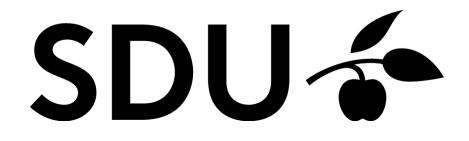
Concurrent Programming

3: Threads and Locks

Fabrizio Montesi <fmontesi@imada.sdu.dk>



Questions from the exercises?

Threads

• Smallest execution unit found in operating systems.

- A single application can have many concurrent threads.
- https://docs.oracle.com/javase/tutorial/essential/concurrency/th reads.html

Scheduling

• The Operating System (OS) decides when a thread executes.

• You have many threads, but only a few CPUs.

• So only a few threads at a time can execute in parallel.

• The scheduler in the OS decides when each thread can execute some of its code for some time.

Interleaving VS True Concurrency

• True Concurrency: multiple actions happening at the same time.

• Interleaving: only one action happens at a time.

• The scheduler makes interleaving "look like" true concurrency.

• What do you have? Depends on how many threads and CPUs.

Interleaving or True Concurrency?

• What do you have?

• nCPU = 1 gives interleaving.

• nCPU >= nThreads gives true concurrency.

• 1 < nCPU < nThreads gives a mix.

• Remember that the system probably has more threads than you run in your application.

Threads share memory

• Threads share the same memory!

- Sharing is the biggest...
 - ...advantage for performance. :-)
 - ...cause of bugs. :-(

Multi-threaded programming is hard

• **Mutable object state** makes multi-threading difficult.

• Mutable = can change at runtime.

• Accessing mutable data from multiple threads is dangerous!

• [Example]

Thread safety

• If a class is accessed by multiple threads, we want it to be thread safe.

• **Thread-safe class:** a class that *behaves correctly* when accessed by multiple threads, regardless of how they are scheduled or how they coordinate with each other.

• The definition of "behaves correctly" depends on the class. (Or rather, the programmer of the class gives it.)

Fixing Concurrency

• To make a class thread-safe, we need to control access to data.

• Important operations on data should be *atomic*: once we start them, we should finish them before the next thread can access the data.

• How can we make an operation atomic?

• Locks!

Synchronized

• Java native support for locking.

 https://docs.oracle.com/javase/tutorial/essential/concurrency/lo cksync.html

Locks and Deadlocks

• Lock objects can be used for programmable locking.

• It is easy to carelessly have deadlocks and get stuck!

Guarded Blocks

• A block of code that waits for some signal before running.

• Implemented via monitors in Java.

 https://docs.oracle.com/javase/tutorial/essential/concurrency/g uardmeth.html

The Producer-Consumer Problem

• Some producers insert elements in a shared data structure.

• Some consumers take elements from the shared data structure.

• Example: a product delivery system.

Questions?

Exercises

• Read the links in the slides.

• Modify the SynchronizedMutableField example such that each thread does 10 (increment or decrement) operations before allowing the other thread to access the counter.

- Same as above, but for LockedMutableField.
- In the Producer-Consumer example, have each consumer add a log message to a StringBuilder shared among all threads when it takes an item for delivery. Ensure thread safety!