1.List和Map底层用的是什么，分别适用于什么场景

List和Map的底层分别用的是数组和链表，

数组的应用场景：数据量比较少，经常做的是按序号访问元素，数组更容易实现，任何高级语言都支持，构建的线性表较稳定。

链表的应用场景：对线性表的长度或者规模难以估计，频繁的做插入删除操作，构建动态性比较强的线性表。

2.ArrayList为什么增删效率低，查询速率高

因为在ArrayList的中间插入或删除一个元素意味着这个列表中剩余的元素都会被移动，所以ArrayList的增删速度比较低，

ArrayList的内部实现是基于基础的对象数组的，所以他在使用get方法访问列表中的任意一个元素时，速度要快。

linkedList为什么增删效率高，查询速率低：

linkedList的底层是基于链表结构的，所以在LinkedList的中插入或者删除一个元素的开销是固定的，

LinkedList中的get方法是按照顺序从列表的一端开始检查，直到另外一端。所以查询速率低。

3.HashMap的键是如何保证唯一性的

通过hashcode（）和equals（）方法保证键的唯一性；

Equals方法用于比较是否重复，hashcode用于减少比较的次数；

4.堆栈结构的特点

先进后出，后进先出；

5.多线程的创建方式以及不同：

1.继承 Thread 类：但 Thread 本质上也是实现了 Runnable 接口的一个实例，它代表一个线程的实例，并且，启动线程的唯一方法就是通过 Thread 类的 start()实例方法。start()方法是一个 native 方法，它将启动一个新线程，并执行 run()方法。这种方式实现多线程很简单，通过自己的类直接 extend Thread，并复写 run()方法，就可以启动新线程并执行自己定义的 run()方法。例如：继承 Thread 类实现多线程，并在合适的地方启动线程

2．实现Runnable接口的方式实现多线程，并且实例化Thread，传入自己的Thread实例，调用run（）方法；

3.使用ExecutorService .Callable .Future 实现有返回结果的多线程：

ExecutorService Callable Future 这些对象实际上都属于Executor框架中的功能类；

6.在java中wait和sleep方法有什么不同：

最大的不同是等待是wait方法会释放锁，而sleep方法一直持有锁，wait方法通常被用于线程间的交互，sleep方法通常被用于暂停执行。

7.说说多线程的几种状态:

New（新建）:线程刚刚被创建，但是并未启动，还没有调用start方法

Runnable（可运行）：线程可以在java虚拟机中运行的状态，可能正在运行自己的代码，也可能没有，这取决与操作系统处理器

Blocked（锁阻塞）：当一个线程试图获取一个对象锁，而该对象锁被其它线程持有，则该线程进入Blocked状态，当该线程持有锁的时候，该线程变成Runnable状态。

Waiting（无线等待）：一个线程在等待另一个线程执行一个（唤醒）动作，该线程进入Waiting状态，进入这个状态后是不能自动唤醒的，必须等待另一个线程调用notify或者notifyAll方法才能够唤醒。

Timed Waiting（计时等待）：同waiting状态，有几个方法有超时参数，调用他们进入Timed Waiting状态，这一状态将一直保持到朝时期满或者接收到唤醒通知。带有超时参数的常用方法有Thread.sleep Object.wait。

