JAVA 面试题集

基础知识：

1. C++或 Java 中的异常处理机制的简单原理和应用。

当 JAVA 程序违反了 JAVA 的语义规则时，JAVA 虚拟机就会将发生的错误表示为一个异常。违反语义规则包括 2 种情况。一种是 JAVA 类库内置的语义检查。例如数组下标越界,会引发 IndexOutOfBoundsException;访问 null 的对象时会引发 NullPointerException。另一种情况就是 JAVA 允许程序员扩展这种语义检查，程序员可以创建自己的异常，并自由选择在何时用 throw 关键字引发异常。所有的异常都是 java.lang.Thowable 的子类。

2. Java 的接口和 C++的虚类的相同和不同处。

由于 Java 不支持多继承，而有可能某个类或对象要使用分别在几个类或对象里面的方法或属性，现有的单继承机制就不能满足要求。与继承相比，接口有更高的灵活性，因为接口中没有任何实现代码。当一个类实现了接口以后，该类要实现接口里面所有的方法和属性，并且接口里面的属性在默认状态下面都是 public static,所有方法默认情况下是 public.一个类可以实现多个接口。

3. 垃圾回收的优点和原理。并考虑 2 种回收机制。

Java 语言中一个显著的特点就是引入了垃圾回收机制，使 c++程序员最头疼的内存管理的问题迎刃而解，它使得 Java 程序员在编写程序的时候不再需要考虑内存管理。由于有个垃圾回收机制，Java 中的对象不再有“作用域”的概念，只有对象的引用才有“作用域”。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低级别的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清楚和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。回收机制有分代复制垃圾回收和标记垃圾回收，增量垃圾回收。

4. 请说出你所知道的线程同步的方法。

wait():使一个线程处于等待状态，并且释放所持有的对象的 lock。

sleep(): 使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要捕捉 InterruptedException 异常。

notify():唤醒一个处于等待状态的线程，注意的是在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由 JVM 确定唤醒哪个线程，而且不是按优先级。

Allnotity():唤醒所有处入等待状态的线程，注意并不是给所有唤醒线程一个对象的锁，而是让它们竞争。

5. 请讲一讲析构函数和虚函数的用法和作用。

6. Error 与 Exception 有什么区别？

Error 表示系统级的错误和程序不必处理的异常，

Exception 表示需要捕捉或者需要程序进行处理的异常。

7. 在 java 中一个类被声明为 final 类型，表示了什么意思？

表示该类不能被继承，是顶级类。

8. 描述一下你最常用的编程风格。

9. heap 和 stack 有什么区别。

栈是一种线形集合，其添加和删除元素的操作应在同一段完成。栈按照后进先出的方式进行处理。

堆是栈的一个组成元素

10. 如果系统要使用超大整数（超过 long 长度范围），请你设计一个数据结构来存储这种超大型数字以及设计一种算法来实现超大整数加法运算）。

```
public class BigInt()
{
   int[] ArrOne = new ArrOne[1000];
   String intString="";
public int[] Arr(String s)
{
   intString = s;
   for(int i=0;i<ArrOne.leght;i++)
{
```

11. 如果要设计一个图形系统，请你设计基本的图形元件(Point,Line,Rectangle,Triangle)的简单实现

12，谈谈 final，finally，finalize 的区别。

final?修饰符（关键字）如果一个类被声明为 final，意味着它不能再派生出新的子类，不能作为父类被继承。因此一个类不能既被声明为 abstract 的，又被声明为 final 的。将变量或方法声明为 final，可以保证它们在使用中不被改变。被声明为 final 的变量必须在声明时给定初值，而在以后的引用中只能读取，不可修改。被声明为 final 的方法也同样只能使用，不能重载。

finally?再异常处理时提供 finally 块来执行任何清除操作。如果抛出一个异常，那么相匹配的 catch 子句就会执行，然后控制就会进入 finally 块（如果有的话）。

finalize?方法名。Java 技术允许使用 finalize() 方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 Object 类中定义的，因此所有的类都继承了它。子类覆盖 finalize() 方法以整理系统资源或者执行其他清理工作。finalize() 方法是在垃圾收集器删除对象之前对这个对象调用的。

13，Anonymous Inner Class (匿名内部类) 是否可以 extends(继承)其它类，是否可以 implements(实现) interface(接口)？

匿名的内部类是没有名字的内部类。不能 extends(继承) 其它类，但一个内部类可以作为一个接口，由另一个内部类实现。

14，Static Nested Class 和 Inner Class 的不同，说得越多越好(面试题有的很笼统)。

Nested Class （一般是 C++的说法），Inner Class (一般是 JAVA 的说法)。Java 内部类与 C++嵌套类最大的不同就在于是否有指向外部的引用上。具体可见 http://www.frontfree.net/articles/services/view.asp?id=704&page=1

注： 静态内部类（Inner Class）意味着 1 创建一个 static 内部类的对象，不需要一个外部类对象，2 不能从一个 static 内部类的一个对象访问一个外部类对象

补充：&和&&的区别。

&是位运算符。&&是布尔逻辑运算符。

15，HashMap 和 Hashtable 的区别。

都属于 Map 接口的类，实现了将惟一键映射到特定的值上。

HashMap 类没有分类或者排序。它允许一个 null 键和多个 null 值。

Hashtable 类似于 HashMap，但是不允许 null 键和 null 值。它也比 HashMap 慢，因为它是同步的。

16，Collection 和 Collections 的区别。

Collections 是个 java.util 下的类，它包含有各种有关集合操作的静态方法。

Collection 是个 java.util 下的接口，它是各种集合结构的父接口。

17，什么时候用 assert。

断言是一个包含布尔表达式的语句，在执行这个语句时假定该表达式为 true。如果表达式计算为 false，那么系统会报告一个 Assertionerror。它用于调试目的：

assert(a > 0); // throws an Assertionerror if a <= 0

断言可以有两种形式：

assert Expression1 ;

assert Expression1 : Expression2 ;

Expression1 应该总是产生一个布尔值。

Expression2 可以是得出一个值的任意表达式。这个值用于生成显示更多调试信息的 String 消息。

断言在默认情况下是禁用的。要在编译时启用断言，需要使用 source 1.4 标记：

javac -source 1.4 Test.java

要在运行时启用断言，可使用 -enableassertions 或者 -ea 标记。

要在运行时选择禁用断言，可使用 -da 或者 -disableassertions 标记。

要系统类中启用断言，可使用 -esa 或者 -dsa 标记。还可以在包的基础上启用或者禁用断言。

可以在预计正常情况下不会到达的任何位置上放置断言。断言可以用于验证传递给私有方法的参数。不过，断言不应该用于验证传递给公有方法的参数，因为不管是否启用了断言，公有方法都必须检查其参数。不过，既可以在公有方法中，也可以在非公有方法中利用断言测试后置条件。另外，断言不应该以任何方式改变程序的状态。

18，GC 是什么？为什么要有 GC？（基础）。

GC 是垃圾收集器。Java 程序员不用担心内存管理，因为垃圾收集器会自动进行管理。要请求垃圾收集，可以调用下面的方法之一：

System.gc()

Runtime.getRuntime().gc()

19，String s = new String(“xyz”);创建了几个 String Object?

两个对象，一个是“xyx”,一个是指向“xyx”的引用对象 s。

20，Math.round(11.5)等於多少？Math.round(-11.5)等於多少？

Math.round(11.5)返回（long）12，Math.round(-11.5)返回（long）-11;

21，short s1 = 1; s1 = s1 + 1;有什么错？short s1 = 1; s1 += 1;有什么错？

short s1 = 1; s1 = s1 + 1;有错，s1 是 short 型，s1+1 是 int 型,不能显式转化为 short 型。可修改为 s1 =(short)(s1 + 1) 。short s1 = 1; s1 += 1 正确。

22，sleep() 和 wait() 有什么区别？搞线程的最爱

sleep()方法是使线程停止一段时间的方法。在 sleep 时间间隔期满后，线程不一定立即恢复执行。这是因为在那个时刻，其它线程可能正在运行而且没有被调度为放弃执行，除非(a)“醒来”的线程具有更高的优先级 (b)正在运行的线程因为其它原因而阻塞。

wait()是线程交互时，如果线程对一个同步对象 x 发出一个 wait()调用，该线程会暂停执行，被调对象进入等待状态，直到被唤醒或等待时间到。

23，Java 有没有 goto?

Goto?java 中的保留字，现在没有在 java 中使用。

24，数组有没有 length()这个方法？String 有没有 length()这个方法？

数组没有 length()这个方法，有 length 的属性。

String 有 length()这个方法。

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

方法的重写 Overriding 和重载 Overloading 是 Java 多态性的不同表现。重写 Overriding 是父类与子类之间多态性的一种表现，重载 Overloading 是一个类中多态性的一种表现。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。子类的对象使用这个方法时，将调用子类

中的定义，对它而言，父类中的定义如同被“屏蔽”了。如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。Overloaded 的方法是可以改变返回值的类型。

26，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?

Set 里的元素是不能重复的，那么用 iterator()方法来区分重复与否。equals()是判读两个 Set 是否相等。

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。

27，给我一个你最常见到的 runtime exception。

ArithmeticException, ArrayStoreException, BufferOverflowException, BufferUnderflowException, CannotRedoException, CannotUndoException, ClassCastException, CMMException, ConcurrentModificationException, DOMException, EmptyStackException, IllegalArgumentException, IllegalMonitorStateException, IllegalPathStateException, IllegalStateException,

ImagingOpException, IndexOutOfBoundsException, MissingResourceException, NegativeArraySizeException, NoSuchElementException, NullPointerException, ProfileDataException, ProviderException, RasterFORMatException, SecurityException, SystemException, UndeclaredThrowableException, UnmodifiableSetException, UnsupportedOperationException

28，error 和 exception 有什么区别?

error 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。

exception 表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。

29，List，Set，Map 是否继承自 Collection 接口?

