1. Java基础部分

基础部分的顺序：基本语法，类相关的语法，内部类的语法，继承相关的语法，异常的语法，线程的语法，集合的语法，io 的语法，虚拟机方面的语法，其他

97、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？

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。

10、&和&&的区别。

&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。

&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 。

&是位运算符。&&是布尔逻辑运算符。

备注：这道题先说两者的共同点，再说出&&和&的特殊之处，并列举一些经典的例子来表明自己理解透彻深入、实际经验丰富。

29、Java有没有goto?

java中的保留字，现在没有在java中使用。

108、在JAVA中，如何跳出当前的多重嵌套循环？

在Java中，要想跳出多重循环，可以在外面的循环语句前定义一个标号，然后在里层循环体的代码中使用带有标号的break 语句，即可跳出外层循环。例如，

```
ok:
 for(int i=0;i<10;i++)
 {
        for(int j=0;j<10;j++)
        {
                System.out.println(“i=” + i + “,j=” + j);
                if(j == 5) break ok;
        }
 }
```

另外，我个人通常并不使用标号这种方式，而是让外层的循环条件表达式的结果可以受到里层循环体代码的控制，例如，要在二维数组中查找到某个数字。

```
boolean found = false;
for(int i=0;i<10 && !found;i++)
 {
        for(int j=0;j<10;j++)
        {
                System.out.println(“i=” + i + “,j=” + j);
```

```
                if(j == 5)
                {
                        found = true;
                        break;
                }
        }
}
```

42、switch是否能作用在byte上，是否能作用在long上，是否能作用在String上?

在switch（expr1）中，expr1只能是一个整数表达式或者枚举常量（更大字体），整数表达式可以是int基本类型或Integer包装类型，由于，byte,short,char都可以隐含转换为int，所以，这些类型以及这些类型的包装类型也是可以的。显然，long和String类型都不符合sitch的语法规定，并且不能被隐式转换成int类型，所以，它们不能作用于swtich语句中。

25、short s1 = 1; s1 = s1 + 1;有什么错? short s1 = 1; s1 += 1;有什么错?

对于short s1 = 1; s1 = s1 + 1;　　由于s1+1运算时会自动提升表达式的类型，所以结果是int型，再赋值给short类型s1时，编译器将报告需要强制转换类型的错误。

对于short s1 = 1; s1 += 1;　由于 += 是java语言规定的运算符，java编译器会对它进行特殊处理，因此可以正确编译。

55、char型变量中能不能存贮一个中文汉字?为什么?

char型变量是用来存储Unicode编码的字符的，unicode编码字符集中包含了汉字，所以，char型变量中当然可以存储汉字啦。不过，如果某个特殊的汉字没有被包含在unicode编码字符集中，那么，这个char型变量中就不能存储这个特殊汉字。补充说明：unicode编码占用两个字节，所以，char类型的变量也是占用两个字节。

备注：后面一部分回答虽然不是在正面回答题目，但是，为了展现自己的学识和表现自己对问题理解的透彻深入，可以回答一些相关的知识，做到知无不言，言无不尽。

44、编程题: 用最有效率的方法算出2乘以8等於几?

2 << 3，

因为将一个数左移n位，就相当于乘以了2的n次方，那么，一个数乘以8只要将其左移3位即可，而位运算cpu直接支持的，效率最高，所以，2乘以8等於几的最效率的方法是2 << 3。

2、请设计一个一百亿的计算器

首先要明白这道题目的考查点是什么，一是大家首先要对计算机原理的底层细节要清楚、要知道加减法的位运算原理和知道计算机中的算术运算会发生越界的情况，二是要具备一定的面向对象的设计思想。

首先，计算机中用固定数量的几个字节来存储的数值，所以计算机中能够表示的数值是有一定的范围的，为了便于讲解和理解，我们先以byte 类型的整数为例，它用1个字节进行存储，表示的最大数值范围

为-128到+127。-1在内存中对应的二进制数据为11111111，如果两个-1相加，不考虑Java运算时的类型提升，运算后会产生进位，二进制结果为1,11111110，由于进位后超过了byte类型的存储空间，所以进位部分被舍弃，即最终的结果为11111110，也就是-2，这正好利用溢位的方式实现了负数的运算。-128在内存中对应的二进制数据为10000000，如果两个-128相加，不考虑Java运算时的类型提升，运算后会产生进位，二进制结果为1,00000000，由于进位后超过了byte类型的存储空间，所以进位部分被舍弃，即最终的结果为00000000，也就是0，这样的结果显然不是我们期望的，这说明**计算机中的算术运算是会发生越界情况的，两个数值的运算结果不能超过计算机中的该类型的数值范围。**由于Java中涉及表达式运算时的类型自动提升，我们无法用byte类型来做演示这种问题和现象的实验，大家可以用下面一个使用整数做实验的例子程序体验一下：

```
int a = Integer.MAX_VALUE;
int b = Integer.MAX_VALUE;
int sum = a + b;
System.out.println("a="+a+",b="+b+",sum="+sum);
```

先不考虑long类型，由于int的正数范围为2的31次方，表示的最大数值约等于2*1000*1000*1000，也就是20亿的大小，所以，要实现一个一百亿的计算器，我们得自己设计一个类可以用于表示很大的整数，并且提供了与另外一个整数进行加减乘除的功能，大概功能如下：

（）这个类内部有两个成员变量，一个表示符号，另一个用字节数组表示数值的二进制数
（）有一个构造方法，把一个包含有多位数值的字符串转换到内部的符号和字节数组中
（）提供加减乘除的功能

```
public class BigInteger
{
    int sign;
    byte[] val;
    public Biginteger(String val)
    {
        sign = ;
        val = ;
    }
    public BigInteger add(BigInteger other)
    {

    }
    public BigInteger subtract(BigInteger other)
    {

    }
    public BigInteger multiply(BigInteger other)
    {

    }
    public BigInteger divide(BigInteger other)
    {

    }

}
```

备注：要想写出这个类的完整代码，是非常复杂的，如果有兴趣的话，可以参看jdk中自带的java.math.BigInteger类的源码。面试的人也知道谁都不可能在短时间内写出这个类的完整代码的，他要的是你是否有这方面的概念和意识，他最重要的还是考查你的能力，所以，你不要因为自己无法写出完整的最终结果就放弃答这道题，你要做的就是你比别人写得多，证明你比别人强，你有这方面的思想意识就可以了，毕竟别人可能连题目的意思都看不懂，什么都没写，你要敢于答这道题，即使只答了一部分，那也与那些什么都不懂的人区别出来，拉开了距离，算是矮子中的高个，机会当然就属于你了。另外，答案中的框架代码也很重要，体现了一些面向对象设计的功底，特别是其中的方法命名很专业，用的英文单词很精准，这也是能力、经验、专业性、英语水平等多个方面的体现，会给人留下很好的印象，在编程能力和其他方面条件差不多的情况下，英语好除了可以使你获得更多机会外，薪水可以高出一千元。

112、使用final关键字修饰一个变量时，是引用不能变，还是引用的对象不能变？

使用final关键字修饰一个变量时，是指引用变量不能变，引用变量所指向的对象中的内容还是可以改变的。例如，对于如下语句：

```
final StringBuffer a=new StringBuffer("immutable");
```

执行如下语句将报告编译期错误：

```
a=new StringBuffer("");
```

但是，执行如下语句则可以通过编译：

```
a.append(" broken!");
```

有人在定义方法的参数时，可能想采用如下形式来阻止方法内部修改传进来的参数对象：

```
public void method(final  StringBuffer  param)
{
}
```

实际上，这是办不到的，在该方法内部仍然可以增加如下代码来修改参数对象：

```
param.append("a");
```

101、"=="和equals方法究竟有什么区别？

==操作符专门用来比较两个变量的值是否相等，也就是用于比较变量所对应的内存中所存储的数值是否相同，要比较两个基本类型的数据或两个引用变量是否相等，只能用==操作符。

如果一个变量指向的数据是对象类型的，那么，这时候涉及了两块内存，对象本身占用一块内存，变量也占用一块内存，此时，变量所对应的内存中存储的数值就是对象占用的那块内存的首地址。对于指向对象类型的变量，如果要比较两个变量是否指向同一个对象，即要看这两个变量所对应的内存中的数值是否相等，这时候就需要用==操作符进行比较。

equals方法是用于比较两个独立对象的内容是否相同，就好比去比较两个人的长相是否相同，它比较的两个对象是独立的。例如，对于下面的代码：

```
String a=new String("foo");
String b=new String("foo");
```

两条new语句创建了两个对象，然后用a,b这两个变量分别指向了其中一个对象，这是两个不同的对象，它们的首地址是不同的，即a和b中存储的数值是不相同的，所以，表达式a==b将返回false，而这两个对象中的内容是相同的，所以，表达式a.equals(b)将返回true。

在实际开发中，我们经常要比较传递进行来的字符串内容是否等，例如，String input = …;input.equals(“quit”)，许多人稍不注意就使用==进行比较了，这是错误的，随便从网上找几个项目实战的教学视频看看，里面就有大量这样的错误。记住，字符串的比较基本上都是使用equals方法。

如果一个类没有自己定义equals方法，那么它将继承Object类的equals方法，Object类的equals方法的实现代码如下：

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

这说明，如果一个类没有自己定义equals方法，它默认的equals方法（从Object 类继承的）就是使用==操作符，也是在比较两个变量指向的对象是否是同一对象，这时候使用equals和使用==会得到同样的结果，如果比较的是两个独立的对象则总返回false。如果你编写的类希望能够比较该类创建的两个实例对象的内容是否相同，那么你必须覆盖equals方法，由你自己写代码来决定在什么情况即可认为两个对象的内容是相同的。

