慧敏：

1. **集合三个，都问**

答案：List、Set、Map，List可以有多个元素引用相同的对象，Set不可以

HashMap采用位桶（数组）+链表+红黑树实现，当链表长度大于阈值8的时候转换成红黑树（其实就是数组里面放链表，然后链表长度大于阈值8（小于64的时候扩容再转）的时候转换成链表）

1. **Final**
2. **项目如何处理的高并发场景**
3. **Spring boot自动装配原理**
4. **Springboot有哪些配置文件？yml(不是xml)?properties？application？**
5. **数据库三范式**

答案：第一范式：数据表中的每一列字段，必须是不可拆分的最小单元，确保每一列的原子性；第二范式：满足第一范式之后要求表中的所有列，都必须依赖主键，而不能有任何一列跟主键没有关系；第三范式：满足第二范式后，要求表中的每一列与主键直接相关，而不能是间接相关（表中的每一列只能依赖于主键）。

1. **Union和union all的区别**

答案：union：对两个结果集进行并集操作，不包括重复行，同时进行默认规则的排序，因为要进行重复行扫描，所以效率低，所以不需要刻意删除重复行的话，选用 union all

Union all:对两个结果集进行并集操作，包括重复行，不进行排序

1. **线程池？线程实现？项目中用到了？**
2. **二分查找手撕**
3. **单例模式线程手撕**
4. **大批数据存储**

答案：要分批多次进行操作；原因：1.大批量操作可能会造成严重的主从延迟，因为只有主库上完成操作才能在从库上执行写操作；2.blog日志为row格式时会产生大量的日志，大批量数据操作会产生大量的日志，尤其是row格式中会记录每一行数据的修改，产生的日志量就会越多，日志的传输和恢复需要的时间也就越长，也是导致主从延迟的原因；3.避免产生大事务操作，大批量修改数据，一定是在一个事务中进行的，会造成表中大量数据进行锁定，从而导致大量的阻塞，对性能影响非常大，长时间的阻塞会沾满数据库的可用连接，会使得其他应用无法连接到数据库，因此大批量数据存储一定要进行分批。

霍霞：

1. **string有几种？string、stringbuffer、stringbuilder的区别**

答案：可变性：string类中使用final关键字修饰字符数组来保存字符串，private final char[] value,所以string对象是不可变的，java9之后string类的实现改用byte数组来存储字符串private final byte[] value; StringBuilder和StringBuffer都继承自AbstractStringBuilder类，在AbstractStringBuilder类中也是用字符数组保存字符串，但是没有用final关键字修饰，所以这两个对象是可变的。

线程安全性：String对象是不可变的，线程安全；StringBuffer对方法加了同步锁或者对调用的方法加了同步锁，所以是线程安全的；StringBuilder并没有对方法加同步锁，所以不是线程安全的。

性能：每次对String进行改变的话，都会生成一个新的String对象，然后把指针指向新的String对象。StringBuffer每次都会对StringBuffer对象 本身进行操作，而不是生成新的对象并改变对象引用。相同情况下，StringBuilder相比StringBuffer仅能获取10%~15%的性能提升，但是要冒线程不安全的风险。

使用总结：

操作少量数据的时候使用String

单线程操作字符串缓冲区下操作大量数据使用StringBuilder

多线程操作字符串缓冲区下操作大量数据使用StringBuffer

1. **抽象类和普通类的区别**

答案：1.普通类可以被实例化，抽象类不能实例化；2.普通类和抽象类都可以被继承，但是抽象类被继承之后子类必须重写抽象方法，除非子类也是抽象方法；3.抽象方法必须为public或protected（因为如果为private，则不能被子类继承，子类无法实现该方法），缺省情况下为public；

1. **抽象类可以被final修饰吗**

答案：抽象类是用来被继承的，但final修饰关键字代表不可修改，不可继承的。

1. **讲一下线程和进程**

答案：线程：线程跟进程相似，但是线程是比进程更小的执行单位，一个进程在其执行的过程当中可以产生多个线程，与进程不同的是多个线程可以共享进程的堆和方法区资源，但每个线程的程序计数器、虚拟机栈和本地方法栈是私有的，所以系统在产生一个线程或是在各个线程之间切换工作时，负担要比进程小得多，线程是轻量级的进程。线程有6中不同的状态（new，runnable，blocked，waiting，time\_waiting，terminated）

