**1. CS与BS架构区别？**

Client/Server是建立在局域网的基础上的。

Browser/Server是建立在广域网的基础上的。



B/S最大的优点就是可以在任何地方进行操作而不用安装任何专门的软件。只要有一台能上网的电脑就能使用，客户端零维护。

C/S的优点是：能充分发挥客户端PC的处理能力，很多工作可以在客户端处理后再提交给服务器，从而响应速度快。

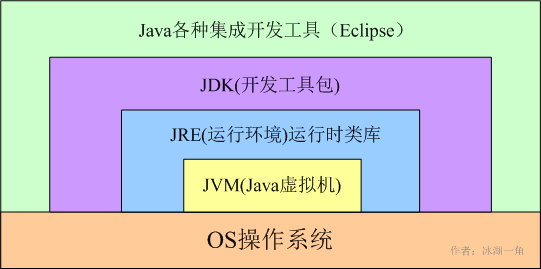
**2.****JavaSE、JavaEE、JavaME？**

Java SE 是Java平台标准版的简称（Java Platform Standard Edition），用于开发和部署桌面、服务器以及嵌入设备和实时环境中的Java应用程序。Java SE包括用于开发Java Web服务的类库，同时，Java SE为Java EE和Java ME提供了基础。Java SE就是基于JDK和JRE的。

Java EE是Java 平台企业版（Java Platform Enterprise Edition），Java EE 可以说是一个框架，也可以说是一种规范，是对 JavaSE 的扩展，增加了一些更加便捷的应用框架。

Java ME是Java微版的简称（Java Platform,Micro Edition），它为移动设备提供了基于Java环境的开发与应用的平台。

**3. JDK、JRE、JVM三者间的联系与区别？**



JDK(Java SE Development Kit)，Java标准开发包，它提供了编译、运行Java程序所需的各种工具和资源，包括Java编译器、Java运行时环境，以及常用的Java类库等。

JRE( Java Runtime Environment) 是Java运行环境，用于解释执行Java的字节码文件。普通用户而只需要安装 JRE来运行 Java 程序。而程序开发者必须安装JDK来编译、调试程序。

JVM(Java Virtual Mechinal)是Java虚拟机，是JRE的一部分。它是整个java实现跨平台的最核心的部分，负责解释执行字节码文件，是可运行java字节码文件的虚拟计算机。所有平台的上的JVM向编译器提供相同的接口，而编译器只需要面向虚拟机，生成虚拟机能识别的代码，然后由虚拟机来解释执行。

**4.Java语言的优点有哪些？**

Java为纯面向对象的语言，开发效率高，易维护，易扩展。

平台无关性，Java语言可以一次编译到处运行，编译后的代码可以在多种平台上运行。由于Java是解释性语言，编译器会把Java代码编译成中间代码，然后在JVM中运行，因此Java可以跨平台运行，具有良好的可移植性。

为完成特定任务内置的多种库。通过使用这些库，简化了开发者的开发难度。

提供了对WEB应用开发的支持。

具有良好的安全性和健壮性。Java的强类型机制，垃圾回收器，异常处理，安全检查机制等都使得Java编写的程序有着良好的健壮性。

去除了C++语言中难以理解，容易混淆的特性，例如：指针、结构、单元、运算符重载、虚拟基础类、多重继承等等，使得程序更加严谨简洁。

**5.****面向对象和面向过程的区别？**

如果要解决问题就需要将问题分解，然后分成很多步骤。如果把每个步骤比作一个点，顺序连接每个点则是一条线，过程就是这条线。所以面向过程侧重的是这条线。这也是大多数人的第一反应。

还是同样的问题，如果我要用面向对象的方法来解决。那我首先会考虑解决一个问题都需要哪些人，哪些工具，物品等。然后再创建人，工具，物品，让他们提供合适的方法。最后再根据我要解决问题的思路来让每个类调用它的接口。这个过程中每个类都是独立的，所以它是一个一个的点。面向过程侧重线，面向对象侧重点。