104、静态变量和实例变量的区别？

在语法定义上的区别：静态变量前要加static关键字，而实例变量前则不加。

在程序运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，才能使用这个实例变量。静态变量不属于某个实例对象，而是属于类，所以也称为类变量，只要程序加载了类的字节码，不用创建任何实例对象，静态变量就会被分配空间，静态变量就可以被使用了。总之，实例变量必须创建对象后才可以通过这个对象来使用，静态变量则可以直接使用类名来引用。

例如，对于下面的程序，无论创建多少个实例对象，永远都只分配了一个staticVar变量，并且每创建一个实例对象，这个staticVar就会加1；但是，每创建一个实例对象，就会分配一个instanceVar，即可能分配多个instanceVar，并且每个instanceVar的值都只自加了1次。

```
public class VariantTest
{
	public static int staticVar = 0;
	public int instanceVar = 0;
	public VariantTest()
	{
		staticVar++;
		instanceVar++;
		System.out.println("staticVar=" + staticVar + ",instanceVar=" + instanceVar);
	}
}
```

备注：这个解答除了说清楚两者的区别外，最后还用一个具体的应用例子来说明两者的差异，体现了自己有很好的解说问题和设计案例的能力，思维敏捷，超过一般程序员，有写作能力！

106、是否可以从一个static方法内部发出对非static方法的调用？

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。

3、什么是内部类？

内部类就是在一个类的内部定义的类，内部类中不能定义静态成员（我想可能是既然静态成员类似c语言的全局变量，而内部类通常是用于创建内部对象用的，所以，把“全局变量”放在内部类中就是毫无意义的事情，既然是毫无意义的事情，就应该被禁止），内部类可以直接访问外部类中的成员变量，内部类可以定义在外部类的方法外面，也可以定义在外部类的方法体中，如下所示：

```
public class Outer
{
	int out_x  = 0;
	public void method()
	{
		Inner1 inner1 = new Inner1();
		class Inner2   //在方法体内部定义的内部类
		{
			public method()
			{
				out_x = 3;
			}
		}
		Inner2 inner2 = new Inner2();
	}

	public class Inner1   //在方法体外面定义的内部类
	{
	}
}
```

在方法体外面定义的内部类的访问类型可以是public,protecte,默认的，private等4种类型，这就好像类中定义的成员变量有4种访问类型一样，它们决定这个内部类的定义对其他类是否可见；对于这种情况，我们也可以在外面创建内部类的实例对象，创建内部类的实例对象时，一定要先创建外部类的实例对象，然后用这个外部类的实例对象去创建内部类的实例对象，代码如下：

```
Outer outer = new Outer();
Outer.Inner1 inner1 = outer.new Innner1();
```

在方法内部定义的内部类前面不能有访问类型修饰符，就好像方法中定义的局部变量一样，这种内部类对其他类是不可见的，虽然其他类无法引用这种内部类，但是这种内部类创建的实例对象可以传递给其他类访问。对于后者，内部类必须是先定义，后使用，即内部类的定义代码必须出现在使用该类之前，这与方法中的局部变量必须先定义后使用的道理也是一样的。这种情况下，在前面可以使用final或abstract修饰符。这种情况下的内部类可以访问方法体中的局部变量，但是，该局部变量前必须加final，

对于这些细节，只要在eclipse写代码试试，根据开发工具提示的各类错误信息就可以马上了解到。

在方法外部定义的内部类前面可以加上static关键字，从而成为静态内部类，这种静态内部类只是在引用时的编程语法上有一些区别，但在运行时的行为上与普通的外部类一样，所以，它也不能直接引用外部类的非static的成员变量。

最后，在方法体内部还可以采用如下语法来创建一种匿名内部类，即定义某一接口或类的子类的同时，还创建了该子类的实例对象，无需为该子类定义名称：

```
public class Outer
{
	public void start()
	{
		new Thread(
			new Runable(){
				public void run(){};
			}
		).start();
	}
}
```

备注：首先根据你的印象说出你对内部类的总体方面的特点：例如，在两个地方可以定义，可以访问外部类的成员变量，不能定义静态成员，这是大的特点。然后再说一些细节方面的知识，例如，几种定义方式的语法区别，静态内部类，以及匿名内部类。

121、内部类可以引用他包含类的成员吗？有没有什么限制？

完全可以。如果不是静态内部类，那没有什么限制！

如果静态内部类，一般情况下是不可以的，因为….，但是如果外部类中的成员是静态的，那也是可以的，例如，下面的代码：

```
class Outer
{
 static int x;
 static class Inner
 {
	void test()
	{
		syso(x);
	}
 }
}
```

如果问静态内部类能否访问外部类的成员这个问题，该如何回答：

答题时，也要能察言观色，揣摩提问者的心思，显然人家希望你说的是静态内部类不能访问外部类的成员，但你一上来就顶牛，这不好，要先顺着人家，让人家满意，然后再说特殊情况，让人家吃惊。

1、Integer与int的区别

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。

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。

26、Math.round(11.5)等於多少? Math.round(-11.5)等於多少?

Math类中提供了三个与取整有关的方法：ceil、floor、round，这些方法的作用与它们的英文名称的含义相对应，例如，ceil的英文意义是天花板，该方法就表示向上取整，Math.ceil(11.3)的结果为12,Math.ceil(-11.3)的结果是-11；floor的英文意义是地板，该方法就表示向下取整，Math.ceil(11.6)的结果为11,Math.ceil(-11.6)的结果是-12；最难掌握的是round方法，它表示“四舍五入”，算法为Math.floor(x+0.5)，即将原来的数字加上0.5后再向下取整，所以，Math.round(11.5)的结果为12，Math.round(-11.5)的结果为-11。

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

这四个作用域的可见范围如下表所示。
说明：如果在修饰的元素上面没有写任何访问修饰符，则表示friendly。

作用域	当前类	同一package	子孙类	其他package
public	√	√	√	√
protected	√	√	√	×
friendly	√	√	×	×
private	√	×	×	×

备注：只要记住了有4种访问权限，4个访问范围，然后将全选和范围在水平和垂直方向上分别按排从小到大或从大到小的顺序排列，就很容易画出上面的图了。

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

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。

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Override。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。

40、构造器Constructor是否可被override?

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

34、接口是否可继承接口? 抽象类是否可实现(implements)接口? 抽象类是否可继承具体类(concrete class)? 抽象类中是否可以有静态的main方法?

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。抽象类中可以有静态的main方法。

备注：只要明白了接口和抽象类的本质和作用，这些问题都很好回答，你想想，如果你是java语言的设计者，你是否会提供这样的支持，如果不提供的话，有什么理由吗？如果你没有道理不提供，那答案就是肯定的了。

107、写clone()方法时，通常都有一行代码，是什么？

clone 有缺省行为，super.clone();因为首先要把父类中的成员复制到位，然后才是复制自己的成员。

6、面向对象的特征有哪些方面

计算机软件系统是现实生活中的业务在计算机中的映射，而现实生活中的业务其实就是一个个对象协作的过程。面向对象编程就是按现实业务一样的方式将程序代码按一个个对象进行组织和编写，让计算机系统能够识别和理解用对象方式组织和编写的程序代码，这样就可以把现实生活中的业务对象映射到计算机系统中。

面向对象的编程语言有封装、继承 、抽象、多态等4个主要的特征。

1封装：

封装是保证软件部件具有优良的模块性的基础，封装的目标就是要实现软件部件的“高内聚、低耦合”，防止程序相互依赖性而带来的变动影响。在面向对象的编程语言中，对象是封装的最基本单位，面向对象的封装比传统语言的封装更为清晰、更为有力。面向对象的封装就是把描述一个对象的属性和行为的代码封装在一个“模块”中，也就是一个类中，属性用变量定义，行为用方法进行定义，方法可以直接访问同一个对象中的属性。通常情况下，只要记住让变量和访问这个变量的方法放在一起，将一个类中的成员变量全部定义成私有的，只有这个类自己的方法才可以访问到这些成员变量，这就基本上实现对象的封装，就很容易找出要分配到这个类上的方法了，就基本上算是会面向对象的编程了。

例如，人要在黑板上画圆，这一共涉及三个对象：人、黑板、圆，画圆的方法要分配给哪个对象呢？由于画圆需要使用到圆心和半径，圆心和半径显然是圆的属性，如果将它们在类中定义成了私有的成员变量，那么，画圆的方法必须分配给圆，它才能访问到圆心和半径这两个属性，人以后只是调用圆的画圆方法、表示给圆发给消息而已，画圆这个方法不应该分配在人这个对象上，**这就是面向对象的封装性，即将对象封装成一个高度自治和相对封闭的个体，对象状态（属性）由这个对象自己的行为（方法）来读取和改变。**一个更便于理解的例子就是，司机将火车刹住了，刹车的动作是分配给司机，还是分配给火车，显然，应该分配给火车，因为司机自身是不可能有那么大的力气将一个火车给停下来的，只有火车自己才能完成这一动作，火车需要调用内部的离合器和刹车片等多个器件协作才能完成刹车这个动作，司机刹车的过程只是给火车发了一个消息，通知火车要执行刹车动作而已。

抽象：

抽象就是找出一些事物的相似和共性之处，然后将这些事物归为一个类，这个类只考虑这些事物的相似和共性之处，并且会忽略与当前主题和目标无关的那些方面，将注意力集中在与当前目标有关的方面。抽象包括行为抽象和状态抽象两个方面。例如，定义一个Person类，如下：

```
class Person
```

```
{
    String name;
    int age;
}
```

