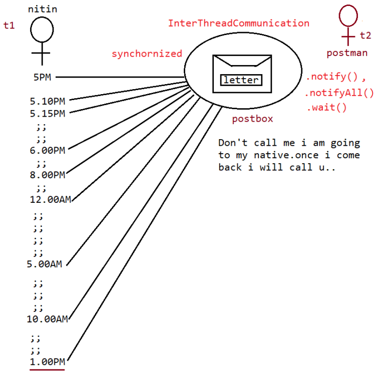


Don't call me i am going
to my native.once i come
back i will call u..

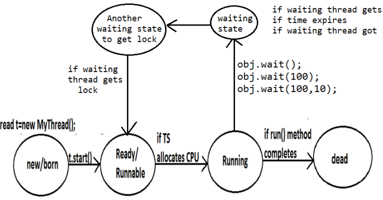
Which ever thread wants to get the resource/information from another thread, then that thread should call "wait()"

Which ever thread updates the resoruce/information to other thread, that thread shoulud call "notify()/notifyAll()".

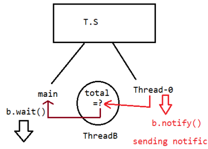
wait()
- |=> thread should be owner
- |=> automatically the thread which calls wait() would release the lock, so other thread can use that lock and perform some updation operation

notify()
- |-> send a notification to the waiting thread
- |-> the thread which calls notify()/notifyAll() may or may not release the lock immediately

Life Cycle of Thread



if waiting thread gets notification
if time expires
if waiting thread got interrupted

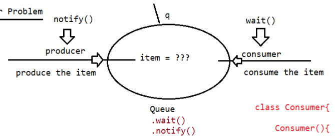


sending notification to the waiting thread.
lock may or may not release immediately

main thread is in waiting state
released the lock and gave it to other thread

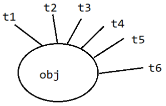
main thread got the notification,waited for the lock to be
released from the other thread, once the lock is released
it will use ThreadB total variable

Producer and Consumer Problem



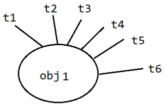
```
class Producer{

     Producer(){

           synchronized(q){

                 produce the item to the queue
                     q.notify();

           }
     }
}
```

```
class Consumer{

     Consumer(){

           synchronized(q){

                 if(q.isEmpty())
                       q.wait();
                 else
                       consume the item
           }
     }
```

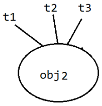


obj.notify()

notify() is used to give notification to one thread.
if there are mulitple waiting threads,then only one
thread will get a chance,remaining threads should wait
for further notification.
But which thread will get a chance is totaly depends
on T.S



obj1.notifyAll()



obj2.notifyAll()

notifyAll() is used to give to notification to all waiting threads. All waiting threads
will be notified. and it wil be exectued one by one.