以学习新知识为主，以复习知识为辅，扎实java基础，构建java基础项目。

准备好近段时间的书籍，在空余时间看。

整理看过的笔记

大神网页：<https://cloud.tencent.com/developer/user/1667832>

Java技术大纲：<https://cloud.tencent.com/developer/article/1356245>

大数据：<https://cloud.tencent.com/developer/article/1349934>

Java面试所需的知识：<https://my.oschina.net/905042249/blog/2878180>

**java面试需要掌握知识点**

JVM内存管理机制和垃圾回收机制（基本每次面试都会问，一定要搞得透彻）

JVM内存调优（了解是怎么回事，一般做项目过程中使用较多）

设计模式（熟悉常见设计模式的应用场景，会画类图，实在不行背几个也是有用的）

多线程（每次都会问，包括线程和进程、线程状态、线程并发的操作等，需要深入复习）

JAVA集合类框架（理解框架图、HashMap、ArrayList、HashSet等的关系和区别，其中HashMap的存储机制几乎每次都有问）

JAVA的异常处理机制（异常的分类、常见的异常有哪些、Try catch finally的使用）

JVM运行机制（理解JVM是如何运行的）

Linux基础（面试笔试中对linux也有一定的要求，建议最好搭建一个linux虚拟机，并练习常用的命令）

**数据结构和算法**

常见的排序算法就不说了，需要理解其原理和会写代码，还有时间空间复杂度也要知道

队列、栈：需要理解其存取结构，并能在某些场景下使用

二叉树：树的遍历、树的深度、按层次输出、平衡二叉树、逆序打印树等

链表：逆序、合并两有序的链表、判断链表是否又环、链表倒数第K个元素等

字符串：KMP算法、动态规划（这个是重点，需要好好理解动态规划，常见的题有：求解最长回文子串、求解最长公共子串等）

海量数据处理：现在好多大公司都会问海量数据的处理，所以需要掌握常见的处理方法，比如Bit-map、分而治之、hash映射等，可以百度看看相关的文章，加深理解

数据库相关（数据库是面试的重点，每家公司都会问）

最基本的数据库CRUD操作要熟悉，能够根据表写出需要的SQL语句

事务、零时表、索引、表锁、行列锁、视图、存储过程等都要深入理解

互联网公司处理的数据量都很大，所以有必要对数据库优化和大数据的处理进行了解和熟悉

建议看看Nosql和redis等非关系数据库相关的书籍和资料

开源框架

像SSH等的开源框架在面试过程中问的很少，但是也是有必要了解SSH的运行原理和使用。面试主要考察的是基础，所以更加偏向于考察数据结构、算法、数据库、JAVA基础知识等。所以，如果你时间有限，可以把精力放在基础知识的学习和复习上，如果你时间充裕，可以学习和复习开源框架相关知识。对于开源框架：最好还是看看源码，对自己的好处是很大的。

**个人建议**

上面比较详细地列出和分析了最近面试过程中的问题，由于时间长久，不可能全部都回忆起来，但是大体上相差不大。对于面试，个人的一些建议：

（1）心态要放好，不能着急。不要看到被人拿到offer，自己没有就抱有心里压力，其实工作谁都可以找到，就看满意不满意，所以着急也是没有用。

（2）身体很重要。不要熬通宵复习、学习，把身体搞垮了得不偿失。保持良好的睡眠。

（3）善于总结面试中的问题。每一次笔试、面试完后，需要进行总结，将自己不会的知识点一定要搞清楚，如果不搞清楚，下次还有可能会问的。其实找工作的过程就是积累知识的过 程，所以善于总结很重要。