进程：进程是程序的一次执行过程，是系统运行程序的基本单位，因此进程是动态的，系统运行一个程序即是一个进程从创建，运行到消亡的过程，在java中我们启动main函数其实就是启动了一个JVM进程，而main所在的线程是这个进程的主线程。各进程是独立的，每个进程还占有某些系统资源如CPU时间，内存空间，文件，输入输出设备的使用权限等等。

扩展：程序计数器为啥是私有的？答案：程序计数器的主要作用就是在多并发条件下，线程切换回来后能恢复到正确的执行位置，用于记录当前线程执行的位置。

虚拟机栈和本地方法栈私有？答案：虚拟机栈：每个java方法在执行的同时会创建一个栈帧用于存储局部变量表，操作数栈，常量池引用等信息，从方法调用到执行完成的过程，就对应着一个栈帧在java虚拟机中入栈和出栈的过程；本地方法栈：和虚拟机栈所发挥的作用非常相似，区别是虚拟机栈为虚拟机执行java方法服务，而本地方法栈为虚拟机用到的Native方法服务。为了保证线程中的局部变量不被别的线程访问到，所以是私有的；堆用于存放新创建的对象，方法区用于存放已被加载的类信息、常量、静态变量、即时编译器编译的代码等等

1. **创建线程的方法**
2. **Mybatis**
3. **#{} 和 ${}的区别**
4. **Spring AOP**

答案：Spring AOP分离业务逻辑和系统服务的高内聚开发，封装那些为业务逻辑代码所经常调用的系统服务（如日志管理、事务管理），避免重复造轮子，易于维护和扩展。

1. **给一个字符串，统计字母个数，按字母表顺序输出（A,a,B,b…）**

**答案：**

1. **倒序输出字符串**
2. **Sql题，到时候做几个**
3. **数组和ArrayList的区别**

答案：声明数组的时候就要初始化并确定其长度，长度不可变，而且只能存放同一个类型的数据，ArrayList，需要先声明，然后再添加数据，长度是根据内容的多少而改变，ArrayList可以存放不同类型的数据，比如

1. **HashMap和HashTable的区别**

答案：1.线程安全：HashMap非线程安全、HashTable安全（synchronized锁修饰），保证线程安全建议用ConcurrentHashMap；2.效率3.对Null Key和Null value的支持，HashMap可以有一个Null key,可以有多个Null value。HashTable不能有null key;4.初始容量大小和每次扩容的不同：创建时如果不指定容量初始值，HashTable的默认大小是11，之后每次扩容变为原来的2n+1,HashMap初始值16，之后每次扩充容量变为2n;如果给定了初始容量，HashTable直接用给定的大小，而HashMap会将其扩充为2的幂次方大小；5.底层数据结构：HashMap在解决哈希冲突的时候，当链表长度大于阈值8时（将链表转换成红黑树前会判断，如果当前数组的长度小于64，那么选择先进行扩容，而不是直接转为红黑树），将链表转为红黑树，以减少搜索时间。

1. **HashSet底层实现**

答案：HashSet底层是HashMap，除了clone（）、writeObject()、readObject()是HashSet自己实现的，其他都是直接用的HashMap的方法。

1. **ArrayList容量可以变吗**

\*\*同学：

1. **Java多态**

答案：多态是指程序中定义的引用变量所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序运行期间才确定。Java中有两种形式可以实现多态：继承（多个子类对同一个方法的重写）和接口（实现接口并覆盖接口中的同一方法）。

1. **接口和抽象类**

答案：1.接口的方法默认是public，所有方法在接口中不能有实现，而抽象类可以有非抽象的方法；

2.接口中除了static、final变量，不能有其他变量，而抽象类中则不一定

3.一个类可以实现多个接口，但只能实现一个抽象类，接口可以通过extends扩展多个接口

4.接口方法默认修饰符是public，抽象方法可以有public、protected、default这些修饰符（抽象方法是为了呗重写，所以不能有private关键字修饰）

5.抽象是对类的抽象，一种模板设计模式，而接口是对行为的抽象，行为规范

1. **Finally**

答案：无论是否捕获或处理异常，finally块里的语句都会被执行，当在try块或catch块中遇到return语句时，finally语句块将在方法返回之前被执行，但是在以下四种情况下finally块不会被执行：1.在finally语句块第一行发生了异常，因为在其他行finally块还是会被执行；2.在前面的代码中用了System.exit(int)退出程序，exit是带参函数，若该语句在异常语句之后，finally会执行；3.程序所在的线程已死亡；4.关闭CPU。注意当try和finally语句都有return时，在返回方法前，finally语句的内容被执行，并且return会被覆盖掉

