

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

# 25 整体设计：队列设计思想、工作中使用场景

更新时间：2019-10-31 19:36:53



“

人生太短，要干的事太多，我要争分夺秒。

——爱迪生

”

## 引导语

本章我们学习了 `LinkedBlockingQueue`、`ArrayBlockingQueue`、`SynchronousQueue`、`DelayQueue` 四种队列，四种队列底层数据结构各不相同，使用场景也不相同，本章我们从设计思想和使用场景两个大的方向做一些对比和总结。

## 1 设计思想

首先我们画出队列的总体设计图：

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

从图中我们可以看出几点：

1. 队列解耦了生产者和消费者，提供了生产者和消费者间关系的多种形式，比如 `LinkedBlockingQueue`、`ArrayBlockingQueue` 两种队列就把解耦了生产者和消费者，比如 `SynchronousQueue` 这种就把生产者和消费者相互对应（生产者的消息被消费者开始消费之后，生产者才能返回，为了方便理解，使用相互对应这个词）；
2. 不同的队列有着不同的数据结构，有链表（`LinkedBlockingQueue`）、数组（`ArrayBlockingQueue`）、堆栈（`SynchronousQueue`）等；
3. 不同的数据结构，决定了入队和出队的姿势是不同的。

接下来我们分别按照这几个方面来总结分析一下。

### 1.1 队列的数据结构

链表结构的队列就是 `LinkedBlockingQueue`，其特征如下：

1. 初始大小默认是 `Integer` 的最大值，也可以设置初始大小；
2. 链表元素通过 `next` 属性关联下一个元素；
3. 新增是从链表的尾部新增，拿是从链表头开始拿。

数组结构的队列是 `ArrayBlockingQueue`，特征如下：

1. 容量大小是固定的，不能动态扩容；
2. 有 `takeIndex` 和 `putIndex` 两个索引记录下下次拿和新增的位置；
3. 当 `takeIndex` 和 `putIndex` 到达数组的最后一个位置时，下次都是从 0 开始循环。

`SynchronousQueue` 有着两种数据结构，分别是队列和堆栈，特征如下：

1. 队列保证了先入先出的数据结构，体现了公平性；
2. 堆栈是先入后出的数据结构，是不公平的，但性能高于先入先出。

### 1.2 入队和出队的方式

不同的队列有着不同的数据结构，导致其入队和出队的方式也不同：

← 慕课专栏	:三 面试官系统精讲Java源码及大厂真题 / 25 整体设计：队列设计思想、工作中使用场景
目录	入队直接指向了 putIndex，出队指向了 takeIndex； 3. 堆栈主要都是围绕栈头进行入栈和出栈的。
第1章 基础	1.3 生产者和消费者之间的通信机制
01 开篇词：为什么学习本专栏	从四种队列我们可以看出来生产者和消费者之间有两种通信机制，一种是强关联，一种是无关联。
02 String、Long 源码解析和面试题	强关联主要是指 SynchronousQueue 队列，生产者往队列中 put 数据，如果这时候没有消费者消费的话，生产者就会一直阻塞住，是无法返回的；消费者来队列里取数据，如果这时候队列中没有数据，消费者也会一直阻塞住，所以 SynchronousQueue 队列模型中，生产者和消费者是强关联的，如果只有其中一方存在，只会阻塞，是无法传递数据的。
03 Java 常用关键字理解	无 关 联 主 要 是 说 有 数 据 存 储 功 能 的 队 列 ， 比 如 说 LinkedBlockingQueue 和 ArrayBlockingQueue，只要队列容器不满，生产者就能放成功，生产者就可以直接返回，和有消费者一点关系都没有，生产者和消费者完全解耦，通过队列容器的储存功能进行解耦。
04 Arrays、Collections、Objects 常用方法源码解析	2 工作中的使用场景
第2章 集合	在日常工作中，我们需要根据队列的特征来匹配业务场景，从而决定使用哪种队列，我们总结下各个队列适合使用的场景：
05 ArrayList 源码解析和设计思路	2.1 LinkedBlockingQueue
06 LinkedList 源码解析	适合对生产的数据大小不定（时高时低），数据量较大的场景，比如说我们在淘宝上买东西，点击下单按钮时，对应着后台的系统叫做下单系统，下单系统会把下单请求都放到一个线程池里面，这时候我们初始化线程池时，一般会选择 LinkedBlockingQueue，并且设置一个合适的大小，此时选择 LinkedBlockingQueue 主要原因在于：在不高于我们设定的阈值内，队列里面的大小可大可小，不会有任何性能损耗，正好符合下单流量的特点，时大时小。
07 List 源码会问哪些面试题	一 般 工 作 中 ， 我 们 大 多 数 都 会 选 择 LinkedBlockingQueue 队 列 ， 但 会 设 置 LinkedBlockingQueue 的最大容量，如果初始化时直接使用默认的 Integer 的最大值，当流量很大，而消费者处理能力很差时，大量请求都会在队列中堆积，会大量消耗机器的内存，就会降低机器整体性能甚至引起宕机，一旦宕机，在队列中的数据都会消失，因为队列的数据是保存在内存中的，一旦机器宕机，内存中的数据都会消失的，所以使用 LinkedBlockingQueue 队列时，建议还是要根据日常的流量设置合适的队列的大小。
08 HashMap 源码解析	2.2 ArrayBlockingQueue
09 TreeMap 和 LinkedHashMap 核心源码解析	一般用于生产数据固定的场景，比如说系统每天会进行对账，对账完成之后，会固定的产生 100 条对账结果，因为对账结果固定，我们就可以使用 ArrayBlockingQueue 队列，大小可以设置成 100。
10 Map源码会问哪些面试题	2.3 DelayQueue
11 HashSet、TreeSet 源码解析	延迟队列，在工作中经常遇到，主要用于任务不想立马执行，想等待一段时间才执行的场景。
12 彰显细节：看集合源码对我们实际工作的帮助和应用	比如说延迟对账，我们在工作中曾经遇到过这样的场景：我们在淘宝上买东西，弹出支付宝付款页面，在我们输入指纹的瞬间，流程主要是前端 -> 交易后端 -> 支付后端，交易后端调用支付后端主要是为了把我们支付宝的钱划给商家，而交易调用支付的过程中，有小概率的情况，因为
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	

