Hoare vs Mesa Monitors

This lesson looks at the different designs for implementing monitors.

Mesa vs Hoare Monitors

We discussed the abstract concept of a monitor in the previous section. Now let's discuss the various designs for monitors.

Mesa Monitors

So far we have determined that the idiomatic usage of a monitor requires using a while loop as follows. Let's see how the design of monitors affects this recommendation.

```
        while condition == false
            condVar.wait()
        end
```

Once the asleep thread is signaled and woken up, you may ask why does it needs to check for the condition being false again if the signaling thread must have just set the condition to true?

In **Mesa monitors** - Mesa is a language developed by Xerox researchers in the 1970s - it is possible that in the time gap between when thread B calls `notify()` and releases its mutex **and** the instant at which the asleep thread A wakes up and reacquires the mutex, ***the predicate is changed back to false by another thread different than the signaler and the awoken threads!*** The woken up thread competes with other threads to acquire the mutex once the signaling thread B **empties** the monitor. On signaling, thread B doesn't give up the monitor just yet; rather it continues to own the monitor until it exits the monitor section.

Hoare Monitors

In contrast in **Hoare monitors** - Hoare being one of the original inventors of monitors - the signaling thread B ***yields*** the monitor to the woken up thread A and thread A ***enters*** the monitor while thread B sits out. This guarantees that the predicate will not have changed and instead of checking for the predicate in a while loop, an if-clause would suffice. The woken-up/released thread A immediately starts execution when the signaling thread B signals that the predicate has changed. No other thread gets a chance to change the predicate since no other thread gets to enter the monitor.

Mesa monitors are more efficient than Hoare monitors.

Several popular programming languages such as Java, Python, C#, etc. subscribe to Mesa monitor semantics and so does Ruby. Thus the developer is always expected to check for condition/predicate in a while loop.