Java面试题1

# Java基础

1. HashMap的源码，实现原理，JDK8中对HashMap做了怎样的优化。

Jdk1.8中HashMap底层是由数组+链表+红黑树实现的，当map中的元素还不多的时候，出现hash冲突时，元素以链表的形式存储，当链表的大小超过8时，就把链表转化成红黑树。计算hash&(n-1)得到在链表数组中的位置。

https://blog.csdn.net/weixin\_37356262/article/details/80543218

1. HaspMap扩容是怎样扩容的，为什么都是2的N次幂的大小。

当size>threshold时进行扩容，hash&(n-1)

1. HashMap，HashTable，ConcurrentHashMap的区别。

底层实现，初始容量，扩容，hash

<https://www.cnblogs.com/heyonggang/p/9112731.html>

1. 极高并发下HashTable和ConcurrentHashMap哪个性能更好，为什么，如何实现的。

ConcurrentHashMap性能更好，其底层采用分段数组+链表来实现，把map分成N个Segment，使用锁分段技术，允许多个线程同时对hash表进行操作

1. HashMap在高并发下如果没有处理线程安全会有怎样的安全隐患，具体表现是什么。

当两个线程都会触发rehash时，第一个线程执行到一半（一个元素链接到下一个还未执行时），另一个线程执行完了整个操作，第一个线程再执行，就会出现循环链表

1.get无限循环，具体表现为CPU使用率100%

2.多线程put时可能导致元素丢失

<https://blog.csdn.net/weixin_37356262/article/details/80554782>

1. Java中四种修饰符的限制范围。

<https://blog.csdn.net/itachiyang/article/details/43647909>

1. Object类中的方法。

1，构造函数

2，hashCode和equals函数用来判断对象是否相同,

3，wait(),wait(long),wait(long,int),notify(),notifyAll()

4，toString()和getClass,

5，clone() 用来另存一个当前存在的对象

6，finalize()用于在垃圾回收

1. 接口和抽象类的区别，注意JDK8的接口可以有实现。

同：都定义方法和属性，都不能创建对象，1.8之前都不能有实现的方法

1. 动态代理的两种方式，以及区别。

一种是JDK反射机制提供的代理，另一种是CGLIB代理。在JDK代理，必须提供接口；而CGLIB则不需要提供接口，cglib可以对任意类生成代理对象，它的原理是对目标对象进行继承代理，如果目标对象被final修饰，那么该类无法被cglib代理。