List，Set 是

Map 不是

30，abstract class 和 interface 有什么区别?

声明方法的存在而不去实现它的类被叫做抽象类（abstract class），它用于要创建一个体现某些基本行为的类，并为该类声明方法，但不能在该类中实现该类的情况。不能创建 abstract 类的实例。然而可以创建一个变量，其类型是一个抽象类，并让它指向具体子类的一个实例。不能有抽象构造函数或抽象静态方法。Abstract 类的子类为它们父类中的所有抽象方法提供实现，否则它们也是抽象类为。取而代之，在子类中实现该方法。知道其行为的其它类可以在类中实现这些方法。

接口（interface）是抽象类的变体。在接口中，所有方法都是抽象的。多继承性可通过实现这样的接口而获得。接口中的所有方法都是抽象的，没有一个有程序体。接口只可以定义 static final 成员变量。接口的实现与子类相似，除了该实现类不能从接口定义中继承行为。当类实现特殊接口时，它定义（即将程序体给予）所有这种接口的方法。然后，它可以在实现了该接口的类的任何对象上调用接口的方法。由于有抽象类，它允许使用接口名作为引用变量的类型。通常的动态联编将生效。引用可以转换到接口类型或从接口类型转换，instanceof 运算符可以用来决定某对象的类是否实现了接口。

31，abstract 的 method 是否可同时是 static,是否可同时是 native，是否可同时是 synchronized?

都不能

32，接口是否可继承接口？抽象类是否可实现(implements)接口？抽象类是否可继承实体类(concrete class)？

接口可以继承接口。抽象类可以实现(implements)接口，抽象类是否可继承实体类，但前提是实体类必须有明确的构造函数。

33，启动一个线程是用 run()还是 start()？

启动一个线程是调用 start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由 JVM 调度并执行。这并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。

34，构造器 Constructor 是否可被 override?

构造器 Constructor 不能被继承，因此不能重写 Overriding，但可以被重载 Overloading。

35，是否可以继承 String 类?

String 类是 final 类故不可以继承。

36，当一个线程进入一个对象的一个 synchronized 方法后，其它线程是否可进入此对象的其它方法?

不能，一个对象的一个 synchronized 方法只能由一个线程访问。

37，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?

会执行，在 return 前执行。

38，编程题: 用最有效率的方法算出 2 乘以 8 等於几?

有 C 背景的程序员特别喜欢问这种问题。

2 << 3

39，两个对象值相同(x.equals(y) == true)，但却可有不同的 hashcode，这句话对不对?

不对，有相同的 hashcode。

40，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?

是值传递。Java 编程语言只由值传递参数。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。

41，swtich 是否能作用在 byte 上，是否能作用在 long 上，是否能作用在 String 上?

switch（expr1）中，expr1 是一个整数表达式。因此传递给 switch 和 case 语句的参数应该是 int、short、 char 或者 byte。long,string 都不能作用于 swtich。

42，编程题: 写一个 Singleton 出来。

Singleton 模式主要作用是保证在 Java 应用程序中，一个类 Class 只有一个实例存在。

一般 Singleton 模式通常有几种种形式：

第一种形式：定义一个类，它的构造函数为 private 的，它有一个 static 的 private 的该类变量，在类初始化时实例话，通过一个 public 的 getInstance 方法获取对它的引用,继而调用其中的方法。

```
public class Singleton {
    private Singleton(){}
    //在自己内部定义自己一个实例，是不是很奇怪？
    //注意这是 private 只供内部调用
    private static Singleton instance = new Singleton();
    //这里提供了一个供外部访问本 class 的静态方法，可以直接访问
    public static Singleton getInstance() {
        return instance;
     }
}
```

第二种形式：

```
public class Singleton {
    private static Singleton instance = null;
    public static synchronized Singleton getInstance() {
    //这个方法比上面有所改进，不用每次都进行生成对象，只是第一次
    //使用时生成实例，提高了效率！
    if (instance==null)
        instance＝new Singleton();
return instance;        }
}
```

其他形式：

定义一个类，它的构造函数为 private 的，所有方法为 static 的。

一般认为第一种形式要更加安全些

Hashtable 和 HashMap

Hashtable 继承自 Dictionary 类，而 HashMap 是 Java1.2 引进的 Map interface 的一个实现

HashMap 允许将 null 作为一个 entry 的 key 或者 value，而 Hashtable 不允许

还有就是，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。

最大的不同是，Hashtable 的方法是 Synchronize 的，而 HashMap 不是，在多个线程访问 Hashtable 时，不需要自己为它的方法实现同步，而 HashMap 就必须为之提供外同步。

Hashtable 和 HashMap 采用的 hash/rehash 算法都大概一样，所以性能不会有很大的差异。

43. 描述一下 JVM 加载 class 文件的原理机制?

44. 试举例说明一个典型的垃圾回收算法？

45. 请用 java 写二叉树算法，实现添加数据形成二叉树功能，并以先序的方式打印出来.

46. 请写一个 java 程序实现线程连接池功能？

47. 给定一个 C 语言函数，要求实现在 java 类中进行调用。

48、编一段代码，实现在控制台输入一组数字后，排序后在控制台输出；

49、列出某文件夹下的所有文件；

50、调用系统命令实现删除文件的操作；

51、实现从文件中一次读出一个字符的操作；

52、列出一些控制流程的方法；

53、多线程有哪些状态？

54、编写了一个服务器端的程序实现在客户端输入字符然后在控制台上显示，直到输入"END"为止，让你写出客户端的程序；

55、作用域 public,private,protected,以及不写时的区别

区别如下：

作用域 当前类 同一 package 子孙类 其他 package

public √ √ √ √

protected √ √ √ ×

friendly √ √ × ×

private √ × × ×

不写时默认为 friendly

56、ArrayList 和 Vector 的区别,HashMap 和 Hashtable 的区别

就 ArrayList 与 Vector 主要从二方面来说：

一.同步性:Vector 是线程安全的，也就是说是同步的，而 ArrayList 是线程序不安全的，不是同步的

二.数据增长:当需要增长时,Vector 默认增长为原来一培，而 ArrayList 却是原来的一半

就 HashMap 与 HashTable 主要从三方面来说：

一.历史原因:Hashtable 是基于陈旧的 Dictionary 类的,HashMap 是 Java 1.2 引进的 Map 接口的一个实现

二.同步性:Hashtable 是线程安全的，也就是说是同步的，而 HashMap 是线程序不安全的,不是同步的

三.值：只有 HashMap 可以让你将空值作为一个表的条目的 key 或 value

57、char 型变量中能不能存贮一个中文汉字?为什么?

是能够定义成为一个中文的，因为 java 中以 unicode 编码，一个 char 占 16 个字节，所以放一个中文是没问题的

58、多线程有几种实现方法,都是什么?同步有几种实现方法,都是什么?

多线程有两种实现方法，分别是继承 Thread 类与实现 Runnable 接口

同步的实现方面有两种，分别是 synchronized,wait 与 notify

59、垃圾回收机制,如何优化程序?

希望大家补上，谢谢

60、float 型 float f=3.4 是否正确?

不正确。精度不准确,应该用强制类型转换，如下所示：float f=(float)3.4

61、介绍 JAVA 中的 Collection FrameWork(包括如何写自己的数据结构)？

Collection FrameWork 如下：

Collection

├List
│ ├LinkedList
│ ├ArrayList
│ └Vector
│ └Stack
└Set

Map

├Hashtable
├HashMap
└WeakHashMap

Collection 是最基本的集合接口，一个 Collection 代表一组 Object，即 Collection 的元素（Elements）
Map 提供 key 到 value 的映射

62、Java 中异常处理机制，事件机制？

11、JAVA 中的多形与继承？

希望大家补上，谢谢

63、抽象类与接口？

答：抽象类与接口都用于抽象，但是抽象类(JAVA 中)可以有自己的部分实现，而接口则完全是一个标识(同时有多重继承的功能)。

编程题：

1．现在输入 n 个数字，以逗号，分开； 然后可选择升或者降序排序；按提交键就在另一页面显示 按什么排序，结果为， ， 提供 reset

答案（1） public static String[] splitStringByComma(String source)

```
{
    if(source==null||source.trim().equals(“”))
    return null;
StringTokenizer commaToker = new StringTokenizer(source,”,”);
String[] result = new String[commaToker.countTokens()];
int i=0;
  while(commaToker.hasMoreTokens())
{
      result[i] = commaToker.nextToken();
      i++;
  }
 return result;
}
```

循环遍历 String 数组

Integer.parseInt(String s)变成 int 类型

组成 int 数组

Arrays.sort(int[] a),

a 数组升序

降序可以从尾部开始输出

2．金额转换，阿拉伯数字的金额转换成中国传统的形式如：

（￥1011）—>（一千零一拾一元整）输出。

3、继承时候类的执行顺序问题,一般都是选择题,问你将会打印出什么？

答:父类：

```
package test;
public class FatherClass
{
public FatherClass()
{
System.out.println(“FatherClass Create”);
}
}
```

子类:

```
package test;
import test.FatherClass;
public class ChildClass extends FatherClass
{
public ChildClass()
{
System.out.println(“ChildClass Create”);
}
public static void main(String[] args)
```

```
{
FatherClass fc = new FatherClass();
ChildClass cc = new ChildClass();
}
}
```
输出结果：
```
C:>java test.ChildClass
FatherClass Create
FatherClass Create
ChildClass Create
```
4、内部类的实现方式?

答：示例代码如下：
```
package test;
public class OuterClass
{
   private class InterClass
   {
      public InterClass()
      {
      System.out.println(“InterClass Create”);
      }
   }
public OuterClass()
   {
   InterClass ic = new InterClass();
   System.out.println(“OuterClass Create”);
   }
public static void main(String[] args)
   {
   OuterClass oc = new OuterClass();
   }
}
```
输出结果:
```
C:>java test/OuterClass
InterClass Create
OuterClass Create
```
再一个例题：
```
public class OuterClass
{
   private double d1 = 1.0;
//insert code here
}
```
You need to insert an inner class declaration at line 3. Which two inner class declarations are valid?(Choose two.)