面向过程的优点：性能比面向对象高，因为类调用时需要实例化，new一个对象是存储在堆里的，我们通过栈中的引用来使用这些对象，开销比较大，比较消耗资源。

面向对象的优点：由于面向对象有封装、继承、多态性的特性，可以设计出低耦合的系统，使系统更加灵活、更加易于扩展和维护。

**6.什么是数据结构？**

数据结构是计算机存储、组织数据的方式。

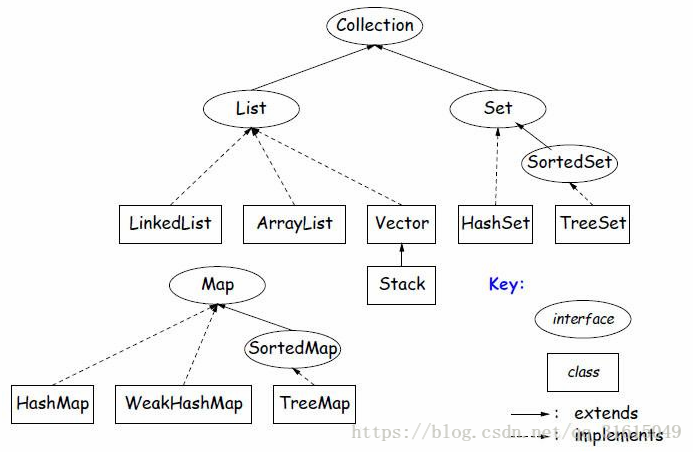
数据结构是指相互之间存在一种或多种特定关系的数据元素的集合。

通常情况下，精心选择的数据结构可以带来更高的运行或者存储效率。

逻辑结构：是指数据对象中数据元素之间的相互关系。包括集合结构（同属于一个集合）、线性结构（一对一）、树形结构（一对多）、图形结构（多对多）。

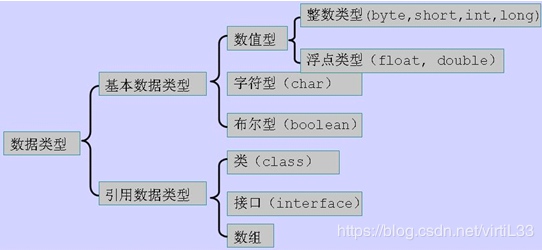
物理结构：是指数据的逻辑结构在计算机中的存储形式。顺序存储和链式存储。

**7.Java的常用数据结构有哪些？**



Collections是集合类的一个帮助类，它包含有各种有关集合操作的静态多态方法，用于实现对各种集合的搜索、排序、线程安全化等操作。此类不能实例化，就像一个工具类，服务于Java的Collection框架。

**8.****Java中的数据类型？**



**9. Java标识符的命名规则与规范？**

应以字母、下划线、美元符开头；

后跟字母、下划线、美元符或数字；

Java标识符大小写敏感，长度无限制。

类名首个字母必须大写，多个单词组成的，每个单词首字母都要大写。

方法名、变量名一般首个字母小写（构造方法例外），多个单词组成方法名，后面单词首字母大写。

**10.instanceof关键字的作用？**

instanceof 运算符是用来在运行时判断对象是否是指定类及其父类的一个实例。

**11.****Java中的数据类型转换？**

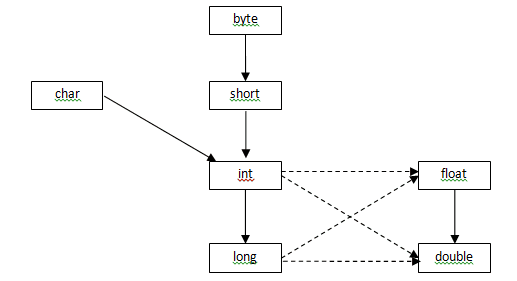


java中整数类型默认的int类型，小数类型默认的double。

String的字符串转化为char类型时，只能用char类型的数组来接收。

自动类型转换是指：数字表示范围小的数据类型可以自动转换成范围大的数据类型。

类型提升就是指在多种不同数据类型的表达式中，类型会自动向范围表示大的值的数据类型提升。



实线表示自动转换时不会造成数据丢失，虚线则可能会出现数据丢失问题。

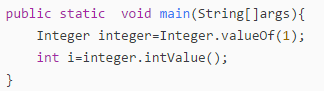
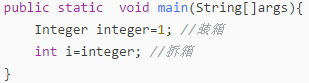
向下转换时可以直接将 int 常量字面量赋值给 byte、short、char 等数据类型，而不需要强制转换，只要该常量值不超过该类型的表示范围都能自动转换。

