent processes, and each process was modeled as a thread. Additionally, we implemented the Producer-Consumer synchronization problem to simulate safe access to a shared bounded buffer.

# **2. Process Simulation with Threads**

Each line of the processes.txt file contained the PID, arrival time, burst time, and priority of a process. We created a ProcessThread class that extended Thread to simulate each process.  
  
Key Implementation Points:

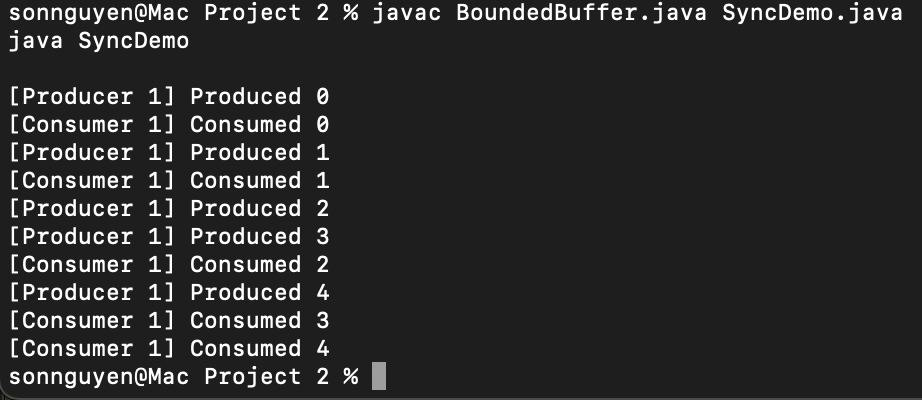
* - The arrival time was simulated using Thread.sleep(arrivalTime \* 1000).
* - The burst time was simulated using another Thread.sleep(burstTime \* 1000).
* - Each process printed logs when starting and finishing, including its PID and priority.

Output:  


# 3. Synchronization Problem: Producer-Consumer

For the synchronization part, we implemented the Producer-Consumer Problem.  
  
Key Implementation Points:

* - A bounded buffer of size 3 was created.
* - Semaphores (full and empty) controlled buffer slot availability.
* - A ReentrantLock (mutex) was used to protect critical sections.
* - A producer thread added items to the buffer.
* - A consumer thread removed items from the buffer.

Output:  


# **4. Challenges Encountered**

* - Initially missing the ReentrantLock import caused compilation issues (import java.util.concurrent.locks.ReentrantLock was needed).  
  - Care was needed to properly acquire and release semaphores to avoid deadlock or resource starvation.  
  - Multiple test runs were used to verify the stability of thread execution and synchronization correctness.

# **5. Conclusion**

Through this project, I gained valuable hands-on experience with:  
- Thread creation and management in Java.  
- Simulating OS-like process scheduling behavior.  
- Solving synchronization problems using semaphores and locks.  
  
The project strengthened my understanding of critical sections, race conditions, and thread-safe programming, which are essential concepts for real-world operating system and concurrent application development.