人本来是很复杂的事物，有很多方面，但因为当前系统只需要了解人的姓名和年龄，所以上面定义的类中只包含姓名和年龄这两个属性，这就是一种抽像，使用抽象可以避免考虑一些与目标无关的细节。我对抽象的理解就是不要用显微镜去看一个事物的所有方面，这样涉及的内容就太多了，而是要善于划分问题的边界，当前系统需要什么，就只考虑什么。

继承：

在定义和实现一个类的时候，可以在一个已经存在的类的基础之上来进行，把这个已经存在的类所定义的内容作为自己的内容，并可以加入若干新的内容，或修改原来的方法使之更适合特殊的需要，这就是继承。继承是子类自动共享父类数据和方法的机制，这是类之间的一种关系，提高了软件的可重用性和可扩展性。

多态：

多态是指程序中定义的引用变量所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序运行期间才确定，即一个引用变量倒底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序运行期间才能决定。因为在程序运行时才确定具体的类，这样，不用修改源程序代码，就可以让引用变量绑定到各种不同的类实现上，从而导致该引用调用的具体方法随之改变，即不修改程序代码就可以改变程序运行时所绑定的具体代码，让程序可以选择多个运行状态，这就是多态性。多态性增强了软件的灵活性和扩展性。例如，下面代码中的UserDao是一个接口，它定义引用变量userDao指向的实例对象由daofactory.getDao()在执行的时候返回，有时候指向的是UserJdbcDao这个实现，有时候指向的是UserHibernateDao这个实现，这样，不用修改源代码，就可以改变userDao指向的具体类实现，从而导致userDao.insertUser()方法调用的具体代码也随之改变，即有时候调用的是UserJdbcDao的insertUser方法，有时候调用的是UserHibernateDao的insertUser方法：

```
UserDao userDao = daofactory.getDao();
userDao.insertUser(user);
```

比喻：人吃饭，你看到的是左手，还是右手？

102、java中实现多态的机制是什么？

靠的是父类或接口定义的引用变量可以指向子类或具体实现类的实例对象，而程序调用的方法就是引用所指向的具体实例对象的方法，也就是内存里正在运行的那个对象的方法，而不是引用变量的类型中定义的方法。

17、abstract class和interface有什么区别?

含有abstract修饰符的class即为抽象类，abstract 类不能创建的实例对象。含有abstract方法的类必须定义为abstract class，abstract class类中的方法不必是抽象的。abstract class类中定义抽象方法必须在具体子类中实现，所以，不能有抽象构造方法或抽象静态方法。如果的子类没有实现抽象父类中的所有抽象方法，那么子类也必须定义为abstract类型。

接口（interface）可以说成是抽象类的一种特例，接口中的所有方法都必须是抽象的。接口中的方法定义默认为public abstract类型，接口中的成员变量类型默认为public static final。

下面比较一下两者的语法区别：

1.抽象类可以有构造方法，接口中不能有构造方法。

2.抽象类中可以有普通成员变量，接口中没有普通成员变量

3.抽象类中可以包含非抽象的普通方法，接口中的所有方法必须都是抽象的，不能有非抽象的普通方法。

4. 抽象类中的抽象方法的访问类型可以是public，protected和默认类型，但接口中的抽象方法只能是public类型的，并且默认即为public abstract类型。

5. 抽象类中可以包含静态方法，接口中不能包含静态方法

6. 抽象类和接口中都可以包含静态成员变量，抽象类中的静态成员变量的访问类型可以任意，但接口中定义的变量只能是public static类型，并且默认即为public static类型。

7. 一个类可以实现多个接口，但只能继承一个抽象类。

下面接着再说说两者在应用上的区别：

接口更多的是在系统架构设计方法发挥作用，主要用于定义模块之间的通信契约。而抽象类在代码实现方面发挥作用，可以实现代码的重用，例如，模板方法设计模式是抽象类的一个典型应用，假设某个项目的所有Servlet类都要用相同的方式进行权限判断、记录访问日志和处理异常，那么就可以定义一个抽象的基类，让所有的Servlet都继承这个抽象基类，在抽象基类的service方法中完成权限判断、记录访问日志和处理异常的代码，在各个子类中完成各自的业务逻辑代码，伪代码如下：

```
public abstract class BaseServlet extends HttpServlet
{
    public void service(HttpServletRequest request, HttpServletResponse response) throws
IOExcetion,ServletException
    {
        记录访问日志
        进行权限判断
        if(具有权限)
        {
            try
            {
                doService(request,response);
            }
            catch(Excetpion e)
            {
                    记录异常信息
            }
        }
    }
    protected abstract void doService(HttpServletRequest request, HttpServletResponse response) throws
IOExcetion,ServletException;
  //注意访问权限定义成protected，显得既专业，又严谨，因为它是专门给子类用的
}

public class MyServlet1 extends BaseServlet
{
   protected void doService(HttpServletRequest request, HttpServletResponse response) throws
IOExcetion,ServletException
    {
        本Servlet处理的具体业务逻辑代码
    }

}
```

父类方法中间的某段代码不确定，留给子类干，就用模板方法设计模式。

备注：这道题的思路是先从总体解释抽象类和接口的基本概念，然后再比较两者的语法细节，最后再说两者的应用区别。比较两者语法细节区别的条理是：先从一个类中的构造方法、普通成员变量和方法（包括抽象方法），静态变量和方法，继承性等6个方面逐一去比较回答，接着从第三者继承的角度的回答，特别是最后用了一个典型的例子来展现自己深厚的技术功底。

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

abstract的method 不可以是static的，因为抽象的方法是要被子类实现的，而static与子类扯不上关系！

native方法表示该方法要用另外一种依赖平台的编程语言实现的，不存在着被子类实现的问题，所以，它也不能是抽象的，不能与abstract混用。例如，FileOutputSteam类要硬件打交道，底层的实现用的是操作系统相关的api实现，例如，在windows用c语言实现的，所以，查看jdk 的源代码，可以发现FileOutputStream的open方法的定义如下：

```
private native void open(String name) throws FileNotFoundException;
```

关于synchronized与abstract合用的问题，我觉得也不行，因为在我几年的学习和开发中，从来没见到过这种情况，并且我觉得synchronized应该是作用在一个具体的方法上才有意义。

21、Static Nested Class 和 Inner Class的不同。

参见前面的什么是内部类的那道题

Inner Class是在类的内部定义，它可以定义在方法体中，也可以定义在方法体外面。在方法体中时，不能家public/protected/private等修饰符，只能加abstract或final修饰符。
在方法体外面，可以加public/protected/private等修饰符。
在内部类中可以访问外部类中的成员变量。如果要在外面创建内部类的实例对象，显得Outer outer = new Outer();Outer.Inner inner = outer.new Inner();

static Nested Class只能在方法体外面定义，它可以在外面直接创建，
例如，Outer.Inner inner = new Outer.Inner ();，所以，static Nested Class 不能访问外部内的成员变量。
这与在外面直接定义一个类没有什么区别，只是这个类的名字更长了，变成了外部类名.内部类名。

Static Nested Class是被声明为静态（static）的内部类，它可以不依赖于外部类实例被实例化。而通常的内部类需要在外部类实例化后才能实例化。

115、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)?

可以继承其他类或完成其他接口，在swing编程中常用此方式。

2、String是最基本的数据类型吗?

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

111、String s = "Hello";s = s + " world!";这两行代码执行后，原始的String对象中的内容到底变了没有？

没有。因为String被设计成不可变(immutable)类，所以它的所有对象都是不可变对

象。在这段代码中，s原先指向一个String对象，内容是 ″Hello″，然后我们对s进行了+操作，那么s所指向的那个对象是否发生了改变呢？答案是没有。这时，s不指向原来那个对象了，而指向了另一个 String对象，内容为″Hello world!″，原来那个对象还存在于内存之中，只是s这个引用变量不再指向它了。
通过上面的说明，我们很容易导出另一个结论，如果经常对字符串进行各种各样的修改，或者说，不可预见的修改，那么使用String来代表字符串的话会引起很大的内存开销。因为 String对象建立之后不能再改变，所以对于每一个不同的字符串，都需要一个String对象来表示。这时，应该考虑使用StringBuffer类，它允许修改，而不是每个不同的字符串都要生成一个新的对象。并且，这两种类的对象转换十分容易。
同时，我们还可以知道，如果要使用内容相同的字符串，不必每次都new一个String。例如我们要在构造器中对一个名叫s的String引用变量进行初始化，把它设置为初始值，应当这样做：

```
public class Demo {
private String s;
...
public Demo {
s = "Initial Value";
}
...
}
```

而非

```
s = new String("Initial Value");
```

后者每次都会调用构造器，生成新对象，性能低下且内存开销大，并且没有意义，因为String对象不可改变，所以对于内容相同的字符串，只要一个String对象来表示就可以了。也就说，多次调用上面的构造器创建多个对象，他们的String类型属性s都指向同一个对象。
上面的结论还基于这样一个事实：对于字符串常量，如果内容相同，Java认为它们代表同一个String对象。而用关键字new调用构造器，总是会创建一个新的对象，无论内容是否相同。
至于为什么要把String类设计成不可变类，是它的用途决定的。其实不只String，很多Java标准类库中的类都是不可变的。在开发一个系统的时候，我们有时候也需要设计不可变类，来传递一组相关的值，这也是面向对象思想的体现。不可变类有一些优点，比如因为它的对象是只读的，所以多线程并发访问也不会有任何问题。当然也有一些缺点，比如每个不同的状态都要一个对象来代表，可能会造成性能上的问题。所以Java标准类库还提供了一个可变版本，即 StringBuffer。

41、是否可以继承String类?