A.
```
class InnerOne{
public static double methoda() {return d1;}
}
```

B. public class InnerOne
{
static double methoda() {return d1;}
}
C. private class InnerOne{
double methoda() {return d1;}
}
D. static class InnerOne
{
protected double methoda() {return d1;}
}
E. abstract class InnerOne{
public abstract double methoda();
}

说明如下：
一.静态内部类可以有静态成员，而非静态内部类则不能有静态成员。 故 A、B 错
二.静态内部类的非静态成员可以访问外部类的静态变量，而不可访问外部类的非静态变量；return d1 出错。
故 D 错
三.非静态内部类的非静态成员可以访问外部类的非静态变量。 故 C 正确
四.答案为 C、E

5、Java 的通信编程，编程题(或问答)，用 JAVA SOCKET 编程，读服务器几个字符，再写入本地显示？

答:Server 端程序:
```
package test;
import java.net.*;
import java.io.*;
public class Server
{
private ServerSocket ss;
private Socket socket;
private BufferedReader in;
private PrintWriter out;
public Server()
{
try
{
ss=new ServerSocket(10000);
while(true)
{
socket = ss.accept();
String RemoteIP = socket.getInetAddress().getHostAddress();
String RemotePort = ":"+socket.getLocalPort();
System.out.println("A client come in!IP:"+RemoteIP+RemotePort);
in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
String line = in.readLine();
```

```
System.out.println(“Cleint send is :” + line);
out = new PrintWriter(socket.getOutputStream(),true);
out.println(“Your Message Received!”);
out.close();
in.close();
socket.close();
}
}catch (IOException e)
{
out.println(“wrong”);
}
}
public static void main(String[] args)
{
new Server();
}
};
```

Client 端程序:

```
package test;
import java.io.*;
import java.net.*;
public class Client
{
Socket socket;
BufferedReader in;
PrintWriter out;
public Client()
{
try
{
System.out.println(“Try to Connect to 127.0.0.1:10000”);
socket = new Socket(“127.0.0.1”,10000);
System.out.println(“The Server Connected!”);
System.out.println(“Please enter some Character:”);
BufferedReader line = new BufferedReader(new
InputStreamReader(System.in));
out = new PrintWriter(socket.getOutputStream(),true);
out.println(line.readLine());
in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
System.out.println(in.readLine());
out.close();
in.close();
socket.close();
}catch(IOException e)
{
out.println(“Wrong”);
}
```

```
}
public static void main(String[] args)
{
new Client();
}
};
```

6、用 JAVA 实现一种排序，JAVA 类实现序列化的方法(二种)？ 如在 COLLECTION 框架中，实现比较要实现什么样的接口？

答:用插入法进行排序代码如下

```
package test;
import java.util.*;
class InsertSort
{
ArrayList al;
public InsertSort(int num,int mod)
{
al = new ArrayList(num);
Random rand = new Random();
System.out.println("The ArrayList Sort Before:");
for (int i=0;i<num ;i++ )
{
al.add(new Integer(Math.abs(rand.nextInt()) % mod + 1));
System.out.println("al["+i+"]="+al.get(i));
}
}
public void SortIt()
{
Integer tempInt;
int MaxSize=1;
for(int i=1;i<al.size();i++)
{
tempInt = (Integer)al.remove(i);
if(tempInt.intValue()>=((Integer)al.get(MaxSize-1)).intValue())
{
al.add(MaxSize,tempInt);
MaxSize++;
System.out.println(al.toString());
} else {
for (int j=0;j<MaxSize ;j++ )
{
if
(((Integer)al.get(j)).intValue()>=tempInt.intValue())
{
al.add(j,tempInt);
MaxSize++;
System.out.println(al.toString());
break;
```

```
}
}
}
}
System.out.println("The ArrayList Sort After:");
for(int i=0;i<al.size();i++)
{
System.out.println("al["+i+"]="+al.get(i));
}
}
public static void main(String[] args)
{
InsertSort is = new InsertSort(10,100);
is.SortIt();
}
}
```

JAVA 类实现序例化的方法是实现 java.io.Serializable 接口

Collection 框架中实现比较要实现 Comparable 接口和 Comparator 接口

7、编程：编写一个截取字符串的函数，输入为一个字符串和字节数，输出为按字节截取的字符串。 但是要保证汉字不被截半个，如“我 ABC”4，应该截为“我 AB”，输入“我 ABC 汉 DEF”，6，应该输出为“我 ABC”而不是“我 ABC+汉的半个”。

答：代码如下:

```
package test;
class SplitString
{
String SplitStr;
int SplitByte;
public SplitString(String str,int bytes)
{
SplitStr=str;
SplitByte=bytes;
System.out.println("The String is:′ "+SplitStr+"′ ;SplitBytes="+SplitByte);
}
public void SplitIt()
{
int loopCount;
loopCount=(SplitStr.length()%SplitByte==0)?(SplitStr.length()/SplitByte):(SplitStr.length()/SplitByte+1);
System.out.println("Will Split into "+loopCount);
for (int i=1;i<=loopCount ;i++)
{
if (i==loopCount){
System.out.println(SplitStr.substring((i-1)*SplitByte,SplitStr.length()));
}
else
{
System.out.println(SplitStr.substring((i-1)*SplitByte,(i*SplitByte)));
}
```

```
}
}
public static void main(String[] args)
{
SplitString ss = new SplitString("test 中 dd 文 dsaf 中男大 3443n 中国 43 中国人 0ewldfls=103",4);
ss.SplitIt();
}
}
```

8、JAVA 多线程编程。 用 JAVA 写一个多线程程序，如写四个线程，二个加 1，二个对一个变量减一，输出。

希望大家补上，谢谢

9、STRING 与 STRINGBUFFER 的区别。

答：STRING 的长度是不可变的，STRINGBUFFER 的长度是可变的。如果你对字符串中的内容经常进行操作，特别是内容要修改时，那么使用 StringBuffer，如果最后需要 String，那么使用 StringBuffer 的 toString()方法

Jsp 方面

1、jsp 有哪些内置对象?作用分别是什么?

答:JSP 共有以下 9 种基本内置组件（可与 ASP 的 6 种内部组件相对应）：

request 用户端请求，此请求会包含来自 GET/POST 请求的参数

response 网页传回用户端的回应

pageContext 网页的属性是在这里管理

session 与请求有关的会话期

application servlet 正在执行的内容

out 用来传送回应的输出

config servlet 的构架部件

page JSP 网页本身

exception 针对错误网页，未捕捉的例外

2、jsp 有哪些动作?作用分别是什么?

答:JSP 共有以下 6 种基本动作

jsp:include：在页面被请求的时候引入一个文件。

jsp:useBean：寻找或者实例化一个 JavaBean。

jsp:setProperty：设置 JavaBean 的属性。

jsp:getProperty：输出某个 JavaBean 的属性。

jsp:forward：把请求转到一个新的页面。

jsp:plugin：根据浏览器类型为 Java 插件生成 OBJECT 或 EMBED 标记

3、JSP 中动态 INCLUDE 与静态 INCLUDE 的区别？

答：动态 INCLUDE 用 jsp:include 动作实现

<jsp:include page="included.jsp" flush="true" />它总是会检查所含文件中的变化，适合用于包含动态页面，并且可以带参数

静态 INCLUDE 用 include 伪码实现,定不会检查所含文件的变化，适用于包含静态页面

<%@ include file="included.htm" %>

4、两种跳转方式分别是什么?有什么区别?

答：有两种，分别为：

<jsp:include page="included.jsp" flush="true">

<jsp:forward page= "nextpage.jsp"/>

前者页面不会转向 include 所指的页面，只是显示该页的结果，主页面还是原来的页面。执行完后还会回来，相当于函数调用。并且可以带参数.后者完全转向新页面，不会再回来。相当于 go to 语句。

Servlet 方面

1、说一说 Servlet 的生命周期?

答:servlet 有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。这个生存期由 javax.servlet.Servlet 接口的 init,service 和 destroy 方法表达。

2、Servlet 版本间(忘了问的是哪两个版本了)的不同?

希望大家补上，谢谢

3、JAVA SERVLET API 中 forward() 与 redirect()的区别？

答:前者仅是容器中控制权的转向，在客户端浏览器地址栏中不会显示出转向后的地址；后者则是完全的跳转，浏览器将会得到跳转的地址，并重新发送请求链接。这样，从浏览器的地址栏中可以看到跳转后的链接地址。所以，前者更加高效，在前者可以满足需要时，尽量使用 forward()方法，并且，这样也有助于隐藏实际的链接。在有些情况下，比如，需要跳转到一个其它服务器上的资源，则必须使用 sendRedirect()方法。

4、Servlet 的基本架构

```
public class ServletName extends HttpServlet
{
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
    }
}
```

Jdbc、Jdo 方面

1、可能会让你写一段 Jdbc 连 Oracle 的程序,并实现数据查询.

答:程序如下：

```
package hello.ant;
import java.sql.*;
public class jdbc
{
String dbUrl="jdbc:oracle:thin:@127.0.0.1:1521:orcl";
String theUser="admin";
String thePw="manager";
Connection c=null;
Statement conn;
ResultSet rs=null;
public jdbc()
{
try{
```

```
Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
c = DriverManager.getConnection(dbUrl,theUser,thePw);
conn=c.createStatement();
}catch(Exception e){
e.printStackTrace();
}
}
public boolean executeUpdate(String sql)
{
try
{
conn.executeUpdate(sql);
return true;
}
catch (SQLException e)
{
e.printStackTrace();
return false;
}
}
public ResultSet executeQuery(String sql)
{
rs=null;
try
{
rs=conn.executeQuery(sql);
}
catch (SQLException e)
{
e.printStackTrace();
}
return rs;
}
public void close()
{
try
{
conn.close();
c.close();
}
catch (Exception e)
{
e.printStackTrace();
}
}
public static void main(String[] args)
{
```

```
ResultSet rs;
jdbc conn = new jdbc();
rs=conn.executeQuery("select * from test");
try{
while (rs.next())
{
System.out.println(rs.getString("id"));
System.out.println(rs.getString("name"));
}
}catch(Exception e)
{
e.printStackTrace();
}
}
}
```

2、Class.forName 的作用?为什么要用?

