# Java

## 1.面向对象编程（oop）

Java 是一个支持并发、基于类和面向对象的计算机编程语言。

下面列出了面向对象软件开发的优点：

代码开发模块化，更易维护和修改；代码复用；增强代码的可靠性和灵活性；增加代码的可理解性。

面向对象编程有很多重要的特性，比如：封装，继承，多态和抽象。

封装

封装给对象提供了隐藏内部特性和行为的能力。对象提供一些能被其他对象访问的方法来改变它内部的数据。在 Java 当中，有 3 种修饰符：public，private 和 protected。每一种修饰符给其他的位于同一个包或者不同包下面对象赋予了不同的访问权限。

下面列出了使用封装的一些好处： 通过隐藏对象的属性来保护对象内部的状态；

提高了代码的可用性和可维护性，因为对象的行为可以被单独的改变或者是扩展；

禁止对象之间的不良交互提高模块化；

多态

多态是编程语言给不同的底层数据类型做相同的接口展示的一种能力。一个多态类型上的操作可以应用到其他类型的值上面。

继承

继承给对象提供了从基类获取字段和方法的能力。继承提供了代码的重用行，也可以在不修改类的情况下给现存的类添加新特性。

抽象

抽象是把想法从具体的实例中分离出来的步骤，因此，要根据他们的功能而不是实现细节来创建类。Java 支持创建只暴漏接口而不包含方法实现的抽象的类。这种抽象技术的主要目的是把类的行为和实现细节分离开。

抽象和封装的不同点

抽象和封装是互补的概念。一方面，抽象关注对象的行为。另一方面，封装关注对象行为的细节。一般是通过隐藏对象内部状态信息做到封装，因此，封装可以看成是用来提供抽象的一种策略。

## 2. JDK 和 JRE 的区别是什么？

Java 运行时环境(JRE)是将要执行 Java 程序的 Java 虚拟机。它同时也包含了执行 applet 需要的浏览器插件。Java 开发工具包(JDK)是完整的 Java 软件开发包，包含了 JRE，编译器和其他

的工具(比如：JavaDoc，Java 调试器)，可以让开发者开发、编译、执行 Java 应用程序。

## 3.说说&和&&的区别

答：

&和&&都可以用作逻辑与的运算符，表示逻辑与（and），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，例如，对于if(str != null && !str.equals(“”))表达式，当str为null时，后面的表达式不会执行，所以不会出现 NullPointerException如果将&&改为&，则会抛出NullPointerException异常。If(x==33 & ++y>0) y会增长，If(x==33 && ++y>0)不会增长

&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，我们通常使用0x0f来与一个整数进行&运算，来获取该整数的最低4个bit位，例如，0x31 & 0x0f的结果为 0x01

## 4.是否可以从一个static方法内部发出对非static方法的调用

答：

不可以。因为非static方法是要与对象关联在一起的，必须创建一个对象后，才可以在该对象上进行方法调用，而static方法调用时不需要创建对象，可以直接调用。也就是说，当一个static方法被调用时，可能还没有创建任何实例对象，如果从一个static方法中发出对非static方法的调用，那个非static方法是关联到哪个对象上的呢？这个逻辑无法成立，所以，一个static方法内部发出对非static方法的调用。

## 5.Integer与int的区别

答：

Java 语言8种基本数据类型是： byte、short、 int、 long 、float、 double、 boolean、char 。

int是java提供的8种原始数据类型之一。Java为每个原始类型提供了封装类，Integer是java为int提供的封装类。int的默认值为0，而Integer的默认值为null，即Integer可以区分出未赋值和值为0的区别，int则无法表达出未赋值的情况，例如，要想表达出没有参加考试和考试成绩为0的区别，则只能使用Integer。在JSP开发中，Integer的默认为null，所以用el表达式在文本框中显示时，值为空白字符串，而int默认的默认值为0，

所以用el表达式在文本框中显示时，结果为0，所以，int不适合作为web层的表单数据的类型。

在Hibernate中，如果将OID定义为Integer类型，那么Hibernate就可以根据其值是否为null而判断一个对象是否是临时的，如果将OID定义为了int类型，还需要在hbm映射文件中设置其unsaved-value属性为0。

另外，Integer提供了多个与整数相关的操作方法，例如，将一个字符串转换成整数，Integer中还定义了表示整数的最大值和最小值的常量。

## 6.String是最基本的数据类型吗?

答：

   基本数据类型包括byte、int、char、long、float、double、boolean和short。

   java.lang.String类是final类型的，因此不可以继承这个类、不能修改这个类。 为了提高效率节省空间，我们应该用StringBuffer类

## 7.是否可以继承String类?

答：String类是final类故不可以继承。

## 8.String 和StringBuffer的区别

答：JAVA平台提供了两个类：String和StringBuffer，它们可以储存和操作字符串，即包含多个字符的字符数据。String类表示内容不可改变的字符串。而StringBuffer类表示内容可以被修改的字符串。当你知道字符数据要改变的时候你就可以使用StringBuffer。典型地，你可以使用StringBuffers来动态构造字符数据。另外，String实现了equals方法，new String(“abc”).equals(new String(“abc”)的结果为true,而StringBuffer没有实现equals方法，所以，new StringBuffer(“abc”).equals(new StringBuffer(“abc”)的结果为false。

 String覆盖了equals方法和hashCode方法，而StringBuffer没有覆盖equals方法和hashCode方法，所以，将StringBuffer对象存储进Java集合类中时会出现问题。

## 9.StringBuffer与StringBuilder的区别

答：StringBuffer和StringBuilder类都表示内容可以被修改的字符串，StringBuilder是线程不安全的，运行效率高，如果一个字符串变量是在方法里面定义，这种情况只可能有一个线程访问它，不存在不安全的因素了，则用StringBuilder。如果要在类里面定义成员变量，并且这个类的实例对象会在多线程环境下使用，那么最好用StringBuffer。

## 10.数组有没有length()这个方法? String有没有length()这个方法？

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

## 11.try {}里有一个return语句，那么紧跟在这个try后的finally {}里的code会不会被执行，什么时候被执行，在return前还是后?

答：

主函数调用子函数并得到结果的过程，好比主函数准备一个空罐子，当子函数要返回结果时，先把结果放在罐子里，然后再将程序逻辑返回到主函数。所谓返回，就是子函数说，我 不运行了，你主函数继续运行吧，这没什么结果可言，结果是在说这话之前放进罐子里的。

## 12.final, finally, finalize的区别

答：final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

  内部类要访问局部变量，局部变量必须定义成final类型，例如，一段代码……

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。JVM不保证此方法总被调用

## 13.运行时异常与一般异常有何异同？

答：

异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。

## 14.JAVA语言如何进行异常处理，关键字：throws,throw,try,catch,finally分别代表什么意义？在try块中可以抛出异常吗？

答：

throws是获取异常  
throw是抛出异常  
try是将会发生异常的语句括起来，从而进行异常的处理，  
catch是如果有异常就会执行他里面的语句，  
而finally不论是否有异常都会进行执行的语句。

**throw和throws的详细区别如下：**

**throw是语句抛出一个异常。**语法：throw (异常对象);  
         throw e;

**throws是方法可能抛出异常的声明**。(用在声明方法时，表示该方法可能要抛出异常)  
语法：[(修饰符)](返回值类型)(方法名)([参数列表])[throws(异常类)]{......}  
            public void doA(int a) throws Exception1,Exception3{......}

## 15.Overload和Override的区别。Overloaded的方法是否可以改变返回值的类型?

答：

Overload是重载的意思，Override是覆盖的意思，也就是重写。

重载Overload表示同一个类中可以有多个名称相同的方法，但这些方法的参数列表各不相同（即参数个数或类型不同）。

重写Override表示子类中的方法可以与父类中的某个方法的名称和参数完全相同，通过子类创建的实例对象调用这个方法时，将调用子类中的定义方法，这相当于把父类中定义的那个完全相同的方法给覆盖了，这也是面向对象编程的多态性的一种表现。子类覆盖父类的方法时，只能比父类抛出更少的异常，或者是抛出父类抛出的异常的子异常，因为子类可以解决父类的一些问题，不能比父类有更多的问题。子类方法的访问权限只能比父类的更大，不能更小。如果父类的方法是private类型，那么，子类则不存在覆盖的限制，相当于子类中增加了一个全新的方法。