String类是final类故不可以继承。

27、String s = new String("xyz");创建了几个String Object?二者之间有什么区别？

两个，一个放在常量区，不管写多少遍，都是同一个。New String每写一遍，就创建一个新。

5、String 和StringBuffer的区别

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。这个String类提供了数值不可改变的字符串。而这个StringBuffer类提供的字符串进行修改。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。

接着要举一个具体的例子来说明，我们要把1到100的所有数字拼起来，组成一个串。

```
StringBuffer sbf = new StringBuffer();
for(int i=0;i<100;i++)
{
    sbf.append(i);
}
```

上面的代码效率很高，因为只创建了一个StringBuffer对象，而下面的代码效率很低，因为创建了101个对象。

```
String str = new String();
for(int i=0;i<100;i++)
{
    str = str + i;
}
```

3、如何把一段逗号分割的字符串转换成一个数组?

如果不查jdk api，我很难写出来！我可以说说我的思路：

1. 用正则表达式，代码大概为：String [] result = orgStr.split(“,”);
2. 用 StingTokenizer ,代码为：StringTokenizer tokener = StringTokenizer(orgStr,”,”);
   String [] result = new String[tokener .countTokens()];
   Int i=0;
   while(tokener.hasNext(){result[i++]=toker.nextToken();}

38、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有有length()这个方法。

43、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。

```
public  class Test {

    /**
     * @param args add by zxx ,Dec 9, 2008
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println(new Test().test());;
    }

    static int test()
    {
        int x = 1;
        try
        {
            return x;
        }
        finally
        {
            ++x;
        }
    }

   }

   ---------执行结果 ---------
   1
```

12、final, finally, finalize的区别。

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

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。

15、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。

50、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。

33、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, NullPointerExcept ion, ProfileDataException, ProviderException, RasterFORMatException, SecurityException, SystemException, UndeclaredThrowableException, UnmodifiableSetException, UnsupportedOperationException

96、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？

Java通过面向对象的方法进行异常处理，把各种不同的异常进行分类，并提供了良好接口。在Java中，每个异常都是一个对象，它是Throwable类或其它子类的实例。当个方法出现异常后便抛出一个异常对象，该对象中包含有异常信息，调用这个对象的法可以捕获到这个异常并进行处理。Java的异常处理是通过5个关键词来实现的：try catch、throw、throws和finally。一般情况下是用try来执行一段程序，如果出现异常，系统会抛出（throws）一个异常，这时候你可以通过它的类型来捕捉（catch）它，或最后（finally）由缺省处理器来处理。

用try来指定一块预防所有"异常"的程序。紧跟在try程序后面，应包含一个catch子句来指定你想要捕捉的"异常"的类型。

throw语句用来明确地抛出一个"异常"。

throws用来标明一个成员函数可能抛出的各种"异常"。
Finally为确保一段代码不管发生什么"异常"都被执行一段代码。
可以在一个成员函数调用的外面写一个try语句，在这个成员函数内部写另一个try语句保护其他代码。每当遇到一个try语句，"异常"的框架就放到堆栈上面，直到所有的try语句都完成。如果下一级的try语句没有对某种"异常"进行处理，堆栈就会展开，直到遇到有处理这种"异常"的try语句。

99、java中有几种方法可以实现一个线程？用什么关键字修饰同步方法? stop()和suspend()方法为何不推荐使用？

有两种实现方法，分别使用new Thread()　和new Thread(runnable)形式，第一种直接调用thread的run方法，所以，我们往往使用Thread子类，即new SubThread()。第二种调用runnable的run方法。

有两种实现方法，分别是继承Thread类与实现Runnable接口
用synchronized关键字修饰同步方法
反对使用stop()，是因为它不安全。它会解除由线程获取的所有锁定，而且如果对象处于一种不连贯状态，那么其他线程能在那种状态下检查和修改它们。结果很难检查出真正的问题所在。suspend()方法容易发生死锁。调用suspend()的时候，目标线程会停下来，但却仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被"挂起"的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就会造成死锁。所以不应该使用suspend()，而应在自己的Thread类中置入一个标志，指出线程应该活动还是挂起。若标志指出线程应该挂起，便用wait()命其进入等待状态。若标志指出线程应当恢复，则用一个notify()重新启动线程。

13、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。

16、同步和异步有何异同，在什么情况下分别使用他们？举例说明。

如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。
当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。

17. 下面两个方法同步吗？（自己发明）

```
class Test
{
    synchronized static void sayHello3()
    {

    }

    synchronized void getX(){}
}
```

56、多线程有几种实现方法?同步有几种实现方法?

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
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。

30、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。

47、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?

分几种情况：
1.其他方法前是否加了synchronized关键字，如果没加，则能。
2.如果这个方法内部调用了wait，则可以进入其他synchronized方法。
3.如果其他个方法都加了synchronized关键字，并且内部没有调用wait，则不能。

58、线程的基本概念、线程的基本状态以及状态之间的关系

线程指在程序执行过程中，能够执行程序代码的一个执行单位，每个程序至少都有一个线程，也就是程序本身。
Java中的线程有四种状态分别是：运行、就绪、挂起、结束。

71、简述synchronized和java.util.concurrent.locks.Lock的异同？

主要相同点：Lock能完成synchronized所实现的所有功能
主要不同点：Lock有比synchronized更精确的线程语义和更好的性能。synchronized会自动释放锁，而Lock一定要求程序员手工释放，并且必须在finally从句中释放。

28、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();
     }
  }
}
}
```

3、ArrayList和Vector的区别

答：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

4、HashMap和Hashtable的区别

HashMap是Hashtable的轻量级实现（非线程安全的实现），他们都完成了Map接口，主要区别在于HashMap允许空（null）键值（key）,由于非线程安全，效率上可能高于Hashtable。
HashMap允许将null作为一个entry的key或者value，而Hashtable不允许。
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继承自Dictionary类，而HashMap是Java1.2引进的Map interface的一个实现。
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。

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

5、List 和 Map 区别?

35、List, Set, Map是否继承自Collection接口?

List，Set是，Map不是

109、List、Map、Set三个接口，存取元素时，各有什么特点？

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。

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。

4、去掉一个Vector集合中重复的元素

```
Vector newVector = new Vector();
For (int i=0;i<vector.size();i++)
{
    Object obj = vector.get(i);
    if(!newVector.contains(obj);
        newVector.add(obj);
}
```

还有一种简单的方式，HashSet set = new HashSet(vector);

9、Collection 和 Collections的区别。

Collection是集合类的上级接口，继承与他的接口主要有Set 和List.
Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

39、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。

53、你所知道的集合类都有哪些？主要方法？

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。

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

对。

如果对象要保存在HashSet或HashMap中，它们的equals相等，那么，它们的hashcode值就必须相等。

如果不是要保存在HashSet或HashMap，则与hashcode没有什么关系了，这时候hashcode不等是可以的，例如arrayList存储的对象就不用实现hashcode，当然，我们没有理由不实现，通常都会去实现的。

100、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。

105、什么是java序列化，如何实现java序列化？

序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时所引发的问题。

序列化的实现：将需要被序列化的类实现Serializable接口，该接口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，然后使用一个输出流(如：FileOutputStream)来构造一个ObjectOutputStream(对象流)对象，接着，使用ObjectOutputStream对象的writeObject(Object obj)方法就可以将参数为obj的对象写出(即保存其状态)，要恢复的话则用输入流。

54、描述一下JVM加载class文件的原理机制?

JVM中类的装载是由ClassLoader和它的子类来实现的,Java ClassLoader 是一个重要的Java运行时系统组件。它负责在运行时查找和装入类文件的类。

18、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

24、GC是什么? 为什么要有GC?

GC是垃圾收集的意思（Gabage Collection）,内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显示操作方法。

51、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。

103、垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收？

对于GC来说，当程序员创建对象时，GC就开始监控这个对象的地址、大小以及使用情况。通常，GC采用有向图的方式记录和管理堆(heap)中的所有对象。通过这种方式确定哪些对象是"可达的"，哪些对象是"不可达的"。当GC确定一些对象为"不可达"时，GC就有责任回收这些内存空间。可以。程序员可以手动执行System.gc()，通知GC运行，但是Java语言规范并不保证GC一定会执行。

23、什么时候用assert。

assertion(断言)在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制。在实现中，assertion就是在程序中的一条语句，它对一个boolean表达式进行检查，一个正确程序必须保证这个boolean表达式的值为true；如果该值为false，说明程序已经处于不正确的状态下，系统将给出警告或退出。一般来说，assertion用于保证程序最基本、关键的正确性。assertion检查通常在开发和测试时开启。为了提高性能，在软件发布后，assertion检查通常是关闭的。

101、java中会存在内存泄漏吗，请简单描述。

会。如：int i,i2; return (i-i2); //when i 为足够大的正数,i2为足够大的负数。结果会造成溢位，导致错误。

7、下面的程序代码输出的结果是多少？

```
public class  smallT
{
      public static void  main(String args[])
      {
            smallT t  = new  smallT();
            int  b  =  t.get();
            System.out.println(b);
      }

      public int  get()
      {
            try
            {
                  return 1 ;
            }
            finally
            {
                  return 2 ;
            }
      }
}
```

返回的结果是2。
我可以通过下面一个例子程序来帮助我解释这个答案，从下面例子的运行结果中可以发现，try中的return语句调用的函数先于finally中调用的函数执行，也就是说return语句先执行，finally语句后执行，所以，返回的结果是2。Return并不是让函数马上返回，而是return语句执行后，将把返回结果放置进函数栈中，此时函数并不是马上返回，它要执行finally语句后才真正开始返回。
在讲解答案时可以用下面的程序来帮助分析：

```
public  class Test {

    /**
     * @param args add by zxx ,Dec 9, 2008
```

```
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println(new Test().test());;
    }

    int test()
    {
        try
        {
            return func1();
        }
        finally
        {
            return func2();
        }
    }

    int func1()
    {
        System.out.println("func1");
        return 1;
    }
    int func2()
    {
        System.out.println("func2");
        return 2;
    }
}
```

----------执行结果----------

```
func1
func2
2
```

结论：finally中的代码比return 和break语句后执行

8、下面程序的输出结果是多少？

```
import java.util.Date;
public  class Test extends Date{

    /**
     * @param args add by zxx ,Dec 9, 2008
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new Test().test();

    }

    public void test()
    {
        System.out.println(
                super.getClass().getName()
        );
```

```
    }
}
```

很奇怪，结果是Test

这属于脑筋急转弯的题目，我同事拿这个开过我的玩笑，所以，我正好知道。
在test方法中，直接调用getClass().getName()方法，返回的是Test类名
由于getClass()在Object类中定义成了final，子类不能覆盖该方法，所以，在
test方法中调用getClass().getName()方法，其实就是在调用父类的getClass()方法，
等效于调用super.getClass().getName()方法，所
以，super.getClass().getName()方法返回的也应该是Test。
如果想得到父类的名称，应该用如下代码：

```
getClass().getSuperClass().getName();
```

112、说出一些常用的类，包，接口，请各举5个

常用的类：BufferedReader BufferedWriter FileReader FileWirter String Integer
常用的包：java.lang java.awt java.io java.util java.sql
常用的接口：Remote List Map Document NodeList

2. Java代码查错

1.