针对浮点型数据运算出现的误差的问题，可以使用Bigdecimal类进行浮点型数据的运算。

**12.Java中的拆装箱？**

除了Integer和Character类以外，其它六个类的类名和基本数据类型一致，只是类名的第一个字母大写即可。

Integer i=10 可以替代 Integer i = new Integer(10);，这就是因为Java帮我们提供了自动装箱的功能，不需要开发者手动去new一个Integer对象。



自动装箱都是通过包装类的valueOf()方法来实现的，自动拆箱都是通过包装类对象的xxxValue()来实现的。

**13.****面向对象的特征有哪些方面？**

面向对象的本质就是将现实世界描绘成一系列完全自治、封闭的对象。

抽象：就是把现实世界中的某一类东西，提取出来，用程序代码表示，抽象出来的一般叫做类或者接口。抽象包括两个方面，一个数据抽象(类的属性)，二是行为抽象(类的方法)。

封装：封装就是把过程和数据包围起来，对数据的访问只能通过已定义的界面。如私有变量，用set，get方法获取。封装保证了模块具有较好的独立性，使得程序维护修改较为容易。

继承：对象的一个新类可以从现有的类中派生，这个过程称为类继承。派生类可以从它的父类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。因此可以说，继承为了重用父类代码，同时为实现多态性作准备。

多态：简单的说就是用同样的对象引用调用同样的方法但是做了不同的事情，也很好的解决了应用程序函数同名问题。多态性分为编译时的多态性和运行时的多态性。总的来说，方法的重写（运行时的多态性），重载（编译时的多态性）与动态链接构成多态性。java引入多态的概念原因之一就是弥补类的单继承带来的功能不足的缺陷。

**14.****访问修饰符？**



Java中，外部类的修饰符只能是public或默认，类的成员（包括内部类）的修饰符可以是以上四种。

**15.****Java中重载和重写的区别？**

方法的重载和重写都是实现多态的方式，区别在于前者实现的是编译时的多态性，而后者实现的是运行时的多态性。重载发生在一个类中，同名的方法如果有不同的参数列表（参数类型不同、参数个数不同或者二者都不同）则视为重载；重写发生在子类与父类之间，重写要求子类被重写方法与父类被重写方法有相同的参数列表，有兼容的返回类型，比父类被重写方法更好访问，不能比父类被重写方法声明更多的异常（里氏代换原则）。重载对返回类型没有特殊的要求，不能根据返回类型进行区分。

**16.程序的结构有哪些？**

顺序结构、选择结构、循环结构

**17.数组实例化有几种方式？**

数组没有length()这个方法，有length的属性。String有length()这个方法。

数组一旦实例化，它的长度就是固定的。

静态实例化：创建数组的时候已经指定数组中的元素。

动态实例化：实例化数组的时候，只指定了数组长度，数组中所有元素都是数组类型的默认值。

**18.Java中各种数据的默认值？**

Byte，short，int，long默认是都是0；

boolean默认值是false ；

char类型的默认值是空字符；

float与double类型的默认是0.0；

对象类型的默认值是null。

**19.Java中的值传递和引用传递？**

值传递和引用传递的区别并不是传递的内容，而是实参到底有没有被复制一份给形参。

理论上说，java都是值传递，对于基本数据类型，传递是值的副本，而不是值本身。对于对象类型，传递是对象的引用的副本，当在一个方法操作参数的时候，其实操作的是引用所指向的对象。

**20.Java中的构造方法？**

构造函数是一种特殊的函数。其主要功能是用来在创建对象时初始化对象， 即为对象成员变量赋初始值，总在创建对象的语句中与new运算符一起使用。构造函数与类名相同，可重载多个不同的构造函数。

**21.Java中的内部类？**

普通内部类：普通内部类对象依赖外部类对象而存在，即在创建一个普通内部类对象时首先需要创建其外部类对象。内部类对象可以访问外部类对象中所有访问权限的字段，同时，外部类对象也可以通过内部类的对象引用来访问内部类中定义的所有访问权限的字段。