Teminated（被终止）：因为run方法正常退出而死亡，或者因为没有捕获的异常终止了run方法而死亡。

8.什么是线程池如何使用：

线程池就是事先将多个线程对象放到一个容器中，当使用的时候就不用new线性而是直接去线程池中拿取线程即可，节省了开辟线程的时间，提高代码的执行效率。

Java.util.concurrent.Excutors中提供了生成多种线程池的静态方法。

9.java中异常的分类：

按照异常异常需要处理的时机分为编译时异常和运行时异常

对于编译时异常的处理方式有两种一种是try（）catch，另一种就是直接抛出。

而对于运行是异常只有当代码在运行的时候才会发生。

说出5个runtimeException：

NullPointerException：空指针异常

ClassNotFoundException：找不到指定的类

NumberFormatException：字符串转换为数字异常

IndexOutOfBoundException：数组索引越界异常

ClassCastException：数据类型转换异常

SQLException：SQL异常

10.Throw和throws的区别：

Throw语句语句用在方法体内，表示抛出异常，由方法体内的语句处理；

Throw是具体向外抛出异常的动作，所以他抛出的是一个异常实例，执行throw一定是抛出某种异常。

Throws语句是用在方法声明后面，表示如果抛出异常由该方法的调用者来进行处理；

Throws主要是声明这个方法会抛出某种类型的异常，让他的使用者要知道需要捕获的异常的类型；

Throws表示出现异常额一种可能性，并不是一定会发生这种异常

11.String. StringBuild. StringBuffer的区别

String是只读字符串，也就意味着String引用的字符串内容是不能被改变的；

StringBuffer/StringBuilder表示的字符串对象可以直接进行修改。

StringBuilder是java5中引入的，它和StringBuffer的方法完全相同，区别在于它是在单线程环境下使用的，因为它的所有方法都没有被synchronize修饰，因此它的效率理论上也比StringBuffer要高；

12.Java的基本数据类型都有那些各占了几个字节



13.java中有几种类型的流

按照流的方向可分为：输入流和输出流

按照处理数据的单位可分为：字节流和字符流

按照实现功能可分为：节点流（可以从或者向一个特定的节点读取数据。如FileRearder）和处理流（对一个与存在的流的连接和封装，通过所封装的流的功能实现数据读写。如BufferedReader）

14.什么是java的序列化，如何实现java序列化：

序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可以将流化后的对象传输于网络之间。序列化是为了解决在对 对象流进行读写操作是所引发的问题。

序列化的实现：将需要被序列化的类 实现Serializable接口，该接口没有需要实现的方法，implements Serializable 只是为了标注该对象是可以被序列化的，然后使用一个输出流来构造一个ObjectOutPutStream对象，接着使用ObjectOutPutStream对象的writeObject（Object obj）方法就可以将参数为obj的对象进行写出；

15.List 和Map Set的区别：

List和Set是存储单列数据的集合，Map是存储键和值这样双列数据的集合；List中存储的数据是有顺序的，并且允许重复；Map中存储的数据是没有顺序的，其键是不能重复的，Set中存储的数据无序且不能重复，但元素在集合中的位置由元素的hashcode决定，

16.抽象类和接口的相同点和不同点：

不同点：

抽象类：

抽象类可以定义构造器

可以有抽象方法和具体方法

抽象类中的成员可以是private，默认，protected Public

抽象类中可以定义成员方法

有抽象方法的类必须被声明为抽象类，而抽象类未必要有抽象方法

抽象类中可以包含静态方法static

一个类只能继承一个抽象类

接口：

接口中不能定义构造器

方法全部都是抽象方法

接口中定义的成员变量实际上都是常量

接口中的成员全都是public的

接口中不能有静态方法

一个类可以实现多个接口

相同：

不能够实例化

可以将抽象类和接口作为引用类型

一个类如果继承了某个抽象类或者实现某个接口都需要对其中的抽象方法全部进行实现，否则该类仍然需要被声明为抽象类。

17.==和equals的区别：

Equals和==最大的区别就是一个是方法一个是运算符。

==：如果比较的对象是基本数据类型，则比较的是数值是否相等；如果比较的是引用类型，则比较的是对象的地址值是否相等。

Equals（）：用来比较方法两个对象的内容是否相等。

注意：equals方法不能用于基本数据类型变量，如果没有对equals方法进行重写，则比较的是引用类型的变量所指向的对象地址。

18.Java中实现多态的机制是什么：

指的是父类或接口定义的引用变量可以指向子类或者具体实现类的实例对象，而程序调用的方法在运行期间才动态绑定，就是引用变量所指的具体实例对象的方法，也就是内存里正在运行的那个对象的方法，而不是引用变量的类型中定义的方法。