<https://blog.csdn.net/weixin_37356262/article/details/80555361>

1. Java序列化的方式。

序列化就是把Java对象储存在某一地方（硬盘、网络），也就是将对象的内容进行流化。

1.Java原生以流的方法进行的序列化

2.Json序列化ObjectMapper.readValue

3.FastJson序列化JSON.toJSONString JSON.parseObject

4.Protobuff序列化

<https://blog.csdn.net/weixin_37356262/article/details/80555452>

1. 传值和传引用的区别，Java是怎么样的，有没有传值引用。

Java的参数传递都是值传递

<http://www.cnblogs.com/binyue/p/3862276.html>

1. 一个ArrayList在循环过程中删除，会不会出问题，为什么。

<https://www.cnblogs.com/androidsuperman/p/9012320.html>

1. @transactional注解在什么情况下会失效，为什么。

1.首先检查配置，在项目的spring配置文件检查是否配置开启事务

2.同一个类中, 一个未标注@Transactional的方法去调用标有@Transactional的方法, 事务会失效

3. 在非public方法上标注@Transactional, 事务无效

<https://blog.csdn.net/u012470138/article/details/73614043>

# 数据结构和算法

1. B+树

非叶子节点不存储数据

1. 快速排序，堆排序，插入排序（其实八大排序算法都应该了解
2. 一致性Hash算法，一致性Hash算法的应用

# JVM

<https://www.cnblogs.com/ityouknow/p/5610232.html>

<https://blog.csdn.net/anjoyandroid/article/details/78609971>

1. JVM的内存结构。

数据：

堆：

方法区：类信息，常量，静态变量，JIT

指令：

程序计数器：指向当前线程正在执行的字节码指令地址

虚拟机栈：用于存储当前线程运行时所需要的数据，指令，返回地址

本地方法栈：用于支持native方法的执行，存储了每个native方法的执行状态

1. JVM方法栈的工作过程，方法栈和本地方法栈有什么区别。

虚拟机栈执行Java方法，本地方法栈执行native方法

1. JVM的栈中引用如何和堆中的对象产生关联。
2. 可以了解一下逃逸分析技术。

逃逸分析，是一种可以有效减少Java 程序中同步负载和内存堆分配压力的跨函数全局数据流分析算法。通过逃逸分析，Java Hotspot编译器能够分析出一个新的对象的引用的使用范围从而决定是否要将这个对象分配到堆上。

1. GC的常见算法，CMS以及G1的垃圾回收过程，CMS的各个阶段哪两个是Stop the world的，CMS会不会产生碎片，G1的优势。

复制、标记-清除和标记-压缩

**初始标记**->并发标记->并发预清理->**重新标记-**>并发清理->并发重置

会产生碎片，G1中采用了比CMS更快的初始快照算法:snapshot-at-the-beginning (SATB)， G1能整理碎片

1. eden survivor区的比例，为什么是这个比例，eden survivor的工作过程。

Eden:survivor0:survivor1=8:1:1

因为年轻代中的对象基本都是朝生夕死的(80%以上)

new出来的对象会存放在eden区，存满后会触发minor GC，若eden区空间仍不够，则将eden区活跃的对象存储在survivor区

<https://blog.csdn.net/z69183787/article/details/52758832>

1. JVM如何判断一个对象是否该被GC，可以视为root的都有哪几种类型。

如果我们的程序无法+再引用到该对象，那么这个对象就肯定可以被回收，这个状态称为不可达。

三类对象在jvm中作为GC roots：虚拟机栈(JVM stack)中的栈帧，方法区中类静态属性引用的对象，本地方法栈(Native Stack)引用的对象。

1. 强软弱虚引用的区别以及GC对他们执行怎样的操作。

1.强引用(Strong) 就是我们平时使用的方式 A a = new A();强引用的对象是不会被回收的

2.软引用(Soft) 在jvm要内存溢出(OOM)时，会回收软引用的对象，释放更多内存

3.弱引用(Weak) 在下次GC时，弱引用的对象是一定会被回收的

4.虚引用(Phantom) 对对象的存在时间没有任何影响，也无法引用对象实力，唯一的作用就是在该对象被回收时收到一个系统通知

1. Java是否可以GC直接内存。
2. Java类加载的过程。

加载，验证，准备，解析，初始化

1. 双亲委派模型的过程以及优势。

某个特定的类加载器在接到加载类的请求时，首先将加载任务委托给父类加载器，依次递归，如果父类加载器可以完成类加载任务，就成功返回；只有父类加载器无法完成此加载任务时，才自己去加载。