至于Overloaded的方法是否可以改变返回值的类型这个问题，要看你倒底想问什么呢？这个题目很模糊。如果几个Overloaded的方法的参数列表不一样，它们的返回者类型当然也可以不一样。但我估计你想问的问题是：如果两个方法的参数列表完全一样，是否可以让它们的返回值不同来实现重载Overload。这是不行的，我们可以用反证法来说明这个问题，因为我们有时候调用一个方法时也可以不定义返回结果变量，即不要关心其返回结果，例如，我们调用map.remove(key)方法时，虽然remove方法有返回值，但是我们通常都不会定义接收返回结果的变量，这时候假设该类中有两个名称和参数列表完全相同的方法，仅仅是返回类型不同，java就无法确定编程者倒底是想调用哪个方法了，因为它无法通过返回结果类型来判断。

override可以翻译为覆盖，从字面就可以知道，它是覆盖了一个方法并且对其重写，以求达到不同的作用。对我们来说最熟悉的覆盖就是对接口方法的实现，在接口中一般只是对方法进行了声明，而我们在实现时，就需要实现接口声明的所有方法。除了这个典型的用法以外，我们在继承中也可能会在子类覆盖父类中的方法。在覆盖要注意以下的几点：

1、覆盖的方法的标志必须要和被覆盖的方法的标志完全匹配，才能达到覆盖的效果；

2、覆盖的方法的返回值必须和被覆盖的方法的返回一致；

3、覆盖的方法所抛出的异常必须和被覆盖方法的所抛出的异常一致，或者是其子类；

4、被覆盖的方法不能为private，否则在其子类中只是新定义了一个方法，并没有对其进行覆盖。

     overload对我们来说可能比较熟悉，可以翻译为重载，它是指我们可以定义一些名称相同的方法，通过定义不同的输入参数来区分这些方法，然后再调用时，VM就会根据不同的参数样式，来选择合适的方法执行。在使用重载要注意以下的几点：

1、在使用重载时只能通过不同的参数样式。例如，不同的参数类型，不同的参数个数，不同的参数顺序（当然，同一方法内的几个参数类型必须不一样，例如可以是fun(int,float)，但是不能为fun(int,int)）；

2、不能通过访问权限、返回类型、抛出的异常进行重载；

3、方法的异常类型和数目不会对重载造成影响；

4、对于继承来说，如果某一方法在父类中是访问权限是priavte，那么就不能在子类对其进行重载，如果定义的话，也只是定义了一个新方法，而不会达到重载的效果。

## 16.接口和抽象类的区别是什么？

Java 提供和支持创建抽象类和接口。它们的实现有共同点，不同点在于：

接口中所有的方法隐含的都是抽象的。而抽象类则可以同时包含抽象和非抽象的方法。

类可以实现很多个接口，但是只能继承一个抽象类

类如果要实现一个接口，它必须要实现接口声明的所有方法。但是，类可以不实现抽象类声明的所有方法，当然，在这种情况下，类也必须得声明成是抽象的。

抽象类可以在不提供接口方法实现的情况下实现接口。

Java 接口中声明的变量默认都是 final 的。抽象类可以包含非 final 的变量。

Java 接口中的成员函数默认是 public 的。抽象类的成员函数可以是 private，protected 或者是 public。

接口是绝对抽象的，不可以被实例化。抽象类也不可以被实例化，但是，如果它包含 main 方法的话是可以被调用的。

## 17.进程和线程的区别是什么？创建线程有几种不同的方式？你喜欢哪一种？为什么？ 概括的解释下线程的几种可用状态。

进程是执行着的应用程序，而线程是进程内部的一个执行序列。一个进程可以有多个线程。

线程又叫做轻量级进程。

**有三种方式可以用来创建线程：**

继承 Thread 类

实现 Runnable 接口

应用程序可以使用 Executor 框架来创建线程池

实现 Runnable 接口这种方式更受欢迎，因为这不需要继承 Thread 类。在应用设计中已经继承了别的对象的情况下，这需要多继承（而 Java 不支持多继承），只能实现接口。同时，线程池也是非常高效的，很容易实现和使用。

**线程在执行过程中，可以处于下面几种状态：**

就绪(Runnable):线程准备运行，不一定立马就能开始执行。

运行中(Running)：进程正在执行线程的代码。

等待中(Waiting):线程处于阻塞的状态，等待外部的处理结束。

睡眠中(Sleeping)：线程被强制睡眠。

I/O 阻塞(Blocked on I/O)：等待 I/O 操作完成。

同步阻塞(Blocked on Synchronization)：等待获取锁。

死亡(Dead)：线程完成了执行。

## 18.如何实现线程间的通讯

答：

[**Java**](http://lib.csdn.net/base/javaee)提供了3个非常重要的方法来巧妙地解决线程间的通信问题。这3个方法分别是：wait()、notify()和notifyAll()。

其中，调用wait()方法可以使调用该方法的线程释放共享资源的锁，然后从运行态退出，进入等待队列，直到被再次唤醒。而调用notify()方法可以唤醒等待队列中第一个等待同一共享资源的线程，并使该线程退出等待队列，进入可运行态。调用notifyAll()方法可以使所有正在等待队列中等待同一共享资源的线程从等待状态退出，进入可运行状态，此时，优先级最高的那个线程最先执行。显然，利用这些方法就不必再循环检测共享资源的状态，而是在需要的时候直接唤醒等待队列中的线程就可以了。这样不但节省了宝贵的CPU资源，也提高了程序的效率。

## 19.sleep() 和 wait() 有什么区别?

sleep是线程类（Thread）的方法，导致此线程暂停执行指定时间，给执行机会给其他线程，但是监控状态依然保持，到时后会自动恢复。调用sleep不会释放对象锁。 wait是Object类的方法，对此对象调用wait方法导致本线程放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象发出notify方法（或notifyAll）后本线程才进入对象锁定池准备获得对象锁进入运行状态。

## 20.启动一个线程是用run()还是start()?

答：

启动一个线程是调用start()方法，使线程就绪状态，以后可以被调度为运行状态，一个线程必须关联一些具体的执行代码，run()方法是该线程所关联的执行代码。

## 21.List 和 Map 区别?

答：

一个是存储单列数据的集合，另一个是存储键和值这样的双列数据的集合，List中存储的数据是有顺序，并且允许重复；Map中存储的数据是没有顺序的，其键是不能重复的，它的值是可以有重复的。

## 22.List, Set, Map是否继承自Collection接口?

答：

List，Set是，Map不是

## 23.说出ArrayList,Vector, LinkedList的存储性能和特性

答：

这样的题属于随意发挥题：这样的题比较考水平，两个方面的水平：一是要真正明白这些内容，二是要有较强的总结和表述能力。如果你明白，但表述不清楚，在别人那里则等同于不明白。

首先，List与Set具有相似性，它们都是单列元素的集合，所以，它们有一个功共同的父接口，叫Collection。Set里面不允许有重复的元素，所谓重复，即不能有两个相等（注意，不是仅仅是相同）的对象 ，即假设Set集合中有了一个A对象，现在我要向Set集合再存入一个B对象，但B对象与A对象equals相等，则B对象存储不进去，所以，Set集合的add方法有一个boolean的返回值，当集合中没有某个元素，此时add方法可成功加入该元素时，则返回true，当集合含有与某个元素equals相等的元素时，此时add方法无法加入该元素，返回结果为false。Set取元素时，没法说取第几个，只能以Iterator接口取得所有的元素，再逐一遍历各个元素。

