Colloquium G - Concurrent Programming

Simon Hasir – 7006072

June 20, 2022

---

1 TE-1

(a) Yes, but only if java is executed with the *-ea* flag which enables Assertions.

(b) No, this behavior would involve too much overhead, but it's implemented for some java classes.

(c) No, not in the Bytecode.

(d) No, comments don't have any semantic, but it's good practice. ;)

(e) Kind of, *await* just continues if it gets notified, BUT you have to check for the negation.

(f) No, Linked Listed have to be monitors as well.

(g) No, *synchronized* can be applied almost as flexible, the only problem is that only one lock per object is used.

(h) Yes

(i) Yes, because multiple locks lead to smaller wait sets, which makes *notifyAll()* faster.

(j) No, only if the class is a monitor that uses only implicit locking.

(k) No, not even if you can ensure that *notify()* is called after *wait()*, because Java could wake up threads without signal as well ⇒ you need a while loop.

(l) Yes, safety violations can be found in finite executions and liveness problem only appear in infinite executions.

(m) No, see the above.

(n) No, *run()* is a sequential function which you can execute in a new Thread with *Thread.start()*

(o) Yes

(p) Yes, *wait()* stops execution.

(q) Yes, *notifyAll()* would be correct.

(r) No, you can only make methods and blocks synchronized.

(s) The *signal()* call is valid, but the functionality may be impaired (e.g. thread that prints wrong result after to signals).

(t) Yes.

(u) True, once it crashes it can't recover. You can give a finite execution.

(v) No, it only sets the boolean flag *interrupted* (fetchable with *isInterrupted()*)

(w) Yes

2 TG-2

1. Monitor pseuCo -> explicit locks Java
   (a) Replace *monitor X* in pseuCo with *class X* in Java.
   (b) Add a *private final* intantice-wise lock *l* in Java.
   (c) Replace condition *c* with *b* in pseuCo with *private final c = l.newCondition()* in Java.
   (d) Add lock at beginning of every function(wrapped in a try) and an unlock in the final block. Add trowInterruptException.
   (e) Replace waitForCondition(*c*) in pseuCo with while (!*b*) c.await(); in Java
2. Explicit locks Java -> Implicit Locks Java
   (a) remove explicit lock
   (b) Replace locking and unlocking with the *synchronized* keyword in the function definition.
   (c) Make every field private
   (d) Replace c.await with wait();
   (e) Replace signal[All] with notify[All].
   (f) Add some kind of indication which was condition should be the one that was triggered.
   (g) Enjoy :)

3 TG-3

HashSet or Ringbuffer (of TreeSets) if you want good runtime, that save the counter for everyone in line, increment on overtake and check that nobody is at the cap.