允许基类的指针或引用类型指向派生类的对象，而在具体访问时实现方法的动态绑定。

19.Break和continue的区别：

Break和continue都是用来控制循环的语句

Break用于完全结束一个循环，跳出循环执行体-执行循环后面的语句。

Continue用于跳过本次循环，执行下次循环。

&和&&的区别：

&运算符有两种用法：（1）按位与；（2）逻辑与；

&&运算符是短路与：逻辑与跟短路与的差别是非常巨大的，虽然二者都要求运算符左右两端的布尔值都是true整的表达式的值才是true。

&&之所以被成为短路运算符是因为，如果&&左边的表达式的值是false，右边的表达式会被直接短路掉，不会进行运算。很多时候我们使用的都是&&而不是&。

20.数据库事务的四个特性及含义

数据库事务transanction正确执行的四个基本要素。ACID,原子性(Atomicity)、一致性(Correspondence)、隔离性(Isolation)、持久性(Durability)。

原子性:整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。

一致性:在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。

隔离性:隔离状态执行事务，使它们好像是系统在给定时间内执行的唯一操作。如果有两个事务，运行在相同的时间内，执行 相同的功能，事务的隔离性将确保每一事务在系统中认为只有该事务在使用系统。这种属性有时称为串行化，为了防止事务操作间的混淆，必须串行化或序列化请 求，使得在同一时间仅有一个请求用于同一数据。

持久性:在事务完成以后，该事务所对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。

21.数据库优化的思路

1.SQL语句优化(至少说出3点,可以百度查)

1）应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。

2）应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如：

select id from t where num is null

可以在num上设置默认值0，确保表中num列没有null值，然后这样查询：

select id from t where num=0

3）很多时候用 exists 代替 in 是一个好的选择

4）用Where子句替换HAVING 子句 因为HAVING 只会在检索出所有记录之后才对结果集进行过滤

22.什么是数据库的三范式?

第一范式，又称1NF，它指的是在一个应用中的数据都可以组织成由行和列的表格形式，且表格的任意一个行列交叉点即单元格，都不可再划分为行和列的形式，实际上任意一张表格都满足1NF； 第二范式，又称2NF，它指的是在满足1NF的基础上，一张数据表中的任何非主键字段都全部依赖于主键字段，没有任何非主键字段只依赖于主键字段的一部分。即，可以由主键字段来唯一的确定一条记录。比如学号+课程号的联合主键，可以唯一的确定某个成绩是哪个学员的哪门课的成绩，缺少学号或者缺少课程号，都不能确定成绩的意义。 第三范式，又称3NF，它是指在满足2NF的基础上，数据表的任何非主键字段之间都不产生函数依赖，即非主键字段之间没有依赖关系，全部只依赖于主键字段。例如将学员姓名和所属班级名称放在同一张表中是不科学的，因为学员依赖于班级，可将学员信息和班级信息单独存放，以满足3NF。

<https://wenku.baidu.com/view/85e160bd294ac850ad02de80d4d8d15abe230013.html>

23.SQL 之聚合函数

聚合函数是对一组值进行计算并返回单一的值的函数，它经常与 select 语句中的 group by 子句一同使 用。

a. avg():返回的是指定组中的平均值，空值被忽略。 b. count():返回的是指定组中的项目个数。

c. max():返回指定数据中的最大值。

d. min():返回指定数据中的最小值。

e. sum():返回指定数据的和，只能用于数字列，空值忽略。

f. group by():对数据进行分组，对执行完 group by 之后的组进行聚合函数的运算，计算每一组的值。 最后用 having 去掉不符合条件的组，having 子句中的每一个元素必须出现在 select 列表中(只针对于 mysql)。