List表示有先后顺序的集合， 注意，不是那种按年龄、按大小、按价格之类的排序。当我们多次调用add(Obj e)方法时，每次加入的对象就像火车站买票有排队顺序一样，按先来后到的顺序排序。有时候，也可以插队，即调用add(int index,Obj e)方法，就可以指定当前对象在集合中的存放位置。一个对象可以被反复存储进List中，每调用一次add方法，这个对象就被插入进集合中一次，其实，并不是把这个对象本身存储进了集合中，而是在集合中用一个索引变量指向这个对象，当这个对象被add多次时，即相当于集合中有多个索引指向了这个对象，如图x所示。List除了可以以Iterator接口取得所有的元素，再逐一遍历各个元素之外，还可以调用get(index i)来明确说明取第几个。

Map与List和Set不同，它是双列的集合，其中有put方法，定义如下：put(obj key,obj value)，每次存储时，要存储一对key/value，不能存储重复的key，这个重复的规则也是按equals比较相等。取则可以根据key获得相应的value，即get(Object key)返回值为key 所对应的value。另外，也可以获得所有的key的结合，还可以获得所有的value的结合，还可以获得key和value组合成的Map.Entry对象的集合。

     List 以特定次序来持有元素，可有重复元素。Set 无法拥有重复元素,内部排序。Map 保存key-value值，value可多值。

HashSet按照hashcode值的某种运算方式进行存储，而不是直接按hashCode值的大小进行存储。例如，"abc" ---> 78，"def" ---> 62，"xyz" ---> 65在hashSet中的存储顺序不是62,65,78，这些问题感谢以前一个叫崔健的学员提出，最后通过查看源代码给他解释清楚，看本次培训学员当中有多少能看懂源码。LinkedHashSet按插入的顺序存储，那被存储对象的hashcode方法还有什么作用呢？学员想想!hashset集合比较两个对象是否相等，首先看hashcode方法是否相等，然后看equals方法是否相等。new 两个Student插入到HashSet中，看HashSet的size，实现hashcode和equals方法后再看size。

同一个对象可以在Vector中加入多次。往集合里面加元素，相当于集合里用一根绳子连接到了目标对象。往HashSet中却加不了多次的

## 24.Set里的元素是不能重复的，那么用什么方法来区分重复与否呢? 是用==还是equals()? 它们有何区别?

答：

Set里的元素是不能重复的，元素重复与否是使用equals()方法进行判断的。

equals()和==方法决定引用值是否指向同一对象equals()在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值

## 25.你所知道的集合类都有哪些？主要方法？

答：

 最常用的集合类是 List 和 Map。 List 的具体实现包括 ArrayList 和 Vector，它们是可变大小的列表，比较适合构建、存储和操作任何类型对象的元素列表。 List 适用于按数值索引访问元素的情形。

 Map 提供了一个更通用的元素存储方法。 Map 集合类用于存储元素对（称作"键"和"值"），其中每个键映射到一个值。

ArrayList/Vector

                  Collection

HashSet/TreeSetSet

PropetiesHashTable

Map

Treemap/HashMap

     我记的不是方法名，而是思想，我知道它们都有增删改查的方法，但这些方法的具体名称，我记得不是很清楚，对于set，大概的方法是add,remove, contains；对于map，大概的方法就是put,remove，contains等，因为，我只要在eclispe下按点操作符，很自然的这些方法就出来了。我记住的一些思想就是List类会有get(int index)这样的方法，因为它可以按顺序取元素，而set类中没有get(int index)这样的方法。List和set都可以迭代出所有元素，迭代时先要得到一个iterator对象，所以，set和list类都有一个iterator方法，用于返回那个iterator对象。map可以返回三个集合，一个是返回所有的key的集合，另外一个返回的是所有value的集合，再一个返回的key和value组合成的EntrySet对象的集合，map也有get方法，参数是key，返回值是key对应的value。

## 26.java中有几种类型的流？JDK为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类

字节流，字符流。字节流继承于InputStream OutputStream，字符流继承于InputStreamReader OutputStreamWriter。在java.io包中还有许多其他的流，主要是为了提高性能和使用方便。

## 27.字节流与字符流的区别

答：

要把一片二进制数据数据逐一输出到某个设备中，或者从某个设备中逐一读取一片二进制数据，不管输入输出设备是什么，我们要用统一的方式来完成这些操作，用一种抽象的方式进行描述，这个抽象描述方式起名为IO流，对应的抽象类为OutputStream和InputStream ，不同的实现类就代表不同的输入和输出设备，它们都是针对字节进行操作的。

在应用中，经常要完全是字符的一段文本输出去或读进来，用字节流可以吗？计算机中的一切最终都是二进制的字节形式存在。对于“中国”这些字符，首先要得到其对应的字节，然后将字节写入到输出流。读取时，首先读到的是字节，可是我们要把它显示为字符，我们需要将字节转换成字符。由于这样的需求很广泛，人家专门提供了字符流的包装类。

底层设备永远只接受字节数据，有时候要写字符串到底层设备，需要将字符串转成字节再进行写入。字符流是字节流的包装，字符流则是直接接受字符串，它内部将串转成字节，再写入底层设备，这为我们向IO设别写入或读取字符串提供了一点点方便。

字符向字节转换时，要注意编码的问题，因为字符串转成字节数组，

其实是转成该字符的某种编码的字节形式，读取也是反之的道理。

## 28.什么是java序列化，如何实现java序列化？或者请解释Serializable接口的作用

答：

我们有时候将一个java对象变成字节流的形式传出去或者从一个字节流中恢复成一个java对象，例如，要将java对象存储到硬盘或者传送给网络上的其他计算机，这个过程我们可以自己写代码去把一个java对象变成某个格式的字节流再传输，但是，jre本身就提供了这种支持，我们可以调用OutputStream的writeObject方法来做，如果要让java 帮我们做，要被传输的对象必须实现serializable接口，这样，javac编译时就会进行特殊处理，编译的类才可以被writeObject方法操作，这就是所谓的序列化。需要被序列化的类必须实现Serializable接口，该接口是一个mini接口，其中没有需要实现的方法，implements Serializable只是为了标注该对象是可被序列化的。

例如，在web开发中，如果对象被保存在了Session中，tomcat在重启时要把Session对象序列化到硬盘，这个对象就必须实现Serializable接口。如果对象要经过分布式系统进行网络传输或通过rmi等远程调用，这就需要在网络上传输对象，被传输的对象就必须实现Serializable接口。

## 29.能不能自己写个类，也叫java.lang.String？

答：

可以，但在应用的时候，需要用自己的类加载器去加载，否则，系统的类加载器永远只是去加载jre.jar包中的那个java.lang.String。由于在tomcat的web应用程序中，都是由webapp自己的类加载器先自己加载WEB-INF/classess目录中的类，然后才委托上级的类加载器加载，如果我们在tomcat的web应用程序中写一个java.lang.String，这时候Servlet程序加载的就是我们自己写的java.lang.String，但是这么干就会出很多潜在的问题，原来所有用了java.lang.String类的都将出现问题。

     虽然java提供了endorsed技术，可以覆盖jdk中的某些类，具体做法是….。但是，能够被覆盖的类是有限制范围，反正不包括java.lang这样的包中的类。

    （下面的例如主要是便于大家学习理解只用，不要作为答案的一部分，否则，人家怀疑是题目泄露了）例如，运行下面的程序：

package java.lang;

public class String {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("string");

}

}

报告的错误如下：

java.lang.NoSuchMethodError: main

Exception in thread "main"

这是因为加载了jre自带的java.lang.String，而该类中没有main方法。

## 30.一个".java"源文件中是否可以包括多个类（不是内部类）？有什么限制？

答：

可以有多个类，但只能有一个public的类，并且public的类名必须与文件名相一致。

## 31.八个基本类型各占多少字节？

答：byte 1字节、short 2字节、int 4字节、long 8字节、float 4字节、double 8字节、char 2字节、boolean 1字节

## 32.HashMap、HashSet、HashTable的区别?

答：

**区别一：**继承的父类不同

Hashtable继承自Dictionary类，而HashMap继承自AbstractMap类。但二者都实现了Map接口。

**区别二：**线程安全性不同

Hashtable 中的方法是Synchronize的，而HashMap中的方法在缺省情况下是非Synchronize的。

**区别三：**是否提供contains方法

HashMap把Hashtable的contains方法去掉了，改成containsValue和containsKey，因为contains方法容易让人引起误解。   
Hashtable则保留了contains，containsValue和containsKey三个方法，其中contains和containsValue功能相同。