好处在于**Java类随着它的类加载器一起具备了一种带有优先级的层次关系**

<https://blog.csdn.net/zy632204200/article/details/80852963>

1. 常用的JVM调优参数。

-Xms设置堆的最小空间大小。

-Xmx设置堆的最大空间大小。

-XX:NewSize设置新生代最小空间大小。

-XX:MaxNewSize设置新生代最大空间大小。

-XX:PermSize设置永久代最小空间大小。

-XX:MaxPermSize设置永久代最大空间大小。

-Xss设置每个线程的堆栈大小。

1. dump文件的分析。

使用jdk中bin目录下的jvisualvm.exe分析dump文件

1. Java有没有主动触发GC的方式（没有）。

# 多线程

1. Java实现多线程有哪几种方式。

继承Thread，实现Runnable，实现Callable，使用线程池

1. Callable和Future的了解。

线程池submit一个实现callable接口的类的实例时会返回一个Future结果。

1. 线程池的参数有哪些，在线程池创建一个线程的过程。

corePollSize：核心线程数

maximumPoolSize：最大线程数

keepAliveTime：空闲的线程保留的时间

TimeUnit：空闲线程的保留时间单位

BlockingQueue<Runnable>：阻塞队列，存储等待执行的任务。参数有ArrayBlockingQueue、LinkedBlockingQueue、SynchronousQueue可选

ThreadFactory：线程工厂，用来创建线程

RejectedExecutionHandler：队列已满，而且任务量大于最大线程的异常处理策略。AbortPolicy，DiscardPolicy，DiscardOldestPolicy，CallerRunsPolicy

<https://blog.csdn.net/hsf15768615284/article/details/78261173>

任务进来时，首先执行判断，判断核心线程是否处于空闲状态，如果不是，核心线程就先就执行任务，如果核心线程已满，则判断任务队列是否有地方存放该任务，若果有，就将任务保存在任务队列中，等待执行，如果满了，在判断最大可容纳的线程数，如果没有超出这个数量，就开创非核心线程执行任务，如果超出了，就调用handler实现拒绝策略。

