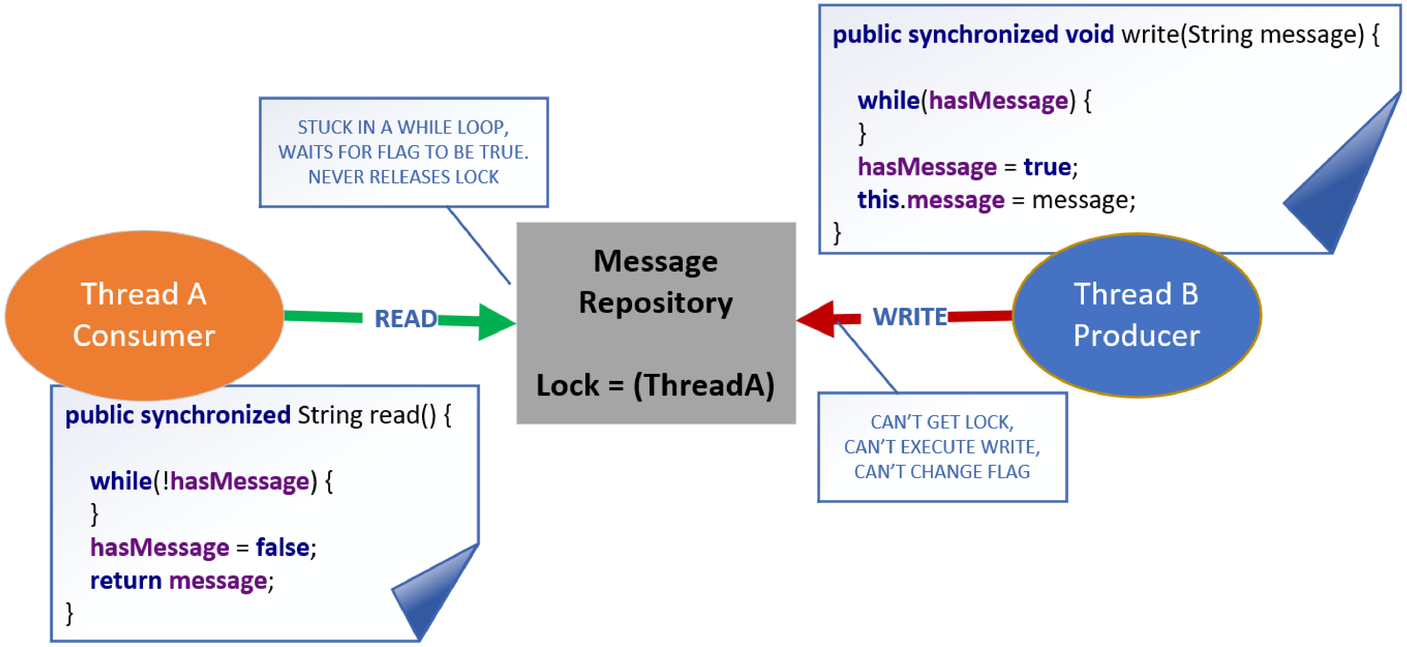
Deadlock in the Consumer Producer Example

In an earlier video, I demonstrated a deadlock, with the Consumer Producer sample code.

Both classes were accessing a single boolean field, on a shared object, in a while loop with no code in it.

We resolved this situation by using the notifyAll, and wait methods on object, in both of these methods.

