### 后端常见问题

**StringBuffer StringBuilder String 区别**

String是一个常量，是不可变的，所以对于每一次+=赋值都会创建一个新的对象，StringBuffer和StringBuilder都是可变的，当进行字符串拼接时采用append方法，在原来的基础上进行追加，所以性能比String要高，又因为StringBuffer是线程安全的而StringBuilder是线程非安全的，所以StringBuilder的效率高于StringBuffer.

String 不可变 使用字符串拼接时是不同的2个空间

StringBuffer 可变 字符串拼接直接在字符串后追加 线程安全

StringBuilder 可变 字符串拼接直接在字符串后追加 非线程安全

**Hashtable与HashMap的区别**

Map是一个以键值对存储的接口。Map下有两个具体的实现，分别是HashMap和HashTable。 其中HashMap是线程非安全的，HashTable是线程安全的，所以HashMap的效率高于 HashTable.。HashMap允许键或值为空，而HashTable不允许键或值为空.

HashMap不是线程安全的，HashTable是线程安全。

HashMap允许空（null）的键和值（key），HashTable则不允许。

HashMap性能优于Hashtable。

**List、Set、Map比较**

Collection 接口：最基本的集合接口 一个Collection代表一组Object

Collection接口派生的两个接口是List和Set

List接口：有序的Collection使用此接口能够精确的控制每个元素插入的位置。用户能够使用 索引（元素在List中的位置，类似于数组下标）来访问List中的元素，这类似于Java的数组

LinkedList实现了List接口，允许null元素

ArrayList实现了可变大小的数组。它允许所有元素，包括null

Set接口：一种不包含重复的元素的Collection 最多有一个null元素 无序

add(obj),remove(int index),get(index) ,set(index,obj)修改

Map接口：没有继承Collection接口 提供key到value的映射

一个Map中不能包含相同的key，每个key只能映射一个 value。

Hashtable类：任何非空的对象都可作为key或者value。是同步的。

HashMap类：Hashtable类似，不同之处在于是非同步的，并且允许null value和null key。

添加数据使用put(key, value)，取出数据使用get(key)

**Java中常用的设计模式（23种）**

Factory（工厂模式），Singleton（单例模式）， Adapter（适配器模式）， Proxy（代理模式），

Visitor（访问者模式），Iterator（迭代子模式）等

**SQL**

Sql优化：

1、SELECT子句中避免使用 \*， 尽量应该根据业务需求按字段进行查询

2、尽量不要用子查询用表连接

3、常用查询条件建索引（主键索引、唯一索引。。。）

4、WHERE子句中的连接顺序，ORACLE采用自下而上的顺序解析WHERE子句，那些可以过滤掉最大数量记录的条件必须写在WHERE子句的末尾

5、用UNION-ALL 替换UNION，因为UNION-ALL不会过滤重复数据，所执行效率要快于UNION,并且UNION可以自动排序，而UNION-ALL不会

建表的三大方式：

第一范式：1NF对属性的原子性约束，要求属性具有原子性，不可再分解；

第二范式：2NF对记录的惟一性约束，要求记录有惟一标识，即实体的惟一性；

第三范式：3NF对字段冗余性的约束，字段没有冗余，即任何字段不能由其他字段派生出来。

冒泡排序：

比较相邻两个元素的值，如果满足条件就交换值，把比较小的元素移动到数组的前面

外层循环控制排轮数 数组长度减1

内层循环用来对比数组中每个相邻元素的大小，随着轮数减少

for(int i=1;i<array.length;i++){

for(int j=0;j<array.length-i;j++){

if(array[j]>array[j+1]{

int temp = array[j];

Array[j]=array[j+1];

Array[j+1]=temp;

}

}

}

oracle 分页：

select \* from (select t.\*, rownum rn from (select \* from menu order by id desc) t where rownum < 5 ) where rn >=0

mysql 分页：

select \* from music where id limit 1,5

**操作系统**

之前公司发布就是需要用到linux系统，所以基本的常用命令还是会一点

cd 进入下个菜单； ls 查询； mkdir 创建； rm 删除； mv 移动； find 搜索

sh.start 启动项目，sh.stop 关闭项目 tail -f 查看日志

**异常问题**

异常类分两大类型：Error类和Exception类

Error类：代表了编译和系统的错误，不允许捕获；

Exception类：代表了标准Java库方法所激发的异常。

Exception类包含运行异常类和非运行异常类这两个直接的子类。

运行异常类对应于编译错误，它是指Java程序在运行时产生的由解释器引发的各种异常。运行异常可能出现在任何地方，且出现频率很高，因此为了避免巨大的系统资源开销，编译器不对异常进行检查。所以Java语言中的运行异常不一定被捕获。出现运行错误往往表示代码有错误，如：算数异常（如被0除）、下标异常（如数组越界）等。

非运行异常时Non\_RuntimeException类及其子类的实例，又称为可检测异常。Java编译器利用分析方法或构造方法中可能产生的结果来检测Java程序中是否含有检测异常的处理程序，对于每个可能的可检测异常，方法或构造方法的throws子句必须列出该异常对应的类。在Java的标准包java.lang java.util 和 java.net 中定义的异常都是非运行异常。

空指针异常类：NullPointerException

类型强制转换异常：ClassCastException

数组负下标异常：NegativeArrayException

数组下标越界异常：ArrayIndexOutOfBoundsException

违背安全原则异常：SecturityException

文件已结束异常：EOFException

文件未找到异常：FileNotFoundException

字符串转换为数字异常：NumberFormatException

操作数据库异常：SQLException

输入输出异常：IOException

方法未找到异常：NoSuchMethodException

**多线程**

一个进程可能包含多个线程

实现线程的两种方式：

1、继承java.lang.Thread类

2、实现java.lng.Runnable接口

线程的生命周期：

出生状态：new Thread();

就绪状态：调用start()方法后

运行状态：得到系统资源后就进入运行状态

等待状态:调用wait()方法进入等待状态，通过notify()方法唤醒

休眠状态：调用sleep()方法进入等待状态

阻塞状态：在运行状态下发出输入或者输出请求时处于阻塞状态 在等待结束时是就绪状态 永 远不能回到运行状态

死亡状态：执行run()方法进入死亡状态