蚂蚁金服面经

1. ArrayList 和 LinkedList 区别？

ArrayList 是一个可改变大小的数组。当更多的元素加入到 ArrayList 中时，其大小将会动态地增长。内部的元素可以直接通过 get 与 set 方法进行访问，因为 ArrayList 本质上就是一个数组。

LinkedList 是一个双链表，在添加和删除元素时具有比 ArrayList 更好的性能.但在 get 与 set 方面弱于 ArrayList。当然，这些对比都是指数据量很大或者操作很频繁的情况下的对比,如果数据和运算量很小，那么对比将失去意义。

2. 什么情况会造成内存泄漏？

在 Java 中，内存泄漏就是存在一些被分配的对象，这些对象有下面两个特点：

首先，这些对象是可达的，即在有向图中，存在通路可以与其相连；

其次，这些对象是无用的，即程序以后不会再使用这些对象。如果对象满足这两个条件，这些对象就可以判定为 Java 中的内存泄漏，这些对象不会被 GC 所回收，然而它却占用内存。

3. 什么是线程死锁，如何解决？

产生死锁的条件有四个：

- 互斥条件：所谓互斥就是进程在某一时间内独占资源。
- 请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。
- 不剥夺条件: 进程已获得资源，在未使用完之前，不能强行剥夺。

- 循环等待条件：若干进程之间形成一种头尾相接的循环等待资源关系。

线程死锁是因为多线程访问共享资源，由于访问的顺序不当所造成的，通常是一个线程锁定了一个资源 A，而又想去锁定资源 B；

在另一个线程中，锁定了资源 B，而又想去锁定资源 A 以完成自身的操作，两个线程都想得到对方的资源，而不愿释放自己的资源，造成两个线程都在等待，而无法执行的情况。

要解决死锁，可以从死锁的四个条件出发，只要破坏了一个必要条件，那么我们的死锁就解决了。在 Java 中使用多线程的时候一定要考虑是否有死锁的问题。

4. 说一下 HashMap 以及它是否线程安全

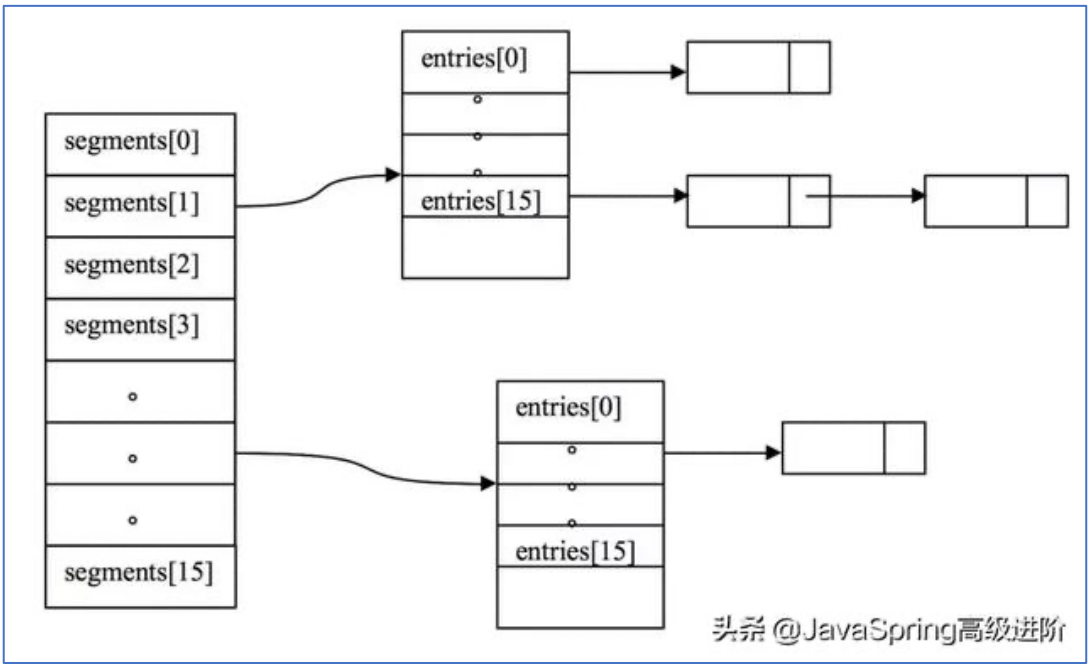
HashMap 基于哈希表的 Map 接口的实现。

HashMap 中，null 可以作为键，这样的键只有一个；可以有一个或多个键所对应的值为 null。

HashMap 中 hash 数组的默认大小是 16，而且一定是 2 的指数。Hashtable、HashMap 都使用了 Iterator。而由于历史原因，Hashtable 还使用了 Enumeration 的方式 。