静态内部类：一个类的静态成员独立于这个类的任何一个对象存在，只要在具有访问权限的地方，我们就可以通过“类名.静态成员名”的形式来访问这个静态成员。在静态内部类中无法直接访问外部类中变量、方法，如果要访问的话，必须要new一个外部类的对象，使用new出来的对象来访问，但是可以直接访问静态的变量、调用静态的方法。

匿名内部类：在匿名内部类中可以使用外部类的属性，但是外部类却不能使用匿名内部类中定义的属性，因为是匿名内部类，因此在外部类中无法获取这个类的类名，也就无法得到属性信息。匿名内部类在实现时必须借助一个接口或者一个抽象类或者一个普通类来构造，从这过层次上讲匿名内部类是实现了接口或者继承了类，但是不能通过extends或implement关键词来继承类或实现接口。

局部内部类：局部内部类使用的比较少，其声明在一个方法体或一段代码块的内部，而且不在定义类的定义域之内便无法使用。

**22.static关键字有什么作用？**

static可以修饰内部类、方法、变量、代码块；

一个类的静态成员独立于这个类的任何一个对象存在，只要在具有访问权限的地方，我们就可以通过“类名.静态成员名”的形式来访问这个静态成员。

静态资源是类初始化的时候加载的，而非静态资源是类new的时候加载的。类的初始化早于类的new，比如Class.forName(“xxx”)方法，就是初始化了一个类，但是并没有new它，只是加载这个类的静态资源罢了。

静态块里面的代码只执行一次，且只在初始化类的时候执行。静态代码块对于定义在它之后的静态变量，可以赋值，但是不能访问。

**23.****Java中final关键字有什么作用？**

被ﬁnal修饰的类不可以被继承；

被ﬁnal修饰的方法不可以被重写，JVM会尝试将其内联,以提高运行效率；

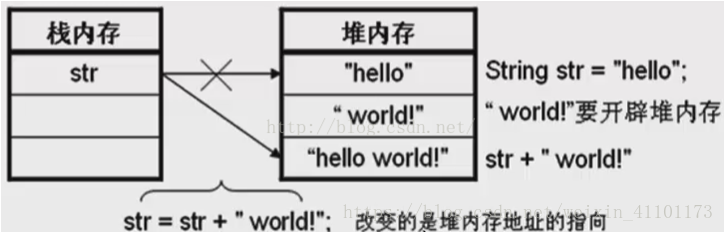
被ﬁnal修饰的变量不可以被改变.如果修饰引用,那么表示引用不可变,引用指向的内容可变；

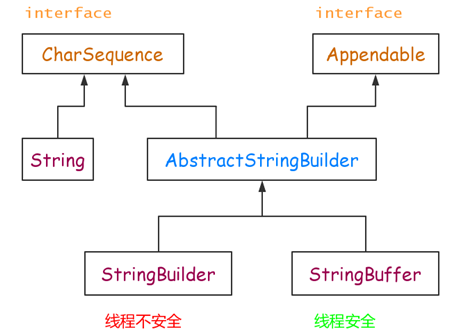
final成员变量必须在声明的时候初始化或者在构造器中初始化，否则就会报编译错误；

被ﬁnal修饰的常量，在编译阶段会存入常量池中。

**24.** **String 、StringBuffer 和 StringBuilder 的区别是什么？**

String的值是不可变的，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 和 StringBuilder 类的对象能够被多次的修改，并且不产生新的未使用对象。由于 StringBuilder 相较于 StringBuffer 有速度优势，所以多数情况下建议使用 StringBuilder 类。然而在应用程序要求线程安全的情况下，则必须使用 StringBuffer 类。

“String a = "aaa";”用这种方式的时候java首先在内存中寻找"aaa"字符串，如果有，就把aaa的地址给它，如果没有则创建。“String a = new String("aaa");”是不管内存中有没有"aaa"都开辟一块新内存保存它。