<https://blog.csdn.net/weixin_40271838/article/details/79998327>

1. volitile关键字的作用，原理。

内存可见性（Memory Visibility）：所有线程都能看到共享内存的最新状态

原理：特殊规则read、load、use动作必须**连续出现，**assign、store、write动作必须**连续出现**

防止指令重排

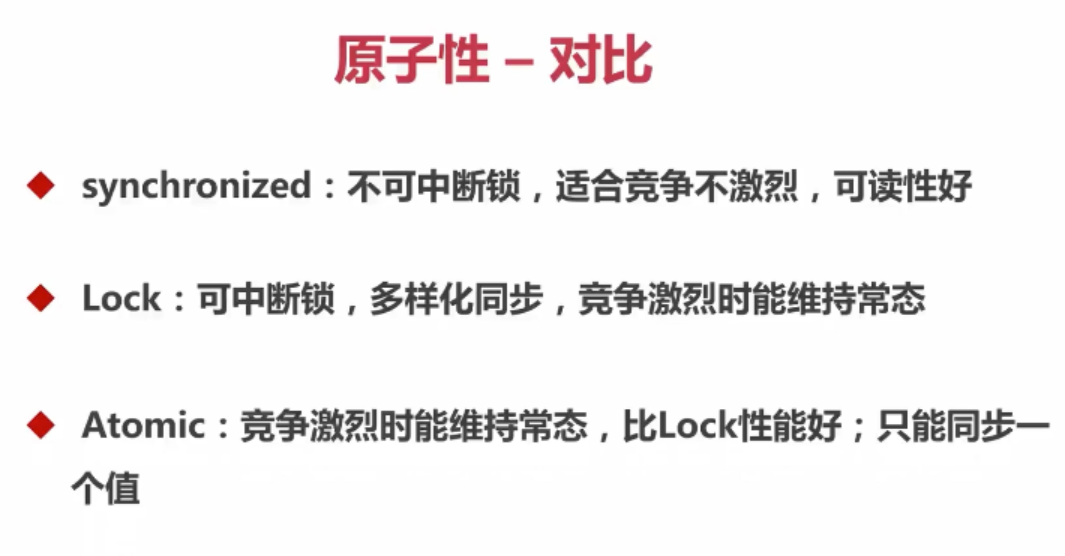
原理：在指令序列中插入内存屏障防止指令重排

<https://www.cnblogs.com/monkeysayhi/p/7654460.html>

1. synchronized关键字的用法，优缺点。

普通方法，静态方法中加synchronized

Synchronized块，（）中传入对象和类



1. Lock接口有哪些实现类，使用场景是什么。

ReentrantLock,

WriteLock(ReadLock) in ReentrantReadWriteLock,

Segment in ConcurrentHashMap

1. 可重入锁的用处及实现原理，写时复制的过程，读写锁，分段锁（ConcurrentHashMap中的segment）。

主要利用CAS+CLH队列来实现: 先通过CAS尝试获取锁。如果此时已经有线程占据了锁，那就加入CLH队列并且被挂起。当锁被释放之后，排在CLH队列队首的线程会被唤醒，然后CAS再次尝试获取锁。

1. 悲观锁，乐观锁，优缺点，CAS有什么缺陷，该如何解决。

乐观锁：每次去拿数据的时候都认为别人不会修改，所以不会上锁，但是在更新的时候会判断一下在此期间别人有没有去更新这个数据，可以使用版本号等机制。

悲观锁：每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会block直到它拿到锁。

<https://blog.csdn.net/hongchangfirst/article/details/26004335>

CPU开销较大，不能保证代码块的原子性，ABA问题

Java中提供了AtomicStampedReference和AtomicMarkableReference来解决ABA问题

<https://blog.csdn.net/qq_41706675/article/details/82794915>

1. ABC三个线程如何保证顺序执行。

Thread.join 线程池Executors.newSingleThreadExecutor()

<https://blog.csdn.net/eene894777/article/details/74942485/>

1. 线程的状态都有哪些。

初始，运行，阻塞，等待，定时等待，结束

1. sleep和wait的区别。

Thread.sleep，不释放锁

Object.wait，释放锁

1. notify和notifyall的区别。

区别在于notify（）只会唤醒一个线程，而notifyAll方法将唤醒所有线程

1. ThreadLocal的了解，实现原理。

ThreadLocal为解决多线程程序的并发问题提供了一种新的思路，线程封闭，内部有个ThreadLocalMap类是以当前线程为key来存储数据的。

# 数据库相关

1. 常见的数据库优化手段
2. 选取最适用的字段属性：将表中字段的宽度设得尽可能小，把字段设置为NOTNULL
3. 使用连接（JOIN）来代替子查询(Sub-Queries)
4. 使用联合(UNION)来代替手动创建的临时表
5. 事务：BEGIN; COMMIT;
6. 锁定表：LOCK TABLE table\_name WRITE ……; UNLOCKTABLES;
7. 使用外键：锁定表的方法可以维护数据的完整性，但是它却不能保证数据的关联性。
8. 使用索引：建立在那些将用于JOIN,WHERE判断和ORDERBY排序的字段上。
9. 优化的查询语句

<https://www.cnblogs.com/zhyunfe/p/6209074.html>

1. 索引的优缺点，什么字段上建立索引

缺点：创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加

很少使用或者参考的列；很少数据值的列；定义为text, image和bit数据类型的列；修改性能远远大于检索性能时，不应该创建索引

用于JOIN,WHERE判断和ORDERBY排序的字段

1. 索引失效
2. 条件中用or，即使其中有条件带索引，也不会使用索引查询（这就是查询尽量不要用or的原因，用in吧）
3. 对于多列索引，不是使用的第一部分，则不会使用索引。
4. like的模糊查询以%开头，索引失效
5. 如果列类型是字符串，那一定要在条件中将数据使用引号引用起来，否则不会使用索引
6. 如果MySQL预计使用全表扫描要比使用索引快，则不使用索引

