**🔹 What is a Condition Variable?**

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

**🔑 What Are Condition Variables?**

**Think of a condition variable as a "waiting room" for threads.**

**Using synchronized, all threads share the same single waiting room (via wait() and notify()).**

**But with ReentrantLock + Condition, you can create multiple waiting rooms, and send signals to specific groups of threads.**

**🧠 Why Is This Useful?**

**In problems like Producer-Consumer, you may want:**

* **Consumers to wait when buffer is empty.**
* **Producers to wait when buffer is full.**

**You want to signal the right group, not all waiting threads.**

**✅ How Does Condition Help?**

**You can do this:**

**Lock lock = new ReentrantLock();**

**Condition notFull = lock.newCondition(); // for producer**

**Condition notEmpty = lock.newCondition(); // for consumer**

**So:**

* **notFull.await() — producer waits when buffer is full**
* **notEmpty.await() — consumer waits when buffer is empty**
* **notFull.signal() — wake a producer**
* **notEmpty.signal() — wake a consumer**

**🎯 Now you control who waits and who gets notified!**

**🧪 Code Example: Producer-Consumer with ReentrantLock and Condition**

**import java.util.\*;**

**import java.util.concurrent.locks.\*;**

**public class ProducerConsumer {**

**private final Queue<Integer> buffer = new LinkedList<>();**

**private final int capacity = 5;**

**private final Lock lock = new ReentrantLock();**

**private final Condition notFull = lock.newCondition();**

**private final Condition notEmpty = lock.newCondition();**

**public void produce(int value) throws InterruptedException {**

**lock.lock();**

**try {**

**while (buffer.size() == capacity) {**

**notFull.await(); // wait if buffer is full**

**}**

**buffer.add(value);**

**System.out.println("Produced: " + value);**

**notEmpty.signal(); // notify consumer**

**} finally {**

**lock.unlock();**

**}**

**}**

**public void consume() throws InterruptedException {**

**lock.lock();**

**try {**

**while (buffer.isEmpty()) {**

**notEmpty.await(); // wait if buffer is empty**

**}**

**int val = buffer.poll();**

**System.out.println("Consumed: " + val);**

**notFull.signal(); // notify producer**

**} finally {**

**lock.unlock();**

**}**

**}**

**}**

**🟡 Summary**

| **Feature** | **synchronized + wait/notify** | **ReentrantLock + Condition** |
| --- | --- | --- |
| **Condition support** | **Only one condition** | **Multiple conditions via newCondition()** |
| **Wait methods** | **wait()** | **await()** |
| **Notify methods** | **notify(), notifyAll()** | **signal(), signalAll()** |
| **Flexibility** | **Limited** | **High** |
| **Thread interruption support** | **No** | **Yes (with lockInterruptibly())** |

**Let me know if you want the full runnable version with main() method and two threads: one producer and one consumer.**