concurrent 包里面的 CountDownLatch 应用

应用场景

主线程想要往下执行，但必须要等到子线程的任务执行完毕后才可以继续往下执行。其中主线程调用个CountDownLatch对象的await()方法，其他子线程的任务执行完自己的任务后调用同一个CountDownLatch对象上的countDown()方法，直到这个CountDownLatch对象的计数值减到0为止，此时主程才能继续往下执行；否则主线程调用await()方法的任务将一直阻塞等待；

Example

- 有三个工人在为老板干活，这个老板有一个习惯，就是当三个工人把一天的活都干完了的时候，他来检查所有工人所干的活。记住这个条件：三个工人先全部干完活，老板才检查。所以在这里用Java码设计两个类，Worker代表工人，Boss代表老板，具体的代码实现如下:

- Boss code

```
public class Boss implements Runnable {

    private CountDownLatch downLatch;

    public Boss(CountDownLatch downLatch){
        this.downLatch = downLatch;
    }

    public void run() {
        System.out.println("老板正在等所有的工人干完活......");
        try {
            this.downLatch.await();
        } catch (InterruptedException e) {
        }
        System.out.println("工人活都干完了，老板开始检查了！");
    }

}
```

- Worker code

```
public class Worker implements Runnable{

    private CountDownLatch downLatch;
    private String name;

    public Worker(CountDownLatch downLatch, String name){
        this.downLatch = downLatch;
        this.name = name;
    }

    public void run() {
        this.doWork();
        try{
            TimeUnit.SECONDS.sleep(new Random().nextInt(10));
        }catch(InterruptedException ie){
        }
        System.out.println(this.name + "活干完了！ ");
        this.downLatch.countDown();

    }

    private void doWork(){
        System.out.println(this.name + "正在干活!");
    }

}
```

- Main code

```
public class CountDownLatchDemo {
    public static void main(String[] args) {
        ExecutorService executor = Executors.newCachedThreadPool();

        CountDownLatch latch = new CountDownLatch(3);

        Worker w1 = new Worker(latch,"张三");
        Worker w2 = new Worker(latch,"李四");
        Worker w3 = new Worker(latch,"王二");

        Boss boss = new Boss(latch);

        executor.execute(w3);
        executor.execute(w2);
        executor.execute(w1);
        executor.execute(boss);

        executor.shutdown();
    }

}
```

- 输出结果

```
王二正在干活!
张三正在干活!
李四正在干活!
老板正在等所有的工人干完活......
王二活干完了！
张三活干完了！
李四活干完了！
工人活都干完了，老板开始检查了！
```

串行执行也可以用到

```
        for (final String st : stArray) {

            CommonThreadPool.getThreadPool("getThread").execute(new Runnable() {
                @Override
                public void run() {
```

```
            try {
               System.out.println(st);
            } catch (RuntimeException e) {
               logger.error(LOGGER,, e));
            } finally {
                overLatch.countDown();
            }
        }
    });
}

try {
    overLatch.await();
} catch (InterruptedException e) {
    logger.error(LOGGER, , e));
}
```

原文链接：concurrent 包里面的 CountDownLatch 应用