100IT 名企 java 面试必考面试题

据官方数据统计，在全球编程语言工程师的数量上,JavaEE 以 900 万的程序员数量位居首位，很多软件的开发都离不开它。

自从黑马程序员授课以来，深受社会上广大编程爱好的喜爱，我们的课程一直被其他机构争相模仿，我们坚信只有教育机构自身能力的不断提升，才能带动整体教育行业的前进。根据黑马程序员毕业学员面试经验，整理 100 家 IT 名（阿里、百度、腾讯、 ）企面试题，黑马程序员为了帮助更多爱好并想学习编程的同学，整理了这篇资源帖子，只为帮助更多的人受益。

更多（Java、android、大数据、python、前端、iOS、PHP ）开发、编程资源、源码、笔记加 QQ:208695827

一、Java 基础部分

1. JAVA 的基本数据类型有哪些 ？ String 是不是基本数据类型 ？

Java 有 8 种基本数据类型: byte int short long double float Boolean char
byte int short long 都属于整数类型.
Double float 属于浮点类型.
Boolean 为布尔类型
Char 为字符型
String 不是基本数据类型.它定义的为对象

2. 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。

3. Java 有没有 goto?

java 中的保留字，现在没有在 java 中使用。

4. 说说&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。If(x==33 & ++y>0) y 会增长，If(x==33 && ++y>0)不会增长
&还可以用作位运算符，当&操作符两边的表达式不是 boolean 类型时，&表示按位与操作，我们通常使用 0x0f 来与一个整数进行&运算，来获取该整数的最低 4 个 bit 位，例如，0x31 & 0x0f 的结果为 0x01。
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。

5. 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。

```
int arr[][] = {{1,2,3},{4,5,6,7},{9}};
boolean found = false;
for(int i=0;i<arr.length && !found;i++){
        for(int j=0;j<arr[i].length;j++){
```

```
                System.out.println("i=" + i + ",j=" + j);
                if(arr[i][j]    == 5) {
                    found = true;
                    break;
                }
            }
        }
```

6. switch 语句能否作用在 byte 上，能否作用在 long 上，能否作用在 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 switch 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。

7. 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。

8. 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。

9. 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。

10. 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：

（1）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

（2）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

（3）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。

11. 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");
```

12. "=="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（堆内存），变量也占用一块内存，例如 Objet obj = new Object();变量 obj 是一个内存，new Object()是另一个内存，此时，变量 obj 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。

13. 静态变量和实例变量的区别？

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。

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！

14. 是否可从一个 static 方法内发出对非 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。

15. 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。

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

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

16. 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。

17. 下面的代码有什么不妥之处?

```
1. if(username.equals(“zxx”){}
2.int x = 1;
return x==1?true:false;
```

18. 请说出作用域 public，private，protected，以及不写时的区别

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。

19. 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。如果父类的方法是 private 类型，那么，子类则不存在覆盖的限制，相当于子类中增加了一个全新的方法。

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

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

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

2、覆盖的方法的返回值必须和被覆盖的方法的返回一致；

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

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

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

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

2、不能通过访问权限、返回类型、抛出的异常进行重载；

3、方法的异常类型和数目不会对重载造成影响；

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

20. 同学贡献的一些题?

1.搞了多个重载方法，参数分别是 int ,char,和 double，然后将 double x = 2，传递进去，会选择哪个方法？

2.说说对 javaee 中的 session 的理解，你是怎么用 session 的？cvs/svn 下载

3.jdk 中哪些类是不能继承的：System,String,StringBuffer 等。

4.在 eclipse 中调试时，怎样查看一个变量的值。

5.判断身份证：要么是 15 位，要么是 18 位，最后一位可以为字母，并写程序提出其中的年月日。

6.一个房子里有椅子，椅子有腿和背，房子与椅子是什么关系，椅子与腿和背是什么关系？

7.如果房子有多个椅子，就是聚合关系，否则是一种关联关系，当然，聚合是一种特殊的关联。椅子与腿和背时组合关系。

8.说说 has a 与 is a 的区别。

9.工厂模式的类图

21. 线程如何同步和通讯？

同学回答说 synchronized 方法或代码块！面试官似乎不太满意！

只有多个 synchronized 代码块使用的是同一个监视器对象，这些 synchronized 代码块之间才具有线程互斥的效果，假如 a 代码块用 obj1 作为监视器对象，假如 b 代码块用 obj2 作为监视器对象，那么，两个并发的线程可以同时分别进入这两个代码块中。 …这里还可以分析一下同步的原理。

对于同步方法的分析，所用的同步监视器对象是 this

接着对于静态同步方法的分析，所用的同步监视器对象是该类的 Class 对象

接着对如何实现代码块与方法的同步进行分析。

22. ClassLoader 如何加载 class ？

jvm 里有多个类加载，每个类加载可以负责加载特定位置的类，例如，bootstrap 类加载负责加载 jre/lib/rt.jar 中的类， 我们平时用的 jdk 中的类都位于 rt.jar 中。extclassloader 负责加载 jar/lib/ext/*.jar 中的类，appclassloader 负责 classpath 指定的目录或 jar 中的类。除了 bootstrap 之外，其他的类加载器本身也都是 java 类，它们的父类是 ClassLoader。

23. Servlet 的生命周期?

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

24. 抽象类的作用？

25. ArrayList 如何实现插入的数据按自定义的方式有序存放？

```
class MyBean implements Comparable{
    public int compareTo(Object obj){
        if(! obj instanceof MyBean)
            throw new ClassCastException() //具体异常的名称，我要查 jdk 文档。
        MyBean other = (MyBean) obj;
        return age > other.age?1:age== other.age?0:-1;
    }
}
class MyTreeSet {
    private ArrayList    datas = new ArrayList();
    public void add(Object obj){
        for(int i=0;i<datas.size();i++){
            if(obj.compareTo(datas.get(i) != 1){
                datas.add(i,obj);
            }
        }
    }
}
```

26. 分层设计的好处？

把各个功能按调用流程进行了模块化，模块化带来的好处就是可以随意组合，举例说明：如果要注册一个用户，流程为显示界面并通过界面接收用户的输入，接着进行业务逻辑处理，在处理业务逻辑又访问数据库，如果我们将这些步骤全部按流水帐的方式放在一个方法中编写，这也是可以的，但这其中的坏处就是，当界面要修改时，由于代码全在一个方法内，可能会碰坏业务逻辑和数据库访问的码，同样，当修改业务逻辑或数据库访问的代码时，也会碰坏其他部分的代码。分层就是要把界面部分、业务逻辑部分、数据库访问部分的代码放在各自独立的方法或类中编写，这样就不会出现牵一发而动全身的问题了。这样分层后，还可以方便切换各层，譬如原来的界面是 Swing，现在要改成 BS 界面，如果最初是按分层设计的，这时候不需要涉及业务和数据访问的代码，只需编写一条 web 界面就可以了。

下面的仅供参考，不建议照搬照套，一定要改成自己的语言，发现内心的感受：
分层的好处：1.实现了软件之间的解耦；2.便于进行分工；3.便于维护；4.提高软件组件的重用；5.便于替换某种产品，比如持久层用的是 hibernate,需要更换产品用 toplink，就不用其他业务代码，直接把配置一改；6.便于产品功能的扩展；7.便于适用用户需求的不断变化

27. 序列化接口的 id 有什么用？

对象经常要通过 IO 进行传送，让你写程序传递对象，你会怎么做？把对象的状态数据用某种格式写入到硬盘，Person->“zxx,male,28,30000”→Person，既然大家都要这么干，并且没

有个统一的干法，于是，sun 公司就提出一种统一的解决方案，它会把对象变成某个格式进行输入和输出，这种格式对程序员来说是透明（transparent）的，但是，我们的某个类要想能被 sun 的这种方案处理，必须实现 Serializable 接口。

