基础

"[Java] hashcode 为啥必须重写",

# Java基础

## 面向对象

### 在一个静态方法内调用一个非静态成员为什么是非法的

### 重载和重写的区别

目的上:

重载:方法之间进行重载，方法的功能相同，但是具体要解决的参数不同。

重写:子类和父类之间。子类继承父类的方法，但是又不想用父类的方法，就可以进行重写。

结构上;

重载: 方法名必须相同，参数，返回值类型，可以不同。Overdloading

重写: 方法名、返回值类型，参数类型全部相同。 Overriding。

从动态角度讲

父类可以引用子类对象，在调用子类方法的时候，意味着这些参数也必须是相等。

### 为什么要重写hashcode方法

hashCode:通过hash算法得出对象的hash值。

从逻辑上讲：

两个对象不相等 <=> equals()不相等 => hashCode()不相等

反之hashcode相等 意味着equals也是相等的。

这是基于java hash 表的存储。

对于一个set集合，底层也是hashmap结构的，当新加一个对象时，需要判断现有集合中是否已经存在与此对象相等的对象，如果没有hashCode()方法，需要将Set进行一次遍历，并逐一用equals()方法判断两个对象是否相等，此种算法时间复杂度为o(n)。

在hashmap结构中，通过对象的hashcode码值与数组的length-1左位于运算，然后去添加到链表中，可以减少遍历的次数。

如果我们在业务上判断equal比较的对象是否相等，例如如果一个对象中的int属性值为偶数的我们就判断他相等，为奇数代表两个相等，在重写hashcode方法的时候，map集合存储的是两个对象，但在业务逻辑上我们判断的这连个对象是相等的，所以就有了冲突，所以java在的要求我们往hash结构存储时，重写euqals，要从写hashcdoe。

String类型是我们最常用作为key的元素，内部也重写了equals和hashcode方法，如果euqals的字符串不相等，那么hashcode方法返回的hash值也一定不相同。

### Java的四种引用，强弱软虚

JVM垃圾回收器根据对象回收的不同时机，将对象分为强软弱虚引用。

目的:当内存空间还足够时，则能保留在内存中；如果内存空间在进行垃圾收集后还是很紧张，则可以抛弃这些对象。

强引用:通过可达性分析算法，发现只要有引用存在，就不会被回收。即使出现oom。

软引用(SoftReference):jvm进行垃圾回收后，发现还没有足够的内存，就会回收这些对象。

弱引用（WeakReference）：只要发生垃圾回收，就会回收的对象。

虚引用（PhantomReference）：最弱的引用，无法获取到引用的对象，随时可能被垃圾回收 。目的是用于追中垃圾回收过程，创建虚引用当需要传递一个队列，当被垃圾回收时，发现当前引用是弱引用，将这个引用扔到队列中，可以从队列中获取该引用，知道他被回收了。

软引用和弱引用的适用场景:

弱引用:

1、ThreadLocalMap防止内存泄漏。

2. 监控对象是否将要被回收。

## 泛型

泛型

泛型目的:

1、集合存储元素，没有指定类型，会返回Object，需要进行强转，强转失败造成ClassCastException。

2、一般的类和方法，只能使用具体的类型：要么是基本类型，要么是自定义的类。如果要编写可以应用于多种类型的代码，这种刻板的限制对代码的束缚就会很大。

3、可以使用接口方式，但是程序的约束也还是太强,使用的时候必须要让类实现接口。

泛型的擦除:

[Kafka] 怎么保证消息可靠性 ★★★★★",

"[Kafka] 如何保证消息不丢失", // YY

"\* [Kafka] Kafka 如何重置消息位移点? ★★", // 爱回收

"[RocketMQ] 如何保证消息不丢失 ★★★★★",

"[RocketMQ] 消息堆积如何处理 ★★★★", // bigo

"[RocketMQ] 事务消息, 使用场景 ★★★★", // 得物

"[RocketMQ] 顺序消息 ★★", // 得物

"[RocketMQ] 延时消息 ★★", // 得物

"\* [MQ] RocketMQ和Kafka如何选型 ★★★★★", // 得物, 趣头条

"\* [Java] ThreadLocal 原理 ★★", // 得物, 快手

"[Java] \* Synchronized 原理 ★★★★★", // 得物

"[Java] \* Synchronized 锁升级, monitorenter、monitorexit 作用时机, 哪种级别的锁会使用这两个指令 ★★★★", // 趣头条

