coursera

1.2 Lecture Summary

1.2 Structured Locks

Lecture Summary: In this lecture, we learned about *structured locks*, and how they can be implemented using `synchronized` statements and methods in Java. Structured locks can be used to enforce *mutual exclusion* and avoid *data races*, as illustrated by the `incr()` method in the `A.count` example, and the `insert()` and `remove()` methods in the the `Buffer` example. A major benefit of structured locks is that their *acquire* and *release* operations are implicit, since these operations are automatically performed by the Java runtime environment when entering and exiting the scope of a `synchronized` statement or method, even if an exception is thrown in the middle.

We also learned about `wait()` and `notify()` operations that can be used to block and resume threads that need to wait for specific conditions. For example, a producer thread performing an `insert()` operation on a *bounded buffer* can call `wait()` when the buffer is full, so that it is only unblocked when a consumer thread performing a `remove()` operation calls `notify()`. Likewise, a consumer thread performing a `remove()` operation on a *bounded buffer* can call `wait()` when the buffer is empty, so that it is only unblocked when a producer thread performing an `insert()` operation calls `notify()`. Structured locks are also referred to as *intrinsic locks* or *monitors*.

Optional Reading:

1. Tutorial on Intrinsic Locks and Synchronization in Java

2. Tutorial on Guarded Blocks in Java

3. Wikipedia article on Monitors

Mark as completed