1. **简述&和&&的区别？**

   &&是逻辑操作符，而&是位操作符。当&&和&作为逻辑操作符时，&表示当运算符两边的表达式的结果都为 true 时，整个运算结果才为 true，否则，只要有一方为 false，则结果为 false。&&还具有短路的功能，即如果第一个表达式为 false，则不再计算第二个表达式。&还可以用作位运算符，当&操作符两边的表达式不是 boolean 类型时，&表示按位与操作。

2. **==和 equals 的区别？**

   它的作用是判断两个对象的地址是不是相等。即，判断两个对象是不是同一个对象(基本数据类型 == 比较的值，引用数据类型 == 比较的是内存地址。String 中的 equals 方法是被重写过的，因为 object 的 equals 方法是比较的对象的内存地址，而 String 的 equals 方法比较的是对象的值。

3. **i++ 和 ++i 的区别？**

   i++：是先把 i 拿出来使用，然后再+1；

   ++i ：是先把 i+1，然后再拿出来使用；

4. **switch 语句能否作用在 byte 上，能否作用在 long 上，能否作用在 String 上？**

   int 基本类型或 Integer 包装类型，由于 byte,short,char 都可以隐含转换为 int，所以这些类型以及这些类型的包装类型也是可以的。显然 long 和 String 类型都不符合 switch 语法的规定，并且不能隐式的转换成 int 类型，所以它们不能作用于 switch 语句中。JDK1.7 之后支持 Sring。

5. **String 为什么是不可变类？**

   String 的成员变量是 private final 的。不可变的 String 类是安全的，多线程的，高性能的

6. **String,StringBuffer 与 StringBuilder 的区别？**

   String 的值是不可变的。 StringBuilder 和 StringBuffer 代表可变字符串对象。不同的是：StringBuffer 是线程安全的，而 StringBuilder 则没有实现线程安全功能，所以性能略高。

7. **简述你对面向对象思想的理解？**

   面向对象是向现实世界模型的自然延伸，这是一种“万物皆对象”的编程思想。在现实生活中的任何物体都可以归为一类事物，而每一个个体都是一类事物的实例。面向对象的三大特征是继承，封装，多态。封装隐蔽了对象内部不需要暴露的细节,使得内部细节的变动跟外界脱离,只依靠接口进行通信.封装性降低了编程的复杂性，使得代码的复用性更高。通过继承,使得新建一个类变得容易,一个类从派生类那里获得其非私有的方法和公用属性的繁琐工作交给了编译器. 而 继承和实现接口和运行时的类型绑定机制 所产生的多态,使得不同的类所产生的对象能够对相同的消息作出不同的反应,极大地提高了代码的通用性.

8. **重载与重写的区别？**

   重载(Overload):发生在本类,方法名相同,参数列表不同,与返回值无关,只和方法名,参数列表,参数的类型有关.是位于一个类之中或者其子类中，具有相同的方法名，但是方法的参数不同，返回值类型可以相同也可以不同。方法名必须相同，方法的参数列表一定不一样。访问修饰符和返回值类型可以相同也可以不同。

   重写（override）：一般都是表示子类和父类之间的关系，其主要的特征是：方法名相同，参数相同，但是具体的实现不同。

9. **Collection 和 Collections 的区别及各自用途。**

   collection 是集合类的上层接口。本身是一个 Interface，里面包含了一些集合的基本操作,是 Set 接口和 List 接口的父接口。Collections 是一个集合框架的帮助类，里面包含一些对集合的排序，搜索以及序列化的操作。

10. ArrayList 和 Vecotor 的区别。

Vector 是多线程安全的，而 ArrayList 不是。两个都是采用的线性连续空间存储元素，但是当空间不足的时候，两个类的增加方式是不同的，Vector 增加原来空间的一倍，ArrayList 增加原来空间的 50%。Vector 可以设置增长因子，而 ArrayList 不可以。

