synchronized vs ReentrantLock + Condition?

**🔹 What is a Condition Variable?**

A **condition variable** allows threads to wait (pause) until a specific condition is true.

In Java, with synchronization, we typically use:

wait()

notify()

notifyAll()

These are **bound to the single intrinsic lock** (monitor) of the object. Meaning:  
🔁 All threads waiting on that lock are **not distinguishable by reason or condition**.

**🔴 Limitation of synchronized:**

synchronized doesn't support multiple condition variables, offering only a single monitor per object with basic wait/notify mechanisms

* Only **one condition queue** per monitor lock.
* If multiple threads are waiting for **different reasons**, you **can’t wake only a specific group**.

❌ No fine-grained control. ❌ You cannot have separate wait() queues for different conditions.

**✅ Example (Problem):**

Suppose you have a **Producer-Consumer** model:

* Consumer should wait if the buffer is empty.
* Producer should wait if the buffer is full.

With synchronized, you only have **one wait() list**.

synchronized (lock) {

while (buffer.isEmpty()) {

lock.wait();

}

// consume

}

Now the problem is: notify() could wake up **any waiting thread**, even a **producer** when it should be the **consumer**'s turn.