Java Concurrency CountDownLatch

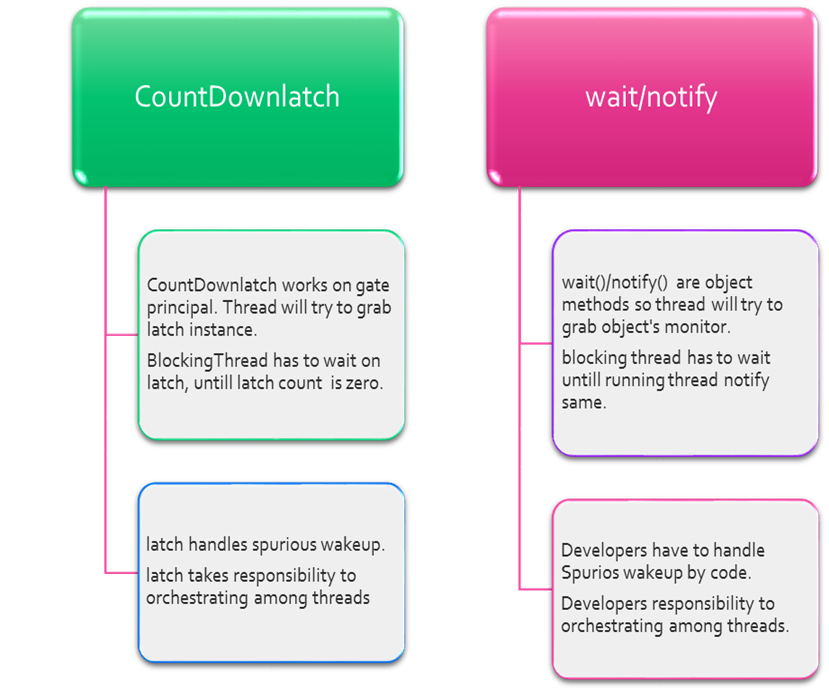
# CountDown Latch

In this article we will discuss about CountDownLatch . Same is introduced in java 5.

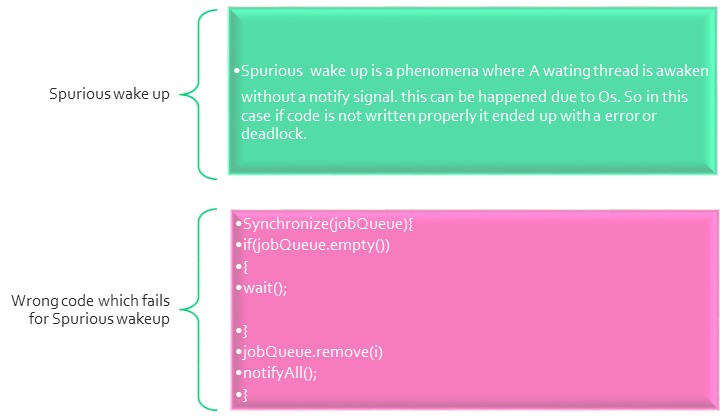
CountDownLatch works very same way as Object’s wait() and notify() does ,but it is a Toplevel class

And very easy to use than wait/notify.

**CountDownLatch VS wait()/ notify()**



Spurious Wake up:



look above code snippet, in case of spurious wake up, Thread goes to JobQueue.remove(i) line as it is suddenly wakes up due to spurious phenomena, but there is no job in queue (as producers does not produce a job), so above code throws **NoSuchElementException**.

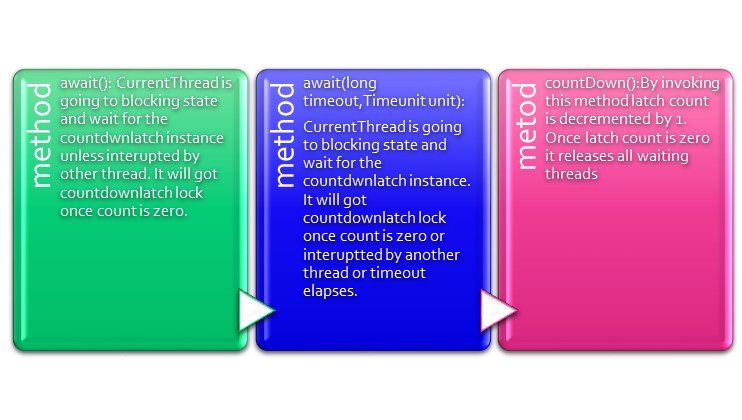
To solve this problem, we are going to make a slight change in above code.

put, **while(JobQueue.empty())** instead of ,**if(JobQueue.empty())** , Now in case of spurious wakes up ,**while(JobQueue.empty())**  always holds true as producer does not produces a job and put the same in JobQueue.

While using CountDownlatch, developers don’t have to bother about spurious wake-up as CountDownlatch implements on gate principle.

CountDownlatch initializes with a positive number and blocking thread has to wait until this number becomes  Zero.   If a Spurious wake up occurs and count is greater than zero obviously blocking thread must has to wait as count is not zero, blocking threads only got chance, when running thread invoke countdown() and count decreases to zero.

Important Methods of CountDownlatch:



CountDownLatch has one demerits once it’s counter is zero it can’t be reset again in order to reset one should use CyclicBarrier.

Example: In a Hotel Food only be served once prepared. And obviously preparing and serving foods are two different service but it has to maintain an order, unless food is prepared it can’t be served. We will solve this problem by CountDownlatch.

Code :

**package** com.example.concurrency.countdownlatch;

**import** java.util.concurrent.CountDownLatch;

**import** java.util.concurrent.ExecutorService;

**import** java.util.concurrent.Executors;

**public** **class** PrepareServeProblem {

**public** **static** **void** main(String[] args) {

ExecutorService service = Executors.*newFixedThreadPool*(2);

CountDownLatch latch = **new** CountDownLatch(1);

Runnable prepareService = **new** Runnable()

{

**public** **void** run()

{

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.***out***.println("Preparing Food");

latch.countDown();

}

};

Runnable serveService = **new** Runnable()

{

**public** **void** run()

{

**try** {

latch.await();

System.***out***.println("wating for prepareService to finish");

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.***out***.println("Serving the Food");

}

};

service.execute(serveService);

service.execute(prepareService);

}

}

Output:

Preparing Food

wating for prepareService to finish

Serving the Food