区别四：\*\*key和value是否允许null值 （面试比较喜欢问）

其中key和value都是对象，并且不能包含重复key，但可以包含重复的value。   
Hashtable中，key和value都不允许出现null值。   
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。因此，在HashMap中不能由get()方法来判断HashMap中是否存在某个键， 而应该用containsKey()方法来判断。

**区别五：**哈希值的计算方法不同，Hashtable直接使用的是对象的hashCode,而HashMap则是在对象的hashCode的基础上还进行了一些变化。

**区别六：**内部实现使用的数组初始化和扩容方式不同，内存初始大小不同，HashTable初始大小是11，而HashMap初始大小是16

## 33.Hashcode和equals

答：

**equals：**

Object类中默认的实现方式是  :   return this == obj  。那就是说，只有this 和 obj引用同一个对象，才会返回true。

而我们往往需要用equals来判断 2个对象是否等价，而非验证他们的唯一性。这样我们在实现自己的类时，就要重写equals

按照约定，equals要满足以下规则。

**自反性**:  x.equals(x) 一定是true

**对null**:  x.equals(null) 一定是false

**对称性**:  x.equals(y)  和  y.equals(x)结果一致

传递性:  a 和 b equals , b 和 c  equals，那么 a 和 c也一定equals。

一致性:  在某个运行时期间，2个对象的状态的改变不会不影响equals的决策结果，那么，在这个运行时期间，无论调用多少次equals，都返回相同的结果。

**Hashcode：**

这个方法返回对象的散列码，返回值是int类型的散列码。  
对象的散列码是为了更好的支持基于哈希机制的Java集合类，例如 Hashtable, HashMap, HashSet 等。

关于hashCode方法，一致的约定是：

重写了euqls方法的对象必须同时重写hashCode()方法。

如果2个对象通过equals调用后返回是true，那么这个2个对象的hashCode方法也必须返回同样的int型散列码  
  
如果2个对象通过equals返回false，他们的hashCode返回的值**允许相同**。(然而，程序员必须意识到，hashCode返回独一无二的散列码，会让存储这个对象的hashtables更好地工作。)

在上面的例子中，Test类对象有2个字段，num和data，这2个字段代表了对象的状态，他们也用在equals方法中作为评判的依据。那么， 在hashCode方法中，这2个字段也要参与hash值的运算，作为hash运算的中间参数。这点很关键，这是为了遵守：2个对象equals，那么 hashCode一定相同规则。

也是说，参与equals函数的字段，也必须都参与hashCode 的计算。

合乎情理的是：同一个类中的不同对象返回不同的散列码。典型的方式就是根据对象的地址来转换为此对象的散列码，但是这种方式对于Java来说并不是唯一的要求的  
的实现方式。通常也不是最好的实现方式。  
  
相比 于 equals公认实现约定，hashCode的公约要求是很容易理解的。有2个重点是hashCode方法必须遵守的。约定的第3点，其实就是第2点的  
细化，下面我们就来看看对hashCode方法的一致约定要求。

**第一：在某个运行时期间，只要对象的（字段的）变化不会影响equals方法的决策结果，那么，在这个期间，无论调用多少次hashCode，都必须返回同一个散列码。**  
**第二：通过equals调用返回true 的2个对象的hashCode一定一样。**  
**第三：通过equasl返回false 的2个对象的散列码不需要不同，也就是他们的hashCode方法的返回值允许出现相同的情况。**  
**总结一句话：等价的(调用equals返回true)对象必须产生相同的散列码。不等价的对象，不要求产生的散列码不相同。**

## 34.方法重载和重写的区别

答：

1.重写必须继承，重载不用。  
2.重写的方法名，参数数目相同，参数类型兼容，重载的方法名相同，参数列表不同。  
3.重写的方法修饰符[大于等于](https://www.baidu.com/s?wd=%E5%A4%A7%E4%BA%8E%E7%AD%89%E4%BA%8E&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBryN9mHIBnyR1n19brAR0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtdnHc4nWR3rf)父类的方法，重载和修饰符无关。  
4.重写不可以抛出父类没有抛出的一般异常，可以抛出运行时异常

## 35.进程和线程的区别

答：

1. 定义

进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动,进程是系统进行资源分配和调度的一个独立单位.

线程是进程的一个实体,是CPU调度和分派的基本单位,它是比进程更小的能独立运行的基本单位.线程自己基本上不拥有系统资源,只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈),但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源.

2.关系

一个线程可以创建和撤销另一个线程;同一个进程中的多个线程之间可以并发执行.

相对进程而言，线程是一个更加接近于执行体的概念，它可以与同进程中的其他线程共享数据，但拥有自己的栈空间，拥有独立的执行序列。

3.区别

进程和线程的主要差别在于它们是不同的操作系统资源管理方式。进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。**但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。**

**1) 简而言之,一个程序至少有一个进程,一个进程至少有一个线程.**

2) 线程的划分尺度小于进程，使得多线程程序的并发性高。

3) 另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。

4) 线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。**但是线程不能够独立执行，**必须依存在应用程序中，由应用程序提供多个线程执行控制。

5) 从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。**这就是进程和线程的重要区别。**

4.优缺点

线程和进程在使用上各有优缺点：线程执行开销小，但不利于资源的管理和保护；而进程正相反。同时，线程适合于在SMP机器上运行，而进程则可以跨机器迁移。

## 36.junit中before和beforeclass区别

答：

@before

在每个测试方法之前都会运行一次，只需声明成public

@beforeclass

在类中只运行一次，必须声明成public static

## 37.单例手写

答：

懒汉式

1. //懒汉式单例类.在第一次调用的时候实例化自己
2. **public** **class** Singleton {
3. **private** Singleton() {}
4. **private** **static** Singleton single=**null**;
5. //静态工厂方法
6. **public** **static** Singleton getInstance() {
7. **if** (single == **null**) {
8. single = **new** Singleton();
9. }
10. **return** single;
11. }
12. }

饿汉式

1. //饿汉式单例类.在类初始化时，已经自行实例化
2. **public** **class** Singleton1 {
3. **private** Singleton1() {}
4. **private** **static** **final** Singleton1 single = **new** Singleton1();
5. //静态工厂方法
6. **public** **static** Singleton1 getInstance() {
7. **return** single;
8. }
9. }

## 38.触发器的作用是什么？

**答：**比如说你emp和dept两张表是有外键关联的，当emp存在相关数据时，dept无法删除数据，这时候就可以写个触发器，让他可以删除的同时并对emp表的依赖数据发生变化

## 39.Static局部变量与全局变量的区别，编译后映射文件是否包含此类变量的地址？

**答：**

全局变量(外部变量)的说明之前再冠以static 就构成了静态的全局变量。全局变量本身就是静态存储方式， 静态全局变量当然也是静态存储方式。 这两者在存储方式上并无不同。这两者的区别虽在于非静态全局变量的作用域是整个源程序， 当一个源程序由多个源文件组成时，非静态的全局变量在各个源文件中都是有效的。 而静态全局变量则限制了其作用域， 即只在定义该变量的源文件内有效， 在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用， 因此可以避免在其它源文件中引起错误。   
　　从以上分析可以看出， 把局部变量改变为静态变量后是改变了它的存储方式即改变了它的生存期。把全局变量改变为静态变量后是改变了它的作用域， 限制了它的使用范围。

　　static函数与普通函数作用域不同。仅在本文件。只在当前源文件中使用的函数应该说明为内部函数(static)，内部函数应该在当前源文件中说明和定义。对于可在当前源文件以外使用的函数，应该在一个头文件中说明，要使用这些函数的源文件要包含这个头文件