```
ObjectOutputStream.writeObject(obj);
Object obj = ObjectInputStream.readObject();
```

假设两年前我保存了某个类的一个对象，这两年来，我修改该类，删除了某个属性和增加了另外一个属性，两年后，我又去读取那个保存的对象，或有什么结果？未知！sun 的 jdk 就会蒙了。为此，一个解决办法就是在类中增加版本后，每一次类的属性修改，都应该把版本号升级一下，这样，在读取时，比较存储对象时的版本号与当前类的版本号，如果不一致，则直接报版本号不同的错!

28. StringBuffer 与 StringBuilder 的区别？

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

29. hashCode 方法的作用？

首先，想要明白 hashCode 的作用，你必须要先知道 Java 中的集合。

总的来说，Java 中的集合（Collection）有两类，一类是 List，再有一类是 Set。你知道它们的区别吗？前者集合内的元素是有序的，元素可以重复；后者元素无序，但元素不可重复。那么这里就有一个比较严重的问题了：要想保证元素不重复，可两个元素是否重复应该依据什么来判断呢？这就是 Object.equals 方法了。但是，如果每增加一个元素就检查一次，那么当元素很多时，后添加到集合中的元素比较的次数就非常多了。也就是说，如果集合中现在已经有 1000 个元素，那么第 1001 个元素加入集合时，它就要调用 1000 次 equals 方法。这显然会大大降低效率。于是，Java 采用了哈希表的原理。可以简单理解，hashCode 方法实际上返回的就是对象存储的物理地址（实际可能并不是）。这样一来，当集合要添加新的元素时，先调用这个元素的 hashCode 方法，就一下子能定位到它应该放置的物理位置上。如果这个位置上没有元素，它就可以直接存储在这个位置上，不用再进行任何比较了；如果这个位置上已经有元素了，就调用它的 equals 方法与新元素进行比较，相同的话就不存了，不相同就散列其它的地址。所以这里存在一个冲突解决的问题。这样一来实际调用 equals 方法的次数就大大降低了，几乎只需要一两次。

所以，Java 对于 eqauls 方法和 hashCode 方法是这样规定的：

1、如果两个对象相同，那么它们的 hashCode 值一定要相同；

2、如果两个对象的 hashCode 相同，它们并不一定相同

上面说的对象相同指的是用 eqauls 方法比较。

你当然可以不按要求去做了，但你会发现，相同的对象可以出现在 Set 集合中。同时，增加新元素的效率会大大下降。

30. webservice 问得很多

31. 设计出计算任意正整数的阶层？

32. 在 oracle 数据库中需要查询出前 8 条记录的 sql 语句怎么写？

33. 什么是 SOA，谈谈你的 SOA 的理解。service orientied architecture？

SOA 是指为了解决在 Internet 环境下业务集成的需要，通过连接能完成特定任务的独立功能

实体实现的一种软件系统架构。

1) 软件系统架构：SOA 不是一种语言，也不是一种具体的技术而是一种软件系统架构，它尝试给出在特定环境下推荐采用的一种架构，从这个角度上来说，它更像一种模式(Pattern)。因此它与很多已有的软件技术比如面向对象技术，是互补的而非互斥的。它们分别面向不同的应用场景，用来满足不同的特定需求。

2) SOA 的使用范围：需求决定同时也限制功能。SOA 并不是包治百病的万灵丹，它最主要的应用场合在于解决在 Internet 环境下的不同商业应用之间的业务集成问题。

34. 如何实现线程间的通讯？

在 Java 语言中，提供了各种各样的输入输出流（stream）,使我们能够很方便的对数据进行操作，其中，管道（pipe）流是一种特殊的流，用于在不同线程（threads）间直接传送数据。一个线程发送数据到输出管道，另一个线程从输入管道中读数据。通过使用管道，实现不同线程间的通讯。无需求助于类似临时文件之类的东西。本文在简要介绍管道的基本概念后，将以一个具体的实例 pipeapp 加以详细说明。

1．管道的创建与使用

Java 提供了两个特殊的专门的类专门用于处理管道，它们就是 pipedinputstream 类和 pipeoutputstream 类。

Pipedinputstream 代表了数据在管道中的输出端，也就是线程向管道读数据的一端；pipeoutputstream 代表了数据在管道中的输入端，也就是线程向管道写数据的一端，这两个类一起使用可以提供数据的管道流。

为了创建一个管道流，我们必须首先创建一个 pipeoutstream 对象，然后，创建 pipeinputstream 对象，实例如下：

pipeout= new pipedyoutstream();

pipein= new pipedputsteam(pipepout);

一旦创建了一个管道后，就可以象操作文件一样对管道进行数据的读写。

2．演示程序： pipeapp

应用程序由三个程序组成：主线程（pipeapp.Java）及由主线程启动的两个二级线程（ythread.Java 和 zthread.Java）,它们使用管道来处理数据。程序从一个内容为一行一行"x"字母的"input.txt"文件中读取数据，使用管道传输数据，第一次是利用线程 ythread 将数据"x"转换为"y"，最后利用线程 zthread 将"y"转换为"z",之后，程序在屏幕上显示修改后的数据。

主线程 （pipeapp.Java）

在 main()方法中，程序首先创建一个应用对象：pipeapp pipeapp=new pipeapp();

由于程序中流操作都需要使用 IOException 异常处理，所以设置了一个 try 块。在 try 中，为了从源文件中读取数据，程序为"input.txt"文件创建了一个输入流 Xfileln,:

fileinputstream xfileln= new fileinputstream("input.txt");

新的输入流传递给 changetoy()方法，让线程 ythread 能读取该文件：

inputstream ylnpipe =pipeapp.changetoy(xfileln);

changetoy()方法创建将输入数据"x"改变到"y"的线程 ythread,并返回该线程的输入管道：

inputstream zlnpipe = pipeapp.changetoz(ylnpipe);

changetoz()方法启动将数据从"y"改变到"z"的线程 zehread,主程序将使用从 changetoz()返回的输入管道。得到以修改的数据。

然后，程序将管道输入流定位到 datainputstream 对象，使程序能够使用 readline()方法读取数据：

datainputstream inputstream = new datainputstream(zlnpiepe);

创建了输入流以后，程序就可以以行一行的读取数据病显示在屏幕上。

String str= inputstream.readline();
While(str!=null){
system.out.println(str);
str=inputstream.readline();
}
显示完成之后，程序关闭输入流：
inputstream.close();
changetoy()方法
changetoy()方法首先通过传递一个参数 inputstream 给 datainputstream 对象来定位资源的输入流，使程序能使用 readline()方法从流中读取数据：
datainputstream xfileln =new datainutstream(inputstream)；
然后，changetoy()创建输出管道和输入管道：
pipeoutstream pipeout = new pipeoutputstream();
pipeinputstream pipeln = new pipedinputsteam(pipeout);
为了能够使用 println()方法输出修改的后的文本行到管道，程序将输出管道定位到 printstream 对象：
printstream printstream = new printstream(pipeout);
现在，程序可以创建将数据从 x 改变到 y 的线程，该线程是 ythread 类的一个对象，他传递两个参数：输入文件（xfileln）和输出管道（调用 printstream）
ythread ythread =new thread(xfileln,printstream);
之后，程序启动线程：
changetoz（）方法
changetoz()方法与 changetoy()方法很相似，他从 changetoy()返回的输入流开始：
datainputstream yfileln= new datainputstream(inputstream);
程序创建一个新的管道：
pipedoutstream pipeout2 = new pipedoutputstream();
pipedinputstream pipeln2 = new pipedinputsream(pipeout2);
该线程通过这个新的管道发出修改后的数据（输入流 pipeln2）给主程序。
源程序如下：

```
//
//pipeapp.Java-pipeapp 的主应用程序
//
import Java.io.*
class pipeapp{
public static void main(string[] args){
pipeapp pipeapp=new pipeapp();
try{
fileinputstream xfile =new fileinputstream("input.txt");
inputstream ylnpipe = pipeapp.changetoy(xfileln);
inputstream zlnpipe=pipeapp.changetoz(ylnpipe);
system.out.println();
system.out.println("here are the results");
system.out.pringln();
datainputstream inputstream = nes datainputstream(zlnpipe);
```

```
string str = inputstream.readline();
while (str!=null){
system.out.println(str);
str=inputstream.readline();
}
inputstream.close();
}
catch(exception e){
system.out.println(e.tostring());
}
}
public inputstream changetoy(inputstream inputstream){
try{
datainputstream pipeout = new datainputsteam(inputstream);
pipedoutstream pipeout = new pipedoutputstream();
pipedlnsteam pipeln = new pipedlnputstream(pipeout);
printstream printstream = new printstream(pipeout);
ythread ythread = new ythread(xfileln,printstream);
ythread.start();
return pipeln;
}
catch(exeption e){
system.out.println(x.tostring());
}
return null;
}
public inputstream changetoz(inputstream inputsteam){
try{
datainputstream yfileln = new datainputstream(inputstream);
pipeoutputstream pipeln2 = new pipedinputstream(pipeout2);
printrstream printstream2 = new printsteam(pipeout2);
zthread zthread = new zthread(yfileln,printstream2);
zthread.start();
return pipeln2;
}
catch(exception e){
system.out.println(e.tostring());
}
return null;
}
}
```

Ythread 类和 Zthread 类

由于 ythread 类与 zthread 类基本一样，在此仅以 ythread 为例加以说明。

Ythread 的构造器接收两个参数：输入的文件和第一个管道的输出端，构造器存储这两

个参数作为类的数据成员：

```
Ythread(datainputstream xfileln,pringstream printstream){
    this.xfileln = xfileln;
    this.printstream = printstream;
}
```

线程通过 run()方法来处理数据。首先读取一行数据，确保 xstring 不为空的情况下循环执行：