11. Throw 和 throws 的区别。

throw 代表动作，表示抛出一个异常的动作；throws 代表一种状态，代表方法可能有异常抛出。throw 用在方法实现中，而 throws 用在方法声明中。throw 只能用于抛出一种异常，而 throws 可以抛出多个异常。

12. Final，finalize,finally 有什么区别？

final 表示不可修改的，可以用来修饰类，方法，变量。final 修饰 class 表示该 class 不可以被继承。final 修饰方法表示方法不可以被 overrride（重写）。final 修饰变量表示变量是不可以修改。finally 是 Java 的异常处理机制中的一部分。finally 块的作用就是为了保证无论出现什么情况，finally 块里的代码一定会被执行。finalize 是 Object 类的一个方法，是 GC 进行垃圾回收前要调用的一个方法。

13. Java 中有几种类型的流？

字节流 InputStream/OutputStream。

FileInputStream/FileOutputStream：文件字节流，用于文件的读写操作。

BufferedInputStream/BufferedOutputStream：加缓冲区的字节流，用于提高效率。

字符流 Reader/Writer

FileReader/FileWriter：文件字符流，用于文本文件的读写操作

BufferedReader/BufferedWriter：加缓冲区的字符流，用于提高效率。

转换流 InputStreamReader/OutputStreamWriter

14. 字节流与字符流的区别？各自应用场景。

字符流处理的单元为 2 个字节的 Unicode 字符，分别操作字符、字符数组或字符串，而字节流处理单元为 1 个字节，操作字节和字节数组。Java 内用 Unicode 编码存储字符，字符流处理类负责将外部的其他编码的字符流和 java 内 Unicode 字符流之间的转换。而类 InputStreamReader 和 OutputStreamWriter 处理字符流和字节流的转换。字符流（一次可以处理一个缓冲区）一次操作比字节流（一次一个字节）效率高。

15. Java 序列化是什么？什么情况下使用序列化？如何实现序列化？

Java 序列化是指将 Java 对象转换成字节流的过程。当 Java 对象需要在网络上传输 或者 持久化存储到文件中时，就需要对 Java 对象进行序列化处理。序列化的实现：类实现 Serializable 接口，这个接口没有需要实现的方法。实现 Serializable 接口是为了告诉 jvm 这个类的对象可以被序列化。

16. 事务的四大特性。

事务是指是程序中一系列严密的逻辑操作，而且所有操作必须全部成功完成，否则在每个操作中所作的所有更改都会被撤消。原子性（Atomicity）一致性（Consistency）隔离性（Isolation）持久性（Durability）

17. 线程和进程的区别。

进程是操作系统资源分配的基本单位，而线程是任务调度和执行的基本单位。进程有自己的独立地址空间，而线程是共享进程中的数据的，使用相同的地址空间。线程之间的通信更方便，进程之间需要以 IPC 通信方式进行。

18. Wait 和 sleep 有什么区别？

wait()方法则是指当前线程让自己暂时退让出同步资源锁，以便其他正在等待该资源的线程得到该资源进而运行，只有调用了 notify()方法，之前调用 wait()的线程才会解除 wait 状态，可以去参与竞争同步资源锁，进而得到执行。sleep()方法可以在任何地方使用；wait()方法则只能在同步方法或同步块中使用；sleep()是线程类（Thread）的方法，调用会暂停此线程指定的时间，但监控依然保持，不会释放对象锁，到时间自动恢复；wait()是 Object 的方法，调用会放弃对象锁，进入等待队列，待调用 notify()/notifyAll()唤醒指定的线程或者所有线程，才会进入锁池，不再次获得对象锁才会进入运行状态；

