Report: Locking-Based Protocol Concurrency in Java

German International University of Applied Sciences
Informatics and Computer Science
Dr. Marwa Zamzam
Distributed Web-based System, Spring 2023
Assignment 2

Introduction

This study details the Java implementation of a distributed web-based system that uses concurrency with a locking-based protocol. The objective is to use the idea of semaphores to address a particular issue involving two threads simultaneously accessing shared data. To prevent the two threads from sharing data concurrently, the issue involves locking the resources. The needed functionality is implemented using the classes Shared, Driver, and MyThread.

Problem Description

The issue scenario calls for the following conditions:

- Give the variable X a definition and a value of 20.
- • Give the variable Y a definition and a 30 value.
- Define the variable `CumSum` to store the cumulative sum of calculations.
- Create the variable CumSum to hold the total of all computations.
- The same data will be accessed simultaneously by two threads.
- The A thread will perform five iterations of the CumSum calculation, calculating ((X + Y) + CumSum) after each iteration.
- Five times will be performed by the B thread's CumSum calculation, with each iteration assessing (CumSum - (X + Y)).
- Following the execution of the two threads, CumSum's value ought to reset to 0.

Class: `Shared`

Static variables corresponding to the shared data are present in the Shared class. It establishes the starting values and cumulative sum variables, X, Y, and CumSum. The class acts as a common resource that both threads can access.

Class: `Driver`

The program's starting point is the Driver class. It makes two threads, A and B, and uses semaphores to coordinate their execution. The primary mechanism for managing the program's flow is implemented by the Driver class.

Class: `MyThread`

The MyThread class defines a unique thread that utilises shared resources to carry out particular calculations. Each MyThread object has a specific name and a pointer to a semaphore. To specify how the thread should behave, the class overrides the run method.

Conclusion

In conclusion, this study has demonstrated a Java implementation of a distributed web-based system that uses concurrency with a locking-based protocol. Semaphores are used by the system to manage access to shared resources and make sure that no two threads are attempting to access the data at the same time. To resolve a particular issue requiring the computation of a cumulative sum, the Shared, Driver, and MyThread classes collaborate. After running the threads, the programme successfully calculates the desired cumulative total and prints the result of CumSum.