```
string xstring = xfileln.readline();
```

每读一行数据，完成一次转换

```
string ystring = xstring.replace('x','y');
```

然后将修改后的数据输出到管道的输出端：

```
prinstream.prinrln(ystring);
```

为了确保所有缓冲区的数据完全进入管道的输出端：

```
pringstram.flush();
```

循环完成后，线程关闭管道输出流：

```
pringstram.close();
```

ythread 类的源程序如下：

```
//
//ythread.Java
//
import Java.io.*;
class ythread exteads thread{
datainputstream xfileln;
pringstream printstream;
ythread(datainputstream xfileln,pringstream.printstream){
this.xfileln = xfileln;
this.printstream = printstream;
}
public void run(){
try{
string xstring = xfileln.readline();
while(xstring!=null){
string ystring= xstring.replace('x','y');
printstream.pringln(ystring);
printstream.flush();
xstring= xfileln.readline();
}
printstream.close();
}
catch{ioexception e}{
system.out.println(e.tostring());
}
}
}
```

35. 编程题:

编程 1.编写一个函数将一个十六进制数的字符串参数转换成整数返回。

```
String str = “13abf”;
int len = str.length;
int sum = 0;
for(int i=0;i<len;i++){
    char c = str.charAt(len-1-i);
    int n = Character.digit(c,16);
    sum += n * (1<<(4*i));
}
```

其实，也可以用 Integer.parseInt(str,16)，但面试官很可能是想考我们的编码基本功。

编程 2 :银行贷款的还款方式中最常用的是一种叫“等额本息”，还款法，即借款人在约定还款期限内的每一期（月）归还的金额（产生的利息+部分本金）都是相等的，现有一笔总额为 T 元的 N 年期住房贷款，年利率为 R，要求算出每一期的还款的本金和利息总额，请写出解决思路和任意一种编程语言实现的主要代码。

思路：既然是按月还款，那我就要将 N 年按月来计算，即要还 N*12 个月，这样就可以求出每月要还的本金。由于每月要还的那部分本金所欠的时间不同，所以，它们所产生的利息是不同的，该部分本金的利息为：部分本金额*所欠月数*月利率。应该是这么个算法，如果利息还计利息，如果月还款不按年利率来算，老百姓算不明白的。

```
int monthMoney = T/N/12;
float monthRate = R/12;
int totalMonth = N * 12;
float totalRate = 0;
for(int i=1;i<=totalMonth;i++){
    totalRate += monthMoney * monthRate * i;
}
int result = monthMoney + totalRate/N/12;
```

36. ****Spring 的 DI 是什么（注：除了 IOC，AOP 这些概念，还不太清楚 DI 的概念）

不管是面向对象，还是面向过程，都需要分成许多的块，然后由这些部件协同工作完成任务要协同工作就会产生依赖，一个方法调用另一个方法，一个对象包含另一个对象

如果对象 A 包含对象 B 的话，就需要在 A 里 new 一个 B

依赖注入从具体类 B 里抽象出接口 IB——IB 的具体实现可能有很多 B,B1,B2...很多种——这样 A 可以不用再 new 具体的 B 了，而是跟 IoC 容器说：我要一个 IB（getBean("IB")）。然后，由容器根据配置文件来做具体的 new 的工作。具体 new 的是哪个，由配置文件从代码外部决定，要更换成 B,B1,或是 B2...修改配置文件就能做到，不用再改代码了

37. *任意数字序列“123456”之类，输出它们所有的排列组合

38. *****什么是 AOP（注：会用，但感觉说不清楚）

AOP: 面向切面编程：Aspect Oriented Programming

AOP 是 OOP 的延续，是（Aspect Oriented Programming）的缩写，意思是面向切面编程。

主要的功能是：日志记录，性能统计，安全控制，事务处理，异常处理等等。

主要的意图是：将日志记录，性能统计，安全控制，事务处理，异常处理等代码从业务逻辑代码中划分出来，通过对这些行为的分离，我们希望可以将它们独立到非指导业务逻辑

的方法中，进而改变这些行为的时候不影响业务逻辑的代码。

可以通过预编译方式和运行期动态代理实现在不修改源代码的情况下给程序动态统一添加功能的一种技术。AOP 实际是 GoF 设计模式的延续，设计模式孜孜不倦追求的是调用者和被调用者之间的解耦，AOP 可以说也是这种目标的一种实现。

在 Spring 中提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务（例如审计（auditing）和事务（transaction）管理）进行内聚性的开发。应用对象只实现它们应该做的——完成业务逻辑——仅此而已。它们并不负责（甚至是意识）其它的系统级关注点，例如日志或事务支持。

应用举例：

假设有在一个应用系统中，有一个共享的数据必须被并发同时访问，首先，将这个数据封装在数据对象中，称为 Data Class，同时，将有多个访问类，专门用于在同一时刻访问这同一个数据对象。

为了完成上述并发访问同一资源的功能，需要引入锁 Lock 的概念，也就是说，某个时刻，当有一个访问类访问这个数据对象时，这个数据对象必须上锁 Locked，用完后就立即解锁 unLocked，再供其它访问类访问。

使用传统的编程习惯，我们会创建一个抽象类，所有的访问类继承这个抽象父类，如下：

```
abstract class Worker{
abstract void locked();
abstract void accessDataObject();
abstract void unlocked();
}
```

我注：由上面这些题，可以看出，思想很重要，只有琢磨思想和原理的人才能很好地回答这些问题！

2 题的答案：

```
String str = “xafdvs”;
char[] arr1 = str.toCharArray();
char[] arr2 = Arrays.copyOf(arr1,arr1.length);
for(int i=0;i<arr1.length-1;i++){
    for(int j = i+1;j<arr2.length;j++){
        syso: arr1[i] + “,” + arr2[j];
    }
}
```

3.题的答案：

1.概念介绍：所谓 AOP，即 Aspect orientied program,就是面向方面的编程，

2.解释什么是方面：贯穿到系统的各个模块中的系统一个功能就是一个方面，

比如，记录日志，统一异常处理，事务处理，全限检查，这些功能都是软件系统

的一个面，而不是一点，在各个模块中都要出现。

3.什么是面向方面编程：把系统的一个方面的功能封装成对象的形式来处理

4.怎么进行面向方面编程：把功能模块对应的对象作为切面嵌入到原来的各个系统模块中，

采用代理技术，代理会调用目标，同时把切面功能的代码（对象）加入进来，所以，

用 spring 配置代理对象时只要要配两个属性，分别表示目标和切面对象（Advisor）。

39. 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。

40. 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，抽象类是否可继承具体类。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。
只有记住抽象类与普通类的唯一区别就是不能创建实例对象和允许有 abstract 方法。

41. 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。

42. 谈谈你对面向对象的理解

面向对象是一种程序的设计方法，或者说它是一种程序设计范型，其基本思想是使用对象，类，继承，封装，消息等基本概念来进行程序设计。
它是从现实世界中客观存在的事物（即对象）出发来构造软件系统，并在系统构造中尽可能运用人类的自然思维方式，强调直接以问题域（现实世界）中的事物为中心来思考问题，认识问题，并根据这些事物的本质特点，把它们抽象地表示为系统中的对象，作为系统的基本构成单位（而不是用一些与现实世界中的事物相关比较远，并且没有对应关系的其它概念来构造系统）。这可以使系统直接地映射问题域，保持问题域中事物及其相互关系的本来面貌。
它可以有不同层次的理解：
从世界观的角度可以认为：面向对象的基本哲学是认为世界是由各种各样具有自己的运动规律和内部状态的对象所组成的；不同对象之间的相互作用和通讯构成了完整的现实世界。因此，人们应当按照现实世界这个本来面貌来理解世界，直接通过对象及其相互关系来反映世界。这样建立起来的系统才能符合现实世界的本来面目。
从方法学的角度可以认为：面向对象的方法是面向对象的世界观在开发方法中的直接运用。它强调系统的结构应该直接与现实世界的结构相对应，应该围绕现实世界中的对象来构造系统，而不是围绕功能来构造系统。
从程序设计的角度来看，面向对象的程序设计语言必须有描述对象及其相互之间关系的语言成分。这些程序设计语言可以归纳为以下几类：系统中一切皆为对象；对象是属性及其操作的封装体；对象可按其性质划分为类，对象成为类的实例；实例关系和继承关系是对象之间的静态关系；消息传递是对象之间动态联系的唯一形式，也是计算的唯一形式；方法是消息的序列。

43. 面向对象的特征有哪些方面？

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。把握一个原则：把对同一事物进行操作的方法和相关的方法放在同一个类中，把方法和它操作的数据放在同一个类中。

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。

2．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。例如，看到一只蚂蚁和大象，你能够想象出它们的相同之处，那就是抽象。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。

3．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。

4．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？

44. 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，而程序调用的方法在运行期才动态绑定，就是引用变量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。

45. 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(Concrete)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,虽然 eclipse 下不报错，但应该也不行），但接口抽象方法只能是 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 final 类型，并且默认即为 public static final 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，在各个子类中只是完成各自的业务逻辑代码，伪代码如下：

