JVM

**JVM内存结构**

堆、栈、方法区、直接内存、堆和栈区别

**Java内存模型**

内存可见性、重排序、顺序一致性、volatile、锁、final

**垃圾回收**

内存分配策略、垃圾收集器（G1）、GC算法、GC参数、对象存活的判定

内存分配策略：

参考链接：<https://www.cnblogs.com/wcd144140/p/5649553.html>

<https://www.cnblogs.com/ygj0930/p/6535893.html>

对象优先分配在线程的本地分配缓冲区

对象优先在Eden分配

大对象直接进入老年代

长期存活的对象将进入老年代

动态对象年龄判定

空间分配担保

GC算法：

参考链接：<https://blog.csdn.net/windcake/article/details/54810052>

<https://www.cnblogs.com/fangfuhai/p/7203468.html?utm_source=itdadao&utm_medium=referral>(推荐)

引用计数法 Reference Counting：

给对象添加一个引用计数器，每过一个引用计数器值就+1，少一个引用就-1。当它的引用变为0时，该对象就不能再被使用。它的实现简单，但是不能解决互相循环引用的问题。

根搜索算法 GC Roots Tracing

以一系列叫“GC Roots”的对象为起点开始向下搜索，走过的路径称为引用链（Reference Chain），当一个对象没有和任何引用链相连时，证明此对象是不可用的，用图论的说法是不可达的。那么它就会被判定为是可回收的对象。

JAVA里可作为GC Roots的对象   
虚拟机栈（栈帧中的本地变量表）中引用的对象   
方法区中的类静态属性引用的对象   
方法区中的常量引用的对象   
本地方法栈中JNI（即Native方法）的引用的对象

标记-清除算法 Mark-Sweep

这是一个非常基本的GC算法，它是现代GC算法的思想基础，分为标记和清除两个阶段：先把所有活动的对象标记出来，然后把没有被标记的对象统一清除掉。但是它有两个问题，一是效率问题，两个过程的效率都不高。二是空间问题，清除之后会产生大量不连续的内存。

复制算法 Copying

复制算法是将原有的内存空间分成两块，每次只使用其中的一块。在GC时，将正在使用的内存块中的存活对象复制到未使用的那一块中，然后清除正在使用的内存块中的所有对象，并交换两块内存的角色，完成一次垃圾回收。它比标记-清除算法要高效，但不适用于存活对象较多的内存，因为复制的时候会有较多的时间消耗

它的致命缺点是会有一半的内存浪费。

标记整理算法 Mark-Compact

标记整理算法适用于存活对象较多的场合，它的标记阶段和标记-清除算法中的一样。整理阶段是将所有存活的对象压缩到内存的一端，之后清理边界外所有的空间。它的效率也不高。

**JVM参数及调优**

**Java对象模型**

oop-klass、对象头

**HotSpot**

即时编译器、编译优化

**类加载机制**

classLoader、类加载过程、双亲委派（破坏双亲委派）、模块化（jboss modules、osgi、jigsaw）

类加载过程：

<https://www.cnblogs.com/fanjie/p/6916784.html>

学完类加载之后，java运行过程就可以分为 编译 》 类加载 》 执行

类加载主要是由jvm虚拟机负责的，过程非常复杂，类加载分三步 加载 》 连接 》初始化，（这里的加载和本文标题的类加载是不同的，标题的类加载包含了完整的三个步骤）下面详细说说每一步的过程

1、加载：这个很简单，程序运行之前jvm会把编译完成的.class二进制文件加载到内存，供程序使用，用到的就是类加载器classLoader ，这里也可以看出java程序的运行并不是直接依 靠底层的操作系统，而是基于jvm虚拟机。如果没有类加载器，java文件就只是磁盘中的一个普通文件。

2、连接：连接是很重要的一步，过程比较复杂，分为三步 验证 》准备 》解析

　　验证：确保类加载的正确性。一般情况由javac编译的class文件是不会有问题的，但是可能有人的class文件是自己通过其他方式编译出来的，这就很有可能不符合jvm的编 译规则，这一步就是要过滤掉这部分不合法文件

准备：为类的静态变量分配内存，将其初始化为默认值 。我们都知道静态变量是可以不用我们手动赋值的，它自然会有一个初始值 比如int 类型的初始值就是0 ；boolean类型初始值为false，引用类型的初始值为null 。 这里注意，只是为静态变量分配内存，此时是没有对象实例的