答：调用该访问返回一个以字符串指定类名的类的对象。

3、Jdo 是什么?

答:JDO 是 Java 对象持久化的新的规范，为 java data object 的简称,也是一个用于存取某种数据仓库中的对象的标准化 API。JDO 提供了透明的对象存储，因此对开发人员来说，存储数据对象完全不需要额外的代码（如 JDBC API 的使用）。这些繁琐的例行工作已经转移到 JDO 产品提供商身上，使开发人员解脱出来，从而集中时间和精力在业务逻辑上。另外，JDO 很灵活，因为它可以在任何数据底层上运行。JDBC 只是面向关系数据库（RDBMS）JDO 更通用，提供到任何数据底层的存储功能，比如关系数据库、文件、XML 以及对象数据库（ODBMS）等等，使得应用可移植性更强。

4、在 ORACLE 大数据量下的分页解决方法。一般用截取 ID 方法，还有是三层嵌套方法。

答:一种分页方法

```
<%
int i=1;
int numPages=14;
String pages = request.getParameter("page") ;
int currentPage = 1;
currentPage=(pages==null)?(1):{Integer.parseInt(pages)}
sql = "select count(*) from tables";
ResultSet rs = DBLink.executeQuery(sql) ;
while(rs.next()) i = rs.getInt(1) ;
int intPageCount=1;
intPageCount=(i%numPages==0)?(i/numPages):(i/numPages+1);
int nextPage ;
int upPage;
nextPage = currentPage+1;
if (nextPage>=intPageCount) nextPage=intPageCount;
upPage = currentPage-1;
if (upPage<=1) upPage=1;
rs.close();
```

```
sql="select * from tables";
rs=DBLink.executeQuery(sql);
i=0;
while((i<numPages*(currentPage-1))&&rs.next()){i++;}
%>
//输出内容
//输出翻页连接
合计:<%=currentPage%>/<%=intPageCount%><a href="List.jsp?page=1">第一页</a><a
href="List.jsp?page=<%=upPage%>">上一页</a>
<%
for(int j=1;j<=intPageCount;j++){
if(currentPage!=j){
%>
<a href="list.jsp?page=<%=j%>">[<%=j%>]</a>
<%
}else{
out.println(j);
}
}
%>
<a href="List.jsp?page=<%=nextPage%>">下一页</a><a href="List.jsp?page=<%=intPageCount%>">最后页
</a>
```

Xml 方面

1、xml 有哪些解析技术?区别是什么?

答:有 DOM,SAX,STAX 等

DOM:处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内存,适合对 XML 的随机访问 SAX:不现于 DOM,SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件，不需要一次全部装载整个文件。当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理 XML 文件，适合对 XML 的顺序访问

STAX:Streaming API for XML (StAX)

2、你在项目中用到了 xml 技术的哪些方面?如何实现的?

答:用到了数据存贮，信息配置两方面。在做数据交换平台时，将不能数据源的数据组装成 XML 文件，然后将 XML 文件压缩打包加密后通过网络传送给接收者，接收解密与解压缩后再同 XML 文件中还原相关信息进行处理。在做软件配置时，利用 XML 可以很方便的进行，软件的各种配置参数都存贮在 XML 文件中。

3、用 jdom 解析 xml 文件时如何解决中文问题?如何解析?

答:看如下代码,用编码方式加以解决

```
package test;
import java.io.*;
public class DOMTest
{
private String inFile = "c:\people.xml";
private String outFile = "c:\people.xml";
public static void main(String args[])
```

```
{
new DOMTest();
}
public DOMTest()
{
try
{
javax.xml.parsers.DocumentBuilder builder =
javax.xml.parsers.DocumentBuilderFactory.newInstance().newDocumentBuilder();
org.w3c.dom.Document doc = builder.newDocument();
org.w3c.dom.Element root = doc.createElement("老师");
org.w3c.dom.Element wang = doc.createElement("王");
org.w3c.dom.Element liu = doc.createElement("刘");
wang.appendChild(doc.createTextNode("我是王老师"));
root.appendChild(wang);
doc.appendChild(root);
javax.xml.transform.Transformer transformer =
javax.xml.transform.TransformerFactory.newInstance().newTransformer();
transformer.setOutputProperty(javax.xml.transform.OutputKeys.ENCODING, "gb2312");
transformer.setOutputProperty(javax.xml.transform.OutputKeys.INDENT, "yes");

transformer.transform(new javax.xml.transform.dom.DOMSource(doc),
new
javax.xml.transform.stream.StreamResult(outFile));
}
catch (Exception e)
{
System.out.println (e.getMessage());
}
}
}
```

4、编程用 JAVA 解析 XML 的方式.

答:用 SAX 方式解析 XML，XML 文件如下：

```
<?xml version="1.0" encoding="gb2312"?>
<person>
<name>王小明</name>
<college>信息学院</college>
<telephone>6258113</telephone>
<notes>男,1955 年生,博士，95 年调入海南大学</notes>
</person>
```

事件回调类 SAXHandler.java

```
import java.io.*;
import java.util.Hashtable;
import org.xml.sax.*;
public class SAXHandler extends HandlerBase
{
private Hashtable table = new Hashtable();
```

```
private String currentElement = null;
private String currentValue = null;
public void setTable(Hashtable table)
{
this.table = table;
}
public Hashtable getTable()
{
return table;
}
public void startElement(String tag, AttributeList attrs)
throws SAXException
{
currentElement = tag;
}
public void characters(char[] ch, int start, int length)
throws SAXException
{
currentValue = new String(ch, start, length);
}
public void endElement(String name) throws SAXException
{
if (currentElement.equals(name))
table.put(currentElement, currentValue);
}
}
```

JSP 内容显示源码,SaxXml.jsp:

```
<HTML>
<HEAD>
<TITLE>剖析 XML 文件 people.xml</TITLE>
</HEAD>
<BODY>
<%@ page errorPage="ErrPage.jsp"
contentType="text/html;charset=GB2312" %>
<%@ page import="java.io.*" %>
<%@ page import="java.util.Hashtable" %>
<%@ page import="org.w3c.dom.*" %>
<%@ page import="org.xml.sax.*" %>
<%@ page import="javax.xml.parsers.SAXParserFactory" %>
<%@ page import="javax.xml.parsers.SAXParser" %>
<%@ page import="SAXHandler" %>
<%
File file = new File("c:\people.xml");
FileReader reader = new FileReader(file);
Parser parser;
SAXParserFactory spf = SAXParserFactory.newInstance();
```

```
SAXParser sp = spf.newSAXParser();
SAXHandler handler = new SAXHandler();
sp.parse(new InputSource(reader), handler);
Hashtable hashTable = handler.getTable();
out.println("<TABLE BORDER=2><CAPTION>教师信息表</CAPTION>");
out.println("<TR><TD>姓名</TD>" + "<TD>" +
(String)hashTable.get(new String("name")) + "</TD></TR>");
out.println("<TR><TD>学院</TD>" + "<TD>" +
(String)hashTable.get(new String("college"))+"</TD></TR>");
out.println("<TR><TD>电话</TD>" + "<TD>" +
(String)hashTable.get(new String("telephone")) + "</TD></TR>");
out.println("<TR><TD>备注</TD>" + "<TD>" +
(String)hashTable.get(new String("notes")) + "</TD></TR>");
out.println("</TABLE>");
%>
</BODY>
</HTML>
```

EJB 方面

1、EJB2.0 有哪些内容?分别用在什么场合? EJB2.0 和 EJB1.1 的区别?

答：规范内容包括 Bean 提供者，应用程序装配者，EJB 容器，EJB 配置工具，EJB 服务提供者，系统管理员。这里面，EJB 容器是 EJB 之所以能够运行的核心。EJB 容器管理着 EJB 的创建，撤消，激活，去活，与数据库的连接等等重要的核心工作。JSP,Servlet,EJB,JNDI,JDBC,JMS.....

2、EJB 与 JAVA BEAN 的区别？

答:Java Bean 是可复用的组件，对 Java Bean 并没有严格的规范，理论上讲，任何一个 Java 类都可以是一个 Bean。但通常情况下，由于 Java Bean 是被容器所创建（如 Tomcat)的，所以 Java Bean 应具有一个无参的构造器，另外，通常 Java Bean 还要实现 Serializable 接口用于实现 Bean 的持久性。Java Bean 实际上相当于微软 COM 模型中的本地进程内 COM 组件，它是不能被跨进程访问的。Enterprise Java Bean 相当于 DCOM，即分布式组件。它是基于 Java 的远程方法调用（RMI）技术的，所以 EJB 可以被远程访问（跨进程、跨计算机）。但 EJB 必须被布署在诸如 Webspere、WebLogic 这样的容器中，EJB 客户从不直接访问真正的 EJB 组件，而是通过其容器访问。EJB 容器是 EJB 组件的代理，EJB 组件由容器所创建和管理。客户通过容器来访问真正的 EJB 组件。

3、EJB 的基本架构

答:一个 EJB 包括三个部分:

Remote Interface 接口的代码

```
package Beans;
import javax.ejb.EJBObject;
import java.rmi.RemoteException;
public interface Add extends EJBObject
{
//some method declare
}
```

Home Interface 接口的代码

```
package Beans;
import java.rmi.RemoteException;
import jaax.ejb.CreateException;
```

```
import javax.ejb.EJBHome;
public interface AddHome extends EJBHome
{
//some method declare
}
```

EJB 类的代码

```
package Beans;
import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javx.ejb.SessionContext;
public class AddBean Implements SessionBean
{
//some method declare
}
```

J2EE,MVC 方面

1、MVC 的各个部分都有那些技术来实现?如何实现?

