**Multithreading – Part-10 – Inter Thread Communication Part-01**

* **Introduction:**

Two threads can communicate with each other by using wait(), notify() and notfiyAll() methods.

The thread which is expecting update is responsible to call wait() method then immediately the thread will enter into waiting state.

The thread which is responsible to perform update, after performing update it is responsible to call notify method. Then waiting thread will get that notification and continue its execution with those updated items.

Note: Recall about the Durga sir waited for letter from her girlfriend.

wait() notify() notifyAll() methods present in Object class but not in thread class, because thread can call these methods on any Java object.

To call wait() notify() or notifyAll() methods on any object, thread should be owner of that object. That is the thread should have lock of that object, that is the thread should be inside synchronized area.

Hence we can call wait(), notify() and notifyAll() methods only from synchronized area, otherwise we will get Runtime Exception saying:

IllegalMonitorStateException

If a thread calls wait() on any object it immediately releases lock of that particular object and it will enter into waiting state.

If a thread calls notify() method on object it releases the lock of that object but may not immediately.

Except wait(), notify() and notifyAll() there is no other method where thread releases the lock.

Reason behind releasing the lock:

We call the wait() to get a notification, notify() or notifyAll() method needs the lock of the object to give the notification, so if we don’t release the lock once we call the wait() method, notify() method won’t be able to notify us.

|  |  |
| --- | --- |
| Method | Is Thread releases Lock? |
| yield() | NO |
| join() | NO |
| sleep() | NO |
| wait() | YES |
| notify() | YES |
| notifyAll() | YES |

* **Which of the following is valid?**

1. If a thread calls wait() immediately it will enter into waiting state without release any lock. //Invalid
2. If a thread calls wait(), it releases the lock of that object but may not immediately. // Invalid
3. If a thread calls wait() on any object, it releases all locks acquired by that thread and immediately enter into waiting state. // Invalid.
4. If a thread calls wait() on any object, it immediately releases the lock of that particular object and enter into waiting state. // Valid
5. If a thread calls notify() on any object it immediately releases the lock of that particular object. // Invalid
6. If a thread calls notify() on any object it releases the lock of that object but may not immediately. // Valid

* **Signature:**

public final void wait() throws InterruptedException

public final native void wait(long ms) throws InterruptedException

public final void wait(long ms, int ns) throws InterruptedException

public final native void notify()

public final native void notifyAll()

Note:

Every wait() method throws InterruptedException which is checked exception. Hence whenever we are using wait() method compulsory we should handle this InterruptedException, either by try/catch or by throws keyword, otherwise we will get compile time error.

* **Change of Thread life cycle with wait().**

MyThread t = new MyThread()

New/Born state

t.start()

Read/Runnable state

If TS allocates processor

Running State

Obj.wait()

Obj.wait(1000)

Obj.wait(1000, 100) (waiting state)

Another waiting state

If got the lock, then Runnable

State.

If run() method completes

Dead State

Another waiting state:

Wait method will not go back to Runnable state directly, in between there is a state, where it will try to get the lock.

It will enter into the get lock state, if the following things

Happen.

1. If waiting thread got notification.
2. If the time expires
3. If the waiting thread got interrupted.