

目录

第1章 基础

01 开篇词：为什么学习本专栏

02 String、Long 源码解析和面试题

03 Java 常用关键字理解

04 Arrays、Collections、Objects 常用方法源码解析

第2章 集合

05 ArrayList 源码解析和设计思路

06 LinkedList 源码解析

07 List 源码会问哪些面试题

08 HashMap 源码解析

09 TreeMap 和 LinkedHashMap 核心源码解析

10 Map源码会问哪些面试题

11 HashSet、TreeSet 源码解析

12 彰显细节：看集合源码对我们实际工作的帮助和应用

13 差异对比：集合在 Java 7 和 8 有何不同和改进

14 简化工作：Guava Lists Maps 实际工作运用和源码

第3章 并发集合类

15 CopyOnWriteArrayList 源码解析和设计思路

16 ConcurrentHashMap 源码解析和设计思路

17 并发 List、Map源码面试题

18 场景集合：并发 List、Map的应用

21 DelayQueue 源码解析

更新时间：2019-10-17 10:46:05



“

如果不想在世界上虚度一生，那就要学习一辈子。

——高尔基

”

引导语

之前我们说的阻塞队列，都是资源足够时立马执行。本章我们说的队列比较特殊，是一种延迟队列，意思是延迟执行，并且可以设置延迟多久之后执行，比如设置过 5 秒钟之后再执行，在一些延迟执行的场景被大量使用，比如说延迟对账等等。

1 整体设计

DelayQueue 延迟队列底层使用的是锁的能力，比如说要在当前时间往后延迟 5 秒执行，那么当前线程就会沉睡 5 秒，等 5 秒后线程被唤醒时，如果能获取到资源的话，线程即可立马执行。原理上似乎很简单，但内部实现却很复杂，有很多难点，比如当运行资源不够，多个线程同时被唤醒时，如何排队等待？比如说在何时阻塞？何时开始执行等等？接下来我们从源码角度来看看是如何实现的。

1.1 类注释

类注释上比较简单，只说了三个概念：

1. 队列中元素将在过期时被执行，越靠近队头，越早过期；

2. 未过期的元素不能够被 take；

3. 不允许空元素。

这三个概念，其实就是三个问题，下文我们会一一看下这三点是如何实现的。

1.2 类图

目录

第1章 基础

01 开篇词：为什么学习本专栏

02 String、Long 源码解析和面试题

03 Java 常用关键字理解

04 Arrays、Collections、Objects 常用方法源码解析

第2章 集合

05 ArrayList 源码解析和设计思路

06 LinkedList 源码解析

07 List 源码会问哪些面试题

08 HashMap 源码解析

09 TreeMap 和 LinkedHashMap 核心源码解析

10 Map源码会问哪些面试题

11 HashSet、TreeSet 源码解析

12 彰显细节：看集合源码对我们实际工作的帮助和应用

13 差异对比：集合在 Java 7 和 8 有何不同和改进

14 简化工作：Guava Lists Maps 实际工作运用和源码

第3章 并发集合类

15 CopyOnWriteArrayList 源码解析和设计思路

16 ConcurrentHashMap 源码解析和设计思路

17 并发 List、Map源码面试题

18 场景集合：并发 List、Map的应用

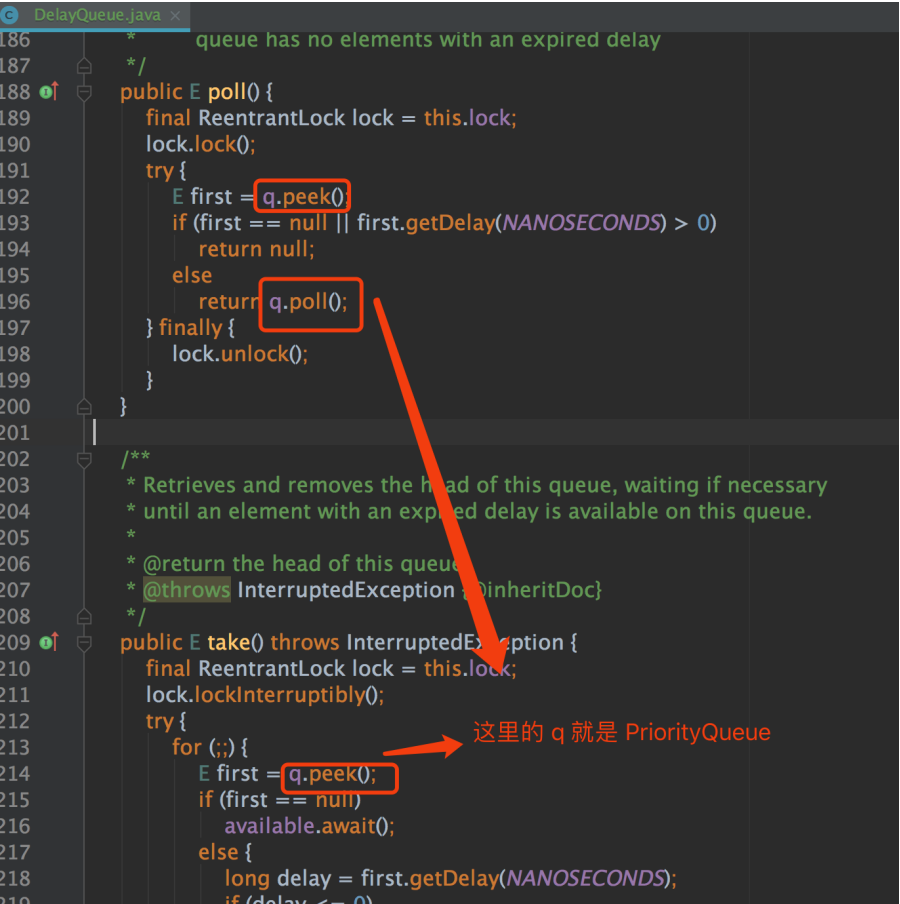
```
public class DelayQueue<E extends Delayed> extends AbstractQueue<E>
    implements BlockingQueue<E> {
```

