Write-up: Project 2

Project Approach

- For this project, my primary objective that I had to fulfill was to implement and create a multi-threaded bank simulation in Java, which would properly ensuring synchronization between tellers and customers using semaphores and locks. After analyzing the project requirements.

Project Organization

- The project was implemented within a single file, BankSimulation.java, for simplicity Key sections within this file are:
    - Teller Class: This will manage teller behavior and the key interaction with customers, including handling deposits and withdrawals.
    - Customer Class: This is to represents customer threads that interact with tellers and request transactions.
    - Main Method: This was used to Initializes and starts the teller and customer threads.

Problems Encountered

- Thread Synchronization: this was used to Ensure that only two tellers can access the safe simultaneously was challenging.
- Random Delays: Managing consistent timing for interactions between tellers and the manager introduced potential thread interruptions.

Solutions Implemented

- Semaphore Implementation: Used semaphores for the manager and safe to manage limited access and prevent resource conflicts.
- CountDownLatch: Employed a CountDownLatch to delay customer entry until all tellers were ready, ensuring proper bank opening.

Lessons Learned

- This project reminded me the importance of thread synchronization and to really understand the Java concurrency primitives. Proper use of semaphores helped ensure efficient communication among threads. I learned strategies for handling thread-safe data structures to maintain consistent output and avoid race conditions.