```
public abstract class BaseServlet extends HttpServlet{
        public final 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 只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。

46. 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 类要和硬件打交道，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;
```

如果我们要用 java 调用别人写的 c 语言函数，我们是无法直接调用的，我们需要按照 java 的要求写一个 c 语言的函数，又我们的这个 c 语言函数去调用别人的 c 语言函数。由于我们的 c 语言函数是按 java 的要求来写的，我们这个 c 语言函数就可以与 java 对接上，java 那边的对接方式就是定义出与我们这个 c 函数相对应的方法，java 中对应的方法不需要写具体的代码，但需要在前面声明 native。

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。而且，方法上的 synchronized 同步所使用的同步锁对象是 this，而抽象方法上无法确定 this 是什么。

47. 什么是内部类？Static Nested Class 和 Inner Class 的不同？

使用 static 声明的内部类就是外部类，可以通过外部类.内部类直接访问。
普通内部类是不能够直接被外部所访问的，需要通过外部类实例在找到内部类实例。
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（静态成员不是对象的特性，只是为了找一个容身之处，所以需要放到一个类中而已，这么一点小事，你还要把它放到类内部的一个类中，过分了啊！提供内部类，不是为让你干这种事情，无聊，不让你干。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 {
                public void method(){
                    out_x = 3;
                }
            }
            Inner2 inner2 = new Inner2();
        }
        public class Inner1 {
        }
}
```

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，但这种内部类的前面可以使用 final 或 abstract 修饰符。这种内部类对其他类是不可见的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。这种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 修饰符。
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 Runnable() {
                public void run() {
                };
            }).start();
        }
}
```

最后，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 Static Nested Class，它不再具有内部类的特性，所有，从狭义上讲，它不是内部类。Static Nested Class 与普通类在运行时的行为和功能上没有什么区别，只是在编程引用时的语法上有一些差别，它可以定义成 public、protected、默认的、private 等多种类型，而普通类只能定义成 public 和默认的这两种类型。在外面引用 Static Nested Class 类的名称为“外部类名.内部类名”。在外面不需要创建外部类的实例对象，就可以直接创建 Static Nested Class，例如，假设 Inner 是定义在 Outer 类中的 Static Nested Class，那么可以使用如下语句创建 Inner 类：

```
Outer.Inner inner = new Outer.Inner();
```

由于 static Nested Class 不依赖于外部类的实例对象，所以，static Nested Class 能访问外部类的非 static 成员变量。当在外部访问 Static Nested Class 时，可以直接使用 Static Nested Class 的名字，而不需要加上外部类的名字了，在 Static Nested Class 中也可以直接引用外部类的 static 的成员变量，不需要加上外部类的名字。

在静态方法中定义的内部类也是 Static Nested Class，这时候不能在类前面加 static 关键字，静态方法中的 Static Nested Class 与普通方法中的内部类的应用方式很相似，它除了可以直接访问外部类中的 static 的成员变量，还可以访问静态方法中的局部变量，但是，该局部变量前必须加 final 修饰符。

备注：首先根据你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。

48. 内部类可以引用它的包含类的成员吗？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！

如果你把静态嵌套类当作内部类的一种特例，那在这种情况下不可以访问外部类的普通成员变量，而只能访问外部类中的静态成员，例如，下面的代码：

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

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。

49. Anonymous Inner Class (匿名内部类) 是否可继承其它类，是否可以 implements 接

口?

可以继承其他类或实现其他接口。不仅是可以，而是必须!因为匿名内部类就是在抽象类和接口的基础上发展起来的。

50. class.forName 的作用？

调用该访问 返回一个以字符串指定类名的类的对象。
返回字节码，返回字节码的方式有几种:
①：这份字节码曾经被加载过已经存在 java 虚拟机中了直接返回。
②：java 虚拟机中还没有这份字节码 用类加载器去加载 把加载进来的字节码缓存在虚拟机中，以后再得到这个字节码就不用再加载。
得到字节码对应的实例对象: 类名.class ; 对象.getClass(),例如 new Date; 类名.class，例如，System.class
对象.getClass()，例如，new Date().getClass() Class.forName(“类名”)，例如，Class.forName(“java.util.Date”);
③：静态方法去查询或者加载这个字符串所对应哪个类的字节码
因为在写源程序的时候还不知道类的名字，在我运行的时候人家传递我一个字符串，这个字符串里面包含了一个类的名字，再写程序的时候把 java.util.Date 换成一个字符串变量， 等程序运行起来的时候这个变量的值从一个配置文件里面装再进来，这个类的名字在写原程序的时候不用知道而是等运行的时候给我临时送进来。
》》》》》》》》》》》》》》》》》》》》》》》》》》》《《《《《《《《《《《《《《《《《《《《《《《《《《《《《
按参数中指定的字符串形式的类名去搜索并加载相应的类，如果该类字节码已经被加载过，则返回代表该字节码的 Class 实例对象，否则，按类加载器的委托机制去搜索和加载该类，如果所有的类加载器都无法加载到该类，则抛出 ClassNotFoundException。加载完这个 Class 字节码后，接着就可以使用 Class 字节码的 newInstance 方法去创建该类的实例对象了。
有时候，我们程序中所有使用的具体类名在设计时（即开发时）无法确定，只有程序运行时才能确定，这时候就需要使用 Class.forName 去动态加载该类，这个类名通常是在配置文件中配置的，例如，spring 的 ioc 中每次依赖注入的具体类就是这样配置的，jdbc 的驱动类名通常也是通过配置文件来配置的，以便在产品交付使用后不用修改源程序就可以更换驱动类名

51. super.getClass()方法调用？

下面程序的输出结果是多少？

```java
import java.util.Date;
public  class Test extends Date{
    public static void main(String[] args) {
        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，在一个 qq 群有个网友正好问过这个问题，我觉得挺有趣，就研究了一下，没想到今天还被你面到了，哈哈。
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，其实就是在调用从父类继承的 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();

52. 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，因此不可继承这个类、不能修改这个类。为提高效率节省空间，我们应用 StringBuffer 类。

53. 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。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，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。

54. 是否可以继承 String 类?

String 类是 final 类故不可以继承.

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

两个或一个，”xyz”对应一个对象，这个对象放在字符串常量缓冲区，常量”xyz”不管出现多少遍，都是缓冲区中的那一个。New String 每写一遍，就创建一个新的对象，它一句那个常量”xyz”对象的内容来创建出一个新 String 对象。如果以前就用过’xyz’，这句代表就不会创建”xyz”自己了，直接从缓冲区拿。

56. 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;
}
```

在讲两者区别时，应把循环的次数搞成 10000，然后用 endTime-beginTime 来比较两者执行的时间差异，最后还要讲讲 StringBuilder 与 StringBuffer 的区别。

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

57. 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：用正则表达式，代码大概为：

```
String [] result = orgStr.split(“,”);
```

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();}
```

58. 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。

59. 下面这条语句一共创建了多少个对象：String s="a"+"b"+"c"+"d"？

答：对于如下代码：

```
String s1 = "a";
String s2 = s1 + "b";
String s3 = "a" + "b";
System.out.println(s2 == "ab");
System.out.println(s3 == "ab");
```

第一条语句打印的结果为 false，第二条语句打印的结果为 true，这说明 javac 编译可以对字符串常量直接相加的表达式进行优化，不必要等到运行期去进行加法运算处理，而是在编译时去掉其中的加号，直接将其编译成一个这些常量相连的结果。

题目中的第一行代码被编译器在编译时优化后，相当于直接定义了一个”abcd”的字符串，所以，上面的代码应该只创建了一个 String 对象。写如下两行代码，

```
String s = "a" + "b" + "c" + "d";
System.out.println(s == "abcd");
```

最终打印的结果应该为 true。

60. try {}里有一 return 语句，那紧跟在这个 try 后的 finally {}里的 code 是否被执行，什么时候被执行，在 return 前还是后?