从泛型中可以看出，DelayQueue 中的元素必须是 Delayed 的子类，Delayed 是表达延迟能力的关键接口，其继承了 Comparable 接口，并定义了还剩多久过期的方法，如下：

```
public interface Delayed extends Comparable<Delayed> {
    long getDelay(TimeUnit unit);
}
```

也就是说 DelayQueue 队列中的元素必须是实现 Delayed 接口和 Comparable 接口的，并覆写了 getDelay 方法和 compareTo 的方法才行，不然在编译时，编译器就会提醒我们元素必须强制实现 Delayed 接口。

除此之外 DelayQueue 还大量使用了 PriorityQueue 队列的大量功能，这个和 SynchronousQueue 队列很像，大量复用了其它基础类的逻辑，代码示例如下：



PriorityQueue 中文叫做优先级队列，在此处的作用就是可以根据过期时间做优先级排序，让先过期的可以先执行，用来实现类注释中的第一点。

这里的复用的思想还是蛮重要的，我们在源码中经常会遇到这种思想，比如说 LinkedHashMap 复用 HashMap 的能力，Set 复用 Map 的能力，还有此处的 DelayQueue 复用 PriorityQueue 的能力。小结一下，如果想要复用需要做到哪些：

- 1. 需要把能遇见可复用的功能尽量抽象，并开放出可扩展的地方，比如说 HashMap 在操作数组的方法中，都给 LinkedHashMap 开放出很多 after 开头的方法，便于

目录	DelayQueue 采用的组合，组合的意思就是把可复用的类给依赖进来。
第1章 基础	2 演示
01 开篇词：为什么学习本专栏	为了方便大家理解，写了一个演示的 demo，演示了一下：
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	
第2章 集合	
05 ArrayList 源码解析和设计思路	
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	

<div><div>← 慕课专栏</div><div>面试官系统精讲Java源码及大厂真题 / 21 DelayQueue 源码解析</div></div>	
目录	
第1章 基础	
01 开篇词：为什么学习本专栏	
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	
第2章 集合	
05 ArrayList 源码解析和设计思路	
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	<pre>@Override public int compareTo(Delayed o) { return (int) (this.getDelay(TimeUnit.MILLISECONDS) - o.getDelay(TimeUnit.MILLISECOND) } public void run(){ log.info("现在已经过了{}秒钟",(System.currentTimeMillis() - beginTime)/1000); } } // demo 运行入口 public static void main(String[] args) throws InterruptedException { BlockingQueue q = new DelayQueue(); DelayQueueDemo.Product p = new DelayQueueDemo.Product(q); DelayQueueDemo.Consumer c = new DelayQueueDemo.Consumer(q); new Thread(c).start(); new Thread(p).start(); } }</pre> <p>打印出来的结果如下：</p> <pre>06:57:50.544 [Thread-0] Consumer begin 06:57:50.544 [Thread-1] begin put 06:57:50.551 [Thread-1] end put 06:57:52.554 [Thread-0] 延迟了2秒钟才执行 06:57:55.555 [Thread-0] 延迟了5秒钟才执行 06:58:00.555 [Thread-0] 延迟了10秒钟才执行 06:58:00.556 [Thread-0] Consumer end</pre>
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	<pre>public boolean offer(E e) { final ReentrantLock lock = this.lock; // 上锁 lock.lock(); try { // 使用 PriorityQueue 的扩容，排序等能力 q.offer(e); // 如果恰好刚放进去的元素正好在队列头 // 立马唤醒 take 的阻塞线程，执行 take 操作 // 如果元素需要延迟执行的话，可以使其更快的沉睡计时 if (q.peek() == e) { leader = null; available.signal(); } } }</pre>