解析：把类中的符号引用转化为直接引用。解释一下符号引用和直接引用。比如在方法A中使用方法B，A（）{B（）；}，这里的B（）就是符号引用，初学java时我们都是知道这是java的引用，以为B指向B方法的内存地址，但是这是不完整的，这里的B只是一个符号引用，它对于方法的调用没有太多的实际意义，可以这么认为，他就是给程序员看的一个标志，让程序员知道，这个方法可以这么调用，但是B方法实际调用时是通过一个指针指向B方法的内存地址，这个指针才是真正负责方法调用，他就是直接引用。

3、初始化：为类的静态变量赋予正确的初始值，上述的准备阶段为静态变量赋予的是虚拟机默认的初始值，此处赋予的才是程序编写者为变量分配的真正的初始值

**虚拟机性能监控与故障处理工具**

jps, jstack, jmap、jstat, jconsole, jinfo, jhat, javap, btrace、TProfiler

编译与反编译

javac 、javap 、jad 、CRF

Java基础知识

**阅读源代码**

String、Integer、Long、Enum、BigDecimal、ThreadLocal、ClassLoader & URLClassLoader、ArrayList & LinkedList、 HashMap & LinkedHashMap & TreeMap & CouncurrentHashMap、HashSet & LinkedHashSet & TreeSet

**Java中各种变量类型**

**熟悉Java String的使用，熟悉String的各种函数**

JDK 6和JDK 7中substring的原理及区别、

replaceFirst、replaceAll、replace区别、

String对“+”的重载、

String.valueOf和Integer.toString的区别、

字符串的不可变性

**自动拆装箱**

Integer的缓存机制

**熟悉Java中各种关键字**

transient、instanceof、volatile、synchronized、final、static、const 原理及用法。

**集合类**

常用集合类的使用

ArrayList和LinkedList和Vector的区别

SynchronizedList和Vector的区别

HashMap、HashTable、ConcurrentHashMap区别

Java 8中stream相关用法

apache集合处理工具类的使用

不同版本的JDK中HashMap的实现的区别以及原因

**枚举**

枚举的用法、枚举与单例、Enum类

**Java IO&Java NIO，并学会使用**

bio、nio和aio的区别、三种IO的用法与原理、netty

**Java反射与javassist**

反射与工厂模式、 java.lang.reflect.\*

**Java序列化**

什么是序列化与反序列化、为什么序列化

序列化底层原理

序列化与单例模式

参考链接

<https://blog.csdn.net/xlgen157387/article/details/79840134>

序列化算法一般会按步骤做如下事情：

（1）将对象实例相关的类元数据输出。

（2）递归地输出类的超类描述直到不再有超类。

（3）类元数据完了以后，开始从最顶层的超类开始输出对象实例的实际数据值。

（4）从上至下递归输出实例的数据

protobuf

为什么说序列化并不安全

**注解**

元注解、自定义注解、Java中常用注解使用、注解与反射的结合

**JMS**

什么是Java消息服务、JMS消息传送模型

**JMX**

java.lang.management.\*、 javax.management.\*

**泛型**

泛型与继承

类型擦除

泛型中K T V E

object等的含义、泛型各种用法

**单元测试**

junit、mock、mockito、内存数据库（h2）

**正则表达式**

java.lang.util.regex.\*

**常用的Java工具库**

commons.lang, commons.\*... guava-libraries netty

**什么是API&SPI**

**异常**

异常类型、正确处理异常、自定义异常

**时间处理**

时区、时令、Java中时间API

**编码方式**

解决乱码问题、常用编码方式

**语法糖**

Java中语法糖原理、解语法糖

Java并发编程

**什么是线程，与进程的区别**

**阅读源代码，并学会使用**

Thread、Runnable、Callable、ReentrantLock、ReentrantReadWriteLock、Atomic\*、Semaphore、CountDownLatch、、ConcurrentHashMap、Executors

**线程池**

自己设计线程池、submit() 和 execute()