也许你的答案是在 return 之前，但往更细地说，我的答案是在 return 中间执行，请看下面程序代码的运行结果：

```
public class Test {
    public static void main(String[] args) {
        System.out.println(new Test().test());
    }
    static int test(){
        int x = 1;
        try{
            return x;
        }
        finally{
            ++x;
            }
        }
}
```

运行结果是 1，为什么呢？主函数调用子函数并得到结果的过程，好比主函数准备一个空罐子，当子函数要返回结果时，先把结果放在罐子里，然后再将程序逻辑返回到主函数。所谓返回，就是子函数说，我不运行了，你主函数继续运行吧，这没什么结果可言，结果是在说这话之前放进罐子里的。

61. 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 语句后才真正返回。
在讲解答案时可以用下面的程序来帮助分析：

```
public    class Test {
        public static void main(String[] args) {
            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

62. 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，是异常的统一出口,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.

63. 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。

64. 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。
Error 表示有 JVM 进行处理的,是 JVM 出错.
Exctption 是可以用程序进行处理的,使用 try…catch 进行处理.

65. Java 中的异常处理机制的简单原理和应用。

异常是指 java 程序运行时（非编译）所发生的非正常情况或错误，与现实生活中的事件很相似，现实生活中的事件可以包含事件发生的时间、地点、人物、情节等信息，可以用一个对象来表示，Java 使用面向对象的方式来处理异常，它把程序中发生的每个异常也都分别封装到一个对象来表示的，该对象中包含有异常的信息。
Java 对异常进行了分类，不同类型的异常分别用不同的 Java 类表示，所有异常的根类为 java.lang.Throwable，Throwable 下面又派生了两个子类：Error 和 Exception，Error 表示应用程序本身无法克服和恢复的一种严重问题，程序只有死的份了，例如，说内存溢出和线程死锁等系统问题。Exception 表示程序还能够克服和恢复的问题，其中又分为系统异常和普通异常，系统异常是软件本身缺陷所导致的问题，也就是软件开发人员考虑不周所导致的问题，软件使用者无法克服和恢复这种问题，但在这种问题下还可以让软件系统继续运行或者让软件死掉，例如，数组脚本越界（ArrayIndexOutOfBoundsException），空指针异常（NullPointerException）、类转换异常（ClassCastException）；普通异常是运行环境的变化或异常所导致的问题，是用户能够克服的问题，例如，网络断线，硬盘空间不够，发生这样的异常后，程序不应该死掉。
java 为系统异常和普通异常提供了不同的解决方案，编译器强制普通异常必须 try..catch 处理或用 throws 声明继续抛给上层调用方法处理，所以普通异常也称为 checked 异常，而系统异常可以处理也可以不处理，所以，编译器不强制用 try..catch 处理或用 throws 声明，所以系统异常也称为 unchecked 异常。

提示答题者：就按照三个级别去思考：虚拟机必须宕机的错误，程序可以死掉也可以不死掉的错误，程序不应该死掉的错误；

66. 请写出你最常见到的 5 个 runtime exception。

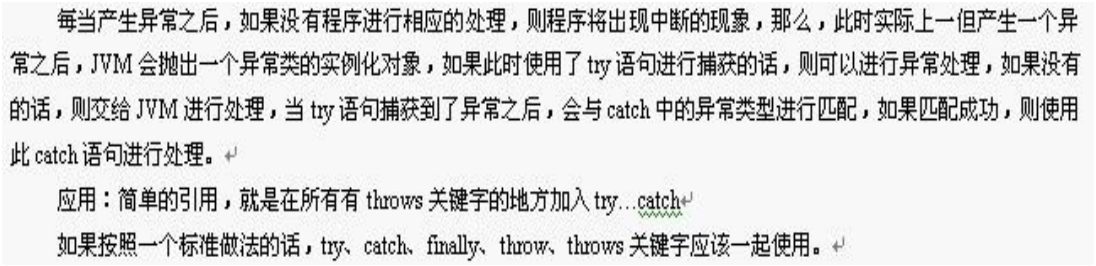
这道题主要考你的代码量到底多大，如果你长期写代码的，应该经常都看到过一些系统方面的异常，你不一定真要回答出 5 个具体的系统异常，但你要能够说出什么是系统异常，以及几个系统异常就可以了，当然，这些异常完全用其英文名称来写是最好的，如果实在写不出，那就用中文吧，有总比没有强！
所谓系统异常，就是.....，它们都是 RuntimeException 的子类，在 jdk doc 中查 RuntimeException 类，就可以看到其所有的子类列表，也就是看到了所有的系统异常。我比较有印象的系统异常有：NullPointerException、ArrayIndexOutOfBoundsException、ClassCastException。

67. java 如何进行异常处理，throws,throw,try,catch,finally 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，并提供了良好的接口。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。当一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，调用这个对象的方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；

try 用来指定一块预防所有“异常”的程序；
catch 子句紧跟在 try 块后面，用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。


每当产生异常之后，如果没有程序进行相应的处理，则程序将出现中断的现象，那么，此时实际上一但产生一个异常之后，JVM 会抛出一个异常类的实例化对象，如果此时使用了 try 语句进行捕获的话，则可以进行异常处理，如果没有的话，则交给 JVM 进行处理，当 try 语句捕获到了异常之后，会与 catch 中的异常类型进行匹配，如果匹配成功，则使用此 catch 语句进行处理。
应用：简单的引用，就是在所有有 throws 关键字的地方加入 try...catch
如果按照一个标准做法的话，try、catch、finally、throw、throws 关键字应该一起使用。

68. 在 java 中如何进行 socket 编程。

答：Sockets 有两种主要的操作方式:面向连接的和无连接的。
无连接的操作使用数据报协议.这个模式下的 socket 不需要连接一个目的的 socket,它只是简单地投出数据报.无连接的操作是快速的和高效的,但是数据安全性不佳.面向连接的操作使用 TCP 协议.一个这个模式下的 socket 必须在发送数据之前与目的地的 socket 取得一个连接.一旦连接建立了,sockets 就可以使用一个流接口:打开-读-写-关闭.所有的发送的信息都会在

另一端以同样的顺序被接收.面向连接的操作比无连接的操作效率更低,但是数据的安全性更高.

在服务器，使用 ServerSocket 监听指定的端口，端口可以随意指定（由于 1024 以下的端口通常属于保留端口，在一些操作系统中不可以随意使用，所以建议使用大于 1024 的端口），等待客户连接请求，客户连接后，会话产生；在完成会话后，关闭连接。在客户端，使用 Socket 对网络上某一个服务器的某一个端口发出连接请求，一旦连接成功，打开会话；会话完成后，关闭 Socket。客户端不需要指定打开的端口，通常临时的、动态的分配一个 1024 以上的端口。

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

java5 以前，有如下两种：

第一种：

new Thread(){}.start();这表示调用 Thread 子类对象的 run 方法，new Thread(){}表示一个 Thread 的匿名子类的实例对象，子类加上 run 方法后的代码如下：

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

第二种：

new Thread(new Runnable(){}).start();这表示调用 Thread 对象接受的 Runnable 对象的 run 方法，new Runnable(){}表示一个 Runnable 的匿名子类的实例对象,runnable 的子类加上 run 方法后的代码如下：

```
new Thread(new Runnable(){
                public void run(){
                }
        }
    ).start();
```

从 java5 开始，还有如下一些线程池创建多线程的方式：

```
ExecutorService pool = Executors.newFixedThreadPool(3)
for(int i=0;i<10;i++){
  pool.execute(new Runable(){public void run(){}});
}
Executors.newCachedThreadPool().execute(new Runable(){public void run(){}});
Executors.newSingleThreadExecutor().execute(new Runable(){public void run(){}});
```

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。

70. sleep() 和 wait() 有什么区别?

sleep：Thread 类中定义的方法，表示线程休眠，会自动唤醒；
wait：Object 中定义的方法，需要手工调用 notify()或者 notifyAll()方法。

（网上的答案：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。）

sleep 就是正在执行的线程主动让出 cpu，cpu 去执行其他线程，在 sleep 指定的时间过后，cpu 才会回到这个线程上继续往下执行，如果当前线程进入了同步锁，sleep 方法并不会释放锁，即使当前线程使用 sleep 方法让出了 cpu，但其他被同步锁挡住了的线程也无法得到执行。wait 是指在一个已经进入了同步锁的线程内，让自己暂时让出同步锁，以便其他正在等待此锁的线程可以得到同步锁并运行，只有其他线程调用了 notify 方法（notify 并不释放锁，只是告诉调用过 wait 方法的线程可以去参与获得锁的竞争了，但不是马上得到锁，因为锁还在别人手里，别人还没释放。如果 notify 方法后面的代码还有很多，需要这些代码执行完后才会释放锁，可以在 notfiy 方法后增加一个等待和一些代码，看看效果），调用 wait 方法的线程就会解除 wait 状态和程序可以再次得到锁后继续向下运行。对于 wait 的讲解一定要配合例子代码来说明，才显得自己真明白。

```
package com.huawei.interview;

public class MultiThread {
    public static void main(String[] args) {
        new Thread(new Thread1()).start();
        try {
            Thread.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        new Thread(new Thread2()).start();
    }

    private static class Thread1 implements Runnable{
        public void run() {
//由于这里的 Thread1 和下面的 Thread2 内部 run 方法要用同一对象作为监视器，我们这里不能用 this，因为在 Thread2 里面的 this 和这个 Thread1 的 this 不是同一个对象。我们用 MultiThread.class 这个字节码对象，当前虚拟机里引用这个变量时，指向的都是同一个对象。
            synchronized (MultiThread.class) {
                System.out.println("enter thread1...");
```