**25. super与this的使用？**

this是自身的一个对象，代表对象本身，可以理解为：指向对象本身的一个指针。普通的直接引用；形参与成员名字重名，用this来区分；引用构造函数，应该为构造函数中的第一条语句。

super可以理解为是指向自己父类对象的一个指针，而这个父类指的是离自己最近的一个父类。用法跟this差不多。

**26.****抽象类和接口？**

抽象类：抽象类使用abstract修饰；抽象类不能实例化，即不能使用new关键字来实例化对象；含有抽象方法（使用abstract关键字修饰的方法）的类是抽象类，必须使用abstract关键字修饰；抽象类可以含有抽象方法，也可以不包含抽象方法，抽象类中可以有具体的方法；如果一个子类实现了父类（抽象类）的所有抽象方法，那么该子类可以不必是抽象类，否则就是抽象类；抽象类中的抽象方法只有方法体，没有具体实现。抽象类可以有构造器、静态方法。构造器能被子类继承并完成初始化，但不会创建抽象类的实例对象。静态方法的好处是不实例化就可由子类类名直接调用。

接口：接口使用interface修饰；接口不能被实例化；一个类只能继承一个类，但是可以实现多个接口；接口中方法均为抽象方法；

**27. HashCode的作用？**

Java的集合有两类，一类是List，还有一类是Set。前者有序可重复，后者无序不重复。当我们在set中插入的时候，可以通过equals方法。但是如果元素太多，用这样的方法就会比较慢。

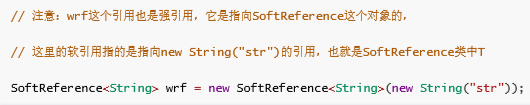
哈希算法也称为散列算法，是将数据依特定算法直接指定到一个地址上。这样一来，当集合要添加新的元素时，先调用这个元素的hashCode方法，就一下子能定位到它应该放置的物理位置上。

**28. Java的四种引用？**

强引用：只要强引用存在，垃圾回收器将永远不会回收被引用的对象，哪怕内存不足时，JVM也会直接抛出OutOfMemoryError，不会去回收。如果想中断强引用与对象之间的联系，可以显示的将强引用赋值为null，这样一来，JVM就可以适时的回收对象了。



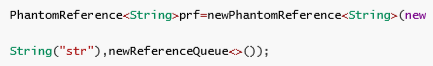
软引用：软引用是用来描述一些非必需但仍有用的对象。在内存足够的时候，软引用对象不会被回收，只有在内存不足时，系统则会回收软引用对象，如果回收了软引用对象之后仍然没有足够的内存，才会抛出内存溢出异常。这种特性常常被用来实现缓存技术，比如网页缓存，图片缓存等。



弱引用：无论内存是否足够，只要 JVM 开始进行垃圾回收，那些被弱引用关联的对象都会被回收。



虚引用：如果一个对象仅持有虚引用，那么它就和没有任何引用一样，它随时可能会被回收。永远无法通过虚引用来获取对象，虚引用必须要和 ReferenceQueue 引用队列一起使用。



引用队列：引用队列可以与软引用、弱引用以及虚引用一起配合使用，当垃圾回收器准备回收一个对象时，如果发现它还有引用，那么就会在回收对象之前，把这个引用加入到与之关联的引用队列中去。程序可以通过判断引用队列中是否已经加入了引用，来判断被引用的对象是否将要被垃圾回收，这样就可以在对象被回收之前采取一些必要的措施。

**29.Java创建对象的方式？**

使用 new 关键字，通过调用指定构造方法进行对象的创建，可以调用任何构造方法；

使用 Class 类的 newInstance 方法；

使用 java.lang.reflect.Constructor 类的 newInstance 方法；

使用 clone 方法，Java 中，所有的对象都具有 clone 方法，一旦该方法被调用，JVM会创建一个新的对象，并且会拷贝原对象中的所有字段，但不会调用任何构造方法；

反序列化：对于实现了 Serializable 接口的类，他就可以被 IO 流序列化，当通过反序列化被还原时，我们就可以重新得到这个类的对象，这也是一个创建对象的过程，但他和 clone 方法一样是由JVM 实现的，他不会调用任何构造方法。

基于序列化和反序列化实现的克隆不仅仅是深度克隆，更重要的是通过泛型限定，可以 检查出要克隆的对象是否支持序列化，这项检查是编译器完成的，不是在运行时抛出异常，这种方案明显优于使用 Object 类的clone方法克隆对象。让问题在编译的时候暴露出来总 是好过把问题留到运行时。

