Java CountDownLatch: Key Methods

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of Java CountDownLatch
- Recognize the key methods in Java CountDownLatch

```java
<<Java Class>>

CountDownLatch

- CountDownLatch(int)
- await():void
- await(long, TimeUnit):boolean
- countDown():void
```
Key Methods in Java
CountDownLatch
Key Methods in Java CountDownLatch

- CountDownLatch has a very simple API
  - i.e., only a handful of methods that are commonly used

```
<<Java Class>>

CountDownLatch

- countDownLatch(int)
- await(): void
- await(long, TimeUnit): boolean
- countDown(): void
```
Key Methods in Java CountDownLatch

- CountDownLatch’s constructor initializes the count

```java
public class CountDownLatch {
    ...
    public CountDownLatch (int count) {
        ...
        this.sync = new Sync(count);
    }
    ...
}
```
Key Methods in Java CountDownLatch

- CountDownLatch’s constructor initializes the count
- This count is simply used to create an instance of the AbstractQueuedSynchronizer

```java
public class CountDownLatch {
    ...
    public CountDownLatch(int count) {
        ...
        this.sync = new Sync(count);
    }
    ...
}
```
• CountDownLatch’s constructor initializes the count
• This count is simply used to create an instance of the AbstractQueuedSynchronizer
• The count cannot be reset without recreating a new instance of CountDownLatch

See upcoming lessons on “Java CyclicBarrier” & “Java Phaser” for alternatives
public class CountDownLatch {
    ...
    public void await() ...
        {
            sync.acquireSharedInterruptibly(1);
        }
    }

    public boolean await(long timeout, TimeUnit unit)
    {
        return sync.
            tryAcquireSharedNanos
            (1, unit.toNanos(timeout));
    }

    public void countDown()
    {
        sync.releaseShared(1);
    }
    ...

• Key methods count down & wait for the count to reach 0
Key Methods in Java CountDownLatch

- Key methods count down & wait for the count to reach 0

```java
public class CountDownLatch {
    ... 
    public void await() ... {
        sync.acquireSharedInterruptibly(1);
    }
}

public boolean await(long timeout, TimeUnit unit) ... {
    return sync.tryAcquireSharedNanos(1, unit.toNanos(timeout));
}

public void countDown() {
    sync.releaseShared(1);
}
... 
```

Methods forward to the underlying methods in the AbstractQueuedSynchronizer

See [gee.cs.oswego.edu/dl/papers/aqs.pdf](gee.cs.oswego.edu/dl/papers/aqs.pdf)
Key Methods in Java CountDownLatch

• Key methods count down & wait for the count to reach 0
• Causes the calling thread to block until the latch’s count reaches 0, at which point await() returns
• Unless the thread is interrupted

```java
public class CountDownLatch {
    ...
    public void await() ... {
        sync.acquire...(1);
    }
    ...
```
Key Methods in Java CountDownLatch

- Key methods count down & wait for the count to reach 0
  - Causes the calling thread to block until the latch’s count reaches 0, at which point await() returns
- Causes the calling thread to block until the latch’s count reaches 0, at which point await() returns
  - Unless waiting time elapses or the thread is interrupted

```java
public class CountDownLatch {
    ...
    public void await() ... {
        sync.acquire...(1);
    }

    public boolean await
        (long timeout,
         TimeUnit unit) ... {
        return sync.
            tryAcquireSharedNanos
                (1, unit.toNanos(timeout));
    }

    ...
}
```
Key Methods in Java CountDownLatch

- Key methods count down & wait for the count to reach 0
  - Causes the calling thread to block until the latch’s count reaches 0, at which point await() returns
  - Causes the calling thread to block until the latch’s count reaches 0, at which point await() returns

```java
cpyublic class CountDownLatch {
    ...
    public void await() ... {
        sync.acquire...(1);
    }
    ...
    pubic boolean await
    (long timeout,
     TimeUnit unit) ... {
        return sync.
            tryAcquireSharedNanos
            (1, unit.toNanos(timeout));
    }
    ...
```
Key Methods in Java CountDownLatch

- Key methods count down & wait for the count to reach 0
  - Causes the calling thread to block until the latch’s count reaches 0, at which point `await()` returns
  - Causes the calling thread to block until the latch’s count reaches 0, at which point `await()` returns
  - Decrements latch count by 1 & releases any threads blocked on `await()` when count reaches 0

```java
class CountDownLatch {
    ...
    public void await() ...
    {
        sync.acquire...(1);
    }
    ...
    public boolean await
    (long timeout,
     TimeUnit unit) ...
    {
        return sync.
        tryAcquireSharedNanos
        (1, unit.toNanos(timeout));
    }
    ...
    public void countDown()
    {
        sync.releaseShared(1);
    }
    ...
}
```
Key Methods in Java CountDownLatch

- Key methods count down & wait for the count to reach 0
- Causes the calling thread to wait until the latch’s count reaches 0, at which point await() returns
- Causes the calling thread to wait until the latch’s count reaches 0, at which point await() returns
- Decrements latch count by 1 & releases any threads blocked on await() when count reaches 0
- Threads calling countDown() don’t block for count to reach 0 before proceeding

```java
class CountDownLatch {
    public void await() {
        sync.acquire...(1);
    }

    public boolean await(long timeout, TimeUnit unit) {
        return sync.
            tryAcquireSharedNanos
                (1, unit.toNanos(timeout));
    }

    public void countDown() {
        sync.releaseShared(1);
    }

    ...
}
```
Key Methods in Java CountDownLatch

- Key methods count down & wait for the count to reach 0
  - Causes the calling thread to wait until the latch’s count reaches 0, at which point await() returns
  - Causes the calling thread to wait until the latch’s count reaches 0, at which point await() returns
  - Decrements latch count by 1 & releases any threads blocked on await() when count reaches 0
  - Threads calling countDown() don’t block for count to reach 0 before proceeding
  - In contrast, threads calling await() must block until count reaches 0 before proceeding

```java
public class CountDownLatch {
    ...

    public void await() {
        sync.acquire...(1);
    }

    public boolean await(long timeout, TimeUnit unit) {
        return sync.
            tryAcquireSharedNanos
            (1, unit.toNanos(timeout))
            ...
    }

    public void countDown() {
        sync.releaseShared(1);
    }

    ...
```
End of Java CountDownLatch: Key Methods