<https://www.cnblogs.com/wxwall/p/7050698.html>

**线程安全**

死锁、死锁如何排查、Java线程调度、线程安全和内存模型的关系

**锁**

CAS、乐观锁与悲观锁、数据库相关锁机制、分布式锁、偏向锁、轻量级锁、重量级锁、monitor、锁优化、锁消除、锁粗化、自旋锁、可重入锁、阻塞锁、死锁

**死锁**

**volatile**

happens-before、编译器指令重排和CPU指令重

三.happens-before规则

1.程序顺序规则：一个线程中的每个操作，happens-before于该线程中的任意后续操作

2.监视器锁规则：对一个锁的解锁，happens-before于随后对这个锁的加锁

　　3.volatile变量规则：对一个volatile域的写，happens-before于任意后续对这个volatile域的读

　　4.传递性：如果A happens-before B，且B happens-before C，那么A happens-before C

　　5.start规则：如果线程A执行操作ThreadB.start()（启动线程B），那么A线程的ThreadB.start()操作happens-before于线程B中的任意操作

　　6.join规则：如果线程A执行操作ThreadB.join()并成功返回，那么线程B中的任意操作happens-before于线程A从ThreadB.join()操作成功返回。

**synchronized**

synchronized是如何实现的？

synchronized和lock之间关系

不使用synchronized如何实现一个线程安全的单例

**sleep 和 wait**

对于sleep()方法，我们首先要知道该方法是属于Thread类中的。而wait()方法，则是属于Object类中的。

sleep()方法导致了程序暂停执行指定的时间，让出cpu该其他线程，但是他的监控状态依然保持者，当指定的时间到了又会自动恢复运行状态。

在调用sleep()方法的过程中，线程不会释放对象锁。

而当调用wait()方法的时候，线程会放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象调用notify()方法后本线程才进入对象锁定池准备

获取对象锁进入运行状态。

**wait 和 notify**

**notify 和 notifyAll**

**ThreadLocal**

**写一个死锁的程序**

**写代码来解决生产者消费者问题**

**守护线程**

守护线程和非守护线程的区别以及用法

二、 进阶篇Java底层知识

**字节码、class文件格式**

**CPU缓存，L1，L2，L3和伪共享**

**尾递归**

**位运算**

用位运算实现加、减、乘、除、取余

设计模式

**了解23种设计模式**

**会使用常用设计模式**

单例、策略、工厂、适配器、责任链。

**实现AOP**

**实现IOC**

**不用synchronized和lock，实现线程安全的单例模式**

**nio和reactor设计模式**

网络编程

**tcp、udp、http、https等常用协议**

三次握手与四次关闭、流量控制和拥塞控制、OSI七层模型、tcp粘包与拆包

**http/1.0 http/1.1 http/2之前的区别**

**Java RMI，Socket，HttpClient**

**cookie 与 session**

cookie被禁用，如何实现session

**用Java写一个简单的静态文件的HTTP服务器**

实现客户端缓存功能，支持返回304 实现可并发下载一个文件 使用线程池处理客户端请求 使用nio处理客户端请求 支持简单的rewrite规则 上述功能在实现的时候需要满足“开闭原则”

**了解nginx和apache服务器的特性并搭建一个对应的服务器**

**用Java实现FTP、SMTP协议**

**进程间通讯的方式**

**什么是CDN？如果实现？**

**什么是DNS？**

**反向代理**

框架知识

**Servlet线程安全问题**

**Servlet中的filter和listener**

**Hibernate的缓存机制**

**Hiberate的懒加载**

**Spring Bean的初始化**

**Spring的AOP原理**

**自己实现Spring的IOC**

**Spring MVC**

**Spring Boot2.0**

Spring Boot的starter原理，自己实现一个starter

**Spring Security**

应用服务器

**JBoss**

**tomcat**

**jetty**

**Weblogic**

工具

**git & svn**

**maven & gradle**

三、 高级篇新技术

**Java 8**

lambda表达式、Stream API、

**Java 9**

Jigsaw、Jshell、Reactive Streams

**Java 10**

局部变量类型推断、G1的并行Full GC、ThreadLocal握手机制

**Spring 5**

响应式编程

**Spring Boot 2.0**

性能优化

使用单例、使用Future模式、使用线程池、选择就绪、减少上下文切换、减少锁粒度、数据压缩、结果缓存

线上问题分析

**dump获取**

线程Dump、内存Dump、gc情况

**dump分析**

分析死锁、分析内存泄露

**自己编写各种outofmemory，stackoverflow程序**

HeapOutOfMemory、 Young OutOfMemory、MethodArea OutOfMemory、ConstantPool OutOfMemory、DirectMemory OutOfMemory、Stack OutOfMemory Stack OverFlow