（4）不要盲目投递简历。不要看见大公司就投递，也不要投递自己不想去的城市，那样就是在瞎折腾，公司就算要你了，你也不去，还不如把时间放在自己想去的地方呢。

（5）复习需要持续。由于记忆过后会遗忘，所以复习过程中，需要反复进行，一边理解一边记忆，达到闭上眼就能看见知识的体系结构。

最后，希望找工作的朋友都能找到一份满意的工作。下面具体列出了面试常见的知识点，供大家参考，希望对你有所帮助。

主要详细知识点

常用算法考察

冒泡排序

快速排序

插入排序

希尔排序

归并排序

堆排序

桶排序

动态规划

最长公共子串

最长回文子串

数组的最大k个值

数字的最大连续子数组之和

左旋转字符串

字符串匹配算法:KMP算法

二分查找

链表

单链表逆序

两个有序单链表合并

两个单链表是否相交

相交处的节点

单链表倒数第K个数

单链表排序

栈和队列

设计包含min函数的栈

两个队列实现栈

两个栈实现队列

一个数组实现栈和队列

树

前序、中序、后续遍历

求二叉树的深度

按层次遍历二叉树

判断二叉树是否为完全二叉树

判断二叉树是否镜面对称

判断两颗树是否相等

卡特兰数

出栈顺序

n个节点构成的二叉树个数

括号化

凸多边形三角划分

设计模式

**Java源码中有哪些设计模式**

http://www.iteye.com/news/18725

**常见的设计模式类图**

http://blog.sina.com.cn/s/blog\_8cc6ff5c0101d4cp.html

**常见的设计模式的示例：**

http://www.jb51.net/article/27973.htm

数据库

数据库的crud语句的书写

数据库优化策略、如何优化

海量数据优化

事务

事务中锁分类

事务特性

数据库表锁

行列锁

临时表

索引

存储空间

视图

Linux常用命令（最好实战）

http://www.php100.com/html/webkaifa/Linux/2009/1106/3485.html

linux上搭建服务器

http://www.cnblogs.com/dudu/archive/2012/12/09/linux-apache-git.html

常用的网络端口

http://www.360doc.com/content/11/1202/09/8209768\_169068538.shtml

Nosql、redis等的熟悉、了解

http://www.yiibai.com/redis/redis\_quick\_guide.html

海量数据的处理

Bit-map、分而治之、hash映射、分布式处理（Hadoop）、Trie树、双层桶排序等

**Java基础**

HashMap的存储机制（很重要，必看）

Set、map区别

Set、list区别

Arraylist、LinkedList区别

String、Stringbufer、StringBuilder区别

HashSet、LinkedHashSet区别

HashMap、TreeMap、LinkedHashMap

熟悉Java集合类的框架图

HashMap、ArrayList、StringBuffer、String等源码的熟悉

线程和进程

线程的特性

线程的状态

多线程并发的实现

线程的创建方式

线程池的使用

Java作用域

final、static、native、等特殊关键字的使用

Java按值传递的过程

重构和重载

抽象类和接口

内部类的使用：匿名内部类、静态内部类等

Java多线程并发、生产者消费者的实现

Java连接数据库的过程

JVM内存管理和存储机制

http://blog.csdn.net/lengyuhong/article/details/5953544

**JVM垃圾回收机制，垃圾回收算法**

http://jbutton.iteye.com/blog/1569746

**JVM加载类过程**

http://www.cnblogs.com/sunada2005/p/3577799.html

**JVM内存调优**

http://www.cnblogs.com/jackyrong/archive/2010/01/21/1653163.html

http://blog.csdn.net/cutesource/article/details/5907418

**框架相关**

对框架的熟悉:SSH （面试过程中问的很少，像jsp、servlet基本不问）

Spring框架组成、AOP、IOC 、用到的设计模式

Hibernate运行机制、缓存的使用、最好看看源码

熟悉和了解JavaScript和jQuery

**java中IO、NIO**

Java NIO和IO之间第一个最大的区别是，IO是面向流的，NIO是面向缓冲区的。 Java IO面向流意味着每次从流中读一个或多个字节，直至读取所有字节，它们没有被缓存在任何地方。此外，它不能前后移动流中的数据。如果需要前后移动从流中读取的数据，需要先将它缓存到一个缓冲区。 Java NIO的缓冲导向方法略有不同。数据读取到一个它稍后处理的缓冲区，需要时可在缓冲区中前后移动。这就增加了处理过程中的灵活性。但是，还需要检查是否该缓冲区中包含所有您需要处理的数据。而且，需确保当更多的数据读入缓冲区时，不要覆盖缓冲区里尚未处理的数据。

**IO流包括字节流和字符流：**

字节流：对应抽象类为InputStream（输入流）和 OutputStream（输出流）。 FileInputStream、FileOutputStream

字符流：对应抽象类为Reader（输入流）和Writer（输出流）。BufferedReader、InputStreamReader、 StringReader

**java中异常的分类及结构**

Java中的异常类，包括内置的异常类以及自定义的异常类，都直接或者间接地继承至java.lang.Throwable类。在java.lang包中，Throwable类有两个直接子类：Error类和Exception类，Error类及其子类描述了java运行时系统的内部错误和资源耗尽错误。出现这样的错误的，除了通知用户，并接尽力使程序安全地终止之外，没有更好的办法。Exception类的层次结构又分为两个分支：一个分支由RuntimeException派生，另外一个分支包含除RuntimeException类之外的异常类。