**static全局变量与普通的全局变量有什么区别：static全局变量只初使化一次，防止在其他文件单元中被引用;**  
**static局部变量和普通局部变量有什么区别：static局部变量只被初始化一次，下一次依据上一次结果值；**  
**static函数与普通函数有什么区别：static函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝**

　　程序的局部变量存在于（堆栈）中，全局变量存在于（静态区 ）中，动态申请数据存在于（ 堆）中。

extern全局变量、static全局变量和static局部变量的生存期都是“永久”，区别只是可见域不同。extern全局变量可见区域是工程，static全局变量可见区域是文件，而static局部变量的可见区域是块。     
  从代码维护角度来看，对extern变量的修改可能会影响所有代码，对static全局变量的修改可能影响一个文件中的代码，而对static变量的修改 可能影响一个块的代码；因此在选择变量类型时，优先级是static局部>static全局>extern全局。但它们有着共同的缺点：使用 了这些类型变量的函数将是不可重入的，不是线程安全的。在C/C++标准库中有很多函数都使用了static局部变量，目前的实现中都为它们提供了两套代 码，单线程版本使用static变量而多线程版本使用“线程全局变量”，比如rand,strtok等。

一个进程可用内存空间为4G，可分在存放静态数据，代码，系统内存，堆，栈等。.活动记录一般存放调用参数、返回地址等内容。堆和栈最大的区别在于 堆是由低地址向高地址分配内存，而栈是由高向低。全局和静态数据存放在全局数据区，其余的在栈中，用malloc 或 new 分配的内存位于堆中。一般来说栈在低地址，堆位于高地址

## 40.用JAVA实现一种排序。（要写出具体的算法实现，不要简单的调用Arrays.sort方法）。

答：

冒泡排序（Bubble Sort）是一种简单的排序算法。它重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。这个算法的名字由来是因为越小的元素会经由交换慢慢“浮”到数列的顶端。

冒泡排序算法的运作如下:

1. 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
2. 对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。
3. 针对所有的元素重复以上的步骤，除了最后一个。
4. 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

## 41.Java中如何实现多继承关系？

答:

1，java中只能利用接口达到多实现而已，跟多继承相仿  
2，java中唯一可以实现多继承的 就是接口与接口之间了。

## 43.当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递?

**答：**

是值传递。Java 编程语言只有值传递参数。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。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的

## 44. Java 集合类框架的基本接口有哪些？

Java 集合类提供了一套设计良好的支持对一组对象进行操作的接口和类。

**Java 集合类里面最基本的接口有：**

Collection：代表一组对象，每一个对象都是它的子元素。

Set：不包含重复元素的 Collection。

List：有顺序的 collection，并且可以包含重复元素。

Map：可以把键(key)映射到值(value)的对象，键不能重复。

## 45.什么是迭代器(Iterator)？

Iterator 接口提供了很多对集合元素进行迭代的方法。每一个集合类都包含了可以返回迭代器实例的

迭代方法。迭代器可以在迭代的过程中删除底层集合的元素。

克隆(cloning)或者是序列化(serialization)的语义和含义是跟具体的实现相关的。因此，应该由集合类的具体实现来决定如何被克隆或者是序列化。

## 46.Iterator 和 ListIterator 的区别是什么？

下面列出了他们的区别：

Iterator 可用来遍历 Set 和 List 集合，但是 ListIterator 只能用来遍历 List。

Iterator 对集合只能是前向遍历，ListIterator 既可以前向也可以后向。

ListIterator 实现了 Iterator 接口，并包含其他的功能，比如：增加元素，替换元素，获取前一个和后一个元素的索引，等等。

## 47.Java 中的 HashMap 的工作原理是什么？

Java 中的 HashMap 是以键值对(key-value)的形式存储元素的。HashMap 需要一个 hash 函数，它使用 hashCode()和 equals()方法来向集合/从集合添加和检索元素。当调用 put()方法的时候，HashMap 会计算 key 的 hash 值，然后把键值对存储在集合中合适的索引上。如果 key 已经存在了，value 会被更新成新值。HashMap 的一些重要的特性是它的容量(capacity)，负载因子(load factor)和扩容极限(threshold resizing)。

## 48.hashCode()和 equals()方法的重要性体现在什么地方？

Java 中的 HashMap 使用 hashCode()和 equals()方法来确定键值对的索引，当根据键获取值的时候也会用到这两个方法。如果没有正确的实现这两个方法，两个不同的键可能会有相同的 hash 值，因此，可能会被集合认为是相等的。而且，这两个方法也用来发现重复元素。所以这两个方法的实现对 HashMap 的精确性和正确性是至关重要的。

## 49.HashMap 和 Hashtable 有什么区别？

HashMap 和 Hashtable 都实现了 Map 接口，因此很多特性非常相似。但是，他们有以下不同点：

HashMap 允许键和值是 null，而 Hashtable 不允许键或者值是 null。

Hashtable 是同步的，而 HashMap 不是。因此，HashMap 更适合于单线程环境，而 Hashtable 适合于多线程环境。

HashMap 提供了可供应用迭代的键的集合，因此，HashMap 是快速失败的。另一方面，

Hashtable 提供了对键的列举(Enumeration)。

一般认为 Hashtable 是一个遗留的类。

## 50.数组(Array)和列表(ArrayList)有什么区别？什么时候应该使用 Array 而不是 ArrayList？

下面列出了 Array 和 ArrayList 的不同点：

Array 可以包含基本类型和对象类型，ArrayList 只能包含对象类型。

Array 大小是固定的，ArrayList 的大小是动态变化的。

ArrayList 提供了更多的方法和特性，比如：addAll()，removeAll()，iterator()等等。

对于基本类型数据，集合使用自动装箱来减少编码工作量。但是，当处理固定大小的基本数据类型的时候，这种方式相对比较慢。

## 51.ArrayList 和 LinkedList 有什么区别？

ArrayList 和 LinkedList 都实现了 List 接口，他们有以下的不同点：

ArrayList 是基于索引的数据接口，它的底层是数组。它可以以 O(1)时间复杂度对元素进行随机访问。与此对应，LinkedList 是以元素列表的形式存储它的数据，每一个元素都和它的前

一个和后一个元素链接在一起，在这种情况下，查找某个元素的时间复杂度是 O(n)。

相对于 ArrayList，LinkedList 的插入，添加，删除操作速度更快，因为当元素被添加到集合任意位置的时候，不需要像数组那样重新计算大小或者是更新索引。

LinkedList 比 ArrayList 更占内存，因为 LinkedList 为每一个节点存储了两个引用，一个指向前一个元素，一个指向下一个元素。也可以参考 ArrayList vs. LinkedList。

## 52.Comparable 和 Comparator 接口是干什么的？列出它们的区别。

Java 提供了只包含一个 compareTo()方法的 Comparable 接口。这个方法可以个给两个对象排序。具体来说，它返回负数，0，正数来表明输入对象小于，等于，大于已经存在的对象。

Java 提供了包含 compare()和 equals()两个方法的 Comparator 接口。compare()方法用来给两个输入参数排序，返回负数，0，正数表明第一个参数是小于，等于，大于第二个参数。equals() 方法需要一个对象作为参数，它用来决定输入参数是否和 comparator 相等。只有当输入参数也是一个 comparator 并且输入参数和当前 comparator 的排序结果是相同的时候，这个方法才返回 true。

## 53.什么是 Java 优先级队列(Priority Queue)？

PriorityQueue 是一个基于优先级堆的无界队列，它的元素是按照自然顺序(natural order)排序的。在创建的时候，我们可以给它提供一个负责给元素排序的比较器。PriorityQueue 不允许 null 值，因为他们没有自然顺序，或者说他们没有任何的相关联的比较器。最后，PriorityQueue 不是线程安全的，入队和出队的时间复杂度是 O(log(n))。