<p>目录</p>	
<p>第1章 基础</p>	
<p>01 开篇词：为什么学习本专栏</p>	
<p>02 String、Long 源码解析和面试题</p>	
<p>03 Java 常用关键字理解</p>	
<p>04 Arrays、Collections、Objects 常用方法源码解析</p>	
<p>第2章 集合</p>	<p>这是一个真实场景，为了方便描述，已经大大简化了，再说明几点：</p>
<p>05 ArrayList 源码解析和设计思路</p>	<p>1. 交易调用支付的接口，这个接口的作用就是为了把小美的 800 元转给商家小明；</p>
<p>06 LinkedList 源码解析</p>	<p>2. 接口调用超时，此时交易系统并不知道 800 元有没有成功转给小明，当然想知道的方式有很多，我们选择了对账的方式，对账的目的就是为了知道当前 800 元有没有成功转给小明；</p>
<p>07 List 源码会问哪些面试题</p>	<p>3. 延迟对账的目的，因为支付系统把 800 元转给商家小明也是需要时间的，如果超时之后立马对账，可能转账的动作还在进行中，导致对账的结果不准确，所以需要延迟几秒后再去对账；</p>
<p>08 HashMap 源码解析</p>	<p>4. 对账之后的结果有几种，比如已经成功的把 800 元转给小明了，这时候需要把对账结果告诉交易系统，交易系统更新数据，前端就能够显示转账成功了。</p>
<p>09 TreeMap 和 LinkedHashMap 核心源码解析</p>	
<p>10 Map源码会问哪些面试题</p>	<p>在这个案例中，延迟对账的核心技术就是 DelayQueue，我们大概这么做的：新建对账任务，设置 3 秒之后执行，把任务放到 DelayQueue 中，过了 3 秒之后，就会自动执行对账任务了。</p>
<p>11 HashSet、TreeSet 源码解析</p>	<p>DelayQueue 延迟执行的功能就在这个场景中得到应用。</p>
<p>12 彰显细节：看集合源码对我们实际工作的帮助和应用</p>	<p>3 总结</p>
<p>13 差异对比：集合在 Java 7 和 8 有何不同和改进</p>	<p>我们不会为了阅读源码而读源码，我们读源码的最初目的，是为了提高我们的技术深度，最终目的是为了在不同的场景中，能够选择合适的技术进行落地，本章中解释的一些队列的场景，我们在工作中其实都会遇到，特别是在使用线程池时，使用哪种队列是我们必须思考的一个问题，所以本章先比较了各个队列的适合使用场景，然后举了几个案例进行具体分析，希望大家也能把技术具体落地到实际工作中，使技术推动、辅助业务。</p>
<p>14 简化工作：Guava Lists Maps 实际工作运用和源码</p>	
<p>第3章 并发集合类</p>	<div><div>←</div><div>24 举一反三：队列在 Java 其它源码中的应用</div><div>26 惊叹面试官：由浅入深手写队列</div><div>→</div></div>
<p>15 CopyOnWriteArrayList 源码解析和设计思路</p>	
<p>16 ConcurrentHashMap 源码解析和设计思路</p>	<p>精选留言 1</p>
<p>17 并发 List、Map源码面试题</p>	<p>欢迎在这里发表留言，作者筛选后可公开显示</p>
<p>18 场景集合：并发 List、Map的应用</p>	

← 慕课专栏

面试官系统精讲Java源码及大厂真题 / 25 整体设计：队列设计思想、工作中使用场景

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

Q、Kafka等？如果可以，实际开发中使用本章讲解的队列好还是消息队列好？

👍 2    回复

2019-11-07

文贺 回复 八卦一百二十八掌

肯定是中间件好，一般中间件能够做到高可用，高性能，高扩展，我们文中只是举一个例子来演示下，实际工作中直接使用中间件就好了，当然中间件的底层 API 也是使用的文中这些队列 API，可能在架构设计上会更加高大尚

回复    2019-11-17 11:09:35

千学不如一看，千看不如一练

www.imooc.com/read/47/article/867

5/5