**Reader-Writer Problem: Semaphore-based Solution**

**Abstract:**

The reader-writer problem involves managing access to shared resources among multiple readers and writers. This report presents a Semaphore-based solution implemented in Java to address this problem. The solution ensures mutual exclusion between writers and provides concurrent access for readers while maintaining the integrity of shared resources.

**Introduction:**

The reader-writer problem aims to regulate access to shared data among concurrent threads. Writers need exclusive access during writing, while multiple readers can access data simultaneously without conflict. The implemented solution employs Semaphores to manage access control, ensuring the prescribed synchronization constraints.

**Solution Overview:**

The solution utilizes Semaphores to implement a **ReadWriteLock** class, comprising methods for **readLock**, **readUnlock**, **writeLock**, and **writeUnlock**. These methods enforce mutual exclusion between writers and coordinate concurrent access for readers.

**Implementation Details:**

* **ReadWriteLock Class**: Contains two Semaphores (**readerLock** and **writerLock**) and maintains a count of active readers (**readerCount**).
* **readLock()**: Grants read access, allowing multiple readers if no writer is present. Acquires the write lock for the first reader to prevent writers.
* **readUnlock()**: Releases the read lock and signals waiting writers if no readers are present.
* **writeLock()**: Grants exclusive write access, blocking other writers and readers.
* **writeUnlock()**: Releases the write lock, allowing waiting writers or readers to access the shared resource.

**Code Execution:**

The provided code was tested using Java's **ExecutorService** and threads for writers and readers. Multiple writer and reader threads were instantiated, demonstrating concurrent access control based on the implemented **ReadWriteLock**.

**Discussion:**

The Semaphore-based solution successfully ensures mutual exclusion for writers and concurrent access for readers, adhering to the specified conditions of the reader-writer problem. However, there might be potential improvements in handling edge cases or optimizing resource utilization.

**Conclusion:**

The implemented Semaphore-based solution effectively manages access control for the reader-writer problem, ensuring synchronization between writers and readers while preserving data integrity. Future enhancements could focus on performance optimizations or handling more complex scenarios.

**References:**

[1] Operating System Concepts - Silberschatz, Galvin, Gagne