**常见问题解决思路**

内存溢出、线程死锁、类加载冲突

**使用工具尝试解决以下问题，并写下总结**

当一个Java程序响应很慢时如何查找问题、

当一个Java程序频繁FullGC时如何解决问题、

如何查看垃圾回收日志、

当一个Java应用发生OutOfMemory时该如何解决、

如何判断是否出现死锁、

如何判断是否存在内存泄露

编译原理知识

**编译与反编译**

**Java代码的编译与反编译**

**Java的反编译工具**

**词法分析，语法分析（LL算法，递归下降算法，LR算法），语义分析，运行时环境，中间代码，代码生成，代码优化**

操作系统知识

**Linux的常用命令**

**进程同步**

**缓冲区溢出**

**分段和分页**

**虚拟内存与主存**

数据库知识

**MySql 执行引擎**

**MySQL 执行计划**

如何查看执行计划，如何根据执行计划进行SQL优化

**SQL优化**

**事务**

事务的隔离级别、事务能不能实现锁的功能

**数据库锁**

行锁、表锁、使用数据库锁实现乐观锁、

**数据库主备搭建**

**binlog**

**内存数据库**

h2

**常用的nosql数据库**

redis、memcached

**分别使用数据库锁、NoSql实现分布式锁**

**性能调优**

数据结构与算法知识

**简单的数据结构**

栈、队列、链表、数组、哈希表、

**树**

二叉树、字典树、平衡树、排序树、B树、B+树、R树、多路树、红黑树

**排序算法**

各种排序算法和时间复杂度 深度优先和广度优先搜索 全排列、贪心算法、KMP算法、hash算法、海量数据处理

大数据知识

**Zookeeper**

基本概念、常见用法

**Solr，Lucene，ElasticSearch**

在linux上部署solr，solrcloud，，新增、删除、查询索引

**Storm，流式计算，了解Spark，S4**

在linux上部署storm，用zookeeper做协调，运行storm hello world，local和remote模式运行调试storm topology。

**Hadoop，离线计算**

HDFS、MapReduce

**分布式日志收集flume，kafka，logstash**

**数据挖掘，mahout**

网络安全知识

**什么是XSS**

XSS的防御

**什么是CSRF**

**什么是注入攻击**

SQL注入、XML注入、CRLF注入

**什么是文件上传漏洞**

**加密与解密**

MD5，SHA1、DES、AES、RSA、DSA

**什么是DOS攻击和DDOS攻击**

memcached为什么可以导致DDos攻击、什么是反射型DDoS

**SSL、TLS，HTTPS**

**如何通过Hash碰撞进行DOS攻击**

**用openssl签一个证书部署到apache或nginx**

四、架构篇分布式

数据一致性、服务治理、服务降级

**分布式事务**

2PC、3PC、CAP、BASE、 可靠消息最终一致性、最大努力通知、TCC

**Dubbo**

服务注册、服务发现，服务治理

**分布式数据库**

怎样打造一个分布式数据库、什么时候需要分布式数据库、mycat、otter、HBase

**分布式文件系统**

mfs、fastdfs

**分布式缓存**

缓存一致性、缓存命中率、缓存冗余

微服务

SOA、康威定律

**ServiceMesh**

**Docker & Kubernets**

**Spring Boot**

**Spring Cloud**

高并发

**分库分表**

**CDN技术**

**消息队列**

ActiveMQ

监控

**监控什么**

CPU、内存、磁盘I/O、网络I/O等

**监控手段**

进程监控、语义监控、机器资源监控、数据波动

**监控数据采集**

日志、埋点

**Dapper**

负载均衡

tomcat负载均衡、Nginx负载均衡

DNS

DNS原理、DNS的设计

CDN

数据一致性

五、 扩展篇云计算

IaaS、SaaS、PaaS、虚拟化技术、openstack、Serverlsess

搜索引擎

Solr、Lucene、Nutch、Elasticsearch

权限管理

Shiro

区块链

哈希算法、Merkle树、公钥密码算法、共识算法、Raft协议、Paxos 算法与 Raft 算法、拜占庭问题与算法、消息认证码与数字签名

**比特币**

挖矿、共识机制、闪电网络、侧链、热点问题、分叉

**以太坊**

**超级账本**

人工智能

数学基础、机器学习、人工神经网络、深度学习、应用场景。

**常用框架**

TensorFlow、DeepLearning4J

其他语言

Groovy、Python、Go、NodeJs、Swift、Rust