**运行时异常、非运行时异常**

运行时异常：RuntimeException类及其子类异常，如NullPointerException、IndexOutOfBoundsException、classCastException等，这些异常是不检查异常，程序中可以选择捕获处理，也可以不处理。这些异常一般是由程序逻辑错误引起的，程序应该从逻辑角度尽可能避免这类异常的发生。

非运行时异常：RuntimeException以外的异常，类型上都属于Exception类及其子类。从程序语法角度讲是必须进行处理的异常，不处理程序不能编译通过。如IOException、SQLException等以及用户自定义的Exception异常。

**动态代理机制**

动态代理可以提供对另一个对象的访问，同时隐藏实际对象的具体事实。代理一般会实现它所表示的实际对象的接口。代理可以访问实际对象，但是延迟实现实际对象的部分功能，实际对象实现系统的实际功能，代理对象对客户隐藏了实际对象。客户不知道它是与代理打交道还是与实际对象打交道。动态代理主要包含以下角色：动态代理类（以下简称为代理类）是一个实现在创建类时在运行时指定的接口列表的类，该类具有下面描述的行为。代理接口 是代理类实现的一个接口。代理实例 是代理类的一个实例。每个代理实例都有一个关联的调用处理程序 对象，它可以实现接口 InvocationHandler。通过其中一个代理接口的代理实例上的方法调用将被指派到实例的调用处理程序的 Invoke 方法，并传递代理实例、识别调用方法的java.lang.reflect.Method 对象以及包含参数的 Object 类型的数组。调用处理程序以适当的方式处理编码的方法调用，并且它返回的结果将作为代理实例上方法调用的结果返回。

**try catch finally的使用**

1. finally 里 始终会被执行到， System.exit(0); 除这种被执行外。

2. 即使try中有return ，也是先执行 return 后面的语句完了之后，不立马return，而是去执行finally中的语句。

3. 当try中与finally里，同时出现return , 则只会返回 finally 中的return 结果。

4. finally中的值不能影响try中 即将返回的结果值。

注意： 若finally中没有return在try或catch中有return，那么在执行return跟着语句之后，会把语句的结果新开辟一内存空间，直接把结果的存放此内存空间中。所以，finally中的值不能影响try或catch中即将return的结果。

**静态成员、非静态成员**

（1）类的静态成员(变量和方法)属于类本身，在类加载的时候就会分配内存，可以通过类名直接去访问；非静态成员（变量和方法）属于类的对象，所以只有在类的对象产生（创建类的实例）时才会分配内存，然后通过类的对象（实例）去访问。

（2）在一个类的静态成员中去访问其非静态成员之所以会出错是因为在类的非静态成员不存在的时候类的静态成员就已经存在了，访问一个内存中不存在的东西当然会出错

**抽象类遵循的原则：**

（1）abstract关键字只能修饰类和方法，不能修饰字段。

（2）抽象类不能被实例化（无法使用new关键字创建对象实例），只能被继承。

（3）抽象类可以包含属性，方法，构造方法，初始化块，内部类，枚举类，和普通类一样，普通方法一定要实现，变量可以初始化、不初始化但不能初始化后在抽象类中重新赋值或操作该变量（只能在子类中改变该变量）。

（4）抽象类中的抽象方法（加了abstract关键字的方法）不能实现。

（5）含有抽象方法的类必须定义成抽象类。

**扩展：抽象类和接口的区别：**

（1）接口是公开的，里面不能有私有的方法或变量，是用于让别人使用的，而抽象类是可以有私有方法或私有变量的。

（2）abstractclass 在Java 语言中表示的是一种继承关系，一个类只能使用一次继承关系。但是，一个类却可以实现多个interface，实现多重继承。接口还有标识（里面没有任何方法，如Remote接口）和数据共享（里面的变量全是常量）的作用。

（3）在abstractclass 中可以有自己的数据成员，也可以有非abstarct的成员方法，而在interface中，只能够有静态的不能被修改的数据成员（也就是必须是static final的，不过在interface中一般不定义数据成员），所有的成员方法默认都是 public abstract 类型的。

（4）abstractclass和interface所反映出的设计理念不同。其实abstractclass表示的是"is-a"关系，interface表示的是"has-a"关系。

（5）实现接口的一定要实现接口里定义的所有方法，而实现抽象类可以有选择地重写需要用到的方法，一般的应用里，最顶级的是接口，然后是抽象类实现接口，最后才到具体类实现。抽象类中可以有非抽象方法。接口中则不能有实现方法。

（6）接口中定义的变量默认是publicstatic final 型，且必须给其初值，所以实现类中不能重新定义，也不能改变其值。抽象类中的变量默认是friendly 型，其值可以在子类中重新定义，也可以在子类中重新赋值。