答:MVC 是 Model－View－Controller 的简写。"Model" 代表的是应用的业务逻辑（通过 JavaBean，EJB 组件实现）， "View" 是应用的表示面（由 JSP 页面产生），"Controller" 是提供应用的处理过程控制（一般是一个 Servlet），通过这种设计模型把应用逻辑，处理过程和显示逻辑分成不同的组件实现。这些组件可以进行交互和重用。

2、应用服务器与 WEB SERVER 的区别？

希望大家补上，谢谢

3、J2EE 是什么？

答:Je22 是 Sun 公司提出的多层(multi-diered),分布式(distributed),基于组件(component-base)的企业级应用模型(enterpriese application model).在这样的一个应用系统中，可按照功能划分为不同的组件，这些组件又可在不同计算机上，并且处于相应的层次(tier)中。所属层次包括客户层(clietn tier)组件,web 层和组件,Business 层和组件,企业信息系统(EIS)层。

4、WEB SERVICE 名词解释。JSWDL 开发包的介绍。JAXP、JAXM 的解释。SOAP、UDDI,WSDL 解释。

答：Web Service 描述语言 WSDL

SOAP 即简单对象访问协议(Simple Object Access Protocol)，它是用于交换 XML 编码信息的轻量级协议。

UDDI 的目的是为电子商务建立标准；UDDI 是一套基于 Web 的、分布式的、为 Web Service 提供的、信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的 Web Service 注册，以使别的企业能够发现的访问协议的实现标准。

5、BS 与 CS 的联系与区别。

希望大家补上，谢谢

6、STRUTS 的应用(如 STRUTS 架构)

答：Struts 是采用 Java Servlet/JavaServer Pages 技术，开发 Web 应用程序的开放源码的 framework。 采用 Struts 能开发出基于 MVC(Model-View-Controller)设计模式的应用构架。 Struts 有如下的主要功能：

一、包含一个 controller servlet，能将用户的请求发送到相应的 Action 对象。

二、JSP 自由 tag 库，并且在 controller servlet 中提供关联支持，帮助开发员创建交互式表单应用。

三、提供了一系列实用对象：XML 处理、通过 Java reflection APIs 自动处理 JavaBeans 属性、国际化的提示和消息。

设计模式方面

1、开发中都用到了那些设计模式?用在什么场合?

答：每个模式都描述了一个在我们的环境中不断出现的问题，然后描述了该问题的解决方案的核心。通过这种方式，你可以无数次地使用那些已有的解决方案，无需在重复相同的工作。主要用到了 MVC 的设计模式。用来开发 JSP/Servlet 或者 J2EE 的相关应用。简单工厂模式等。

2、UML 方面

答：标准建模语言 UML。用例图,静态图(包括类图、对象图和包图),行为图,交互图(顺序图,合作图),实现图,

JavaScript 方面

1、如何校验数字型?

```
var re=/^d{1,8}$|.d{1,2}$/;
var str=document.form1.all(i).value;
var r=str.match(re);
if (r==null)
{
sign=-4;
break;
}
else{
document.form1.all(i).value=parseFloat(str);
}
```

CORBA 方面

1、CORBA 是什么?用途是什么?

答：CORBA 标准是公共对象请求代理结构(Common Object Request Broker Architecture)，由对象管理组织(Object Management Group，缩写为 OMG)标准化。它的组成是接口定义语言(IDL), 语言绑定(binding:也译为联编)和允许应用程序间互操作的协议。 其目的为：

用不同的程序设计语言书写

在不同的进程中运行

为不同的操作系统开发

LINUX 方面

1、LINUX 下线程，GDI 类的解释。

答：LINUX 实现的就是基于核心轻量级进程的"一对一"线程模型，一个线程实体对应一个核心轻量级进程，而线程之间的管理在核外函数库中实现。

GDI 类为图像设备编程接口类库。

JAVA 华为面试题

JAVA 方面

1、面向对象的特征有哪些方面

2 String 是最基本的数据类型吗?

3 int 和 Integer 有什么区别

4 String 和 StringBuffer 的区别

5 运行时异常与一般异常有何异同?

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。

6 说出一些常用的类，包,接口，请各举 5 个

7 说出 ArrayList,Vector, LinkedList 的存储性能和特性

ArrayList 和 Vector 都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector 由于使用了 synchronized 方法（线程安全），通常性能上较 ArrayList 差，而 LinkedList 使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

8 设计 4 个线程，其中两个线程每次对 j 增加 1，另外两个线程对 j 每次减少 1。写出程序。

以下程序使用内部类实现线程，对 j 增减的时候没有考虑顺序问题。

```
public class ThreadTest1{
private int j;
public static void main(String args[]){
ThreadTest1 tt=new ThreadTest1();
Inc inc=tt.new Inc();
Dec dec=tt.new Dec();
for(int i=0;i<2;i++){
Thread t=new Thread(inc);
t.start();
t=new Thread(dec);
t.start();
}
}
private synchronized void inc(){
j++;
System.out.println(Thread.currentThread().getName()+"-inc:"+j);
```

```
}
private synchronized void dec(){
j--;
System.out.println(Thread.currentThread().getName()+"-dec:"+j);
}

class Inc implements Runnable{
public void run(){
for(int i=0;i<100;i++){
inc();
}
}
}
class Dec implements Runnable{
public void run(){
for(int i=0;i<100;i++){
dec();
}
}
}
}
```

9． JSP 的内置对象及方法。

request request 表示 HttpServletRequest 对象。它包含了有关浏览器请求的信息，并且提供了几个用于获取 cookie, header, 和 session 数据的有用的方法。

response response 表示 HttpServletResponse 对象，并提供了几个用于设置送回 浏览器的响应的方法（如 cookies,头信息等）

out out 对象是 javax.jsp.JspWriter 的一个实例，并提供了几个方法使你能用于向浏览器回送输出结果。
pageContext pageContext 表示一个 javax.servlet.jsp.PageContext 对象。它是用于方便存取各种范围的名字空间、servlet 相关的对象的 API，并且包装了通用的 servlet 相关功能的方法。

session session 表示一个请求的 javax.servlet.http.HttpSession 对象。Session 可以存贮用户的状态信息

application applicaton 表示一个 javax.servle.ServletContext 对象。这有助于查找有关 servlet 引擎和 servlet 环境的信息

config config 表示一个 javax.servlet.ServletConfig 对象。该对象用于存取 servlet 实例的初始化参数。

page page 表示从该页面产生的一个 servlet 实例

10.用 socket 通讯写出客户端和服务器端的通讯，要求客户发送数据后能够回显相同的数据。

参见课程中 socket 通讯例子。

11 说出 Servlet 的生命周期，并说出 Servlet 和 CGI 的区别。
Servlet 被服务器实例化后，容器运行其 init 方法，请求到达时运行其 service 方法，service 方法自动派遣

运行与请求对应的 doXXX 方法（doGet，doPost）等，当服务器决定将实例销毁的时候调用其 destroy 方法。
与 cgi 的区别在于 servlet 处于服务器进程中，它通过多线程方式运行其 service 方法，一个实例可以服务于多个请求，并且其实例一般不会销毁，而 CGI 对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于 servlet。

12.EJB 是基于哪些技术实现的?并说出 SessionBean 和 EntityBean 的区别，StatefulBean 和 StatelessBean 的区别。

13．EJB 包括（SessionBean,EntityBean）说出他们的生命周期，及如何管理事务的？

14．说出数据连接池的工作机制是什么？

15 同步和异步有和异同，在什么情况下分别使用他们？举例说明。

16 应用服务器有那些？

17 你所知道的集合类都有哪些？主要方法？

18 给你一个:驱动程序 A,数据源名称为 B,用户名称为 C,密码为 D,数据库表为 T，请用 JDBC 检索出表 T 的所有数据。

19．说出在 JSP 页面里是怎么分页的?
页面需要保存以下参数：
总行数：根据 sql 语句得到总行数
每页显示行数：设定值
当前页数：请求参数
页面根据当前页数和每页行数计算出当前页第一行行数，定位结果集到此行，对结果集取出每页显示行数的行即可。

数据库方面：

1. 存储过程和函数的区别
存储过程是用户定义的一系列 sql 语句的集合，涉及特定表或其它对象的任务，用户可以调用存储过程，而函数通常是数据库已定义的方法，它接收参数并返回某种类型的值并且不涉及特定用户表。
2. 事务是什么？
事务是作为一个逻辑单元执行的一系列操作，一个逻辑工作单元必须有四个属性，称为 ACID（原子性、一致性、隔离性和持久性）属性，只有这样才能成为一个事务：
原子性
事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。
一致性
事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都必须是正确的。
隔离性
由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状态的数据。

这称为可串行性，因为它能够重新装载起始数据，并且重播一系列事务，以使数据结束时的状态与原始事务执行的状态相同。

持久性

事务完成之后，它对于系统的影响是永久性的。该修改即使出现系统故障也将一直保持。

3. 游标的作用？如何知道游标已经到了最后？

游标用于定位结果集的行，通过判断全局变量@@FETCH_STATUS 可以判断是否到了最后，通常此变量不等于 0 表示出错或到了最后。

4. 触发器分为事前触发和事后触发，这两种触发有和区别。语句级触发和行级触发有何区别。

事前触发器运行于触发事件发生之前，而事后触发器运行于触发事件发生之后。通常事前触发器可以获取事件之前和新的字段值。

语句级触发器可以在语句执行前或后执行，而行级触发在触发器所影响的每一行触发一次。

中 远 面 试 题

一、JAVA

1、面向对象的三个基本特征

2、方法重载和方法重写的概念和区别

3、接口和内部类、抽象类的特性

4、文件读写的基本类

**5、串行化的注意事项以及如何实现串行化

6、线程的基本概念、线程的基本状态以及状态之间的关系

7、线程的同步、如何实现线程的同步

8、几种常用的数据结构及内部实现原理。

9、Socket 通信(TCP、UDP 区别及 Java 实现方式)

**10、Java 的事件委托机制和垃圾回收机制

11、JDBC 调用数据库的基本步骤

**12、解析 XML 文件的几种方式和区别

13、Java 四种基本权限的定义

14、Java 的国际化

二、JSP

1、至少要能说出 7 个隐含对象以及他们的区别
** 2、forward 和 redirect 的区别
3、JSP 的常用指令

三、servlet

1、什么情况下调用 doGet()和 doPost()?

