集合：

List : 有序，可重复，有下标索引信息。

ArrayList ：线程不安全，底层是数组，初始容量10，扩容原来的1.5倍。

LinkedList：底层是双向链表比ArrayList添加删除快，查询没有ArrayList快。

CopyOnWriteArrayList ：是线程安全的，适合大量的读操作。

Set：无序，不可重复，没有下标索引信息。

HashSet ：

LinkedHashSet：

TreeSet :

Map :key – value

HashMap : 线程不安全的，初始容量是16，负载因子0.75，就是当存储键到达12个是开始扩容，通过rehash方法扩容为原来的两倍，HashMap最大容量是2的30幂。

LinkedHashMap：是HashMap的子类，保留了添加的顺序。

TreeMap：key类型必须是可排序的。

HashTable：线程安全的初始容量11，负载因子0.75，扩容是原数组的2倍加1，不能有null键和null值。之所以HashTable效率低是因为它在方法上添加sychornized。

CurrentHashMap：是JUC下线程安全的容器，它的性能比HashTable效率高，因为它中分为16个类似于HashMap的容器。假如添加一个元素，首先算出key的hashcode值，然后在hash算出到底存入哪个小的HashMap，只对这个小的HashMap加锁。

IOC的理解：ioc加做控制反转，他是一种设计思想。那么“正”是什么意思？假如我一人在家想吃宫保鸡丁，我需要考虑家中有没有鸡胸，葱，黄瓜，没有我还要去菜市场买，离得远我还得开车，很麻烦这是 “正向的”。现在反向过来有一家饭店为你做宫保鸡丁，你不用管它是怎么做的，你只需要打个电话说我要吃宫爆鸡丁，饭店就会为你送过来，你只负责吃。就像好莱坞设计思想：“你不用找我们，我们找你。”

DI的理解：依赖注入，是ioc的一种重要实现，谁依赖谁？应用程序依赖ioc。依赖注入并没有为系统提供更多的功能，而是让系统更加灵活，可扩展，降低耦合，不用任何代码就可指定目标需要的资源。

AOP的理解：面向切面编程的目的就是分离关注点。假如你是个公子哥，没啥人生目标，天天就是衣来伸手，饭来张口，整天只知道玩一件事！那么，每天你一睁眼，就光想着吃完饭就去玩（你必须要做的事），但是在玩之前，你还需要穿衣服、穿鞋子、叠好被子、做饭等等等等事情，这些事情就是你的关注点，但是你只想吃饭然后玩，那么怎么办呢？这些事情通通交给别人去干。在你走到饭桌之前，有一个专门的仆人A帮你穿衣服，仆人B帮你穿鞋子，仆人C帮你叠好被子，仆人C帮你做饭，然后你就开始吃饭、去玩（这就是你一天的正事），你干完你的正事之后，回来，然后一系列仆人又开始帮你干这个干那个，然后一天就结束了！

        AOP的好处就是你只需要干你的正事，其它事情别人帮你干。也许有一天，你想裸奔，不想穿衣服，那么你把仆人A解雇就是了！也许有一天，出门之前你还想带点钱，那么你再雇一个仆人D专门帮你干取钱的活！这就是AOP。每个人各司其职，灵活组合，达到一种可配置的、可插拔的程序结构。Aop的底层实现是动态代理。

        从Spring的角度看，AOP最大的用途就在于提供了事务管理的能力。事务管理就是一个关注点，你的正事就是去访问数据库，而你不想管事务（太烦），所以，Spring在你访问数据库之前，自动帮你开启事务，当你访问数据库结束之后，自动帮你提交/回滚事务！

MySql：

隔离级别：

1. 读未提交（read uncommitted）
2. 读已提交（read committed）
3. 不可重复读（repeatable read）
4. 可串行化（serializable）

索引：

什么情况建立索引：

1. 主键自动建立索引
2. 频繁作为条件查询的字段应该建立索引
3. 查询中与其他表关联的字段，外键关系建立索引
4. 单键/复合键选择问题，复合键性价比高
5. 排序的字段添加索引
6. 统计或分组的字段添加索引

什么情况不建立索引：

1. 表中数据经常添加删除
2. 表中的数据太小
3. Where条件用不到的字段不要创建索引。

Explain ：模拟sql语句，从而知道mysql是如何处理你的sql语句。

查看哪些重要信息：

1. Id：相同，顺序由上到下执行。不相同id值大的先执行。
2. Type：一般来说查询至少达到range，最好能到达到ref。
3. Key：实际使用的索引。
4. Rows：实际物理扫描的行数，越少越好。
5. Extra：其他信息，using where索引用到了条件查询；using filesort 没有使用索引进行排序。

索引失效：

1. 再索引列上进行了计算，函数，会导致索引失效。
2. 使用范围条件右边的索引失效。
3. Is not null 无法使用索引。
4. Like “%abc” 会导致索引失效。
5. 字符串不加单引号会导致索引失效。

项目中redis是怎么用的？

Redis存放商品信息，用到了zset这个数据类型，商品过滤需要用到取交集的功能所以用zset数据类型。

Zset的key存放的是商品的二级分类的id，value是sku商品的集合。

取交集需要把某件东西下所有不同属性的商品分别存放到redis中。具体key是

Key = av\_28\_17\_18

二级分类的id 属性名的id 属性值的id

取交集方法：zinterstore

多线程：

说道多线程就要知道什么事并发什么事并行。

并发：指在同一台计算机上同一时刻做同样的事情。

并行：在不同的计算机同一时刻做同样的事情。

所以多线程就一个目的就是更好的利用cpu资源。

首先说线程池：线程的工具类 Executors

1. FixedThreadPool（5）：固定的5个线程
2. singleThreadExecutor（）单利的线程池
3. cachedthreadpool（） 自适应的线程池
4. 调度的线程池，我不太清楚。

举例说明线程池：银行不管大厅中有多少人办理业务都开4个窗口，这种叫固定的线程池。银行不管大厅中有少人哪怕是爆满也只开一个窗口，这种叫单利线程池。银行根据大厅办理业务人数的多少合理的开设窗口，这种叫自适应线程池。

说完线程池再说其他三种实现线程的方法，Thread类，runnable接口，callable接口，callable实现call方法，他可以有返回值，可以抛出异常，只执行一次。

然后说锁 Lock类：ReentrantLock ReadLock WriteLock

ReentrantLock 加锁之后要解锁，解锁必须放到finnaly中，有结论表明既是出现了oom堆内存溢出也会保证finnally中的代码被执行。

ReadLock 与WriteLock 读写锁：写写，写读互斥，读读不互斥，这样的好处是提高了大量读的性能。

ReentrantLock 与sychornized用哪个？

ReentrantLock 是类，它有sychornized所有的功能。

在并发量比较小的情况下，使用synchronized是个不错的选择，但是在并发量比较高的情况下，其性能下降很严重，此时ReentrantLock是个不错的方案。