## 54..Java 集合类框架的最佳实践有哪些？

根据应用的需要正确选择要使用的集合的类型对性能非常重要，比如：假如元素的大小是固定的，而且能事先知道，我们就应该用 Array 而不是 ArrayList。

有些集合类允许指定初始容量。因此，如果我们能估计出存储的元素的数目，我们可以设置初始容量来避免重新计算 hash 值或者是扩容。

为了类型安全，可读性和健壮性的原因总是要使用泛型。同时，使用泛型还可以避免运行时的 ClassCastException。

使用 JDK 提供的不变类(immutable class)作为 Map 的键可以避免为我们自己的类实现 hashCode()和 equals()方法。编程的时候接口优于实现。

底层的集合实际上是空的情况下，返回长度是 0 的集合或者是数组，不要返回 null。

## 55..Enumeration 接口和 Iterator 接口的区别有哪些？

Enumeration 速度是 Iterator 的 2 倍，同时占用更少的内存。但是，Iterator 远远比 Enumeration 安全，因为其他线程不能够修改正在被 iterator 遍历的集合里面的对象。同时，Iterator 允许调用者删除底层集合里面的元素，这对 Enumeration 来说是不可能的。

## 56..HashSet 和 TreeSet 有什么区别？

HashSet 是由一个 hash 表来实现的，因此，它的元素是无序的。add()，remove()，contains() 方法的时间复杂度是 O(1)。

另一方面，TreeSet 是由一个树形的结构来实现的，它里面的元素是有序的。因此，add()， remove()，contains()方法的时间复杂度是 O(logn)。

# 垃圾收集器(Garbage Collectors)

## 57..Java 中垃圾回收有什么目的？什么时候进行垃圾回收？

垃圾回收的目的是识别并且丢弃应用不再使用的对象来释放和重用资源。

## 58..System.gc()和 Runtime.gc()会做什么事情？

这两个方法用来提示 JVM 要进行垃圾回收。但是，立即开始还是延迟进行垃圾回收是取决于 JVM 的。

## 59.finalize()方法什么时候被调用？析构函数(finalization)的目的是什么？

在释放对象占用的内存之前，垃圾收集器会调用对象的 finalize()方法。一般建议在该方法中释放对象持有的资源。

## 60.如果对象的引用被置为 null，垃圾收集器是否会立即释放对象占用的内存？

不会，在下一个垃圾回收周期中，这个对象将是可被回收的。

## 61.Java 堆的结构是什么样子的？什么是堆中的永久代(Perm Gen space)?

JVM 的堆是运行时数据区，所有类的实例和数组都是在堆上分配内存。它在 JVM 启动的时

候被创建。对象所占的堆内存是由自动内存管理系统也就是垃圾收集器回收。

堆内存是由存活和死亡的对象组成的。存活的对象是应用可以访问的，不会被垃圾回收。死亡的对象是应用不可访问尚且还没有被垃圾收集器回收掉的对象。一直到垃圾收集器把这些对象回收掉之前，他们会一直占据堆内存空间。

## 62.串行(serial)收集器和吞吐量(throughput)收集器的区别是什么？

吞吐量收集器使用并行版本的新生代垃圾收集器，它用于中等规模和大规模数据的应用程序。而串行收集器对大多数的小应用(在现代处理器上需要大概 100M 左右的内存)就足够了。

## 63.在 Java 中，对象什么时候可以被垃圾回收？

当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。

## 64.JVM 的永久代中会发生垃圾回收么？

垃圾回收不会发生在永久代，如果永久代满了或者是超过了临界值，会触发完全垃圾回收 (Full GC)。如果你仔细查看垃圾收集器的输出信息，就会发现永久代也是被回收的。这就是为什么正确的永久代大小对避免 Full GC 是非常重要的原因。请参考下 Java8：从永久代到元数据区

(译者注：Java8 中已经移除了永久代，新加了一个叫做元数据区的 native 内存区)

# 异常处理

## 65.Java 中的两种异常类型是什么？他们有什么区别？

Java 中有两种异常：受检查的(checked)异常和不受检查的(unchecked)异常。不受检查的异常不需要在方法或者是构造函数上声明，就算方法或者是构造函数的执行可能会抛出这样的异常，并且不受检查的异常可以传播到方法或者是构造函数的外面。相反，受检查的异常必须

要用 throws 语句在方法或者是构造函数上声明。这里有 Java 异常处理的一些小建议。

## 66.Java 中 Exception 和 Error 有什么区别？

Exception 和 Error 都是 Throwable 的子类。Exception 用于用户程序可以捕获的异常情况。Error 定义了不期望被用户程序捕获的异常。

## 67 throw 和 throws 有什么区别？

throw 关键字用来在程序中明确的抛出异常，相反，throws 语句用来表明方法不能处理的异常。每一个方法都必须要指定哪些异常不能处理，所以方法的调用者才能够确保处理可能发生的异常，多个异常是用逗号分隔的。

## 68.异常处理的时候，finally 代码块的重要性是什么？

无论是否抛出异常，finally 代码块总是会被执行。就算是没有 catch 语句同时又抛出异常的情况下，finally 代码块仍然会被执行。最后要说的是，finally 代码块主要用来释放资源，比如：I/O 缓冲区，数据库连接。

## 69.异常处理完成以后，Exception 对象会发生什么变化？

Exception 对象会在下一个垃圾回收过程中被回收掉。

## 70.finally 代码块和 finalize()方法有什么区别？

无论是否抛出异常，finally 代码块都会执行，它主要是用来释放应用占用的资源。finalize()

方法是 Object 类的一个 protected 方法，它是在对象被垃圾回收之前由 Java 虚拟机来调用的。

# JDBC

## 71.什么是 JDBC？

JDBC 是允许用户在不同数据库之间做选择的一个抽象层。JDBC 允许开发者用 JAVA 写数据库应用程序，而不需要关心底层特定数据库的细节。

## 72.解释下驱动(Driver)在 JDBC 中的角色。

JDBC 驱动提供了特定厂商对 JDBC API 接口类的实现，驱动必须要提供 java.sql 包下面这些类的实现：Connection, Statement, PreparedStatement,CallableStatement, ResultSet 和 Driver。

## 73.Class.forName()方法有什么作用？

这个方法用来载入跟数据库建立连接的驱动。

## 74.PreparedStatement 比 Statement 有什么优势？

PreparedStatements 是预编译的，因此，性能会更好。同时，不同的查询参数值，

PreparedStatement 可以重用。

## 75.什么时候使用 CallableStatement？用来准备 CallableStatement 的方法是什么？

CallableStatement 用来执行存储过程。存储过程是由数据库存储和提供的。存储过程可以接受输入参数，也可以有返回结果。非常鼓励使用存储过程，因为它提供了安全性和模块化。

准备一个 CallableStatement 的方法是：

CallableStament.prepareCall();

## 76.数据库连接池是什么意思？

像打开关闭数据库连接这种和数据库的交互可能是很费时的，尤其是当客户端数量增加的时候，会消耗大量的资源，成本是非常高的。可以在应用服务器启动的时候建立很多个数据库连接并维护在一个池中。连接请求由池中的连接提供。在连接使用完毕以后，把连接归还到池中，以用于满足将来更多的请求。

# Servlet

## 77.什么是 Servlet？

Servlet 是用来处理客户端请求并产生动态网页内容的 Java 类。Servlet 主要是用来处理或者是存储 HTML 表单提交的数据，产生动态内容，在无状态的 HTTP 协议下管理状态信息。

## 78.说一下 Servlet 的体系结构。

所有的 Servlet 都必须要实现的核心的接口是 javax.servlet.Servlet。每一个 Servlet 都必须要直接或者是间接实现这个接口，或者是继承 javax.servlet.GenericServlet 或者 javax.servlet.http.HTTPServlet。最后，Servlet 使用多线程可以并行的为多个请求服务。

## 79.Applet 和 Servlet 有什么区别？

Applet 是运行在客户端主机的浏览器上的客户端 Java 程序。而 Servlet 是运行在 web 服务器上的服务端的组件。applet 可以使用用户界面类，而 Servlet 没有用户界面，相反，Servlet 是等待客户端的 HTTP 请求，然后为请求产生响应。