2、servlet 的 init()方法和 service()方法的区别

3、servlet 的生命周期

4、如何现实 servlet 的单线程模式

5、servlet 的配置

6、四种会话跟踪技术

四、EJB

**1、EJB 容器提供的服务
主要提供声明周期管理、代码产生、持续性管理、安全、事务管理、锁和并发行管理等服务。
2、EJB 的角色和三个对象
EJB 角色主要包括 Bean 开发者 应用组装者 部署者 系统管理员 EJB 容器提供者 EJB 服务器提供者
三个对象是 Remote（Local）接口、Home（LocalHome）接口，Bean 类
2、EJB 的几种类型
会话（Session）Bean ，实体（Entity）Bean 消息驱动的（Message Driven）Bean
会话 Bean 又可分为有状态（Stateful）和无状态（Stateless）两种
实体 Bean 可分为 Bean 管理的持续性（BMP）和容器管理的持续性（CMP）两种
3、bean 实例的生命周期
对于 Stateless Session Bean、Entity Bean、Message Driven Bean 一般存在缓冲池管理，而对于 Entity Bean 和 Statefull Session Bean 存在 Cache 管理，通常包含创建实例，设置上下文、创建 EJB Object（create）、业务方法调用、remove 等过程，对于存在缓冲池管理的 Bean，在 create 之后实例并不从内存清除，而是采用缓冲池调度机制不断重用实例，而对于存在 Cache 管理的 Bean 则通过激活和去激活机制保持 Bean 的状态并限制内存中实例数量。
4、激活机制
以 Statefull Session Bean 为例：其 Cache 大小决定了内存中可以同时存在的 Bean 实例的数量，根据 MRU 或 NRU 算法，实例在激活和去激活状态之间迁移，激活机制是当客户端调用某个 EJB 实例业务方法时，如果对应 EJB Object 发现自己没有绑定对应的 Bean 实例则从其去激活 Bean 存储中（通过序列化机制存储实例）回复（激活）此实例。状态变迁前会调用对应的 ejbActive 和 ejbPassivate 方法。
5、remote 接口和 home 接口主要作用
remote 接口定义了业务方法，用于 EJB 客户端调用业务方法
home 接口是 EJB 工厂用于创建和移除查找 EJB 实例
6、客服端调用 EJB 对象的几个基本步骤
(1) 设置 JNDI 服务工厂以及 JNDI 服务地址系统属性

(2) 查找 Home 接口
(3) 从 Home 接口调用 Create 方法创建 Remote 接口
(4) 通过 Remote 接口调用其业务方法

五、数据库

1、存储过程的编写

2、基本的 SQL 语句

六、weblogic

1、 如何给 weblogic 指定大小的内存?
在启动 Weblogic 的脚本中（位于所在 Domian 对应服务器目录下的 startServerName），增加 set MEM_ARGS=-Xms32m -Xmx200m，可以调整最小内存为 32M，最大 200M
2、 如何设定的 weblogic 的热启动模式(开发模式)与产品发布模式?
可以在管理控制台中修改对应服务器的启动模式为开发或产品模式之一。或者修改服务的启动文件或者 commenv 文件，增加 set PRODUCTION_MODE=true。
3、 如何启动时不需输入用户名与密码?
修改服务启动文件，增加 WLS_USER 和 WLS_PW 项。也可以在 boot.properties 文件中增加加密过的用户名和密码.
4、在 weblogic 管理制台中对一个应用域(或者说是一个网站,Domain)进行 jms 及 ejb 或连接池等相关信息进行配置后,实际保存在什么文件中?
保存在此 Domain 的 config.xml 文件中，它是服务器的核心配置文件。
5、说说 weblogic 中一个 Domain 的缺省目录结构?比如要将一个简单的 helloWorld.jsp 放入何目录下,然的在浏览器上就可打入 http://主机:端口号//helloword.jsp 就可以看到运行结果了？又比如这其中用到了一个自己写的 javaBean 该如何办?
Domain 目录\服务器目录\applications，将应用目录放在此目录下将可以作为应用访问，如果是 Web 应用，应用目录需要满足 Web 应用目录要求，jsp 文件可以直接放在应用目录中，Javabean 需要放在应用目录的 WEB-INF 目录的 classes 目录中，设置服务器的缺省应用将可以实现在浏览器上无需输入应用名。
6、 如何查看在 weblogic 中已经发布的 EJB?
可以使用管理控制台，在它的 Deployment 中可以查看所有已发布的 EJB
7、 如何在 weblogic 中进行 ssl 配置与客户端的认证配置或说说 j2ee(标准)进行 ssl 的配置
缺省安装中使用 DemoIdentity.jks 和 DemoTrust.jks KeyStore 实现 SSL，需要配置服务器使用 Enable SSL，配置其端口，在产品模式下需要从 CA 获取私有密钥和数字证书，创建 identity 和 trust keystore，装载获得的密钥和数字证书。可以配置此 SSL 连接是单向还是双向的。
8、在 weblogic 中发布 ejb 需涉及到哪些配置文件
不同类型的 EJB 涉及的配置文件不同，都涉及到的配置文件包括 ejb-jar.xml,weblogic-ejb-jar.xmlCMP 实体 Bean 一般还需要 weblogic-cmp-rdbms-jar.xml
9、EJB 需直接实现它的业务接口或 Home 接口吗,请简述理由.
远程接口和 Home 接口不需要直接实现，他们的实现代码是由服务器产生的，程序运行中对应实现类会作为对应接口类型的实例被使用。
10、说说在 weblogic 中开发消息 Bean 时的 persistent 与 non-persisten 的差别
persistent 方式的 MDB 可以保证消息传递的可靠性,也就是如果 EJB 容器出现问题而 JMS 服务器依然会将消息在此 MDB 可用的时候发送过来，而 non－persistent 方式的消息将被丢弃。
11、说说你所熟悉或听说过的 j2ee 中的几种常用模式?及对设计模式的一些看法

Session Facade Pattern：使用 SessionBean 访问 EntityBean
Message Facade Pattern：实现异步调用
EJB Command Pattern：使用 Command JavaBeans 取代 SessionBean，实现轻量级访问
Data Transfer Object Factory：通过 DTO Factory 简化 EntityBean 数据提供特性
Generic Attribute Access：通过 AttibuteAccess 接口简化 EntityBean 数据提供特性
Business Interface：通过远程（本地）接口和 Bean 类实现相同接口规范业务逻辑一致性
ＥＪＢ架构的设计好坏将直接影响系统的性能、可扩展性、可维护性、组件可重用性及开发效率。项目越复杂，项目队伍越庞大则越能体现良好设计的重要性

复习：

1、原子类型有哪些？

数组不属于原子类型；

原子类型属于内建对像；

2、为什么要为原子类型对应一个包装类？

原子类都有包装类；

3、java 的包在哪里？

一个目录体系就是包；

Jar 文件就是一个目录体系。

4、怎样生成一个 java 包？

5、包的根目录在哪里？

echo %classpath%

6、怎样避免类名重复的问题？

把文件放在一个包下，采用域名倒置法

7、类与对象是什么关系？

类是等待着去实例化的一个模板；

类—对象

1 ：N

static 永久存在，永远只有一个；

8、为什么说对象的引用不是对象本身？

在程序中永远不可能拿到对像，拿到的只是对象的一个引用；

一个引用“=”另一个引用，就是用另一个引用的值来覆盖它；

9、如何复制一个对象？

机械复制；克隆；

10、面向对象的 3 大特征是什么？

1、 封装

2、 继承

3、多态

11、为什么要封装？

封装的目的：（1）怕破坏 （2）改结构

12、如何理解 this 引用？

this 就是一个对象的引用;

this 就是目前正在执行这项任务(这个方法)的那个对象;

13、this 引用有何作用？

this 在重名的时候，无法区分的时候;

测试传进来的参数是不是自己；

14、静态方法有什么特征？

1、如果一个函数前面用了 static，不用对象就可以调用；static 没有 this 指针,不能调用非 static 的方法；

与对象无关的，可以单独执行；

static 只有一份，是公共的。

15、可以从多个父类继承吗？

不可以。Java 里是单根继承；

16、子类可以访问父类的哪些东西？

17、哪种现象称为多态？

站在抽象的高度来设计程序；

18、多态的目的是什么？

能站在抽象的高度来考虑问题;

19、抽象方法有什么用？

定义抽象方法就是为了继承；

20、什么是抽象类？

21、什么是接口？

22、接口有什么用处？

约定了函数名；实现的接口越多，适应能力就越强；

23、什么是内部类？

24、使用内部类有什么好处？

25、什么是匿名类？

是普通类的简单写法。

26、Java 为了支持数据结构，提供了哪些接口？

27、Set 接口与 List 接口有哪些共同的特征？

28、Map 接口怎样描述一对多的关系？

为了确定两个物体间的关系；把 Set 和 Map 联合起来

29、为什么 java 定义那么多关于流的类？

观点不同；

基本元素的类型不同；

字节，char;int;double 串

30、如何流化（序列化）一个对象

把一个对象折成了一个个的字节，再把字节合成一个对像；

31、线程与进程有什么区别？

最主要的是：是否共享内存；

线程是共享内存的；

32、为什么要使用线程？

为了均匀响应；

33、使用线程能提高效率吗？

不会；线程不可能提高效率；

34、如何创建线程？

1)、extends Thread

2）、implements Runnable

35、如何在线程中实现排斥?

synchronized

36、何为线程安全对象？

37、什么叫线程协作？如何实现？

38、hashCode?