<div><div>← 慕课专栏</div><div>面试官系统精讲Java源码及大厂真题 / 21 DelayQueue 源码解析</div></div>	
目录	<pre>// 释放锁 lock.unlock(); } }</pre>
第1章 基础	
01 开篇词：为什么学习本专栏	可以看到其实底层使用的是 PriorityQueue 的 offer 方法，我们来看下：
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	
第2章 集合	
05 ArrayList 源码解析和设计思路	<pre>// 新增元素 public boolean offer(E e) { // 如果是空元素的话，抛异常 if (e == null) throw new NullPointerException(); modCount++; int i = size; // 队列实际大小大于容量时，进行扩容 // 扩容策略是：如果老容量小于 64，2 倍扩容，如果大于 64，50 % 扩容 if (i >= queue.length) grow(i + 1); size = i + 1; // 如果队列为空，当前元素正好处于队头 if (i == 0) queue[0] = e; else // 如果队列不为空，需要根据优先级进行排序 siftUp(i, e); return true; } // 按照从小到大的顺序排列 private void siftUpComparable(int k, E x) { Comparable<? super E> key = (Comparable<? super E>) x; // k 是当前队列实际大小的位置 while (k > 0) { // 对 k 进行减倍 int parent = (k - 1) >>> 1; Object e = queue[parent]; // 如果 x 比 e 大，退出，把 x 放在 k 位置上 if (key.compareTo((E) e) >= 0) break; // x 比 e 小，继续循环，直到找到 x 比队列中元素大的位置 queue[k] = e; k = parent; } queue[k] = key; }</pre>
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	<pre>(int) (this.getDelay(TimeUnit.MILLISECONDS) - o.getDelay(TimeUnit.MILLISECONDS)));</pre>
18 场景集合：并发 List、Map的应用	这样便可实现越快过期的元素越能排到队头。

目录

第1章 基础

01 开篇词：为什么学习本专栏

02 String、Long 源码解析和面试题

03 Java 常用关键字理解

04 Arrays、Collections、Objects 常用方法源码解析

第2章 集合

05 ArrayList 源码解析和设计思路

06 LinkedList 源码解析

07 List 源码会问哪些面试题

08 HashMap 源码解析

09 TreeMap 和 LinkedHashMap 核心源码解析

10 Map源码会问哪些面试题

11 HashSet、TreeSet 源码解析

12 彰显细节：看集合源码对我们实际工作的帮助和应用

13 差异对比：集合在 Java 7 和 8 有何不同和改进

14 简化工作：Guava Lists Maps 实际工作运用和源码

第3章 并发集合类

15 CopyOnWriteArrayList 源码解析和设计思路

16 ConcurrentHashMap 源码解析和设计思路

17 并发 List、Map源码面试题

18 场景集合：并发 List、Map的应用

4 拿数据

取数据时，如果发现有元素的过期时间到了，就能拿出数据来，如果没有过期元素，那么线程就会一直阻塞，我们以 take 为例子，来看一下核心源码：

```
for (;;) {
    // 从队头中拿数据出来
    E first = q.peek();
    // 如果为空，说明队列中，没有数据，阻塞住
    if (first == null)
        available.await();
    else {
        // 获取队头数据的过期时间
        long delay = first.getDelay(NANOSECONDS);
        // 如果过期了，直接返回队头数据
        if (delay <= 0)
            return q.poll();
        // 引用置为 null，便于 gc，这样可以让线程等待时，回收 first 变量
        first = null;
        // leader 不为空的话，表示当前队列元素之前已经被设置过阻塞时间了
        // 直接阻塞当前线程等待。
        if (leader != null)
            available.await();
        else {
            // 之前没有设置过阻塞时间，按照一定的时间进行阻塞
            Thread thisThread = Thread.currentThread();
            leader = thisThread;
            try {
                // 进行阻塞
                available.awaitNanos(delay);
            } finally {
                if (leader == thisThread)
                    leader = null;
            }
        }
    }
}
```

可以看到阻塞等待的功能底层使用的是锁的能力，这个我们在后面章节中会说到。

以上演示的 take 方法是会无限阻塞，直到队头的过期时间到了才会返回，如果不想无限阻塞，可以尝试 poll 方法，设置超时时间，在超时时间内，队头元素还没有过期的话，就会返回 null。

5 总结

DelayQueue 是非常有意思的队列，底层使用了排序和超时阻塞实现了延迟队列，排序使用的是 PriorityQueue 排序能力，超时阻塞使用得是锁的等待能力，可以看出 DelayQueue 其实就是为了满足延迟执行的场景，在已有 API 的基础上进行了封装，我们在工作中，可以学习这种思想，对已有的功能能复用的尽量复用，减少开发的工作量。

<div>← 慕课专栏</div> <div>≡ 面试官系统精讲Java源码及大厂真题 / 21 DelayQueue 源码解析</div>	
目录	欢迎在这里发表留言，作者筛选后可公开显示
第1章 基础	
01 开篇词：为什么学习本专栏	<div>慕盖茨4571687</div> <div>对key减倍应该怎么理解</div> <div>👍 0 回复</div> <div>2019-11-06</div>
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	<div>文贺 回复 慕盖茨4571687</div> <div>同学你好，可以简单理解成二分查找法</div> <div>回复</div> <div>2019-11-07 10:47:44</div>
第2章 集合	
05 ArrayList 源码解析和设计思路	千学不如一看，千看不如一练
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	