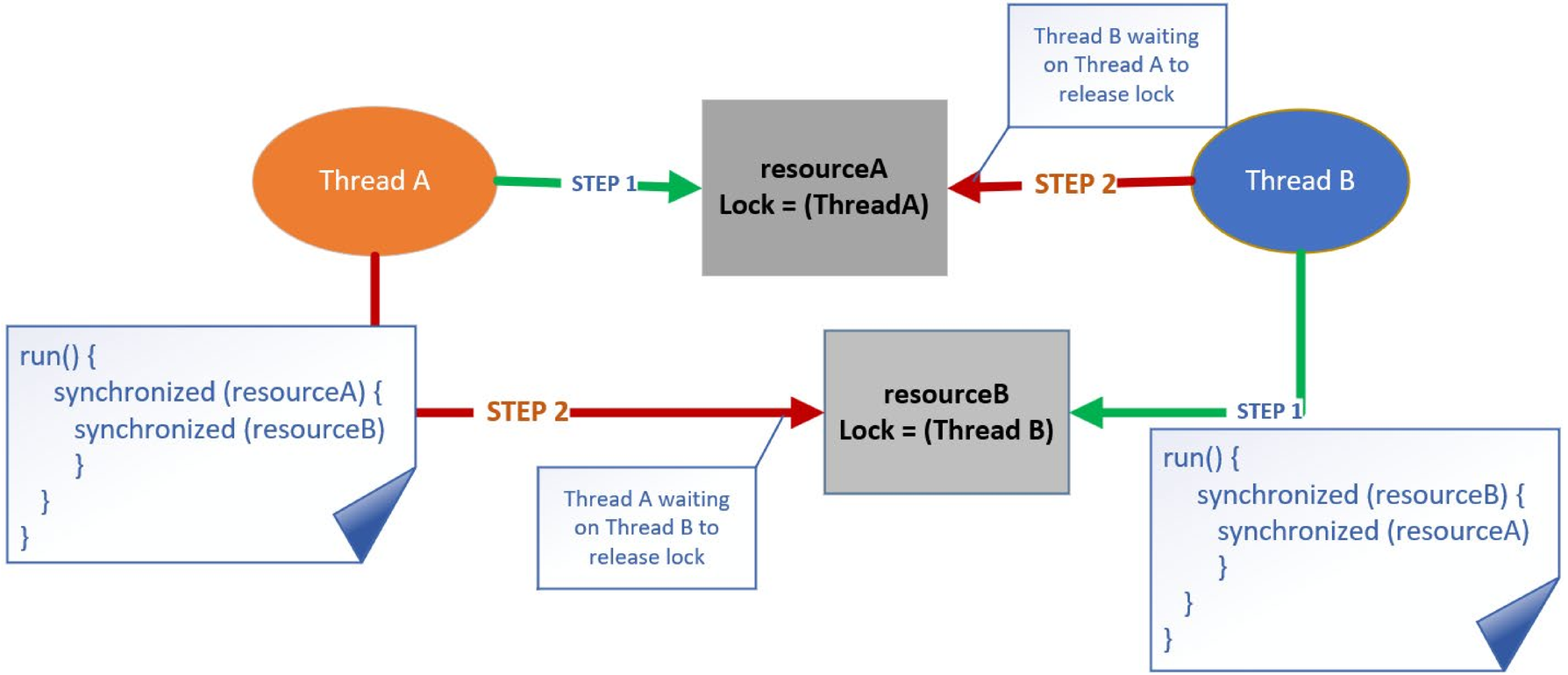


The Common Problems in a Multi-Threaded Application

Problem	Description
Deadlock	Two or more threads are blocked, waiting for each other to release a resource.
Livelock	Two or more threads are continuously looping, each waiting for the other thread to take some action.
Starvation	A thread is not able to obtain the resources it needs to execute.

A Second Deadlock Scenario

When both threads need access to multiple resources in a certain order, this can cause contention, and produce a deadlock.



Preventing Deadlocks

A couple of common ways to avoid this kind of deadlock situation, are listed here.

- Organize your locks into a hierarchy, and ensure that all threads acquire locks in the same order to avoid circular waiting, which is a common cause of deadlocks. This approach helps establish a global lock order that all threads must follow.
- Instead of using traditional synchronized blocks or methods, you can use the tryLock method on the Lock interface. This method allows you to attempt to acquire a lock. If it fails, you can handle the situation without causing a deadlock.