散列表是最快的查找方式；

散列存是为了最快的查找；

总结:

■final ◆ 修饰的变量

类中声明为final的变量，有两种初始化方式：

1、声明时就初始化。

2、2。在类的构造方法中初始化

初始化以后就不能在修改了

◆ 修饰的方法

不能被子类重写，但可以被本类方法重载

◆ 修饰类

我不会继承这个类，

■static ◆ 修饰的变量

类中声明为static的变量，在类的第一次实例化就存在于内存，以后的实例化就不会重复产生内存空间了

◆ 修饰的方法

只能操作本类的静态成员属性

■static final◆ 共同声明的成员属性，在申明时就要初始化。

`

■public ◆ 类中的成员属性

可以被包内包外的的类的实例化对象方法使用(类A中有public方法，类B和A在同一包中，类B通过在自己的实例化对象的方法中在申明类A的对象对类A的方法进行调用)

■protected ◆类中的成员属性

可以被包外继承本类的类的实例化对象使用和包内的类的实例化对象方法使用

■private ◆类中的成员属性

只能被本类的成员方法使用

■interface ◆功能的集合，但方法都没有实体，默认都是public 方法和默认的public final 属性。

■abstract ◆本身不能被实例化，但继承它的子类在实例化自己的时候会初始化一些抽象类的属性。

■多态 ◆继承了同一接口的不同类。在声明时声明成接口类型，在实例化时可以实例化成这几个不同类(但实例化后的对象只能使用接口本身的方法)

■继承 ◆子类可以继承父类的全部protected和public的方法和属性

例子：

com.org.bluedot包中一个类

```
public class A
      {
        protected int age;
      }
```

com.org.bluedot.test包中一个类

```
import com.org.buledot.*;
public class B extends A
{
```

```
    public static void main(String [] args)
    {
        B b = new B();
        b.age;       //可以这样引用
        A a = new A();
        a.age;       //不可以这样引用
        }
}
```

protected类型的成员属性和方法只能有本包内的方法操作和外包的子类对象操作（外包子类中new的父类对象不能操作父类的protected）

■内部类 ◆内部类的实例化发生于外部类已被实例化以后可有两种实例方法

1、在外部类实例化以后，用实例化的对象（My1.MyIn2 a = a0.new MyIn2();）在实例化内部对象。

2、在外部类的方法中加入实例化内部类。

内部类和外部类可以互相操作对方的private的成员属性和成员方法,不用this可直接操作，但在内部类前加上static后，内部类就不能访问外部类的属性了

内部类之间可以继承

内部类的public,private,protected修饰符同外部类的属性定义权限一样！

public的内部类可以被实例化了外部类的所有类实例化

private的内部类只能被本类的对象实例化

protected的内部类能被本包中实例化了外部类的所有类实例化和外包中继承了该外部类的实例化对象实例化

内部类隐藏的是它的类型！

◆例子

```
class My1
{
  // 帮助类，工具类，辅助类
  private class MyIn
  {
      private int m;
      public void f()
      {
          a++;  //可访问外部
          System.out.println("MyIn...f");
      }
  }

  private class MyIn3 extends MyIn
  {
      public void f()
      {
          super.f();
          System.out.println("MyIn3 ... ...");
      }
  }
```

```
//复杂内部类
public class MyIn2
{
    public void f()
    {
        a++;  //可访问外部
        System.out.println("MyIn...f");
    }
}

public static class MyIn4
{
    //a++;  // 没有外部对象的指针
    public void f()
    {
        System.out.println("MyIn4...f");
    }
}

private int a;

public void add()
{
    a++;
}

public void show()
{
    System.out.println(a);
    MyIn x = new MyIn();
    x.f();
    x.m++;  //外部可访问内部类
}

public void g()
{
    //MyIn a = new MyIn3();
    //a.f();

    // 只用一次的类不需要定义
    MyIn x = new MyIn() {
        public void f(){
            super.f();
            System.out.println("Ni Ming");
        }
    };
    x.f();
```

```
    }
}

public class InnerC
{
    public static void main(String av[])
    {
        /*
        My1 a = new My1();
        a.add();
        a.show();
        a.show();
        */

        /*
        My1 a0 = new My1();
        //My1.MyIn2 a = new My1.MyIn2(); 经典错误
        My1.MyIn2 a = a0.new MyIn2();
        a.f();
        */

        My1 a = new My1();
        a.g();
    }
}
```

■存储　◆类在实例化后，成员属性放在堆中，方法中的局部变量放在栈中

■合成　◆在新的类里面直接创建旧的类的对象，新的类是由旧的类合成而来，所复用的是代码的功能，不是代码的形式

■toString()　◆由一个对象调用，返回一个字符串！如果默认则使用由Object继承下来的toString(),返回这个对象的内存地址Object.toString()方法内容:

```
return getClass().getName() + "@" + Integer.toHexString(hashCode());
```

一个典型的重写toString()的例子：

```
public static String toString(int [] a)
{
    StringBuffer result = new StringBuffer("[");
    for(int i=0;i<a.length;i++)
    {
        result.append(a[i]);
        if(i<a.length-1)
        {
            result.append(",");
        }
    }
    result.append("]");
    return result.toString();
}
```

■上传　　◆在某个类的应用中，把一个派生类的对象当做基类的对象来使用

■HasMap　　◆ Key　value

■Map　　◆ Key 的set和value的collection

■List　　◆Value　有序的集合,继承于collection(非排序)

■collection　◆总的集合(无规则集合)(Map不是从这里继承的)

■Set　　Key　value 没有重复的集合,继承于collection

(如果要做很多随机访问，请用ArrayList，但是如果要在List的中间做很多插入和删除的话，就应该用LinkedList,LinkedList能提供队列，双向队列和栈的功能)

(Map提供的不是对象与数组的关联，而是对象和对象的关联。HashMap看重的是访问速度，而TreeMap看重键的顺序，因而它不如HashMap那么快，而LinkedHashMap则保持对象插入的顺序，但是也可以用LRU算法为它重新排序)

(Set只接受不重复的对象，HashSet提供了最快的查询速度，而TreeSet则保持元素有序，LinkedHashSet保持元素的插入顺序)

■判断两个对象是否相同　　◆equals,hashCode返回一个int

■iterator　◆集合的通用接口,有三个方法

◆例子：

```
import java.util.*;
import java.util.List;
public class TestMap {
  public TestMap()
  {
      List ls= new ArrayList();
      List ls1= new ArrayList();

      Map map = new HashMap();

      ls.add("李科长");//里面都是对象
      ls.add("刘科长");
      ls.add("赵科长");

      ls1.add("处长");
      ls1.add("科长");
      ls1.add("赵科长");

      map.put("李局长",ls);
      map.put("王局长",ls1);

      List show=(List)map.get("李局长");//返回是Object，所以要转换成List
      List show1=(List)map.get("王局长");

          Iterator it=show.iterator();//迭代接口，里面有三个方法可调用，是相对于集合的
      Iterator it1=show1.iterator();

      while(it.hasNext())
```

```
        {
            System.out.println((String)it.next());
        }
        while(it1.hasNext())
        {
            System.out.println((String)it1.next());
        }

    }

    public static void main(String[] args)
    {
        new TestMap();
    }
}
```

■有效和正确定义hashCode()和equals()

有效和正确定义hashCode()和equals()
作者：unknown 更新时间： 2005-03-22

有效和正确定义hashCode()和equals()

级别：入门级
Brian Goetz (brian@quiotix.com)
Quiotix Corp首席顾问
2003 年 8 月

每个Java对象都有hashCode()和 equals()方法。许多类忽略（Override)这些方法的缺省实施,以在对象实例之间提供更深层次的语义可比性。在Java理念和实践这一部分，Java开发人员Brian Goetz向您介绍在创建Java类以有效和准确定义hashCode()和equals()时应遵循的规则和指南。您可以在讨论论坛与作者和其它读者一同探讨您对本文的看法。(您还可以点击本文顶部或底部的讨论进入论坛。)

虽然Java语言不直接支持关联数组 -- 可以使用任何对象作为一个索引的数组 -- 但在根Object类中使用hashCode()方法明确表示期望广泛使用HashMap(及其前辈Hashtable)。理想情况下基于散列的容器提供有效插入和有效检索；直接在对象模式中支持散列可以促进基于散列的容器的开发和使用。

定义对象的相等性

Object类有两种方法来推断对象的标识：equals()和hashCode()。一般来说，如果您忽略了其中一种，您必须同时忽略这两种，因为两者之间有必须维持的至关重要的关系。特殊情况是根据equals() 方法，如果两个对象是相等的，它们必须有相同的hashCode()值(尽管这通常不是真的)。

特定类的equals()的语义在Implementer的左侧定义；定义对特定类来说equals()意味着什么是其设计工作的一部分。Object提供的缺省实施简单引用下面等式：

```
public boolean equals(Object obj) { return (this == obj); }
```

在这种缺省实施情况下，只有它们引用真正同一个对象时这两个引用才是相等的。同样，Object提供的hashCode()的缺省实施通过将对象的内存地址对映于一个整数值来生成。由于在某些架构上，地址空间大于int值的范围，两个不同的对象有相同的hashCode()是可能的。如果您忽略了hashCode()，您仍旧可以使用System.identityHashCode()方法来接入这类缺省值。

忽略 equals() -- 简单实例

缺省情况下，equals()和hashCode()基于标识的实施是合理的，但对于某些类来说，它们希望放宽等式的定义。例如，Integer类定义equals() 与下面类似：

```
public boolean equals(Object obj) {
return (obj instanceof Integer
&& intValue() == ((Integer) obj).intValue());
}
```

在这个定义中，只有在包含相同的整数值的情况下这两个Integer对象是相等的。结合将不可修改的Integer，这使得使用Integer作为HashMap中的关键字是切实可行的。这种基于值的Equal方法可以由Java类库中的所有原始封装类使用，如Integer、Float、Character和Boolean以及String(如果两个String对象包含相同顺序的字符，那它们是相等的)。由于这些类都是不可修改的并且可以实施hashCode()和equals()，它们都可以做为很好的散列关键字。

为什么忽略 equals()和hashCode()?