19. 阐述 java 垃圾回收机制及工作流程。

JVM 实现了 java 垃圾的自动回收机制。自动垃圾回收是一种在堆内存中找出哪些对象在被使用，还有哪些对象没被使用，并且将后者删掉的机制。在 JVM 中，new 出来的对象会被放在堆区。将对象实例运行数据放在栈区。堆区分为“年轻代”，“年老代”，“持久代”3 个区域。“年轻代”又分为伊甸区（刚出生的地方）和 2 个缓冲区。刚 new 出来的对象一般先放到伊甸区。持久代中存放的是 java 类 import 的类信息。垃圾回收主要涉及年轻代和年老代。年轻代中的回收属于轻量级回收，年老代和持久代存满会触发重量级（FULL GC）。通过判断对象上有没有强引用来确定是否回收，JDK 早期使用‘引用计数法’，后期使用‘根搜索算法’。程序员也可以手动调用 System.gc() 来启动 FULL GC。

20. 单例模式的实现。

```
public class Singleton {
    private final static Singleton INSTANCE = null;

    private Singleton() {}

    public static Singleton getInstance() {
        return new Singleton();
    }
  }
```

21. map 底层实现原理。

底层使用数组实现，数组中每一项是个单向链表，即数组和链表的结合表。当链表长度大于一定阈值（默认 8）时，链表转换为红黑树，这样减少链表查询时间。HashMap 在底层将 key-value 当成一个整体进行处理，这个整体就是一个 Node 对象。HashMap 底层采用一个 Node[]数组来保存所有的 key-value 对，当需要存储一个 Node 对象时，会根据 key 的 hash 算法来决定其在数组中的存储位置，在根据 equals 方法决定其在该数组位置上的链表中的存储位置；

22. 事务级别。

READ UNCOMMITTED（读未提交数据）： 允许事务读取未被其他事务提交的变更数据，会出现脏读、不可重复读和幻读问题。

READ COMMITTED（读已提交数据）： 只允许事务读取已经被其他事务提交的变更数据，可避免脏读，仍会出现不可重复读和幻读问题。（oracle 数据库默认）

REPEATABLE READ（可重复读）： 确保事务可以多次从一个字段中读取相同的值，在此事务持续期间，禁止其他事务对此字段的更新，可以避免脏读和不可重复读，仍会出现幻读问题。（Mysql InnoDB 引擎默认）

SERIALIZABLE（序列化）： 确保事务可以从一个表中读取相同的行，在这个事务持续期间，禁止其他事务对该表执行插入、更新和删除操作，可避免所有并发问题，但性能非常低。

23. 垂直分表和水平分表的区别？

垂直分表：适用于一个表有很多个字段的情况，将原先一个表中的某些字段拆出来，单独放到一个或者多个表。

水平分表：根据具体的切分规则，将数据划分到不同的表上面。

24. arraylist 和 linkedlist 的底层实现

ArrayList 默认初始容量为 10,基于 1.5 倍动态扩大容量。

ArrrayList 底层的数据结构是数组，支持随机访问，而 LinkedList 的底层数据结构是双向循环链表。

25. HashMap 与 Hashtable 的区别？

Hashtable 是 java 一开始发布时就提供的键值映射的数据结构，而 HashMap 产生于 JDK1.2，两个作者不同。

继承的父类不同。HashMap 是继承自 AbstractMap 类，而 HashTable 是继承自 Dictionary 类。不过它们都实现了同时实现了 map、Cloneable（可复制）、Serializable（可序列化）这三个接口。

Hashtable 既不支持 Null key 也不支持 Null value。Hashtable 的 put()方法的注释中有说明。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。当 get()方法返回 null 值时，可能是 HashMap 中没有该键，也可能使该键所对应的值为 null。

Hashtable 是线程安全的，它的每个方法中都加入了 Synchronize 方法。HashMap 不是线程安全的，在多线程并发的环境下，可能会产生死锁等问题。虽然 HashMap 不是线程安全的，但是它的效率会比 Hashtable 要好很多。

初始容量大小和每次扩充容量大小的不同。Hashtable 默认的初始大小为 11，之后每次扩充，容量变为原来的 2n+1。HashMap 默认的初始化大小为 16。之后每次扩充，容量变为原来的 2 倍。