## 80.GenericServlet 和 HttpServlet 有什么区别？

GenericServlet 是一个通用的协议无关的 Servlet，它实现了 Servlet 和 ServletConfig 接口。继承自 GenericServlet 的 Servlet 应该要覆盖 service()方法。最后，为了开发一个能用在网页上服务于使用 HTTP 协议请求的 Servlet，你的 Servlet 必须要继承自 HttpServlet。这里有 Servlet 的例子。

## 81.解释下 Servlet 的生命周期。

对每一个客户端的请求，Servlet 引擎载入 Servlet，调用它的 init()方法，完成 Servlet 的初始化。然后，Servlet 对象通过为每一个请求单独调用 service()方法来处理所有随后来自客户端的请求，最后，调用Servlet(译者注：这里应该是Servlet 而不是server)的destroy()方法把Servlet 删除掉。

## 82.doGet()方法和 doPost()方法有什么区别？

doGet：GET 方法会把名值对追加在请求的 URL 后面。因为 URL 对字符数目有限制，进而限制了用在客户端请求的参数值的数目。并且请求中的参数值是可见的，因此，敏感信息不能用这种方式传递。

doPOST：POST 方法通过把请求参数值放在请求体中来克服 GET 方法的限制，因此，可以发送的参数的数目是没有限制的。最后，通过 POST 请求传递的敏感信息对外部客户端是不可见的。

## 83.什么是 Web 应用程序？

Web 应用程序是对 Web 或者是应用服务器的动态扩展。有两种类型的 Web 应用：面向表现的和面向服务的。面向表现的 Web 应用程序会产生包含了很多种标记语言和动态内容的交互的web页面作为对请求的响应。而面向服务的Web应用实现了 Web服务的端点(endpoint)。一般来说，一个 Web 应用可以看成是一组安装在服务器 URL 名称空间的特定子集下面的

Servlet 的集合。

## 84.什么是服务端包含(Server Side Include)？

服务端包含(SSI)是一种简单的解释型服务端脚本语言，大多数时候仅用在 Web 上，用 servlet 标签嵌入进来。SSI 最常用的场景把一个或多个文件包含到 Web 服务器的一个 Web 页面中。当浏览器访问 Web 页面的时候，Web 服务器会用对应的 servlet 产生的文本来替换 Web 页面中的 servlet 标签。

## 85.什么是 Servlet 链(Servlet Chaining)？

Servlet 链是把一个 Servlet 的输出发送给另一个 Servlet 的方法。第二个 Servlet 的输出可以发

送给第三个 Servlet，依次类推。链条上最后一个 Servlet 负责把响应发送给客户端。

## 86.如何知道是哪一个客户端的机器正在请求你的 Servlet？

ServletRequest 类可以找出客户端机器的 IP 地址或者是主机名。getRemoteAddr()方法获取客户端主机的 IP 地址，getRemoteHost()可以获取主机名。看下这里的例子。

## 87.HTTP 响应的结构是怎么样的？

HTTP 响应由三个部分组成：

状态码(Status Code)：描述了响应的状态。可以用来检查是否成功的完成了请求。请求失败的情况下，状态码可用来找出失败的原因。如果 Servlet 没有返回状态码，默认会返回成功的状态码 HttpServletResponse.SC\_OK。

HTTP 头部(HTTP Header)：它们包含了更多关于响应的信息。比如：头部可以指定认为响应过期的过期日期，或者是指定用来给用户安全的传输实体内容的编码格式。如何在 Serlet 中检索 HTTP 的头部看这里。

主体(Body)：它包含了响应的内容。它可以包含 HTML 代码，图片，等等。主体是由传输在

HTTP 消息中紧跟在头部后面的数据字节组成的。

## 88.什么是 cookie？session 和 cookie 有什么区别？

cookie 是 Web 服务器发送给浏览器的一块信息。浏览器会在本地文件中给每一个 Web 服务器存储 cookie。以后浏览器在给特定的 Web 服务器发请求的时候，同时会发送所有为该服务器存储的 cookie。**下面列出了 session 和 cookie 的区别**：

无论客户端浏览器做怎么样的设置，session 都应该能正常工作。客户端可以选择禁用 cookie，但是，session 仍然是能够工作的，因为客户端无法禁用服务端的 session。

在存储的数据量方面 session 和 cookies 也是不一样的。session 能够存储任意的 Java 对象， cookie 只能存储 String 类型的对象。

## 89.浏览器和 Servlet 通信使用的是什么协议？

浏览器和 Servlet 通信使用的是 HTTP 协议。

## 90.什么是 HTTP 隧道？

HTTP 隧道是一种利用 HTTP 或者是 HTTPS 把多种网络协议封装起来进行通信的技术。因此，

HTTP 协议扮演了一个打通用于通信的网络协议的管道的包装器的角色。把其他协议的请求掩盖成 HTTP 的请求就是 HTTP 隧道。

## 91.sendRedirect()和 forward()方法有什么区别？

sendRedirect()方法会创建一个新的请求，而 forward()方法只是把请求转发到一个新的目标上。重定向(redirect)以后，之前请求作用域范围以内的对象就失效了，因为会产生一个新的请求，而转发(forwarding)以后，之前请求作用域范围以内的对象还是能访问的。一般认为 sendRedirect()比 forward()要慢。

## 92.什么是 URL 编码和 URL 解码？

URL 编码是负责把 URL 里面的空格和其他的特殊字符替换成对应的十六进制表示，反之就是解码。

# JSP

## 93.什么是 JSP 页面？

JSP 页面是一种包含了静态数据和 JSP 元素两种类型的文本的文本文档。静态数据可以用任何基于文本的格式来表示，比如：HTML 或者 XML。JSP 是一种混合了静态内容和动态产生的内容的技术。这里看下 JSP 的例子。

## 94.JSP 请求是如何被处理的？

浏览器首先要请求一个以.jsp 扩展名结尾的页面，发起 JSP 请求，然后，Web 服务器读取这个请求，使用 JSP 编译器把 JSP 页面转化成一个 Servlet 类。需要注意的是，只有当第一次请求页面或者是 JSP 文件发生改变的时候 JSP 文件才会被编译，然后服务器调用 servlet 类，处理浏览器的请求。一旦请求执行结束，servlet 会把响应发送给客户端。这里看下如何在 JSP 中获取请求参数。

## 95.JSP 有什么优点？

下面列出了使用 JSP 的优点：

JSP 页面是被动态编译成 Servlet 的，因此，开发者可以很容易的更新展现代码。

JSP 页面可以被预编译。

JSP 页面可以很容易的和静态模板结合，包括：HTML 或者 XML，也可以很容易的和产生动态内容的代码结合起来。

开发者可以提供让页面设计者以类 XML 格式来访问的自定义的 JSP 标签库。

开发者可以在组件层做逻辑上的改变，而不需要编辑单独使用了应用层逻辑的页面。

## 96.什么是 JSP 指令(Directive)？JSP 中有哪些不同类型的指令？

Directive 是当 JSP 页面被编译成 Servlet 的时候，JSP 引擎要处理的指令。Directive 用来设置页面级别的指令，从外部文件插入数据，指定自定义的标签库。Directive是定义在<%@ 和 %> 之间的。下面列出了不同类型的 Directive：

包含指令(Include directive)：用来包含文件和合并文件内容到当前的页面。

页面指令(Page directive)：用来定义 JSP 页面中特定的属性，比如错误页面和缓冲区。

Taglib 指令： 用来声明页面中使用的自定义的标签库。

## 

## 97.什么是 JSP 动作(JSP action)？

JSP 动作以 XML 语法的结构来控制 Servlet 引擎的行为。当 JSP 页面被请求的时候，JSP 动作会被执行。它们可以被动态的插入到文件中，重用 JavaBean 组件，转发用户到其他的页面，或者是给 Java 插件产生 HTML 代码。下面列出了可用的动作：

jsp:include-当 JSP 页面被请求的时候包含一个文件。

jsp:useBean-找出或者是初始化 Javabean。

jsp:setProperty-设置 JavaBean 的属性。 jsp:getProperty-获取 JavaBean 的属性。

jsp:forward-把请求转发到新的页面。

jsp:plugin-产生特定浏览器的代码。

## 98.什么是 Scriptlets？

JSP 技术中，scriptlet 是嵌入在 JSP 页面中的一段 Java 代码。scriptlet 是位于标签内部的所有的东西，在标签与标签之间，用户可以添加任意有效的 scriplet。