```
abstract class Name {
  private String name;
  public abstract boolean isStupidName(String name) {}
}
```

大侠们，这有何错误?
答案: 错。abstract method必须以分号结尾，且不带花括号。
2.

```
public class Something {
  void doSomething () {
     private String s = "";
     int l = s.length();
  }
}
```

有错吗?
答案: 错。局部变量前不能放置任何访问修饰符 (private，public，和protected)。final可以用来修饰局部变量
(final如同abstract和strictfp，都是非访问修饰符，strictfp只能修饰class和method而非variable)。
3.

```
abstract class Something {
    private abstract String doSomething ();
}
```

这好像没什么错吧?
答案: 错。abstract的methods不能以private修饰。abstract的methods就是让子类implement(实现)具体细节的，怎么可以用private把abstract
method封锁起来呢? (同理，abstract method前不能加final)。
4.

```
public class Something {
```

```
  public int addOne(final int x) {
    return ++x;
  }
}
```
这个比较明显。
答案: 错。int x被修饰成final，意味着x不能在addOne method中被修改。
5.
```
public class Something {
  public static void main(String[] args) {
    Other o = new Other();
    new Something().addOne(o);
  }
  public void addOne(final Other o) {
    o.i++;
  }
}
class Other {
  public int i;
}
```
和上面的很相似，都是关于final的问题，这有错吗?
答案: 正确。在addOne method 中，参数o被修饰成final。如果在addOne method里我们修改了o的reference
(比如: o = new Other();)，那么如同上例这题也是错的。但这里修改的是o的member vairable (成员变量)，而o的reference并没有改变。
6.
```
class Something {
   int i;
   public void doSomething() {
     System.out.println("i = " + i);
   }
}
```
有什么错呢? 看不出来啊。
答案: 正确。输出的是"i = 0" 。int i 属於instant variable (实例变量，或叫成员变量)。instant variable有default value。int的default value是0。
7.
```
class Something {
   final int i;
   public void doSomething() {
     System.out.println("i = " + i);
   }
}
```
和上面一题只有一个地方不同，就是多了一个final。这难道就错了吗?
答案: 错。final int i是个final的instant variable (实例变量，或叫成员变量)。final的instant variable没有default value ，必须在constructor (构造器)结束之前被赋予一个明确的值。可以修改为"final int i = 0;"。
8.
```
public class Something {
    public static void main(String[] args) {
      Something s = new Something();
      System.out.println("s.doSomething() returns " + doSomething());
   }
   public String doSomething() {
      return "Do something ...";
   }
}
```
看上去很完美。

答案:　　错。看上去在main里call doSomething没有什么问题，毕竟两个methods都在同一个class里。但仔细看，main是static的。static method 不能直接call non-static methods 。可改成"System.out.println("s.doSomething() returns " + s.doSomething());" 。同理，static method不能访问non-static instant variable。

9.

此处，Something类的文件名叫OtherThing.java

```
class Something {
   private static void main(String[] something_to_do) {
      System.out.println("Do something ...");
   }
}
```

这个好像很明显。

答案: 正确。从来没有人说过Java的Class名字必须和其文件名相同。但public class的名字必须和文件名相同。

10.

```
interface  A{
  int x = 0;
}
class B{
  int x =1;
}
class C extends B implements A {
  public void pX(){
    System.out.println(x);
  }
  public static void main(String[] args) {
    new C().pX();
  }
}
```

答案：错误。在编译时会发生错误(错误描述不同的JVM有不同的信息，意思就是未明确的x调用，两个x都匹配（就象在同时import java.util和java.sql两个包时直接声明Date一样）。对于父类的变量,可以用super.x来明确，而接口的属性默认隐含为 public static final. 所以可以通过A.x来明确。

11.

```
interface Playable {
   void play();
}
interface Bounceable {
   void play();
}
interface Rollable extends Playable, Bounceable {
   Ball ball = new Ball("PingPang");
}
class Ball implements Rollable {
   private String name;
   public String getName() {
      return name;
   }
   public Ball(String name) {
      this.name = name;
   }
  public void play() {
      ball = new Ball("Football");
      System.out.println(ball.getName());
   }
}
```

这个错误不容易发现。
答案: 错。"interface Roll able extends Playable, Bounceable" 没有问题。interface可继承多个interfaces，所以这里没错。问题出在interface Rollable 里的"Ball ball = new Ball("PingPang");"。任何在interface里声明的interface variable (接口变量，也可称成员变量)，默认为public static final 。也就是说"Ball ball = new Ball("PingPang");" 实际上是"public static final Ball ball = new Ball("PingPang");" 。在Ball类的Play()方法中，"ball = new Ball("Football");"改变了ball的reference，而这里的ball来自Rollable interface ，Rollable interface里的ball是public static final 的，final的object是不能被改变reference的。因此编译器将在"ball = new Ball("Football");"这里显示有错。

2. 算法与编程

48、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

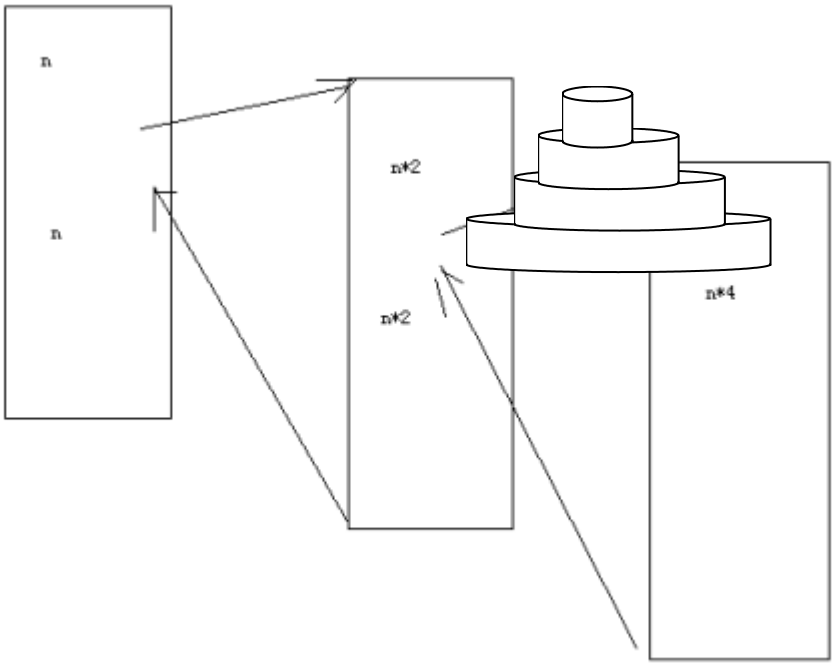
7、递归算法题1

一个整数，大于0，不用循环和本地变量，按照n，2n，4n，8n的顺序递增，当值大于5000时，把值按照指定顺序输出来。
例：n=1237
则输出为：

1237，
2474，
4948，
9896，
9896，
4948，
2474，
1237，

```
    public static void multiply(int n)
    {
        if(n>5000) return;
        System.out.println(n);
        multiply(n*2);
        System.out.println(n);
    }
```

Gaibaota(N) = Gaibaota(N-1) + n



7、递归算法题2

第1个人10，第2个比第1个人大2岁，依次地退，第8个人多大？

```
package cn.itcast;

import java.util.Date;

public class A1 {

    public static void main(String [] args)
    {
        System.out.println(computeAge(8));
    }
```

```
    public static int computeAge(int n)
    {
        if(n==1) return 10;
        return computeAge(n-1) + 2;
    }
}

    public static void toBinary(int n,StringBuffer result)
    {

        if(n/2 != 0)
            toBinary(n/2,result);
        result.append(n%2);
    }
```

94、排序都有哪几种方法？请列举。用JAVA实现一个快速排序。

本人只研究过冒泡排序、选择排序和快速排序，下面是快速排序的代码：

```
public class QuickSort {
/**
* 快速排序
* @param strDate
* @param left
* @param right
*/
public void quickSort(String[] strDate,int left,int right){
String middle,tempDate;
int i,j;
i=left;
j=right;
middle=strDate[(i+j)/2];
do{
while(strDate[i].compareTo(middle)<0&& i<right)
i++; //找出左边比中间值大的数
while(strDate[j].compareTo(middle)>0&& j>left)
j--; //找出右边比中间值小的数
if(i<=j){ //将左边大的数和右边小的数进行替换
tempDate=strDate[i];
strDate[i]=strDate[j];
strDate[j]=tempDate;
i++;
j--;
```