计算 hash 值的方法不同。Hashtable 比 HashMap 多提供了 elments() 和 contains() 两个方法。

26. ConcurrentHashMap 的并发度是什么？

ConcurrentHashMap 的并发度就是 segment 的大小，默认为 16，这意味着最多同时可以有 16 条线程操作 ConcurrentHashMap，这也是 ConcurrentHashMap 对 Hashtable 的最大优势。

27. String,StringBuffer,StringBuilder

String:（不可变字符对象）因为 String 的值是不可变的（final 修饰的），这就导致每次对 String 的操作都会生成新的 String 对象，这样不仅效率低下，而且大量浪费有限的内存空间。

StringBuffer: （可变字符对象、效率低、线程安全）每个 StringBuffer 对象都有一定的缓冲区容量，当字符串大小没有超过容量时，不会分配新的容量，当字符串大小超过容量时，会自动增加容量。

StringBuilder:（可变字符对象、效率高、线程不安全）

28. java 反射机制原理

反射就是把 java 类中的各种成分映射成一个个的 Java 对象 例如：一个类有：成员变量、方法、构造方法、包等等信息，利用反射技术可以对一个类进行解剖，把各个组成部分映射成一个个对象。运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意方法和属性；

29. autowired 与 resource 区别

autowired 默认根据类型装配 bytype 与 Qualifier 联合只用可按名称装配

resource 默认根据名称匹配 byname 指定 type 时按类型装配

@Autowired 能够用在：构造器、方法、参数、成员变量和注解上，而@Resource 能用在：类、成员变量和方法上。

@Autowired 是 spring 定义的注解，而@Resource 是 JSR-250（JDK 底层）定义的注解。

30. java8 新特性

Lambda 表达式　Stream API　Date Time API　Option 封装类

31. 线程 run 和 start 区别

start 方法来启动线程，真正实现了多线程运行，这时无需等待 run 方法体代码执行完毕而直接继续执行下面的代码。通过调用 Thread 类的 start()方法来启动一个线程，这时此线程处于就绪（可运行）状态，并没有运行，一旦得到 cpu 时间片，就开始执行 run()方法，这里方法 run()称为线程体，它包含了要执行的这个线程的内容，run 方法运行结束，此线程随即终止。

run()方法只是类的一个普通方法而已，如果直接调用 Run 方法，程序中依然只有主线程这一个线程，其程序执行路径还是只有一条，还是要顺序执行，还是要等待 run 方法体执行完毕后才可继续执行下面的代码，这样就没有达到写线程的目的。

32. http 状态码

200 - 请求成功
400 -客户端请求的语法错误，服务器无法理解
401 -Unauthorized　请求要求用户的身份认证
403 -服务器理解请求客户端的请求，但是拒绝执行此请求
404 - 请求的资源（网页等）不存在
500 - 内部服务器错

33. 为什么使用数据索引能提高效率?

数据索引的存储是有序的。

在有序的情况下，通过索引查询一个数据是无需遍历索引记录的。

极端情况下，数据索引的查询效率为二分法查询效率，趋近于 log2。

34. MySQL 索引使用的注意事项?

更新频繁的列不要加索引。

数据量小的表不要重复

数据多的字段不要加索引,比如性别字段。

首先应该考虑对 where 和 order by 涉及的列上建立索引。

35. 索引的优缺点

优点：

大大加快数据的检索速度；

创建唯一性索引，保证数据库表中每一行数据的唯一性；

加速表和表之间的连接；

在使用分组和排序子句进行数据检索时，可以显著减少查询中分组和排序的时间。

缺点：

索引需要占物理空间；

当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，降低数据的维护速度。

36. 数据库索引失效

1) 没有查询条件，或者查询条件没有建立索引

2) 在查询条件上没有使用引导列（没用到对应的列）

3) 查询的数量是大表的大部分，应该是 30%以上。

4) 索引本身失效 （名称不匹配或没有启用）

5)隐式转换导致索引失效.这一点应当引起重视.也是开发中经常会犯的错误. 由于表的字段 tu_mdn 定义为 varchar2(20),

6)使用内部函数导致索引失效

37. 数据库分页语句

MYSQL select *from table limit start,pageNum

oracle select *from (select rownum from table where rownum <=endIndex) where rownum> startIndex