"[Java] ReentrantLock 原理, AQS 原理 ★★★",

"[Java] CAS 原理和优缺点", // 得物

"[Java] volatile 原理", // 得物

"[Java] JMM 是什么? 如何保证可见性? hapen-before 原则", // 趣头条, 腾讯, 腾讯, 得物

"[Java] HashMap 原理, put/get/resize ★★★", // 得物

"\* [Java] JUC: ConcurrentHashMap ★★★★", // 得物, bigo

"[Java] JUC: CopyOnWriteArrayList ★★", // 得物

"[Java] hashcode 为啥必须重写",

"[Java] JUC: 线程池原理",

"[Java] JVM 内存布局",

"[Java] JVM: new Object 过程",

"[Java] \* JVM: new Object 内存布局 ★★★★", // 趣头条

"[Java] 双亲委派模型", // 快手

"[Java] 怎么判断一个对象是垃圾对象 \*\*",

"[Java] 垃圾回收算法",

"[Java] G1和CMS相比, 有什么不同? 有什么优点?", // 趣头条

"[Java] 什么时候会触发YGC、FGC? 对象什么时候进入老年代",

"[Java] JVM 调优手段 ★★★★", // 阿里, bigo

"[Spring] Bean 生命周期 ★", // 得物

"[Spring] 循环引用 ★★★", // 得物

"[Spring] SpringBoot 启动流程 ★★★", // 得物

"[SpringCloud] Hystrix 原理 ★★★", // 趣头条

"\* [MySQL] 索引: 聚簇索引, 非聚簇索引, 覆盖索引, 联合索引, 索引下推 ★★", // 阿里, 得物

"\* [MySQL] 锁: 共享锁, 排他锁, 意向锁, 记录锁, 间隙锁, 临键锁 ★★",

"\* [MySQL] MVCC ★★★",

"\* [MySQL] 分库分表, shardingKey 怎么选? 分表后ID怎么办? 怎么处理非 shardingKey的查询 ★★★★", // 得物

"\* [MySQL] 分页为啥越来越慢(limit分页性能差), 怎么解决? ★★★", // 得物

"\* [MySQL] 索引失效, SQL优化 ★★★", // 得物

"\* [MySQL] 主从同步原理 ★★",

"[MySQL] 优化器",

"[Redis] ziplist 压缩列表",

"[Redis] Zset 跳跃表 ★★★", // 得物, 阿里

"[Redis] 分布式锁 ★★★", // 趣头条, bigo

"\* [Redis] 热key怎么解决", // 得物

"[Redis] 缓存击穿, 缓存穿透, 缓存雪崩 ★★",

"[ES] 倒排索引 ★★",

"[ES] 写入过程 ★★",

"[ES] 联合索引 ★★★",

"[算法] 一致性哈希",

"[算法] LRU 算法",

"[算法] 链表相加", // bigo

"[数据结构] 红黑树 ★★★★", // 得物

"\* [分布式] 分布式事务 XA, 2PC, 3PC, TCC ★★★★",

"\* [分布

"[Kafka] 怎么保证消息可靠性 ★★★★★",

"[Kafka] 如何保证消息不丢失", // YY

"\* [Kafka] Kafka 如何重置消息位移点? ★★", // 爱回收

"[RocketMQ] 如何保证消息不丢失 ★★★★★",

"[RocketMQ] 消息堆积如何处理 ★★★★", // bigo

"[RocketMQ] 事务消息, 使用场景 ★★★★", // 得物

"[RocketMQ] 顺序消息 ★★", // 得物

"[RocketMQ] 延时消息 ★★", // 得物

"\* [MQ] RocketMQ和Kafka如何选型 ★★★★★", // 得物, 趣头条

"\* [Java] ThreadLocal 原理 ★★", // 得物, 快手

"[Java] \* Synchronized 原理 ★★★★★", // 得物

"[Java] \* Synchronized 锁升级, monitorenter、monitorexit 作用时机, 哪种级别的锁会使用这两个指令 ★★★★", // 趣头条