```
                System.out.println("thread1 is waiting");
                try {
            //释放锁有两种方式，第一种方式是程序自然离开监视器的范围，也就是离开了 synchronized 关键字管辖的代码范围，另一种方式就是在 synchronized 关键字管辖的代码内部调用监视器对象的 wait 方法。这里，使用 wait 方法释放锁。
                    MultiThread.class.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                System.out.println("thread1 is going on...");
                System.out.println("thread1 is being over!");
            }
        }
    }

    private static class Thread2 implements Runnable{
        public void run() {
            synchronized (MultiThread.class) {
                System.out.println("enter thread2...");
                System.out.println("thread2 notify other thread can release wait status..");
//由于 notify 方法并不释放锁，即使 thread2 调用下面的 sleep 方法休息了 10 毫秒，但 thread1 仍然不会执行，因为 thread2 没有释放锁，所以 Thread1 无法得不到锁。

                MultiThread.class.notify();
                System.out.println("thread2 is sleeping ten millisecond...");
                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                System.out.println("thread2 is going on...");
                System.out.println("thread2 is being over!");
            }
        }
    }
}
```

71. 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。

72. 下面两个方法同步吗？（自己发明）

```
class Test{
```

```
synchronized static void sayHello3()     {
        }
        synchronized void getX(){}
}
```

73. 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。

74. 启动一个线程是用 run()还是 start()? .

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

75. 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。
4.如果其他方法是 static，它用的同步锁是当前类的字节码，与非静态的方法不能同步，因为非静态的方法用的是 this。

76. 线程的基本概念、线程的基本状态以及状态之间的关系

一个程序中可以有多条执行线索同时执行，一个线程就是程序中的一条执行线索，每个线程上都关联有要执行的代码，即可以有多段程序代码同时运行，每个程序至少都有一个线程，即 main 方法执行的那个线程。如果只是一个 cpu，它怎么能够同时执行多段程序呢？这是从宏观上来看的，cpu 一会执行 a 线索，一会执行 b 线索，切换时间很快，给人的感觉是 a,b 在同时执行，好比大家在同一个办公室上网，只有一条链接到外部网线，其实，这条网线一会为 a 传数据，一会为 b 传数据，由于切换时间很短暂，所以，大家感觉都在同时上网。

状态：就绪，运行，synchronize 阻塞，wait 和 sleep 挂起，结束。wait 必须在 synchronized 内部调用。

调用线程的 start 方法后线程进入就绪状态，线程调度系统将就绪状态的线程转为运行状态，遇到 synchronized 语句时，由运行状态转为阻塞，当 synchronized 获得锁后，由阻塞转为运行，在这种情况可以调用 wait 方法转为挂起状态，当线程关联的代码执行完后，线程变为结束状态。

77. 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。Lock 还有更强大的功能，例如，它的 tryLock 方法可以非阻塞方式去拿锁。
举例说明（对下面的题用 lock 进行了改写）：

```
package com.huawei.interview;

import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class ThreadTest {
    private int j;
    private Lock lock = new ReentrantLock();
    public static void main(String[] args) {
        ThreadTest tt = new ThreadTest();
        for(int i=0;i<2;i++){
            new Thread(tt.new Adder()).start();
            new Thread(tt.new Subtractor()).start();
        }
    }
    private class Subtractor implements Runnable{
        public void run() {
            while(true){
                /*synchronized (ThreadTest.this) {
                    System.out.println("j--=" + j--);
                    //这里抛异常了，锁能释放吗？
                }*/
                lock.lock();
                try{
                    System.out.println("j--=" + j--);
                }finally{
                    lock.unlock();
                }
            }
        }
    }

    private class Adder implements Runnable{
        public void run() {
            while(true){
                /*synchronized (ThreadTest.this) {
                System.out.println("j++=" + j++);
                }*/
                lock.lock();
```

```
                try{
                    System.out.println("j++=" + j++);
                }finally{
                    lock.unlock();
                }
            }
        }
    }
}
```

78. 设计 4 个线程，其中两线程每次对 j 增加 1，另外两线程对 j 每次减少 1。

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();
                    }
                }
            }
}
---------随手再写的一个-------------
class A{
JManger j =new JManager();
main(){
    new A().call();
}

void call{
    for(int i=0;i<2;i++){
        new Thread(
            new Runnable(){ public void run(){while(true){j.accumulate()}}}
        ).start();
        new Thread(new Runnable(){ public void run(){while(true){j.sub()}}}).start();
        }
}
}

class JManager{
    private j = 0;

    public synchronized void subtract(){
        j--
    }

    public synchronized void accumulate(){
        j++;
    }
}
```

79. 子线程循环 10 次，接着主线程循环 100，接着又回到子线程循环 10 次，接着再回到主线程又循环 100，如此循环 50 次，请写出程序。

最终的程序代码如下：

```
public class ThreadTest {
    public static void main(String[] args) {
        new ThreadTest().init();
    }
    public void init(){
```

```
        final Business business = new Business();
        new Thread(
                new Runnable(){
                    public void run() {
                        for(int i=0;i<50;i++){
                            business.SubThread(i);
                        }
                    }
                }

        ).start();

        for(int i=0;i<50;i++){
            business.MainThread(i);
        }
    }

    private class Business{
        boolean bShouldSub = true;//这里相当于定义了控制该谁执行的一个信号灯
        public synchronized void MainThread(int i){
            if(bShouldSub)
                try {
                    this.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

            for(int j=0;j<5;j++){
                System.out.println(Thread.currentThread().getName() + ":i=" + i +",j=" + j);
            }
            bShouldSub = true;
            this.notify();
        }

        public synchronized void SubThread(int i){
            if(!bShouldSub)
                try {
                    this.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

            for(int j=0;j<10;j++){
                System.out.println(Thread.currentThread().getName() + ":i=" + i +",j=" + j);
```

```
                }
                bShouldSub = false;
                this.notify();
            }
        }
    }
```

备注：不可能一上来就写出上面的完整代码，最初写出来的代码如下，问题在于两个线程的代码要参照同一个变量，即这两个线程的代码要共享数据，所以，把这两个线程的执行代码搬到同一个类中去：

```
package com.huawei.interview.lym;

public class ThreadTest {

    private static boolean bShouldMain = false;

    public static void main(String[] args) {
        /*new Thread(){
        public void run(){
            for(int i=0;i<50;i++){
                for(int j=0;j<10;j++){
                    System.out.println("i=" + i + ",j=" + j);
                }
            }
        }

    }.start();*/

        //final String str = new String("");
        new Thread(
                new Runnable(){
                    public void run(){
                        for(int i=0;i<50;i++){
                            synchronized (ThreadTest.class) {
                                if(bShouldMain){
                                    try {
                                        ThreadTest.class.wait();}
                                    catch (InterruptedException e) {
                                        e.printStackTrace();
                                    }
                                }
                                for(int j=0;j<10;j++){

System.out.println(Thread.currentThread().getName() + "i=" + i + ",j=" + j);
```

```
                                        }
                                        bShouldMain = true;
                                        ThreadTest.class.notify();
                                    }
                                }
                            }
                        }
                ).start();

                for(int i=0;i<50;i++){
                    synchronized (ThreadTest.class) {
                        if(!bShouldMain){
                            try {
                                ThreadTest.class.wait();}
                            catch (InterruptedException e) {
                                e.printStackTrace();
                            }
                        }
                        for(int j=0;j<5;j++){
                            System.out.println(
                                    Thread.currentThread().getName() +
                                    "i=" + i + ",j=" + j);
                        }
                        bShouldMain = false;
                        ThreadTest.class.notify();
                    }
                }
            }
}
```

下面使用 jdk5 中的并发库来实现的：

```
import java.util.concurrent.Executors;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.concurrent.locks.Condition;

public class ThreadTest{
    private static Lock lock = new ReentrantLock();
    private static Condition subThreadCondition = lock.newCondition();
    private static boolean bBhouldSubThread = false;
    public static void main(String [] args){
        ExecutorService threadPool = Executors.newFixedThreadPool(3);
        threadPool.execute(new Runnable(){
            public void run(){
```

```
                    for(int i=0;i<50;i++){
                        lock.lock();
                        try{
                            if(!bBhouldSubThread)
                                subThreadCondition.await();
                            for(int j=0;j<10;j++){
                                System.out.println(Thread.currentThread().getName() + ",j=" +
j);
                            }
                            bBhouldSubThread = false;
                            subThreadCondition.signal();
                        }catch(Exception e){
                        }
                        finally{
                            lock.unlock();
                        }
                    }
                }
            });
            threadPool.shutdown();
            for(int i=0;i<50;i++){
                    lock.lock();
                    try{
                        if(bBhouldSubThread)
                                subThreadCondition.await();

                        for(int j=0;j<10;j++){
                        System.out.println(Thread.currentThread().getName() +",j=" + j);
                        }
                        bBhouldSubThread = true;
                        subThreadCondition.signal();
                    }catch(Exception e){
                    }
                    finally{
                        lock.unlock();
                }
            }
        }
}
```