```
}
}while(i<=j); //当两者交错时停止

if(i<right){
quickSort(strDate,i,right);//从
}
if(j>left){
quickSort(strDate,left,j);
}
}
/**
 * @param args
 */
public static void main(String[] args){
String[] strVoid=new String[]{"11","66","22","0","55","22","0","32"};
QuickSort sort=new QuickSort();
sort.quickSort(strVoid,0,strVoid.length-1);
for(int i=0;i<strVoid.length;i++){
System.out.println(strVoid[i]+" ");
}
}
}
```

7、有数组a[n]，用java代码将数组元素顺序颠倒

```
package cn.itcast.lecture2;

import java.util.Arrays;
import java.util.Collections;
public class ReverseTest {
    public static void main(String [] args)
    {
        //产生若干0到1000的随机数，作为数组的初始值
        int data[] = new int[]{
            (int)(Math.random() * 1000),
            (int)(Math.random() * 1000),
            (int)(Math.random() * 1000),
            (int)(Math.random() * 1000),
            (int)(Math.random() * 1000),
            (int)(Math.random() * 1000),
            (int)(Math.random() * 100),
        };
        System.out.println(Math.random());
        System.out.print("交换前的数据: ");
        System.out.println(
                Arrays.toString(data));
```

```
        reverse(data);
        System.out.print("交换后的数据：");
        System.out.println(Arrays.toString(data));
    }

    //方法执行完后，参数data中的数据顺序即被颠倒
    //实现思路是第1个和第n个交换，第2个和第n-1个交换，依次类推...
    public static void reverse(int[] data)
    {

        int len = data.length;
        for(int i=0;i<len/2;i++)
        {
            int temp = data[i];
            data[i] = data[len-1-i];
            data[len-1-i] = temp;
        }
    }
}
```

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。

```
public class RenMingBi {
    /**
     * @param args add by zxx ,Nov 29, 2008
     */
    private static final char[] data = new char[]{
            '零','壹','贰','叁','肆','伍','陆','柒','捌','玖'
        };
    private static final char[] units = new char[]{
        '元','拾','佰','仟','万','拾','佰','仟','亿'
    };
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println(
                convert(135689123));
    }
    public static String convert(int money)
    {
        StringBuffer sbf = new StringBuffer();
        int unit = 0;
        while(money!=0)
        {
            sbf.insert(0,units[unit++]);
            int number = money%10;
            sbf.insert(0, data[number]);
            money /= 10;
        }
        return sbf.toString();
    }
}
```

2. html&JavaScript部分

1. 用table显示n条记录，每3行换一次颜色，即1，2，3用红色字体，4，5，6用绿色字体，7，8，9用红颜色字体。

1、HTML 的 form 提交之前如何验证数值不为空? 为空的话提示用户并终止提交?

2、请写出用于校验HTML文本框中输入的内容全部为数字的javascript代码

```
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);
}
```

3. Java web部分

1、HTTP请求的GET与POST方式的区别

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。

62、解释一下什么是servlet;

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。

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。

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。
web容器加载servlet，生命周期开始。通过调用servlet的init()方法进行servlet的初始化。通过调用service()方法实现，根据请求的不同调用不同的do***()方法。结束服务，web容器调用servlet的destroy()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 {
}
}
```

3、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。

60、什么情况下调用doGet()和doPost()？

Jsp页面中的FORM标签里的method属性为get时调用doGet()，为post时调用doPost()。

66、Request对象的主要方法：

setAttribute(String name,Object)：设置名字为name的request的参数值
getAttribute(String name)：返回由name指定的属性值
getAttributeNames()：返回request对象所有属性的名字集合，结果是一个枚举的实例
getCookies()：返回客户端的所有Cookie对象，结果是一个Cookie数组
getCharacterEncoding()：返回请求中的字符编码方式
getContentLength()：返回请求的Body的长度
getHeader(String name)：获得HTTP协议定义的文件头信息
getHeaders(String name) ：返回指定名字的request Header的所有值，结果是一个枚举的实例
getHeaderNames()：返回所以request Header的名字，结果是一个枚举的实例
getInputStream()：返回请求的输入流，用于获得请求中的数据
getMethod()：获得客户端向服务器端传送数据的方法
getParameter(String name)：获得客户端传送给服务器端的有name指定的参数值
getParameterNames()：获得客户端传送给服务器端的所有参数的名字，结果是一个枚举的实例
getParametervalues(String name)：获得有name指定的参数的所有值
getProtocol()：获取客户端向服务器端传送数据所依据的协议名称
getQueryString()：获得查询字符串
getRequestURI()：获取发出请求字符串的客户端地址
getRemoteAddr()：获取客户端的IP地址
getRemoteHost()：获取客户端的名字
getSession([Boolean create])：返回和请求相关Session
getServerName()：获取服务器的名字
getServletPath()：获取客户端所请求的脚本文件的路径
getServerPort()：获取服务器的端口号
removeAttribute(String name)：删除请求中的一个属性

19、forward 和redirect的区别

forward是服务器请求资源，服务器直接访问目标地址的URL，把那个URL的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。
redirect就是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以session,request参数都可以获取。

4、request.getAttribute() 和 request.getParameter() 有何区别?

1. jsp有哪些内置对象?作用分别是什么? 分别有什么方法?

答:JSP共有以下9个内置的对象：
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

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 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
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表示从该页面产生的一个servlet实例

2. jsp有哪些动作?作用分别是什么?

（这个问题似乎不重要，不明白为何有此题）
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

59、JSP的常用指令

isErrorPage(是否能使用Exception对象)，isELIgnored(是否忽略表达式)

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?

（下面的回答严重错误，应该是想问forward和sendRedirect 的区别，毕竟出题的人不是专业搞文字艺术的人，可能表达能力并不见得很强，用词不一定精准，加之其自身的技术面也可能存在一些问题，不一定真正将他的意思表达清楚了，严格意思上来讲，一些题目可能根本就无人能答，所以，答题时要掌握主动，只要把自己知道的表达清楚就够了，而不要去推敲原始题目的具体含义是什么，不要一味想着是在答题）
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。

63、页面间对象传递的方法

request，session，application，cookie等

64、JSP和Servlet有哪些相同点和不同点，他们之间的联系是什么？

JSP是Servlet技术的扩展，本质上是Servlet的简易方式，更强调应用的外表表达。JSP编译后是"类servlet"。Servlet和JSP最主要的不同点在于，Servlet的应用逻辑是在Java文件中，并且完全从表示层中的HTML里分离开来。而JSP的情况是Java和HTML可以组合成一个扩展名为.jsp的文件。JSP侧重于视图，Servlet主要用于控制逻辑。

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。

68、我们在web应用开发过程中经常遇到输出某种编码的字符，如iso8859-1等，如何输出一个某种编码的字符串？

```
Public String translate (String str) {
  String tempStr = "";
  try {
    tempStr = new String(str.getBytes("ISO-8859-1"), "GBK");
    tempStr = tempStr.trim();
  }
  catch (Exception e) {
    System.err.println(e.getMessage());
  }
  return tempStr;
}
```

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

4. 数据库部分

1、数据库三范式是什么?

2、union和union all有什么不同?

3、oracle关联查询题目1

数据库中有3个表 teacher 表，student表，tea_stu关系表。
teacher 表 teaID name age
student 表 stuID name age
teacher_student表 teaID stuID
要求用一条sql查询出这样的结果
  1.显示的字段要有老师name, age 每个老师所带的学生人数
2 只列出老师age为40以下学生age为12以上的记录

实验准备：

```
drop table if exists tea_stu;
drop table if exists teacher;
drop table if exists student;
```

```
create table teacher(teaID int primary key,name varchar(50),age int);
create table student(stuID int primary key,name varchar(50),age int);
create table tea_stu(teaID int references teacher(teaID),stuID int references student(stuID));
 insert into teacher values(1,'zxx',45), (2,'lhm',25) , (3,'wzg',26) , (4,'tg',27);
 insert into student values(1,'wy',11), (2,'dh',25) , (3,'ysq',26) , (4,'mxc',27);
 insert into tea_stu values(1,1), (1,2), (1,3);
 insert into tea_stu values(2,2), (2,3), (2,4);
  insert into tea_stu values(3,3), (3,4), (3,1);
 insert into tea_stu values(4,4), (4,1), (4,2) , (4,3);
```

结果：2→3,3→2,4→3

解题思路：（真实面试答题时，也要写出每个分析步骤，如果纸张不够，就找别人要）

1要会统计分组信息：

```
select teaid,count(*) from tea_stu group by teaid;
```

2.要会筛选大于40的老师

先讲给1号学生带课的所有老师的名字，要知道1号学生的老师，要从哪个表获得？从tea_stu里才可以，从这个表里获得的是老师的什么？是老师的teaID，如果要得到老师的名称，则必须拿着teaID去teacher里找。所以，这两个表要进行关联，关联就引出迪卡尔既的问题。假设我们好获得teaID为1的老师的名称，我们要关联的是哪条记录？是1那一条，但是，迪卡尔既的结果有几条啊？4条，显然，我接着要做的就是把其他垃圾的3条去掉。

```
select * from tea_stu,teacher where tea_stu.teaID=teacher.teaID;
```

再接着去掉大于40的老师：

```
select * from tea_stu,teacher where tea_stu.teaID=teacher.teaID and teacher.age<=40;
```