如果Integer不忽略equals() 和 hashCode()情况又将如何?如果我们从未在HashMap或其它基于散列的集合中使用Integer作为关键字的话，什么也不会发生。但是，如果我们在HashMap中使用这类Integer对象作为关键字，我们将不能够可靠地检索相关的值，除非我们在get()调用中使用与put()调用中极其类似的Integer实例。这要求确保在我们的整个程序中，只能使用对应于特定整数值的Integer对象的一个实例。不用说，这种方法极不方便而且错误频频。

Object的interface contract要求如果根据 equals()两个对象是相等的，那么它们必须有相同的hashCode()值。当其识别能力整个包含在equals()中时，为什么我们的根对象类需要hashCode()？ hashCode()方法纯粹用于提高效率。Java平台设计人员预计到了典型Java应用程序中基于散列的集合类（Collection Class)的重要性--如Hashtable、HashMap和HashSet，并且使用equals()与许多对象进行比较在计算方面非常昂贵。使所有Java对象都能够支持hashCode()并结合使用基于散列的集合，可以实现有效的存储和检索。

实施equals()和hashCode()的需求

实施equals()和 hashCode()有一些限制，Object文件中列举出了这些限制。特别是equals()方法必须显示以下属性：

Symmetry：两个引用，a和 b,a.equals(b) if and only if b.equals(a)
Reflexivity：所有非空引用， a.equals(a)
Transitivity：If a.equals(b) and b.equals(c), then a.equals(c)
Consistency with hashCode()：两个相等的对象必须有相同的hashCode()值

Object的规范中并没有明确要求equals()和 hashCode() 必须一致 -- 它们的结果在随后的调用中将是相同的，假设“不改变对象相等性比较中使用的任何信息。”这听起来象“计算的结果将不改变，除非实际情况如此。”这一模糊声明通常解释为相等性和散列值计算应是对象的可确定性功能，而不是其它。

对象相等性意味着什么？

人们很容易满足Object类规范对equals() 和 hashCode() 的要求。决定是否和如何忽略equals()除了判断以外，还要求其它。在简单的不可修值类中，如Integer(事实上是几乎所有不可修改的类)，选择相当明显 -- 相等性应基于基本对象状态的相等性。在Integer情况下，对象的唯一状态是基本的整数值。

对于可修改对象来说，答案并不总是如此清楚。equals() 和hashCode() 是否应基于对象的标识(象缺省实施)或对象的状态(象Integer和String)？没有简单的答案 -- 它取决于类的计划使用。对于象List和Map这样的容器来说，人们对此争论不已。Java类库中的大多数类，包括容器类，错误出现在根据对象状态来提供equals()和hashCode()实施。

如果对象的hashCode()值可以基于其状态进行更改，那么当使用这类对象作为基于散列的集合中的关键字时我们必须注意，确保当它们用于作为散列关键字时，我们并不允许更改它们的状态。所有基于散列的集合假设，当对象的散列值用于作为集合中的关键字时它不会改变。如果当关键字在集合中时它的散列代码被更改，那么将产生一些不可预测和容易混淆的结果。实践过程中这通常不是问题 -- 我们并不经常使用象List这样的可修改对象做为HashMap中的关键字。

一个简单的可修改类的例子是Point，它根据状态来定义equals()和hashCode()。如果两个Point 对象引用相同的(x, y)座标，Point的散列值来源于x和y座标值的IEEE 754-bit表示，那么它们是相等的。

对于比较复杂的类来说，equals()和hashCode()的行为可能甚至受到superclass或interface的影响。例如，List接口要求如果并且只有另一个对象是List，而且它们有相同顺序的相同的Elements(由Element上的Object.equals() 定义)，List对象等于另一个对象。hashCode()的需求更特殊--list的hashCode()值必须符合以下计算：

```
hashCode = 1;
Iterator i = list.iterator();
while (i.hasNext()) {
Object obj = i.next();
hashCode = 31*hashCode + (obj==null ? 0 : obj.hashCode());
}
```

不仅仅散列值取决于list的内容，而且还规定了结合各个Element的散列值的特殊算法。(String类规定类似的算法用于计算String的散列值。)

编写自己的equals()和hashCode()方法

忽略缺省的equals()方法比较简单，但如果不违反对称（Symmetry）或传递性（Transitivity）需求，忽略已经忽略的equals() 方法极其棘手。当忽略equals()时，您应

该总是在equals()中包括一些Javadoc注释，以帮助那些希望能够正确扩展您的类的用户。

作为一个简单的例子，考虑以下类：

```
class A {
final B someNonNullField;
C someOtherField;
int someNonStateField;
}
```

我们应如何编写该类的equals()的方法？这种方法适用于许多情况：

```
public boolean equals(Object other) {
// Not strictly necessary, but often a good optimization
if (this == other)
return true;
if (!(other instanceof A))
return false;
A otherA = (A) other;
return
(someNonNullField.equals(otherA.someNonNullField))
&& ((someOtherField == null)
? otherA.someOtherField == null
: someOtherField.equals(otherA.someOtherField)));
}
```

现在我们定义了equals()，我们必须以统一的方法来定义hashCode()。一种统一但并不总是有效的定义hashCode()的方法如下：

```
public int hashCode() { return 0; }
```

这种方法将生成大量的条目并显著降低HashMaps的性能，但它符合规范。一个更合理的hashCode()实施应该是这样：

```
public int hashCode() {
int hash = 1;
hash = hash * 31 + someNonNullField.hashCode();
hash = hash * 31
+ (someOtherField == null ? 0 : someOtherField.hashCode());
return hash;
}
```

注意：这两种实施都降低了类状态字段的equals()或hashCode()方法一定比例的计算能力。根据您使用的类，您可能希望降低superclass的equals()或hashCode()功能一部分计算能力。对于原始字段来说，在相关的封装类中有helper功能，可以帮助创建散列值，如Float.floatToIntBits。

编写一个完美的equals()方法是不现实的。通常，当扩展一个自身忽略了equals()的instantiable类时，忽略equals()是不切实际的，而且编写将被忽略的equals()方法(如在抽象类中)不同于为具体类编写equals()方法。关于实例以及说明的更详细信息请参阅Effective Java Programming Language Guide, Item 7 (参考资料) 。

有待改进?

将散列法构建到Java类库的根对象类中是一种非常明智的设计折衷方法 -- 它使使用基于散列的容器变得如此简单和高效。但是，人们对Java类库中的散列算法和对象相等性的方法和实施提出了许多批评。java.util中基于散列的容器非常方便和简便易用，但可能不适用于需要非常高性能的应用程序。虽然其中大部分将不会改变，但当您设计严重依赖于基于散列的容器效率的应用程序时必须考虑这些因素，它们包括：

太小的散列范围。使用int而不是long作为hashCode()的返回类型增加了散列冲突的几率。

糟糕的散列值分配。短strings和小型integers的散列值是它们自己的小整数，接近于其它“邻近”对象的散列值。一个循规导矩（Well-behaved）的散列函数将在该散列范围内更均匀地分配散列值。

无定义的散列操作。虽然某些类，如String和List，定义了将其Element的散列值结合到一个散列值中使用的散列算法，但语言规范不定义将多个对象的散列值结合到新散列值中的任何批准的方法。我们在前面编写自己的equals()和hashCode()方法中讨论的List、String或实例类A使用的诀窍都很简单，但算术上还远远不够完美。类库不提供任何散列算法的方便实施，它可以简化更先进的hashCode()实施的创建。

当扩展已经忽略了equals()的 instantiable类时很难编写equals()。当扩展已经忽略了equals()的 instantiable类时，定义equals()的“显而易见的”方式都不能满足equals()方法的对称或传递性需求。这意味着当忽略equals()时，您必须了解您正在扩展的类的结构和实施详细信息，甚至需要暴露基本类中的机密字段，它违反了面向对象的设计的原则。

结束语

通过统一定义equals()和hashCode()，您可以提升类作为基于散列的集合中的关键字的使用性。有两种方法来定义对象的相等性和散列值：基于标识，它是Object提供的缺省方法；基于状态，它要求忽略equals()和hashCode()。当对象的状态更改时如果对象的散列值发生变化，确信当状态作为散列关键字使用时您不允许更更改其状态。

■Throwable ◆分为Exception , Error(内存错误，栈益处)

■Exception ◆分为RuntimeException（java编译器不强制其和它的派生类抛出异常）（编译器认为不会犯这种错误）,和checkedException（java编译器强制其抛出异常）

■java.util.Arrays ◆包括一组可用于数组的static方法，其中基本方法是：1。用来比较两个数组是否相等的equals()2。用来填充数组的fill()3。用来对数组进行排序的sort()4。用于在一个已排序的数组中查找元素的binarySearch()5。用于接受一个数组，然后把它转化成一个List容器

■Thread.join() ◆调用此函数的线程是相对于主线程的.主线程只有在调用这个函数的线程结束后，才能执行

■int.class == Integer.TYPE ◆true

总结

■各种类型直接转换

String.split("@",0) @位置不能是"+"得改写成"[+]", 正则表达式!!!!!!!!!!!!!!

==和equals

集合

异常

流

线程

socket

■java中的参数传递全部都是COPY,都是传值的,没有传地址的

■通过内部类可是实现多重继承