38. hibernate 运行原理。

hibernate 里面提供了 3 个核心接口

Configuration、SessionFactory、Session

1、hibernate 启动的时候利用 Configuration 读取 xml 配置文件

2、通过配置文件创建 SessionFactory 对象，初始化 hibernate 基本信息

3、获取 session 然后调用 CRUD 方法进行数据操作，hibernate 会把我们的数据进行三种状态的划分，然后根据状态进行管理我们的数据，对应的发送 SQL 进行数据操作

4、关闭 session，如果有事务的情况下，需要手动获取事务并开启，然后事务结束后提交事务。

5、在提交事务的时候，去验证我们的快照里面的数据和缓存数据是否一致，如果不一致，发送 SQL 进行修改。

39. 什么是 Mybatis 及其优点？

Mybatis 是一个半 ORM（对象关系映射）框架，它内部封装了 JDBC，开发时只需要关注 SQL 语句本身，不需要花费精力去处理加载驱动、创建连接、创建 statement 等繁杂的过程。程序员直接编写原生态 sql，可以严格控制 sql 执行性能，灵活度高。

基于 SQL 语句编程，相当灵活，不会对应用程序或者数据库的现有设计造成任何影响，SQL 写在 XML 里，解除 sql 与程序代码的耦合，便于统一管理；提供 XML 标签，支持编写动态 SQL 语句，并可重用，能够与 Spring 很好的集成；提供映射标签，支持对象与数据库的 ORM 字段关系映射；提供对象关系映射标签，支持对象关系组件维护。与 JDBC 相比，减少了 50%以上的代码量，消除了 JDBC 大量冗余的代码，不需要手动开关连接；能很好的与各种数据库兼容。

40. MyBatis 与 Hibernate 有哪些不同？

Mybatis 直接编写原生态 sql，可以严格控制 sql 执行性能，灵活度高，非常适合对关系数据模型要求不高的软件开发，因为这类软件需求变化频繁，一但需求变化要求迅速输出成果。但是灵活的前提是 mybatis 无法做到数据库无关性，如果需要实现支持多种数据库的软件，则需要自定义多套 sql 映射文件，工作量大。Hibernate 对象/关系映射能力强，数据库无关性好，对于关系模型要求高的软件，如果用 hibernate 开发可以节省很多代码，提高效率。

41. #{}和${}的区别是什么？

#{}是预编译处理，${}是字符串替换。Mybatis 在处理#{}时，会将 sql 中的#{}替换为?号，调用 PreparedStatement 的 set 方法来赋值；Mybatis 在处理${}时，就是把${}替换成变量的值。使用#{}可以有效的防止 SQL 注入，提高系统安全性。

42. Spring 是什么?

Spring 是一个轻量级的 IoC 和 AOP 容器框架。是为 Java 应用程序提供基础性服务的一套框架，目的是用于简化企业应用程序的开发，它使得开发者只需要关心业务需求。常见的配置方式有三种：基于 XML 的配置、基于注解的配置、基于 Java 的配置。

IOC 就是控制反转，是指创建对象的控制权的转移，以前创建对象的主动权和时机是由自己把控的，而现在这种权力转移到 Spring 容器中，并由容器根据配置文件去创建实例和管理各个实例之间的依赖关系，对象与对象之间松散耦合，也利于功能的复用。DI 依赖注入，和控制反转是同一个概念的不同角度的描述，应用程序在运行时依赖 IoC 容器来动态注入对象需要的外部资源。