3.再对上面的结果去掉小于12的学生

```
select * from
        (select tea_stu.* from tea_stu,teacher where tea_stu.teaID=
        teacher.teaID and teacher.age<40) as t,
        student
where t.stuid=student.stuid and student.age>12;
```

4.再对上面的结果进行统计，显示的是老师的id和组信息

```
select t.teaID,count(*) from
        (select tea_stu.* from tea_stu,teacher where tea_stu.teaID=
        teacher.teaID and teacher.age<40) as t,
        student
where t.stuid=student.stuid and student.age>12
group by t.teaID;
```

5.然后对上面的东西进行改写，改写成显示老师的名字

```
    select teacher.name,t2.c from
        (select t.teaID,count(*) c from
            (select tea_stu.* from tea_stu,teacher where tea_stu.teaID=
            teacher.teaID and teacher.age<40) as t,
            student
        where t.stuid=student.stuid and student.age>12
        group by t.teaID) as t2,
        teacher
    where teacher.teaID=t2.teaID;
```

第二种写法：

```
select teacher.teaID, teacher.name, t1.total from teacher,
        (select teaID,count(tea_stu.stuID) total from tea_stu, student
        where tea_stu.stuID = student.stuID and student.age>12
        group by teaID ) as t1
where teacher.teaID = t1.teaID and teacher.age<40 ;
```

求出发帖最多的人：

```
select authorid,count(*) total from articles
group by authorid
having total=
    (select max(total2) from (select count(*) total2 from articles group by authorid) as t);
```

1、什么是存储过程和如何编写

1、注册Jdbc驱动程序的三种方式

1、用JDBC如何调用存储过程

1、JDBC中的PreparedStatement相比Statement的好处

1. 写一个用jdbc连接并访问oracle数据的程序代码

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、大数据量下的分页解决方法。

4、用 JDBC 查询学生成绩单,把主要代码写出来.

5、这段代码有什么不足之处?

```
try {
Connection conn = ...;
Statement stmt = ...;
ResultSet rs = stmt.executeQuery("select * from table1");
while(rs.next()) {
}
} catch(Exception ex) {
}
```

36、说出数据连接池的工作机制是什么?

J2EE服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。

4、为什么要用 ORM? 和 JDBC 有何不一样?

5. XML部分

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)
讲解这些区别是不需要特别去比较，就像说传智播客与其他培训机构的区别时，我们只需说清楚传智播客有什么特点和优点就行了，这就已经间接回答了彼此的区别。

