**If you call wait() and notify() on an object, it must be inside of a block synchronized on the same object. What happens if not? Why is it necessary to own the lock on an object before calling wait() and notify() on it?**

This could be explained by analyzing the vote counting example provided in class:

public void setVotes (int[] votes) {

this.votes = votes;

synchronized (this) {

this.votes = votes;

notifyAll();

}

}

. . .

public void run () {

synchronized (this) { //mind the order of synchronized and while

while (votes == null) {

try {

wait();

} catch (InterruptedException e) {

// Auto-generated catch block

e.printStackTrace();

}

}

}

countVotes();

}

The most apparent reason for why notify() and wait() should be inside a synchronized block is that the wait and notify methods should be mutually exclusive – only one can happen at a time, instead of wait and notify being called at the same time. To achieve this, the two calls need to share the same lock by synchronized statements with the same object lock.

Another reason is that, in addition to being notified from waiting, a process needs to check whether a condition is fulfilled before they can wake up. For example, the condition for run() is votes==null, but votes is set by another method setVotes(). To ensure that the condition is thread safe, we need synchronized block for both parts of the code that sets/checks the same condition.