1. **Try with resource知道吗**

答案：所有被打开过的系统资源（如流、文件、socket连接等），都需要被开发者手动关闭，否则随着程序的不断运行，资源泄露将会累积成重大的生产事故。之前都是在finally块中关闭资源，try with resource语法来打开资源，可以理解为是一个声明一个或多个资源的try语句（用分号隔开），try with resource语句确保在语句执行完毕后，每个资源都被自动关闭。

1. **Try能不能抛出异常**

答案：用于捕获异常，其后可以接0个或者多个catch块，如果没有catch块，必须接一个finally块

1. **JVM内存了解吗？介绍下**

答案：

1. **Java里面的流了解吗**

答案：JAVA IO流，1.按照流的流向分输入流和输出流；2.按照操作单元分为字节流和字符流；3.按照流的角色分为节点流和处理流。所有的类都是从以下四个类派生出来的：InputStream/OutputStream属于字节流（输入、输出），Reader/Writer属于字符流。

还可以继续问：既然有了字节流，为什么还要有字符流？答案：字符流是由java虚拟机将字节转换之后得到的，问题就出在这个过程非常耗时且容易出现乱码的问题，所以IO流干脆提供了一个操作字符的接口，方便操作，如果是音频、图片等媒体文件用字节流比较好，如果涉及到字符的话用字符流比较好。BIO：Blocking IO同步阻塞模型，数据的读取和写入必须阻塞在一个线程内完成，使用于活动链接数不是特别高（小于单机1000）的情况由socket和serversocket两种不同的套接字通道来实现；NIO：NON-Blocking IO同步非阻塞模型，他支持面向缓冲的，基于通道的IO操作方法，提供了socketchannel和serversocketchannel（跟BIO的对应）两种不同的套接字通道实现，两种通道都支持阻塞和非阻塞模式，适用于高并发、高负载的（网络）应用。AIO：Asynchronous IO异步非阻塞模型，异步IO是基于事件和回调机制实现的，当应用操作处理完成后直接返回，不会堵塞，当后台处理完成后，操作系统会通知相应的线程进行后续的操作。AIO目前应用不是很广泛，Netty尝试过，放弃了

1. **说下线程池执行任务的过程**

答案：线程池是为了减少每次获取资源的消耗，提高对资源的利用率；好处：降低资源消耗，提高响应速率，提高线程的可管理性。过程：

1. **连接复用了解吗**
2. **https的具体过程**
3. **https一定是安全的吗**
4. **TCP四次挥手，及其状态**
5. **TCP有大量的处于time wait状态的连接是怎么回事**
6. **Redis的底层数据结构**

答案：1.简单动态字符串2.链表3.字典4.跳跃表5.整数集合6.压缩列表7.对象

1. **Redis的参数**
2. **MySQL的索引**
3. **MySQL有哪些索引**

答案：哈希索引和BTree索引，哈希索引底层的数据结构是哈希表，因此在绝大多数需求为单条记录查询的时候，可以选择哈希索引，查询性能最快，其余场景都用BTree索引，BTree索引使用是B+Tree

1. **说下聚簇索引和非聚簇索引**

答案：B+Tree对于主要的两种存储引擎的实现方式不一样

MyISAM：B+Tree叶节点的data域存放的是数据记录的地址，在索引检索的时候，首先按照B+Tree搜索算法搜索索引，如果指定的key存在，则取出其中的data域的值，然后以data域的值作为地址读取相应的数据记录，称为非聚簇索引

InnoDB:其数据文件本身就是索引文件，相比MyISAM，数据文件和索引文件是相互分离的，其表数据文件本身就是按照B+Tree组织的索引结构，树的叶节点data域保存了完整的数据记录，这个索引的key是数据表的主键，因此数据文件本身就是主索引，称为聚簇索引；而其余的索引作为辅助索引，其data域存储相应记录主键的值而不是地址；在根据主索引搜索时，直接找到key所在的节点取出数据，根据辅助索引查找时，先取出主键的值，然后再走一遍主索引。不建议用过长的字段作为主索引，也不建议用非单调的字段作为主索引

1. **为什么用B+树**

答案：1.索引很大，不可能全部存在内存中，因此索引往往以索引文件的形式存在磁盘上

2.索引的结构组织要尽量减少查找过程中磁盘I/O的存取次数

1. **Spring security**
2. **Spring的常用注解**
3. **@Autowired 和@Resource的区别**
4. **Spring中bean的生命过程**
5. **Spring的AOP**
6. **AOP中的动态代理**

答案：如果方法实现了接口，就用JDK动态代理

如果没有实现接口，就用CGLib动态代理

1. **Spring boot了解吗**