**一，问答题**

**1，介绍下四种会话跟踪技术并对其加以说明？**

**标签隐藏域：标签隐藏域就是在用户看到的地方，隐藏标签里面存放信息，发送到数据库**

**URL重写：url可以进行拼接，添加元素信息，然后服务器可以接收信息**

**Cookie：cookie的信息保存在浏览器，是浏览器端的会话技术，服务器可以读取这些信息**

**Session：session基于cookie实现，将一些敏感信息保存在服务器端，服务器可以随时读取**

**2，垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收？**

**1，垃圾回收器：当对象没有任何引用的时候，他会被扫描，经过算法扫描，确认不被使用才会被回收**

**2，垃圾回收期不会马上回收内存，而是会经过新生代，老年代，永久代的垃圾回收算法扫描，只有真的不被使用才会被进行回收**

**3，可以调用system.gc()主动进行**

**3，什么是java序列化，如何实现java序列化？**

**Java的序列化操作是将对象从内存中，保存到磁盘本地，需要实现serlizable接口，或者用objectoutputstream进行主动序列化**

**4，说出一些常用的类，包，接口，请各举5个**

**类：hashmap，arrarylist，string，runnable，linkedlist**

**包：spring-context，mybatis，druid，aspjwawer，mysql-connection-5.1**

**接口：serlizable，list，map，applicationContext，sqlsessionfactorybuild**

**5，开发中都用到了那些设计模式?用在什么场合?**

**单例模式：当你只需要获取只需要获取一个实体类的单个对象的时候，例如spring容器里面的对象，默认是单例的，不管你怎么获取，他在内存中只存在一份**

**代理模式：代理模式可以在不修改源码的情况下，增强原来方法的功能，例如spring AOP的事务**

**工厂模式：**

**构建者模式：**

**6，BS与CS的联系与区别。**

**BS:browser/server，浏览器访问服务器，基于HTTP的协议实现，只要用户有网络，有一个浏览器就可以进行访问，访问简便，访问压力主要在 服务器端。**

**CS:client/server，客户端访问服务器，用户访问需要下载客户端，才可以访问，访问压力由客户端和服务器一起承担。**

**7，请大概描述一下Linkedlist和ArrayList的区别，Hashtable和HashMap的区别，线程安全与否**

**答：Arrarylist：存储的时候在内存中是连续的一段地址，当你获取可以按照索引快速的获得；当你要末尾增加的时候，直接添加新的内存在尾部即可；但是当你要插入或者删除的时候，那么你必须要在插入或者删除的索引处，向后移动内存或者向前移动内存，消耗的时间很多。**

**总结：ArrayList 查找快，删除慢，插入慢，适合经常查找数据，不经常修改数据的场景。**

**Linkedlist：存储的时候在内存中不是连续的地址，当你获取的时候不可以按照索引快速的获得，你必须从头还是一个个寻找到你要查找的位置；当你要末尾增加的时候，原先的末尾会直接将他的nextNode指向你，同时链表的lastNode也会指向你，你的preNode也会指向原来的末尾；当你要插入或者删除的时候，那么先从头开始遍历到你要插入或者删除的索引处，同时对前一个节点和后一个节点的nextNode和preNode进行指向新的逻辑关系。例如删除是将索引的上一个节点的nextNode指向索引的下一个节点。**

**总结：Linkedlist查找慢，删除快，插入快，适合经常修改数据，但是不经常查找数据的场景**

**Hashtable和HashMap：**

**1，hashmap是线程不安全的，hashtable是线程安全的。**

**2，hashmap的效率没有hashtable高，**

**8，请阐述一下你对JAVA多线程中“锁”的概念的理解**

**答：锁的出现主要是为了解决多线程处理的时候同时访问同一块资源的处理问题，当多个线程处理一个变量的时候，每个线程申请到的CPU时间是不一样的，可能你运行到一半就会停止，由其他线程执行，那么前一个线程处理的变量会再次被处理，这个变量就被改变。所以对于多线程的类，不允许定义类成员变量，或者必须定义不可以修改的类成员变量。**

**为了解决多线程操作的问题，锁出现了，锁主要分两种，悲观锁和乐观锁。**

**悲观锁：当线程A处理数据的时候，A把要处理的数据全部上锁，其他线程必须等到我处理完了，其他线程才能够处理我的数据，A悲观的认为其他人一定会动我的数据。**

**乐观锁：当线程A处理数据的时候，A会记录他处理过得变量的值，当他下一次再处理这个变量的时候，他会进行对比，是否和他上一次处理的一致，如果一致则继续处理，如果不一致则返回不处理，A乐观的认为，我的数据不会有人处理，但是我要确认一遍，防止有人动了。**

**悲观锁适合工程大，并发的场景，线程之间的多个处理，乐观锁适合工程小，并发小的场景，没有那么多线程压力。**

**9，请阐述一下你对“面向接口编程”的理解**

**1，接口定义需要的功能，具体实现的话可以由子类进行继承实现。**

**2，接口可以实现多态，所有的实现都由父类接口接收，然后由父类接口调用功能。**

**10， Try to explain Singleton to us? Is it thread safe? If no, how to make it thread safe?**

**尝试向我们解释Singleton吗？ 它是线程安全的吗？ 如果没有，如何使它线程安全？**

**答：以懒汉式为例，最开始的时候是线程不安全的，可以对其getInstance进行加锁操作保证其的多线程安全性，**

**当你对懒汉式进行加锁操作后，他理论上是线程安全的，但是多线程执行的时候，还会需要注意代码指令重排和CPU优化的问题，所以对定义的变量要进行volatile描述，可以使其不受代码重排的影响**

**Private volatile static Singleton singleton**

**二，数据库题**

**表结构如下：**

**flight{flightID,StartCityID ,endCityID,StartTime}**

**city{cityID, CityName)**

**实验环境：**

**create table city(cityID int auto\_increment primary key,cityName varchar(20));**

**create table flight (flightID int auto\_increment primary key,**

**StartCityID int references city(cityID),**

**endCityID int references city(cityID),**

**StartTime timestamp);**

**插入的数据：**

**insert into city values(null,'北京'),(null,'上海'),(null,'广州');**

**insert into flight values**

**(null,1,2,'9:37:23'),(null,1,3,'9:37:23'),(null,1,2,'10:37:23'),(null,2,3,'10:37:23');**

**1、查询起飞城市是北京的所有航班，按到达城市的名字排序**

**SELECT \* FROM flight WHERE flight.StartCityID = (SELECT city.cityID FROM city WHERE city.cityName = '北京') ORDER BY flight.endCityID**

**2、查询北京到上海的所有航班纪录（起飞城市，到达城市，起飞时间，航班号）**

**SELECT '北京', '上海', flight.StartTime, flight.flightID FROM flight WHERE**

**flight.StartCityID = (SELECT city.cityID FROM city WHERE city.cityName = '北京') AND**

**flight.endCityID = (SELECT city.cityID FROM city WHERE city.cityName = '上海')**