<https://stackoverflow.com/questions/1038007/why-should-wait-always-be-called-inside-a-loop>

# [Why should wait() always be called inside a loop](https://stackoverflow.com/questions/1038007/why-should-wait-always-be-called-inside-a-loop)

Q: I have read that we should always call a wait() from within a loop:

while (!condition) { obj.wait(); }

It works fine without a loop so why is that?

A1:

You need not only to loop it but check your condition in the loop. Java does not guarantee that your thread will be woken up only by a notify()/notifyAll() call or the right notify()/notifyAll() call at all. Because of this property the loop-less version might work on your development environment and fail on the production environment unexpectedly.

For example, you are waiting for something:

synchronized (theObjectYouAreWaitingOn) {

while (!carryOn) {

theObjectYouAreWaitingOn.wait();

}

}

An evil thread comes along and:

theObjectYouAreWaitingOn.notifyAll();

If the evil thread does not/can not mess with the carryOn you just continue to wait for the proper client.

**Edit:** Added some more samples. The wait can be interrupted. It throws InterruptedException and you might need to wrap the wait in a try-catch. Depending on your business needs, you can exit or suppress the exception and continue waiting.

A2:

It's answered in [documentation for Object.wait(long milis)](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Object.html#wait(long))

A thread can also wake up without being notified, interrupted, or timing out, a so-called spurious wakeup. While this will rarely occur in practice, applications must guard against it by testing for the condition that should have caused the thread to be awakened, and continuing to wait if the condition is not satisfied. In other words, waits should always occur in loops, like this one:

synchronized (obj) {

while (<condition does not hold>)

obj.wait(timeout);

... // Perform action appropriate to condition

}

(For more information on this topic, see Section 3.2.3 in Doug Lea's "Concurrent Programming in Java (Second Edition)" (Addison-Wesley, 2000), or Item 50 in Joshua Bloch's "Effective Java Programming Language Guide" (Addison-Wesley, 2001).