A `java.util.concurrent.CountDownLatch` is a concurrency construct that allows one or more threads to wait for a given set of operations to complete.

A `CountDownLatch` is initialized with a given count. This count is decremented by calls to the `countDown()` method. Threads waiting for this count to reach zero can call one of the `await()` methods. Calling `await()` blocks the thread until the count reaches zero.

Below is a simple example. After the `Decrementer` has called `countDown()` 3 times on the`CountDownLatch`, the waiting `Waiter` is released from the `await()` call.

```
CountDownLatch latch = new CountDownLatch(3);

Waiter      waiter      = new Waiter(latch);
Decrementer decrementer = new Decrementer(latch);

new Thread(waiter)     .start();
new Thread(decrementer).start();

Thread.sleep(4000);
```

```
public class Waiter implements Runnable{

    CountDownLatch latch = null;

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

    public void run() {
        try {
            latch.await();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Waiter Released");
    }
}
```

```
public class Decrementer implements Runnable {

    CountDownLatch latch = null;

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

    public void run() {

        try {
            Thread.sleep(1000);
            this.latch.countDown();

            Thread.sleep(1000);
            this.latch.countDown();

            Thread.sleep(1000);
            this.latch.countDown();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```