HashMap 实现 Iterator，支持 fast-fail。

哈希表是由数组+链表组成的，它是通过把 key 值进行 hash 来定位对象的，这样可以提供比线性存储更好的性能。



HashMap 不是线程安全的。

5. 说一下 Innodb 和 MySIAM 的区别？

MyISAM 类型不支持事务处理等高级处理，而 InnoDB 类型支持。MyISAM 类型的表强调的是性能，其执行数度比 InnoDB 类型更快，但是不提供事务支持，而 InnoDB 提供事务支持以及外部键等高级数据库功能。

InnoDB 不支持 FULLTEXT 类型的索引。

InnoDB 中不保存表的具体行数，也就是说，执行 select count(*) from table 时，InnoDB 要扫描一遍整个表来计算有多少行，但是 MyISAM 只要简单的读出保存好的行数即可。注意的是，当 count(*)语句包含 where 条件时，两种表的操作是一样的。

对于 AUTO_INCREMENT 类型的字段，InnoDB 中必须包含只有该字段的索引，但是在 MyISAM 表中，可以和其他字段一起建立联合索引。

DELETE FROM table 时，InnoDB 不会重新建立表，而是一行一行的删除。

LOAD TABLE FROM MASTER 操作对 InnoDB 是不起作用的，解决方法是首先把 InnoDB 表改成 MyISAM 表，导入数据后再改成 InnoDB 表，但是对于使用的额外的 InnoDB 特性(例如外键)的表不适用。

6. 访问淘宝网页的一个具体流程，从获取 IP 地址，到怎么返回相关内容？

先通过 DNS 解析到服务器地址，然后反向代理、负载均衡服务器等，寻找集群中的一台机器来真正执行你的请求。还可以介绍 CDN、页面缓存、Cookie 以及 session 等。

这个过程还包括三次握手、HTTP request 中包含哪些内容，状态码等，还有 OSI 七层分层可以介绍。

服务器接到请求后，会执行业务逻辑，执行过程中可以按照 MVC 来分别介绍。

服务处理过程中是否调用其他 RPC 服务或者异步消息，这个过程包含服务发现与注册，消息路由。

最后查询数据库，会不会经过缓存？是不是关系型数据库？是会分库分表还是做哪些操作？

对于数据库，分库分表如果数据量大的话是有必要的，一般业务根据一个分表字段进行取模进行分表，而在做数据库操作的时候，也根据同样的规则，决定数据的读写操作对应哪张表。这种也有开源的实现的，如阿里的 TDDL 就有这种功能。分库分表还涉及到很多技术，比如 sequence 如何设置 ，如何解决热点问题等。

最后再把处理结果封装成 response，返回给客户端。浏览器再进行页面渲染。

7. 介绍一下并发

这里可以把整个并发的体系都说下，包括 volatile、synchronized、lock、乐观悲观锁、锁膨胀、锁降级、线程池等。

8. 说一下关系型数据库和非关系型数据库的区别

非关系型数据库的优势：

性能：NOSQL 是基于键值对的，可以想象成表中的主键和值的对应关系，而且不需要经过 SQL 层的解析，所以性能非常高。

可扩展性：同样也是因为基于键值对，数据之间没有耦合性，所以非常容易水平扩展。

使用场景：日志、埋点、论坛、博客等。

关系型数据库的优势：

复杂查询：可以用 SQL 语句方便的在一个表以及多个表之间做非常复杂的数据查询

事务支持:使得对于安全性能很高的数据访问要求得以实现。

使用场景：所有有逻辑关系的数据存储。

9. 如何访问链表中间节点

对于这个问题，我们首先能够想到的就是先遍历一遍整个的链表，然后计算出链表的长度，进而遍历第二遍找出中间位置的数据。这种方式非常简单。

若题目要求只能遍历一次链表，那又当如何解决问题？

可以采取建立两个指针，一个指针一次遍历两个节点，另一个节点一次遍历一个节点，当快指针遍历到空节点时，慢指针指向的位置为链表的中间位置，这种解决问题的方法称为

快慢指针方法。

10. 说下进程间通信，以及各自的区别

进程间通信是指在不同进程之间传播或交换信息。方式通常有管道（包括无名管道和命名管道）、消息队列、信号量、共享存储、Socket、Streams 等。

面经资料来自网络