80. 介绍 Collection 框架的结构

答：随意发挥题，天南海北谁便谈，只要让别觉得你知识渊博，理解透彻即可。

```
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

81. Collection 框架中实现比较要实现什么接口

comparable/comparator

82. ArrayList 和 Vector 的区别

这两个类都实现了 List 接口（List 接口继承了 Collection 接口），他们都是有序集合，即存储在这两个集合中的元素的位置都是有顺序的，相当于一种动态的数组，我们以后可以按位置索引号取出某个元素，并且其中的数据是允许重复的，这是 HashSet 之类的集合的最大不同处，HashSet 之类的集合不可以按索引号去检索其中的元素，也不允许有重复的元素（本来题目问的与 hashset 没有任何关系，但为了说清楚 ArrayList 与 Vector 的功能，我们使用对比方式，更有利于说明问题）。

接着才说 ArrayList 与 Vector 的区别，这主要包括两个方面：.

（1）同步性：

Vector 是线程安全的，也就是说是它的方法之间是线程同步的，而 ArrayList 是线程序不安全的，它的方法之间是线程不同步的。如果只有一个线程会访问到集合，那最好是使用 ArrayList，因为它不考虑线程安全，效率会高些；如果有多个线程会访问到集合，那最好是使用 Vector，因为不需要我们自己再去考虑和编写线程安全的代码。

(备注：对于 Vector&ArrayList、Hashtable&HashMap，要记住线程安全问题，记住 Vector 与 Hashtable 是旧的，是 java 一诞生就提供的，它们是线程安全的，ArrayList 与 HashMap 是 java2 时才提供的，它们是线程不安全的。所以，我们讲课时先讲老的。)

（2）数据增长：

ArrayList 与 Vector 都有一个初始的容量大小，当存储进它们里面的元素的个数超过了容量时，就需要增加 ArrayList 与 Vector 的存储空间，每次要增加存储空间时，不是只增加一个存储单元，而是增加多个存储单元，每次增加的存储单元的个数在内存空间利用与程序效率之间要取得一定的平衡。Vector 默认增长为原来两倍，而 ArrayList 的增长策略在文档中没有明确规定（从源代码看到的是增长为原来的 1.5 倍）。ArrayList 与 Vector 都可以设置初始的空间大小，Vector 还可以设置增长的空间大小，而 ArrayList 没有提供设置增长空间的方法。

总结：即 Vector 增长原来的一倍，ArrayList 增加原来的 0.5 倍。

83. HashMap 和 Hashtable 的区别。

HashMap：JDK1.2 之后推出，是新的类。采用异步处理方式，性能较高，但是属于非线程安全。允许设置 null。

Hashtable:JDK1.0 时推出，是旧的类。采用同步处理方式，性能较低，但是属于非线程安全。允许设置 null。

（条理上还需要整理，也是先说相同点，再说不同点）
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，在只有一个线程访问的情况下，效率要高于 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 。

84. List 和 Map 区别?

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

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

List，Set 是，Map 不是

86. List、Map、Set 三个接口，存取元素时，各有什么特点?

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

首先，List 与 Set 具有相似性，它们都是单列元素的集合，所以，它们有一个功共同的父接口，叫 Collection。Set 里面不允许有重复的元素，所谓重复，即不能有两个相等（注意，不是仅仅是相同）的对象 ，即假设 Set 集合中有了一个 A 对象，现在我要向 Set 集合再存入一个 B 对象，但 B 对象与 A 对象 equals 相等，则 B 对象存储不进去，所以，Set 集合的 add 方法有一个 boolean 的返回值，当集合中没有某个元素，此时 add 方法可成功加入该元素时，则返回 true，当集合含有与某个元素 equals 相等的元素时，此时 add 方法无法加入该元素，返回结果为 false。Set 取元素时，没法说取第几个，只能以 Iterator 接口取得所有的元素，再逐一遍历各个元素。
List 表示有先后顺序的集合， 注意，不是那种按年龄、按大小、按价格之类的排序。当我们多次调用 add(Obj e)方法时，每次加入的对象就像火车站买票有排队顺序一样，按先来后

到的顺序排序。有时候，也可以插队，即调用 add(int index,Obj e)方法，就可以指定当前对象在集合中的存放位置。一个对象可以被反复存储进 List 中，每调用一次 add 方法，这个对象就被插入进集合中一次，其实，并不是把这个对象本身存储进了集合中，而是在集合中用一个索引变量指向这个对象，当这个对象被 add 多次时，即相当于集合中有多个索引指向了这个对象，如图 x 所示。List 除了可以以 Iterator 接口取得所有的元素，再逐一遍历各个元素之外，还可以调用 get(index i)来明确说明取第几个。

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

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

HashSet 按照 hashcode 值的某种运算方式进行存储，而不是直接按 hashCode 值的大小进行存储。例如，"abc" ---> 78，"def" ---> 62，"xyz" ---> 65 在 hashSet 中的存储顺序不是 62,65,78，LinkedHashSet 按插入的顺序存储，那被存储对象的 hashcode 方法还有什么作用呢？我们想想!hashset 集合比较两个对象是否相等，首先看 hashcode 方法是否相等，然后看 equals 方法是否相等。new 两个 Student 插入到 HashSet 中，看 HashSet 的 size，实现 hashcode 和 equals 方法后再看 size。

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

87. 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。

LinkedList 也是线程不安全的，LinkedList 提供了一些方法，使得 LinkedList 可以被当作堆栈和队列来使用。

88. 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);
89.     if(!newVector.contains(obj);
            newVector.add(obj);
}
```

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

90. 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。

91. 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，元素重复与否是使用 equals()方法进行判断的。
equals()和==方法决定引用值是否指向同一对象 equals()在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值。

92. 链表和数组的区别？

创建数组时，必须明确说明数组的长度,(即数组中元素的个数),以便在内存中留出一块空间存放所有的数组元素,数组中各数据元素在内存中是顺序存放的。
创建链表时，不需要给出链表中元素(称为节点)的个数,可以先只创建一个链表头,其他元素在需要时动态地创建并加入到链表,链表的数据无素在内存中不是连续存放的。

93. 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。

ArrayList/Vector→List
→Collection
HashSet/TreeSet→Set
Propeties→HashTable
→Map
Treemap/HashMap

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

94. 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。

95. TreeSet 里面放对象，如同时放入父类和子类实例对象，比较时使用的是父类的 compareTo 方法，还是子类的 compareTo 方法，还是抛异常！

（应该是没有针对问题的确切的答案，当前的 add 方法放入的是哪个对象，就调用哪个对象的 compareTo 方法，至于这个 compareTo 方法怎么做，就看当前这个对象的类中是如何编写这个方法的）

实验代码：

```
public class Parent implements Comparable {
    private int age = 0;
    public Parent(int age){
        this.age = age;
    }
    public int compareTo(Object o) {
        System.out.println("method of parent");
        Parent o1 = (Parent)o;
        return age>o1.age?1:age<o1.age?-1:0;
    }
}
public class Child extends Parent {
    public Child(){
        super(3);
    }
    public int compareTo(Object o) {
            System.out.println("method of child");
            Child o1 = (Child)o;
            return 1;
    }
}
public class TreeSetTest {
    public static void main(String[] args) {
        TreeSet set = new TreeSet();
        set.add(new Parent(3));
        set.add(new Child());
        set.add(new Parent(4));
        System.out.println(set.size());
    }
}
```

96. 说出一些常用的类，包，接口，请各举 5 个

要让人家感觉你对 java ee 开发很熟，所以，不能仅仅只列 core java 中的那些东西，要多列你在做 ssh 项目中涉及的那些东西。就写你最近写的那些程序中涉及的那些类。

常用的类：BufferedReader BufferedWriter FileReader FileWirter String Integer java.util.Date，System，Class，List,HashMap

常用的包：java.lang java.io java.util java.sql ,javax.servlet,org.apache.strtuts.action,org.hibernate
常用的接口：Remote List Map Document NodeList ,Servlet,HttpServletRequest,HttpServletResponse,Transaction(Hibernate)、Session(Hibernate),HttpSession

97. 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。

98. 字节流与字符流的区别

要把一片二进制数据数据逐一输出到某个设备中，或者从某个设备中逐一读取一片二进制数据，不管输入输出设备是什么，我们要用统一的方式来完成这些操作，用一种抽象的方式进行描述，这个抽象描述方式起名为 IO 流，对应的抽象类为 OutputStream 和 InputStream ，不同的实现类就代表不同的输入和输出设备，它们都是针对字节进行操作的。
在实际应用中，经常需要把完全是字符的一段文本输出去或读进来，用字节流可以吗？计算机中的一切最终都是二进制的字节形式存在。对于“中国”这些字符，首先要得到其对应的字节，然后将字节写入到输出流。读取时，首先读到的是字节，可是我们要把它显示为字符，我们需要将字节转换成字符。由于这样的需求很广泛，人家专门提供了字符流的包装类。
底层设备永远只接受字节数据，有时候要写字符串到底层设备，需要将字符串转成字节再进行写入。字符流是字节流的包装，字符流则是直接接受字符串，它内部将串转成字节，再写入底层设备，这为我们向 IO 设别写入或读取字符串提供了一点点方便。
字符向字节转换时，要注意编码的问题，因为字符串转成字节数组，其实是转成该字符的某种编码的字节形式，读取也是反之的道理。