"[Java] ReentrantLock 原理, AQS 原理 ★★★",

"[Java] CAS 原理和优缺点", // 得物

"[Java] volatile 原理", // 得物

"[Java] JMM 是什么? 如何保证可见性? hapen-before 原则", // 趣头条, 腾讯, 腾讯, 得物

"[Java] HashMap 原理, put/get/resize ★★★", // 得物

"\* [Java] JUC: ConcurrentHashMap ★★★★", // 得物, bigo

"[Java] JUC: CopyOnWriteArrayList ★★", // 得物

"[Java] JUC: 线程池原理",

"[Java] JVM 内存布局",

"[Java] JVM: new Object 过程",

"[Java] \* JVM: new Object 内存布局 ★★★★", // 趣头条

"[Java] 双亲委派模型", // 快手

"[Java] 怎么判断一个对象是垃圾对象 \*\*",

"[Java] 垃圾回收算法",

"[Java] G1和CMS相比, 有什么不同? 有什么优点?", // 趣头条

"[Java] 什么时候会触发YGC、FGC? 对象什么时候进入老年代",

"[Java] JVM 调优手段 ★★★★", // 阿里, bigo

"[Spring] Bean 生命周期 ★", // 得物

"[Spring] 循环引用 ★★★", // 得物

"[Spring] SpringBoot 启动流程 ★★★", // 得物

"[SpringCloud] Hystrix 原理 ★★★", // 趣头条

"\* [MySQL] 索引: 聚簇索引, 非聚簇索引, 覆盖索引, 联合索引, 索引下推 ★★", // 阿里, 得物

"\* [MySQL] 锁: 共享锁, 排他锁, 意向锁, 记录锁, 间隙锁, 临键锁 ★★",

"\* [MySQL] MVCC ★★★",

"\* [MySQL] 分库分表, shardingKey 怎么选? 分表后ID怎么办? 怎么处理非 shardingKey的查询 ★★★★", // 得物

"\* [MySQL] 分页为啥越来越慢(limit分页性能差), 怎么解决? ★★★", // 得物

"\* [MySQL] 索引失效, SQL优化 ★★★", // 得物

"\* [MySQL] 主从同步原理 ★★",

"[MySQL] 优化器",

"[Redis] ziplist 压缩列表",

"[Redis] Zset 跳跃表 ★★★", // 得物, 阿里

"[Redis] 分布式锁 ★★★", // 趣头条, bigo

"\* [Redis] 热key怎么解决", // 得物

"[Redis] 缓存击穿, 缓存穿透, 缓存雪崩 ★★",

"[ES] 倒排索引 ★★",

"[ES] 写入过程 ★★",

"[ES] 联合索引 ★★★",

"[算法] 一致性哈希",

"[算法] LRU 算法",

"[算法] 链表相加", // bigo

"[数据结构] 红黑树 ★★★★", // 得物

"\* [分布式] 分布式事务 XA, 2PC, 3PC, TCC ★★★★",

"\* [分布式] 分布式锁 ★★★",

"[分布式] Raft 协议 ★★★★★", // 得物

"[系统设计] 缓存和数据库一致性 ★★★", // 得物

"[系统设计] 怎么提高并发",

"[系统设计] 秒杀方案", // 阿里, 得物

"[系统设计] 设计一个IM系统包括群聊单聊 ★★★★★", // 阿里, 得物

"[系统设计] 链路跟踪",

"[生产问题] 线上问题排查你们使用什么工具? 排查的过程是什么? ★★★", // 趣头条

"[生产问题] CPU飙升很高, 排查工具 Jstack|Arthas|Bistoury ★★★" // 得物, 饿了么

可以画一下你们系统的架构图吗？