<https://blog.csdn.net/sdujava2011/article/details/78310426?locationNum=8&fps=1>

1. 数据库连接池。

数据库连接池的基本原理是在内部对象池中维护一定数量的数据库连接，并对外暴露数据库连接获取和返回方法。

<https://www.cnblogs.com/wym789/p/6374440.html>

1. durid的常用配置。

<https://blog.csdn.net/dengtaowei/article/details/45698367>

# 设计模式

1. 观察者模式

发布订阅模式，发布者发布信息，订阅者获取信息，订阅了就能收到信息，没订阅就收不到信息。

<https://www.cnblogs.com/luohanguo/p/7825656.html>

1. 代理模式

代理模式给某一个对象提供一个代理对象，并由代理对象控制对原对象的引用。

<https://www.cnblogs.com/daniels/p/8242592.html>

1. 单例模式，有五种写法，可以参考文章[单例模式的五种实现方式](https://segmentfault.com/a/1190000010755849)

饿汉模式(线程安全)

懒汉模式(非线程安全)

懒汉模式方法上加synchronized(线程安全，性能低)

懒汉模式使用synchronized块双重同步锁(非线程安全)

懒汉模式使用synchronized块双重同步锁加volatile (线程安全)

1. 可以考Spring中使用了哪些设计模式

简单工厂，工厂方法，单例模式，适配器，包装器，代理，观察者，策略，模板方法

<https://www.cnblogs.com/yuefan/p/3763898.html>

# 分布式相关

1. 分布式事务的控制。

基于XA协议的两阶段提交

消息事务+最终一致性

TCC编程模式

[**http://www.cnblogs.com/betterboyz/p/8556875.html**](http://www.cnblogs.com/betterboyz/p/8556875.html)

1. 分布式锁如何设计。

<https://blog.csdn.net/WuLex/article/details/78791498>

1. 分布式session如何设计。

放在Redis缓存中

1. dubbo的组件有哪些，各有什么作用。

四大基本组件：Registry、Monitor、Provider、Consumer

1. zookeeper的负载均衡算法有哪些。
2. dubbo是如何利用接口就可以通信的。

# 缓存相关

1. redis和memcached的区别。

Redis支持的类型除了K/V形式还有String，set，list；memcached只支持K/V

Redis单线程但存小数据时性能不比多线程的memcached低

Redis并不是所有的数据都一直存储在内存中；memcached数据都是一直存在内存中

Redis支持数据持久化

1. redis支持哪些数据结构。

String,hash,list,set,sorted set

1. redis是单线程的么，所有的工作都是单线程么。

redis是单线程的

1. redis如何存储一个String的。

SET key value

1. redis的部署方式，主从，集群。

<https://blog.csdn.net/RobertoHuang/article/details/70766809>

1. redis的哨兵模式，一个key值如何在redis集群中找到存储在哪里。
2. redis持久化策略。

提供两种主要的持久化策略：

将当前的数据快照存放成一个数据文件的持久化机制，即RDB快照；

AOF持久化的实现可以分为命令追加（append），文件写入，文件同步（sync）。

<https://www.cnblogs.com/hanfei-1005/p/5692455.html>

# 框架相关

1. SpringMVC的Controller是如何将参数和前端传来的数据一一对应的。
2. Mybatis如何找到指定的Mapper的，如何完成查询的。

mybatis通过JDK的动态代理方式，在启动加载配置文件时，根据配置mapper的xml去生成。读取核心配置文件<mappers>，根据namespace找到对应的class

1. Quartz是如何完成定时任务的。
2. 自定义注解的实现。
3. Spring使用了哪些设计模式。

简单工厂，工厂方法，单例模式，适配器，包装器，代理，观察者，策略，模板方法

<https://www.cnblogs.com/yuefan/p/3763898.html>

1. Spring的IOC有什么优势。

因为把对象生成放在了XML里定义，所以当我们需要换一个实现子类将会变成很简单（一般这样的对象都是现实于某种接口的），只要修改XML就可以了，这样我们甚至可以实现对象的热插拨（有点象USB接口和SCIS硬盘了）

1. Spring如何维护它拥有的bean。
2. 服务启动解析配置文件，并保存配置文件中的元素；
3. 实例化所有元素，并提供获取实例方法。

<https://blog.csdn.net/magicianjun/article/details/78716190>