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>
```

70、XML文档定义有几种形式？它们之间有何本质区别？解析XML文档有哪几种方式？

a: 两种形式 dtd schema ，b: 本质区别:schema本身是xml的，可以被XML解析器解析(这也是从DTD上发展schema的根本目的)，c: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)

6. j2ee部分

117、BS与CS的联系与区别。

C/S是Client/Server的缩写。服务器通常采用高性能的PC、工作站或小型机，并采用大型数据库系统，如Oracle、Sybase、InFORMix或 SQL Server。客户端需要安装专用的客户端软件。

B/S是Brower/Server的缩写，客户机上只要安装一个浏览器（Browser），如Netscape Navigator或Internet Explorer，服务器安装Oracle、Sybase、InFORMix或 SQL Server等数据库。在这种结构下，用户界面完全通过WWW浏览器实现，一部分事务逻辑在前端实现，但是主要事务逻辑在服务器端实现。浏览器通过Web Server 同数据库进行数据交互。

C/S 与 B/S 区别：

1．硬件环境不同:

C/S 一般建立在专用的网络上, 小范围里的网络环境, 局域网之间再通过专门服务器提供连接和数据交换服务.

B/S 建立在广域网之上的, 不必是专门的网络硬件环境,例与电话上网, 租用设备. 信息自己管理. 有比C/S更强的适应范围, 一般只要有操作系统和浏览器就行

2．对安全要求不同

C/S 一般面向相对固定的用户群, 对信息安全的控制能力很强. 一般高度机密的信息系统采用C/S 结构适宜. 可以通过B/S发布部分可公开信息.

B/S 建立在广域网之上, 对安全的控制能力相对弱, 可能面向不可知的用户。

3．对程序架构不同

C/S 程序可以更加注重流程, 可以对权限多层次校验, 对系统运行速度可以较少考虑.

B/S 对安全以及访问速度的多重的考虑, 建立在需要更加优化的基础之上. 比C/S有更高的要求 B/S 结构的程序架构是发展的趋势, 从MS的.Net系列的BizTalk 2000 Exchange 2000 等, 全面支持网络的构件搭建的系统. SUN 和IBM推的JavaBean 构件技术等,使 B/S更加成熟.

4．软件重用不同

C/S 程序可以不可避免的整体性考虑, 构件的重用性不如在B/S要求下的构件的重用性好.

B/S 对的多重结构,要求构件相对独立的功能. 能够相对较好的重用.就入买来的餐桌可以再利用,而不是做在墙上的石头桌子

5．系统维护不同

C/S 程序由于整体性, 必须整体考察, 处理出现的问题以及系统升级. 升级难. 可能是再做一个全新的系统

B/S 构件组成,方面构件个别的更换,实现系统的无缝升级. 系统维护开销减到最小. 用户从网上自己下载安装就可以实现升级.

6．处理问题不同

C/S 程序可以处理用户面固定, 并且在相同区域, 安全要求高需求, 与操作系统相关. 应该都是相同的系统

B/S 建立在广域网上, 面向不同的用户群, 分散地域, 这是C/S无法作到的. 与操作系统平台关系最小.

7．用户接口不同

C/S 多是建立的Window平台上,表现方法有限,对程序员普遍要求较高

B/S 建立在浏览器上, 有更加丰富和生动的表现方式与用户交流. 并且大部分难度减低,减低开发成本.

8．信息流不同

C/S 程序一般是典型的中央集权的机械式处理, 交互性相对低

B/S 信息流向可变化, B-B B-C B-G等信息、流向的变化, 更像交易中心。

2、应用服务器与WEB SERVER的区别？

应用服务器：Weblogic、Tomcat、Jboss
WEB SERVER：IIS、 Apache

32、应用服务器有那些？

BEA WebLogic Server ，IBM WebSphere Applicatio n Server ，Oracle9i Application Server，jBoss，Tomcat

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。

一个另类的回答：j2ee就是增删改查。

67、J2EE是技术还是平台还是框架？ 什么是J2EE

J2EE本身是一个标准，一个为企业分布式应用的开发提供的标准平台。
J2EE也是一个框架，包括JDBC、JNDI、RMI、JMS、EJB、JTA等技术。

95、请对以下在J2EE中常用的名词进行解释(或简单描述)

web容器：给处于其中的应用程序组件（JSP，SERVLET）提供一个环境，使JSP,SERVLET直接更容器中的环境变量接口交互，不必关注其它系统问题。主要有WEB服务器来实现。例如：TOMCAT,WEBLOGIC,WEBSPHERE等。该容器提供的接口严格遵守J2EE规范中的WEB APPLICATION 标准。我们把遵守以上标准的WEB服务器就叫做J2EE中的WEB容器。

EJB容器：Enterprise java bean 容器。更具有行业领域特色。他提供给运行在其中的组件EJB各种管理功能。只要满足J2EE规范的EJB放入该容器，马上就会被容器进行高效率的管理。并且可以通过现成的接口来获得系统级别的服务。例如邮件服务、事务管理。

JNDI：（Java Naming & Directory Interface）JAVA命名目录服务。主要提供的功能是：提供一个目录系统，让其它各地的应用程序在其上面留下自己的索引，从而满足快速查找和定位分布式应用程序的功能。

JMS：（Java Message Service ）JAVA消息服务。主要实现各个应用程序之间的通讯。包括点对点和广播。

JTA：（Java Transaction API）JAVA事务服务。提供各种分布式事务服务。应用程序只需调用其提供的接口即可。

JAF：（Java Action FrameWork ）JAVA安全认证框架。提供一些安全控制方面的框架。让开发者通过各种部署和自定义实现自己的个性安全控制策略。

RMI/IIOP:（Remote Method Invocation /internet 对象请求中介协议）他们主要用于通过远程调用服务。例如，远程有一台计算机上运行一个程序，它提供股票分析服务，我们可以在本地计算机上实现对其直接调用。当然这是要通过一定的规范才能在异构的系统之间进行通信。RMI是JAVA特有的。

80、如何给weblogic指定大小的内存?

（这个问题不作具体回答，列出来只是告诉读者可能会遇到什么问题，你不需要面面俱到，什么都精通。）

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

81、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。

82、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.

83、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。

84、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。

85、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

86、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。

87、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. ejb部分

8、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。

EJB 包括Session Bean、Entity Bean、Message Driven Bean，基于JNDI、RMI、JAT等技术实现。
SessionBean在J2EE应用程序中被用来完成一些服务器端的业务操作，例如访问数据库、调用其他EJB组件。EntityBean被用来代表应用系统中用到的数据。
对于客户机，SessionBean是一种非持久性对象，它实现某些在服务器上运行的业务逻辑。
对于客户机，EntityBean是一种持久性对象，它代表一个存储在持久性存储器中的实体的对象视图，或是一个由现有企业应用程序实现的实体。
Session Bean 还可以再细分为 Stateful Session Bean 与 Stateless Session Bean ，这两种的 Session Bean 都可以将系统逻辑放在 method 之中执行，不同的是 Stateful Session Bean 可以记录呼叫者的状态，因此通常来说，一个使用者会有一个相对应的 Stateful Session Bean 的实体。Stateless Session Bean 虽然也是逻辑组件，但是他却不负责记录使用者状态，也就是说当使用者呼叫 Stateless Session Bean 的时候，EJB Container 并不会找寻特定的 Stateless Session Bean 的实体来执行这个 method 。换言之，很可能数个使用者在执行某个 Stateless Session Bean 的 methods 时，会是同一个 Bean 的 Instance 在执行。从内存方面来看，Stateful Session Bean 与 Stateless Session Bean 比较， Statef ul Session Bean 会消耗 J2EE Server 较多的内存，然而 Stateful Session Bean 的优势却在于他可以维持使用者的状态。

2、简要讲一下 EJB 的 7 个 Transaction Level?

3、EJB与JAVA BEAN的区别？

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。

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

SessionBean：Stateless Session Bean 的生命周期是由容器决定的，当客户机发出请求要建立一个Bean的实例时，EJB容器不一定要创建一个新的Bean的实例供客户机调用，而是随便找一个现有的实例提供给客户机。当客户机第一次调用一个Stateful Session Bean 时，容器必须立即在服务器中创建一个新的Bean实例，并关联到客户机上，以后此客户机调用Stateful Session Bean 的方法时容器会把调用分派到与此客户机相关联的Bean实例。

EntityBean：Entity Beans能存活相对较长的时间，并且状态是持续的。只要数据库中的数据存在，Entity beans就一直存活。而不是按照应用程序或者服务进程来说的。即使EJB容器崩溃了，Entity beans 也是存活的。Entity Beans 生命周期能够被容器或者Beans自己管理。

EJB通过以下技术管理实务：对象管理组织（OMG）的对象实务服务（OTS），Sun Microsystems的Transaction Service （JTS）、Java Transaction API （JTA），开发组（X/Open）的XA接口。

73、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。

77、EJB的激活机制

以Stateful 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。

78、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

79、客服端调用EJB对象的几个基本步骤

设置JNDI服务工厂以及JNDI服务地址系统属性，查找Home接口，从Home接口调用Create方法创建Remote接口，通过Remote接口调用其业务方法。

8. webservice部分

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

Web ServiceWeb Service 是基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得Web Service能与其他兼容的组件进行互操作。

JAXP(Java API for XML Parsing) 定义了在Java中使用DOM, SAX, XSLT的通用的接口。这样在你的程序中你只要使用这些通用的接口，当你需要改变具体的实现时候也不需要修改代码。

JAXM(Java API for XML Messaging) 是为SOAP通信提供访问方法和传输机制的API。

WSDL是一种 XML 格式，用于将网络服务描述为一组端点，这些端点对包含面向文档信息或面向过程信息的消息进行操作。这种格式首先对操作和消息进行抽象描述，然后将其绑定到具体的网络协议和消息格式上以定义端点。相关的具体端点即组合成为抽象端点（服务）。

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。

88、CORBA是什么?用途是什么?

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。

5. 流行的框架与新技术

11、谈谈Struts中的Action servlet。

12、Struts优缺点

优点：

1. 实现MVC模式，结构清晰,使开发者只关注业务逻辑的实现.

2．有丰富的tag可以用 ,Struts的标记库(Taglib)，如能灵活动用，则能大大提高开发效率

3. 页面导航

使系统的脉络更加清晰。通过一个配置文件，即可把握整个系统各部分之间的联系，这对于后期的维护有着莫大的好处。尤其是当另一批开发者接手这个项目时，这种优势体现得更加明显。

4. 提供Exception处理机制 .
5. 数据库链接池管理
6. 支持I18N

缺点

1、 转到展示层时，需要配置forward，如果有十个展示层的jsp，需要配置十次struts，而且还不包括有时候目录、文件变更，需要重新修改forward，注意，每次修改配置之后，要求重新部署整个项目，而tomcate这样的服务器，还必须重新启动服务器
2、 二、 Struts 的Action必需是thread－safe方式，它仅仅允许一个实例去处理所有的请求。所以action用到的所有的资源都必需统一同步，这个就引起了线程安全的问题。
3、 测试不方便. Struts的每个Action都同Web层耦合在一起，这样它的测试依赖于Web容器，单元测试也很难实现。不过有一个Junit的扩展工具Struts TestCase可以实现它的单元测试。
4、 类型的转换. Struts的FormBean把所有的数据都作为String类型，它可以使用工具Commons-Beanutils进行类型转化。但它的转化都是在Class级别，而且转化的类型是不可配置的。类型转化时的错误信息返回给用户也是非常困难的。
5、 对Servlet的依赖性过强. Struts 处理Action时必需要依赖ServletRequest和ServletResponse，所有它摆脱不了Servlet容器。
6、 前端表达式语言方面.Struts集成了JSTL，所以它主要使用JSTL的表达式语言来获取数据。可是JSTL的表达式语言在Collection和索引属性方面处理显得很弱。
7、 对Action执行的控制困难. Struts创建一个Action，如果想控制它的执行顺序将会非常困难。甚至你要重新去写Servlet来实现你的这个功能需求。
8、 对Action 执行前和后的处理. Struts处理Action的时候是基于class的hierarchies，很难在action处理前和后进行操作。
9、 对事件支持不够. 在struts中，实际是一个表单Form对应一个Action类(或DispatchAction)，换一句话说：在Struts中实际是一个表单只能 对应一个事件，struts这种事件方式称为application event ， application event 和component event相比是一种粗粒度的事件

119、STRUTS的应用(如STRUTS架构)

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。

110、hibernate中的update()和saveOrUpdate()的区别，session的load()和get()的区别。

110、简述 Hibernate 和 JDBC 的优缺点? 如何书写一个 one to many 配置文件.

7、iBatis与Hibernate有什么不同?

111、Spring 的依赖注入是什么意思? 给一个 Bean 的 message 属性, 字符串类型, 注入值为 "Hello" 的 XML 配置文件该怎么写?

120、Jdo是什么?

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。

什么是spring的IOC AOP

STRUTS的工作流程!

spring 与EJB的区别！！

9. 软件工程与设计模式

111、UML方面

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。

112. 软件开发的

92、j2ee常用的设计模式？说明工厂模式。

总共23种，分为三大类：创建型，结构型，行为型
我只记得其中常用的6、7种，分别是：
创建型（工厂、工厂方法、抽象工厂、单例）
结构型（包装、适配器，组合，代理）
行为（观察者，模版，策略）
然后再针对你熟悉的模式谈谈你的理解即可。

Java中的23种设计模式：
Factory（工厂模式），Builder（建造模式），Factory Method（工厂方法模式），
Prototype（原始模型模式），Singleton（单例模式），Facade（门面模式），
Adapter（适配器模式），Bridge（桥梁模式），Composite（合成模式），
Decorator（装饰模式），Flyweight（享元模式），Proxy（代理模式），
Command（命令模式），Interpreter（解释器模式），Visitor（访问者模式），
Iterator（迭代子模式），Mediator（调停者模式），Memento（备忘录模式），
Observer（观察者模式），State（状态模式），Strategy（策略模式），
Template Method（模板方法模式），Chain Of Responsibleity（责任链模式）
工厂模式：工厂模式是一种经常被使用到的模式，根据工厂模式实现的类可以根据提供的数据生成一组类中某一个类的实例，通常这一组类有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后需要定义一个工厂类，工厂类可以根据条件生成不同的子类实例。当得到子类的实例后，开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例。

113、开发中都用到了那些设计模式?用在什么场合?

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。

10. Linux

118、LINUX下线程，GDI类的解释。

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。

10. 问得稀里糊涂的题

65、四种会话跟踪技术

会话作用域ServletsJSP 页面描述
page否是代表与一个页面相关的对象和属性。一个页面由一个编译好的 Java servlet 类（可以带有任何的 include 指令，但是没有 include 动作）表示。这既包括 servlet 又包括被编译成 servlet 的 JSP 页面
request是是代表与 Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个 Web 组件（由于 forward 指令和 include 动作的关系）
session是是代表与用于某个 Web 客户机的一个用户体验相关的对象和属性。一个 Web 会话可以也经常会跨越多个客户机请求
application是是代表与整个 Web 应用程序相关的对象和属性。这实质上是跨越整个 Web 应用程序，包括多个页面、请求和会话的一个全局作用域

69、简述逻辑操作(&,|,^)与条件操作(&&,||)的区别。

区别主要答两点：a.条件操作只能操作布尔型的,而逻辑操作不仅可以操作布尔型,而且可以操作数值型
b.逻辑操作不会产生短路

10. 其他

1、请用英文简单介绍一下自己.

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

2、请把 http://tomcat.apache.org/ 首页的这一段话用中文翻译一下?

Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process.

Apache Tomcat is developed in an open and participatory environment and released under the Apache Software License. Apache Tomcat is intended to be a collaboration of the best-of-breed developers from around the world. We invite you to participate in this open development project. To learn more about getting involved, click here.

Apache Tomcat powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. Some of these users and their stories are listed on the PoweredBy wiki page.

3、美资软件公司JAVA工程师电话面试题目

1. Talk about overriding, overloading.
2. Talk about JAVA design patterns you known.
3. Talk about the difference between LinkList, ArrayList and Victor.
4. Talk about the difference between an Abstract class and an Interface.
5. Class a = new Class(); Class b = new Class();
 if(a == b) returns true or false, why?
6. Why we use StringBuffer when concatenating strings?
7. Try to explain Singleton to us? Is it thread safe? If no, how to make it thread safe?
8. Try to explain Ioc?
9. How to set many-to-many relationship in Hibernate?
10. Talk about the difference between INNER JOIN and LFET JOIN.
11. Why we use index in database? How many indexes is the maximum in one table as your suggestion?
12. When ‘Final’ is used in class, method and property, what dose it mean?
13. Do you have any experience on XML? Talk about any XML tool you used ,e.g. JAXB, JAXG.
14. Do you have any experience on Linux?
15. In OOD what is the reason when you create a Sequence diagram?