43. Spring 注入 bean 的几种方式?

Set 方法注入。构造器注入：①通过 index 设置参数的位置，②通过 type 设置参数类型。

静态工厂注入。

44. SpringMVC 的流程?

用户发送请求至前端控制器 DispatcherServlet;

DispatcherServlet 收到请求后，调用 HandlerMapping 处理器映射器，请求获取 Handle;

处理器映射器根据请求 url 找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给 DispatcherServlet;

DispatcherServlet 调用 HandlerAdapter 处理器适配器;

HandlerAdapter 经过适配调用 具体处理器(Handler，也叫后端控制器);

Handler 执行完成返回 ModelAndView;

HandlerAdapter 将 Handler 执行结果 ModelAndView 返回给 DispatcherServlet;

DispatcherServlet 将 ModelAndView 传给 ViewResolver 视图解析器进行解析;

ViewResolver 解析后返回具体 View;

DispatcherServlet 对 View 进行渲染视图（即将模型数据填充至视图中）

DispatcherServlet 响应用户。

45. spring 循环依赖是怎么解决的?

第一级缓存（也叫单例池）：Map<String, Object> singletonObjects，存放已经经历了完整生命周期的 Bean 对象。

第二级缓存：Map<String, Object> earlySingletonObjects，存放早期暴露出来的 Bean 对象，Bean 的生命周期未结束（属性还未填充完）。

第三级缓存：Map<String, ObjectFactory<?>> singletonFactories，存放可以生成 Bean 的工厂。

Spring 管理的 Bean 其实默认都是单例的，也就是说 Spring 将最终可以使用的 Bean 统一放入第一级缓存中，也就是 singletonObjects（单例池）里，以后凡是用到某个 Bean 了都从这里获取就行了

46. spring 自动配置是如何实现的

Spring Boot 启动的时候会通过 @EnableAutoConfiguration 注解找到 META-INF/spring.factories 配置文件中的所有自动配置类，并对其进行加载，而这些自动配置类都是以 AutoConfiguration 结尾来命名的，它实际上就是一个 JavaConfig 形式的 Spring 容器配置类,它能通过以 Properties 结尾命名的类中取得在全局配置文件中配置的属性如:server.port，而 XxxxProperties 类是通过@ConfigurationProperties 注解与全局配置文件中对应的属性进行绑定的。

47. spring bean 作用域

- singleton 在 Spring 容器中仅存在一个 Bean 实例， Bean 以单例的形式存在。
- prototype 每次从容器中调用 Bean 时，都会返回一个新的实例，即相当于执行 new

XxxBean() 的实例化操作。

- request 每次 http 请求都会创建一个新的 Bean ， 仅用于 WebApplicationContext 环境。
- session 同一个 http Session 共享一个 Bean ，不同的 http Session 使用不同的 Bean，仅用于 WebApplicationContext 环境。
- globalSession 同一个全局 Session 共享一个 bean, 用于 Porlet, 仅用于 WebApplication 环境。

48. Spring Boot 的优点是什么?

减少开发、测试的时间和工作量。
使用 JavaConfig 有助于避免使用 XML。
避免大量 maven 导入和各种版本冲突。
提供可选的开发方法。
通过提供默认开发方式进行快速开发。
不需要单独的 Web 服务器。
由于没有 web.xml 文件，所以需要更少的配置。

49. 什么是 springcloud 及其优点？

Spring cloud 流应用程序启动器是基于 Spring Boot 的 Spring 集成应用程序，提供与外部系统的集成。

每个服务直接足够内聚，代码容易理解
开发效率高，一个服务只做一件事，适合小团队开发
松耦合，有功能意义的服务。
可以用不同语言开发，面向接口编程。
易于第三方集成
微服务只是业务逻辑的代码，不会和 HTML,CSS 或其他界面结合.
可以灵活搭配，连接公共库/连接独立库