## 99.声明(Decalaration)在哪里？

声明跟 Java 中的变量声明很相似，它用来声明随后要被表达式或者 scriptlet 使用的变量。添加的声明必须要用开始和结束标签包起来。

## 100.什么是表达式(Expression)？

JSP 表达式是 Web 服务器把脚本语言表达式的值转化成一个 String 对象，插入到返回给客户端的数据流中。表达式是在<%=和%>这两个标签之间定义的。

## 101.隐含对象是什么意思？有哪些隐含对象？

JSP 隐含对象是页面中的一些 Java 对象，JSP 容器让这些 Java 对象可以为开发者所使用。开发者不用明确的声明就可以直接使用他们。JSP 隐含对象也叫做预定义变量。下面列出了 JSP 页面中的隐含对象：

application 、page 、request 、response 、session 、exception 、out 、config 、pageContext

# JVM

## 102.Jvm如何调优？

答：

观察内存释放情况、集合类检查、对象树

可查看堆空间大小分配（年轻代、年老代、持久代分配）

提供即时的垃圾回收功能

垃圾监控（长时间监控回收情况）

查看堆内类、对象信息查看：数量、类型等

对象引用情况查看

有了堆信息查看方面的功能，我们一般可以顺利解决以下问题：

  --年老代年轻代大小划分是否合理

  --内存泄漏

  --垃圾回收算法设置是否合理

线程信息监控：系统线程数量。

线程状态监控：各个线程都处在什么样的状态下

Dump线程详细信息：查看线程内部运行情况

死锁检查

**内存泄漏检查**

    内存泄漏是比较常见的问题，而且解决方法也比较通用，这里可以重点说一下，而线程、热点方面的问题则是具体问题具体分析了。

    内存泄漏一般可以理解为系统资源（各方面的资源，堆、栈、线程等）在错误使用的情况下，导致使用完毕的资源无法回收（或没有回收），从而导致新的资源分配请求无法完成，引起系统错误。

    内存泄漏对系统危害比较大，因为他可以直接导致系统的崩溃。

需要区别一下，内存泄漏和系统超负荷两者是有区别的，虽然可能导致的最终结果是一样的。内存泄漏是用完的资源没有回收引起错误，而系统超负荷则是系统确实没有那么多资源可以分配了（其他的资源都在使用）

**解决：**

    这种方式解决起来也比较容易，一般就是根据垃圾回收前后情况对比，同时根据对象引用情况（常见的集合对象引用）分析，基本都可以找到泄漏点。

**持久代被占满**

**异常：**java.lang.OutOfMemoryError: PermGen space

**说明：**

    Perm空间被占满。无法为新的class分配存储空间而引发的异常。这个异常以前是没有的，但是在Java反射大量使用的今天这个异常比较常见了。主要原因就是大量动态反射生成的类不断被加载，最终导致Perm区被占满。

    更可怕的是，不同的classLoader即便使用了相同的类，但是都会对其进行加载，相当于同一个东西，如果有N个classLoader那么他将会被加载N次。因此，某些情况下，这个问题基本视为无解。当然，存在大量classLoader和大量反射类的情况其实也不多。

**解决：**

    1. -XX:MaxPermSize=16m

    2. 换用JDK。比如JRocket。

**堆栈溢出**

**异常：**java.lang.StackOverflowError

**说明：**这个就不多说了，一般就是递归没返回，或者循环调用造成

**线程堆栈满**

**异常**：Fatal: Stack size too small

**说明**：java中一个线程的空间大小是有限制的。JDK5.0以后这个值是1M。与这个线程相关的数据将会保存在其中。但是当线程空间满了以后，将会出现上面异常。

**解决**：增加线程栈大小。-Xss2m。但这个配置无法解决根本问题，还要看代码部分是否有造成泄漏的部分。

**系统内存被占满**

**异常**：java.lang.OutOfMemoryError: unable to create new native thread

**说明**：

    这个异常是由于操作系统没有足够的资源来产生这个线程造成的。系统创建线程时，除了要在Java堆中分配内存外，操作系统本身也需要分配资源来创建线程。因此，当线程数量大到一定程度以后，堆中或许还有空间，但是操作系统分配不出资源来了，就出现这个异常了。

分配给Java虚拟机的内存愈多，系统剩余的资源就越少，因此，当系统内存固定时，分配给Java虚拟机的内存越多，那么，系统总共能够产生的线程也就越少，两者成反比的关系。同时，可以通过修改-Xss来减少分配给单个线程的空间，也可以增加系统总共内生产的线程数。

**解决：**

    1. 重新设计系统减少线程数量。

    2. 线程数量不能减少的情况下，通过-Xss减小单个线程大小。以便能生产更多的线程。

## 103.Jvm如何加载类？如何分配空间。

答：

指的是将class文件的二进制数据读入到运行时数据区（JVM在内存中划分的）中，并在方法区内创建一个class对象，JVM运行起来时就给内存划分空间，这块空间就称为运行时数据区。

运行时数据区被划分为以下几块内容

**1.栈：**

每一个线程运行起来的时候就会对应一个栈（线程栈），栈中存放的数据被当前

线程所独享（不会产生资源共享情况，所以线程是安全的）。而栈当中存放的是栈帧，

当线程调用方法时，就是形成一个栈帧，并将这个栈帧进行压栈操作。方法执行完后，

进行出栈操作。这个栈帧里面包括（局部变量，操作数栈，指向当前方法对应类的常

量池引用，方法返回地址等信息）。

**2.本地方法栈：**

　　本地方法栈的机制和栈的相似，区别在于，栈运行的是Java实现的方法，而本地

方法栈运行的是本地方法。本地方法指的是JVM需要调用非Java语言所实现的方法，

例如C语言。在JVM规范中，没有强化性要求实现方一定要划分出本地方法栈（例如：

HotSpot虚拟机将本地方法栈和栈合二为一）和具体实现（不同的操作系统，对JVM

规范的具体实现都不一样）。

**3.程序计数器：**

　　程序计数器也可以称为PC寄存器（通俗讲就是  指令缓存）。它主要用于缓存当前

程序下一条指令的指令地址，CPU根据这个地址找到将要执行的指令。这个寄存器是JVM

内部实现的，不是物理概念上的计数器，不过和JVM的实现逻辑一样。

**4.堆：**

　　堆内存主要存放创建的对象和数组。堆内存在JVM中是唯一的，能被多个线程所共享。

堆里面的每一个对象都存放着实例的实例变量。堆内存的对象没有被引用，会自动被Java

垃圾回收机制回收。

　　当在方法中定义了局部变量，如果这个变量是基本数据类型，那么这个变量的值就直接

存放在栈中；如果这个变量是引用数据类型，那么变量值就存放在堆内存中，而栈中存放的是

指向堆中的引用地址。

**5.方法区：**

　　方法区在JVM也是一个非常重要的一块内存区域，和堆一样，可以被多个线程多共享。

主要存放每一个加载class的信息。class信息主要包含魔数（确定是否是一个class文件），常量

池，访问标志（当前的类是普通类还是接口，是否是抽象类，是否被public修饰，是否使用了final

修饰等描述信息......），字段表集合信息（使用什么访问修饰符，是实例变量还是静态变量，是否

使用了final修饰等描述信息.....），方法表集合信息（使用什么访问修饰符，是否静态方法，是否

使用了final修饰，是否使用了synchronized修饰，是否是native方法......）等内容。当一个类加

载器加载了一个类的时候，会根据这个class文件创建一个class对象，class对象就包含了上述的信息。

后续要创建这个类的实例，都根据这个class对象创建出来的。

**6.常量池：**

　　常量池是方法区中的一部分，存放class对象中最重要的资源。JVM为每一个class对象都维护一个

常量池。