How is CountDownLatch used in Java Multithreading?

Asked 7 years, 1 month ago Active 1 year ago Viewed 117k times

184

79

Can someone help me to understand what Java `CountDownLatch` is and when to use it?

I don't have a very clear idea of how this program works. As I understand all three threads start at once and each Thread will call CountDownLatch after 3000ms. So count down will decrement one by one. After latch becomes zero the program prints "Completed". Maybe the way I understood is incorrect.

```
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Processor implements Runnable {
    private CountDownLatch latch;

    public Processor(CountDownLatch latch) {
        this.latch = latch;
    }

    public void run() {
        System.out.println("Started.");

        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        latch.countDown();
    }
}


// ----------------------------------------------------


public class App {

    public static void main(String[] args) {

        CountDownLatch latch = new CountDownLatch(3); // coundown from 3 to 0

        ExecutorService executor = Executors.newFixedThreadPool(3); // 3 Threads in pool

        for(int i=0; i < 3; i++) {
            executor.submit(new Processor(latch)); // ref to latch. each time call new
Processes latch will count down by 1
        }

        try {
            latch.await();  // wait until latch counted down to 0
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

```
            System.out.println("Completed.");
        }

    }
```

java multithreading countdown countdownlatch

edited Sep 6 '17 at 21:16

ROMANIA_engineer
44.5k 23 184 168

asked Jul 24 '13 at 6:47

amal
2,960 8 23 40

2 docs.oracle.com/javase/7/docs/api/java/util/concurrent/… – Christophe Roussy Mar 2 '16 at 10:18

9 I just used your question sample code for an android parallel service batch and it worked like a charm. Thank you so much! – Roisgoen Jul 8 '16 at 4:05

Got here from this video from 2012, which shows *remarkable resemblance* to the example shown here. For anyone interested, this is part of a Java multi-threading tutorial series from a guy named John. I like John. Highly recommended. – Elia Grady May 30 '18 at 19:58

12 Answers

Active | Oldest | Votes

197

Yes, you understood correctly. `CountDownLatch` works in latch principle, the main thread will wait until the gate is open. One thread waits for *n* threads, specified while creating the `CountDownLatch`.

Any thread, usually the main thread of the application, which calls `CountDownLatch.await()` will wait until count reaches zero or it's interrupted by another thread. All other threads are required to count down by calling `CountDownLatch.countDown()` once they are completed or ready.

As soon as count reaches zero, the waiting thread continues. One of the disadvantages/advantages of `CountDownLatch` is that it's not reusable: once count reaches zero you cannot use `CountDownLatch` any more.

Edit:

Use `CountDownLatch` when one thread (like the main thread) requires to wait for one or more threads to complete, before it can continue processing.

A classical example of using `CountDownLatch` in Java is a server side core Java application which uses services architecture, where multiple services are provided by multiple threads and the application cannot start processing until all services have started successfully.

P.S. OP's question has a pretty straightforward example so I didn't include one.

edited Apr 4 '18 at 19:13 answered Jul 24 '13 at 7:06

NikolaB
3,779 2 25 34

---

1 Thank you for reply. Could you give me an example where to apply CountDown latch ? – amal Jul 24 '13 at 7:12

11 a tutorial of how to use CountDownLatch is here howtodoinjava.com/2013/07/18/… – thiagoh Oct 7 '13 at 19:30

1 @NikolaB But in this given example we can achieve same result by using join method isn't it ? – Vikas Verma Sep 27 '14 at 11:32

3 I would consider the non-reusability an advantage: you're sure that no one can reset it, or increase the count. – ataulm Mar 16 '15 at 9:38

@Spiderman if i use the same mechanism but instead of using threads i make the main application wait for services to complete in android,will it work the same way? – Jude Fernandes May 25 '17 at 18:25

Wrt your point: *One thread waits for n number of threads specified while creating CountDownLatch in Java.* Do you mean ordering as mentioned here? – overexchange Oct 24 '17 at 17:43

3 Nice explanation. But I would slightly disagree on the point `One thread waits for n number of threads specified while creating CountDownLatch in Java` . If you need such mechanism, then it is prudent to use `CyclicBarrier` . The fundamental conceptual difference between these two, as given in `Java concurrency in Practice` is: `Latches are for waiting for events; barriers are for waiting for other threads` . `cyclicBarrier.await()` goes into a blocking state. – RDM Nov 13 '17 at 13:02

From the book: `Threads call await when they reach the barrier point, and await blocks until all the threads have reached the barrier point` . Whereas in `CountdownLatch` , a single thread can generate `n` numbers of events to open the latch. Check this link:tutorials.jenkov.com/java-util-concurrent/countdownlatch.html – RDM Nov 13 '17 at 13:02

do we use the `new CountDownLatch(3)` as we have 3 threads from the `newFixedThreadPool` defined? – Chaklader Asfak Arefe Jan 30 '18 at 6:51