**30.当hash冲突产生时的处理方式？**

拉链法：每个哈希表节点都有一个next指针，多个哈希表节点可以用next指针构成一个单向链表，被分配到同一个索引上的多个节点可以用这个单向链表进行存储。

开放定址法：一旦发生了冲突，就去寻找下一个空的散列地址，只要散列表足够大，空的散列地址总能找到，并将记录存入。

再哈希：又叫双哈希法，有多个不同的hash函数，当发生冲突时，使用第二个，第三个…等哈希函数计算地址，直到无冲突。

**31.Java的浅拷贝和深拷贝？**

浅拷贝：浅拷贝是按位拷贝对象，它会创建一个新对象，这个对象有着原始对象属性值的一份精确拷贝。如果属性是基本类型，拷贝的就是基本类型的值；如果属性是内存地址（引用类型），拷贝的就是内存地址 ，因此如果其中一个对象改变了这个地址，就会影响到另一个对象。

深拷贝：在拷贝引用类型成员变量时，为引用类型的数据成员另辟了一个独立的内存空间，实现真正内容上的拷贝。对于有多层对象的，每个对象都需要实现 Cloneable 并重写 clone()方法，进而实现了对象的串行层层拷贝。

**32.****JDBC操作数据库的基本步骤？**

加载（注册）数据库驱动到JVM。

建立（获取）数据库连接。

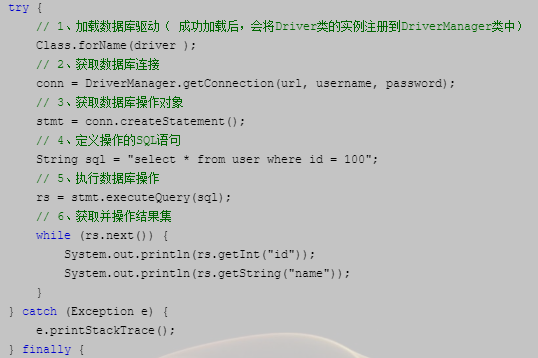
创建（获取）数据库操作对象。

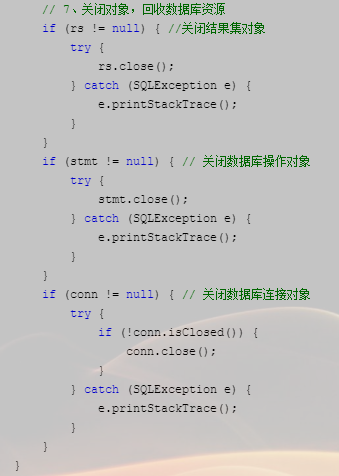
定义操作的SQL语句。

执行数据库操作。

获取并操作结果集。

关闭对象，回收数据库资源（关闭结果集-->关闭数据库操作对象-->关闭连接）。





**33.** **在使用JDBC的时候，如何防止出现SQL注入的问题？**

所谓SQL注入，就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。

预编译防止SQL注入：PreperedStatement是Statement的子类。Statement的对象会使数据库频繁编译SQL，可能造成数据库缓冲区溢出。PreparedStatement的对象可对SQL进行预编译，从而提高数据库的执行效率。并且PreperedStatement对于SQL中的参数， 允许使用"？"等占位符的形式进行对输入数据的替换，不会改变SQL语句的原有语法，简化SQL语句的编写。

**34. 怎么在JDBC内调用一个存储过程？**

存储过程相当于数据库中的函数，一组为了完成特定功能的SQL语句集，存储在数据库中，经过第一次编译后以后再调用任意次都不需要重新编译了。

在JDBC中通过Connection对象，调用prepareCall(…)方法，在prepareCall方法中的参数为字符串，内容应该为”{CALL存储过程名(?(占位符，个数根据存储过程参数来定)...)}”，调用这个方法后返回CallableStatement对象。

**35. 数据库连接池的好处?**

基本原理：在内部对象池中，维护一定数量的数据库连接，并对外暴露数据库连接的获取和返回方法。如外部使用者可通过getConnection方法获取数据库连接，使用完毕后再通过releaseConnection方法将连接返回，注意此时的连接并没有关闭，而是由连接池管理器回收，并为下一次使用做好准备。

资源重用：由于数据库连接得到重用，避免了频繁创建、释放连接引起的大量性能开销。在减少系统消耗的基础上，增进了系统环境的平稳性（减少内存碎片以及数据库临时进程、线程的数量）。

