**Java Thread Communication Interview Questions with Detailed Answers**

**1. What is thread communication in Java?**

**Answer:** Thread communication refers to the interaction between threads to coordinate the execution of code that accesses shared resources. Java provides methods like wait(), notify(), and notifyAll() for inter-thread communication. A classic use case is the Producer-Consumer problem, where producer and consumer threads share a common buffer.

**2. What is the difference between wait() and sleep()?**

**Answer:**

* wait() is used for inter-thread communication. It must be called from within a synchronized block and releases the monitor lock.
* sleep() is used to pause a thread for a specific time. It does not release the lock.

|  |  |  |
| --- | --- | --- |
| **Feature** | **wait()** | **sleep()** |
| Lock release | Yes | No |
| Use case | Thread communication | Pause execution |
| Needs sync? | Yes | No |

**3. How does wait() work in synchronized blocks/methods?**

**Answer:** wait() can only be called from a block/method that is synchronized. When a thread calls wait(), it gives up the monitor lock and goes into a waiting state until another thread calls notify() or notifyAll() on the same monitor.

**4. Why do we use while instead of if with wait()?**

**Answer:** Due to spurious wakeups (where threads wake up without notification), it is recommended to use while instead of if to recheck the condition:

while (buffer.isEmpty()) {

wait();

}

This ensures correctness even if the thread is awakened unexpectedly.

**5. What are notify() and notifyAll()?**

**Answer:**

* notify(): Wakes up one waiting thread.
* notifyAll(): Wakes up all threads waiting on the object's monitor. Use notifyAll() when multiple types of threads (e.g., producers and consumers) may be waiting.

**6. Implement Producer-Consumer using wait/notify**

**Answer:** Already done in earlier examples. Key points:

* Use synchronized methods.
* Use wait() when buffer is full/empty.
* Use notifyAll() instead of notify() when multiple producers/consumers exist.

**7. What will happen if two producers use notify() instead of notifyAll()?**

**Answer:** If one producer notifies and wakes up another producer while the buffer is full, that thread will again wait, causing all producers to be stuck. Deadlock can occur if consumers are not notified.

**8. What are the limitations of wait/notify?**

**Answer:**

* All threads wait on the same monitor.
* Risk of deadlock if used incorrectly.
* Difficult to manage complex conditions.
* Cannot interrupt waiting threads easily.

**9. Why is notifyAll() safer than notify()?**

**Answer:** Because notify() may wake a thread not ready to proceed (e.g., a producer when buffer is full), leading to starvation. notifyAll() wakes all waiting threads, letting only the eligible ones proceed after rechecking their conditions.

**10. How does BlockingQueue internally handle producer-consumer?**

**Answer:** BlockingQueue handles thread safety and blocking internally:

* put() blocks if the queue is full.
* take() blocks if the queue is empty. It uses locks and Condition objects to manage thread communication, avoiding manual use of wait() and notify().

**11. When would you use thread communication in real-world microservices?**

**Answer:** In scenarios like:

* Internal caching mechanisms.
* In-memory job queues.
* Event batching and processing.

**12. What is the difference between busy-waiting and condition-waiting?**

**Answer:**

* **Busy-waiting:** Thread continuously checks a condition using a loop. Wasteful on CPU.
* **Condition-waiting:** Thread goes to sleep using wait() or await() and wakes only on notification.

**13. How does Condition improve over wait/notify?**

**Answer:**

* Allows multiple condition queues.
* Better separation of producer and consumer signals.
* More flexible and readable with await(), signal(), signalAll().

**14. Design a thread-safe task scheduler using wait-notify**

**Answer:** Use a shared queue:

* Producer adds tasks.
* Worker threads wait if empty.
* On task arrival, notify waiting workers.

**15. Have you faced thread communication issues in your project?**

**Answer:** *(Sample)* Yes, once a deadlock occurred due to using notify() instead of notifyAll() with multiple producers. After debugging, we switched to notifyAll() and later replaced the custom logic with a BlockingQueue.

**16. Can notify() wake a thread of a different condition?**

**Answer:** Yes. All threads wait on the same monitor, so notify() might wake a thread not ready to proceed. This is why we prefer notifyAll().

**17. What happens if you call notify() and no threads are waiting?**

**Answer:** Nothing happens. No error is thrown. The notification is lost.

**18. Can two threads call wait() on different monitors and still coordinate?**

**Answer:** No. wait() and notify() must be used on the same object monitor to coordinate.

Prepared by: ChatGPT Based on discussion with Gyanenjay (2025)

What is the purpose of `BlockingQueue`?

36. How does `LinkedBlockingQueue` differ from `ArrayBlockingQueue`?