更快的系统响应速度：数据库连接池在初始化过程中，往往已经创建了若干数据库连接置于池内备用。此时连接池的初始化操作均已完成。对于业务请求处理而言，直接利用现有可用连接，避免了数据库连接初始化和释放过程的时间开销，从而缩减了系统整体响应时间。

新的资源分配手段：对于多应用共享同一数据库的系统而言，可在应用层通过数据库连接的配置，实现数据库连接池技术。例如：某一应用最大可用数据库连接数的限制，避免某一应用独占所有数据库资源。

统一的连接管理，避免数据库连接泄露：在较为完备的数据库连接池实现中，可根据预先的连接占用超时设定，强制收回被占用的连接，从而避免了常规数据库连接操作中可能出现的资源泄露。

**36.使用数据源有什么好处？**

数据源是实际数据库的替代品，或者说是实际数据库的一个引用。数据源中并无真正的数据，它记录的仅仅是连接到哪个数据库以及如何连接，例如：c3p0，druid等。

通过配置文件的形式来维护数据库信息，而不是通过硬编码。当连接的数据库信息发生改变时，不需要再更改程序代码就实现了数据库信息的更新。

**37.** **排序都有哪几种方法？**

插入排序（直接插入排序、希尔排序）：

直接插入排序：对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。

希尔排序：希尔排序是把记录按下标的一定增量分组，对每组使用直接插入排序算法排序，随着增量逐渐减少至1时，算法便终止。

交换排序（冒泡排序、快速排序）：

冒泡排序：它重复地走访过要排序的数列，一次比较两个元素，如果它们的顺序错误就把它们交换过来。

快速排序：从序列中选择一个轴点元素，利用轴点元素将序列分割成2个子序列，小于轴点元素放左侧，大于轴点元素放右侧，等于轴点元素放哪边都可以，直到子序列中只剩下一个元素。

选择排序（直接选择排序、堆排序）：

直接选择排序：首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。

堆排序：对序列进行原地建堆（heapify），交换堆顶元素与尾元素，堆的元素数量减1，对0位置进行1次siftDown操作，直到堆的元素数量为1。

归并排序：将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。

分配排序（计数排序、基数排序、桶排序）

计数排序：适合对一定范围内的整数进行排序。根据序列中最大的数确定数组的大小，数组的最大索引等于序列中最大的数，数的出现次数为对应索引的值，最后根据数组对序列进行排序。

基数排序：非常适合用于非负整数排序。依次对个位数、百位数、千位数…进行排序，排序利用的是计数排序。

桶排序：按照一定的规则（不同类型的数据，规则不同），将序列中的元素均匀分配到对应的桶，分别对每个桶进行单独排序（计数排序），将所有非空桶的元素合并成有序序列。

**38. Java 对于 eqauls 方法和 hashCode 方法的规定？**

如果两个对象相同（equals 方法返回 true），那么它们的 hashCode 值一定要相同；

如果两个对象的 hashCode 相同，它们并不一定相同。当然，你未必要按照要求去做，但是如果你违背了上述原则就会发现在使用容器时，相同的对象可以出现在 Set 集合中，同时增加新元素的效率会大大下降（对于使用哈希存储的系统，如果哈希码频繁的冲突将会造成存取性能急剧下降）。

**39.Java中方法的参数传递问题？**

Java语言的方法调用只支持参数的值传递。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。

Java 中没有传引用实在是非常的不方便，正是如此在 Java 编写的代码中才会出现大量的 Wrapper 类。

**40. char 型变量中能不能存贮一个中文汉字，为什么？**

char 类型可以存储一个中文汉字，因为Java中使用的编码是 Unicode（不选择任何特定的编码，直接使用字符在字符集中的编号，这是统一的唯一方法），一个 char 类型占 2 个字节（16 比特），所以放一个中文是没问题的。

使用 Unicode 意味着字符在 JVM 内部和外部有不同的表现形式，在 JVM内部都是 Unicode，当这个字符被从 JVM 内部转移到外部时（例如存入文件系统中），需要进行编码转换。所以 Java 中有字节流和字符流，以及在字符流和字节流之间进行转换的转换流。