讲解字节流与字符流关系的代码案例：

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.io.PrintWriter;

public class IOTest {
	public static void main(String[] args) throws Exception {
		String str = "中国人";
		/*FileOutputStream fos	= new FileOutputStream("1.txt");

		fos.write(str.getBytes("UTF-8"));
		fos.close();*/

		/*FileWriter fw = new FileWriter("1.txt");
```

```
            fw.write(str);
            fw.close();*/
            PrintWriter pw = new PrintWriter("1.txt","utf-8");
            pw.write(str);
            pw.close();

            /*FileReader fr = new FileReader("1.txt");
            char[] buf = new char[1024];
            int len = fr.read(buf);
            String myStr = new String(buf,0,len);
            System.out.println(myStr);*/
            /*FileInputStream fr = new FileInputStream("1.txt");
            byte[] buf = new byte[1024];
            int len = fr.read(buf);
            String myStr = new String(buf,0,len,"UTF-8");
            System.out.println(myStr);*/
            BufferedReader br = new BufferedReader(
                    new InputStreamReader(
                            new FileInputStream("1.txt"),"UTF-8"
                            )
                    );
            String myStr = br.readLine();
            br.close();
            System.out.println(myStr);
    }
}
```

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

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

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

100. 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。

101. heap 和 stack 有什么区别。

java 的内存分为两类，一类是栈内存，一类是堆内存。栈内存是指程序进入一个方法时，会为这个方法单独分配一块私属存储空间，用于存储这个方法内部的局部变量，当这个方法结束时，分配给这个方法的栈会释放，这个栈中的变量也将随之释放。
堆是与栈作用不同的内存，一般用于存放不放在当前方法栈中的那些数据，例如，使用 new 创建的对象都放在堆里，所以，它不会随方法的结束而消失。方法中的局部变量使用 final 修饰后，放在堆中，而不是栈中。

102. GC 是什么？为什么要有 GC?

GC：垃圾回收，使用 GC 可以进行垃圾空间释放操作。
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。

103. 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。
简单答法:将无用的空间对象进行释放.
俩中回收机制:自动回收和手工调用 System.gc()方法,实际上调用 System.gc 就相当于调用了 Runtime.getRuntime().gc()方法.

104. 垃圾回收器的基本原理是？垃圾回收器可以马上回收内存吗？如何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。

105. 什么时候用 assert。

JDK1.4 之后增加的新关键字——assert，表示断言，即程序执行到某个地方之后值肯定是预计好的；一般开发中很少使用 assert；要想使用断言，就必须使用-ea 参数。
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，assert 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。

```
package com.huawei.interview;
```

```
public class AssertTest {
    public static void main(String[] args) {
        int i = 0;
        for(i=0;i<5;i++){
            System.out.println(i);
        }
        //假设程序不小心多了一句--i;
        --i;
        assert i==5;
    }
}
```

106. java 中会存在内存泄漏吗，请简单描述。

所谓内存泄露就是指一个不再被程序使用的对象或变量一直被占据在内存中。java 中有垃圾回收机制，它可以保证一对象不再被引用的时候，即对象编程了孤儿的时候，对象将自动被垃圾回收器从内存中清除掉。由于 Java 使用有向图的方式进行垃圾回收管理，可以消除引用循环的问题，例如有两个对象，相互引用，只要它们和根进程不可达的，那么 GC 也是可以回收它们的，例如下面的代码可以看到这种情况的内存回收：

```
package com.huawei.interview;

import java.io.IOException;

public class GarbageTest {
    public static void main(String[] args) throws IOException {
        try {
            gcTest();
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("has exited gcTest!");
        System.in.read();
        System.in.read();
        System.out.println("out begin gc!");
        for(int i=0;i<100;i++){
            System.gc();
            System.in.read();
            System.in.read();
        }
    }

    private static void gcTest() throws IOException {
        System.in.read();
        System.in.read();
        Person p1 = new Person();
```

```
            System.in.read();
            System.in.read();
            Person p2 = new Person();
            p1.setMate(p2);
            p2.setMate(p1);
            System.out.println("before exit gctest!");
            System.in.read();
            System.in.read();
            System.gc();
            System.out.println("exit gctest!");
        }

        private static class Person{
            byte[] data = new byte[20000000];
            Person mate = null;
            public void setMate(Person other){
                mate = other;
            }
        }
}
```

java 中的内存泄露的情况：长生命周期的对象持有短生命周期对象的引用就很可能发生内存泄露，尽管短生命周期对象已经不再需要，但是因为长生命周期对象持有它的引用而导致不能被回收，这就是 java 中内存泄露的发生场景，通俗地说，就是程序员可能创建了一个对象，以后一直不再使用这个对象，这个对象却一直被引用，即这个对象无用但是却无法被垃圾回收器回收的，这就是 java 中可能出现内存泄露的情况，例如，缓存系统，我们加载了一个对象放在缓存中(例如放在一个全局 map 对象中)，然后一直不再使用它，这个对象一直被缓存引用，但却不再被使用。
检查 java 中的内存泄露，一定要让程序将各种分支情况都完整执行到程序结束，然后看某个对象是否被使用过，如果没有，则才能判定这个对象属于内存泄露。

如果一个外部类的实例对象的方法返回了一个内部类的实例对象，这个内部类对象被长期引用了，即使那个外部类实例对象不再被使用，但由于内部类持久外部类的实例对象，这个外部类对象将不会被垃圾回收，这也会造成内存泄露。

下面内容来自于网上（主要特点就是清空堆栈中的某个元素，并不是彻底把它从数组中拿掉，而是把存储的总数减少，本人写得可以比这个好，在拿掉某个元素时，顺便也让它从数组中消失，将那个元素所在的位置的值设置为 null 即可）：
我实在想不到比那个堆栈更经典的例子了,以致于我还要引用别人的例子，下面的例子不是我想到的，是书上看到的，当然如果没有在书上看到，可能过一段时间我自己也想的到，可是那时我说是我自己想到的也没有人相信的。

```
import java.util.EmptyStackException;

public class Stack {
```

```
    private Object[] elements = new Object[10];
    private int size = 0;

    public void push(Object e) {
        ensureCapacity();
        elements[size++] = e;
    }

    public Object pop() {
        if (size == 0)
            throw new EmptyStackException();
        return elements[--size];
    }
    private void ensureCapacity() {
        if (elements.length == size) {
            Object[] oldElements = elements;
            elements = new Object[2 * elements.length + 1];
            System.arraycopy(oldElements, 0, elements, 0, size);
            }
        }
}
```

上面的原理应该很简单，假如堆栈加了 10 个元素，然后全部弹出来，虽然堆栈是空的，没有我们要的东西，但是这是个对象是无法回收的，这个才符合了内存泄露的两个条件：无用，无法回收。

但是就是存在这样的东西也不一定会导致什么样的后果，如果这个堆栈用的比较少，也就浪费了几个 K 内存而已，反正我们的内存都上 G 了，哪里会有什么影响，再说这个东西很快就会被回收的，有什么关系。下面看两个例子。

例子 1

```
import java.util.Stack;

public class Bad {
    public static Stack s = Stack();
    static {
        s.push(new Object());
        s.pop(); // 这里有一个对象发生内存泄露
        s.push(new Object()); // 上面的对象可以被回收了，等于是自愈了
    }
}
```

因为是 static，就一直存在到程序退出，但是我们也可以看到它有自愈功能，就是说如果你的 Stack 最多有 100 个对象，那么最多也就只有 100 个对象无法被回收其实这个应该很容易理解，Stack 内部持有 100 个引用，最坏的情况就是他们都是无用的，因为我们一旦放新的进取，以前的引用自然消失！

内存泄露的另外一种情况：当一个对象被存储进 HashSet 集合中以后，就不能修改这个对象

中的那些参与计算哈希值的字段了，否则，对象修改后的哈希值与最初存储进 HashSet 集合中时的哈希值就不同了，在这种情况下，即使在 contains 方法使用该对象的当前引用作为的参数去 HashSet 集合中检索对象，也将返回找不到对象的结果，这也会导致无法从 HashSet 集合中单独删除当前对象，造成内存泄露。

107. 能不能自己写个类，也叫 java.lang.String?

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

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

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

```
package java.lang;

public class String {
    public static void main(String[] args) {
        System.out.println("string");
    }
}
```

报告的错误如下：

```
java.lang.NoSuchMethodError: main
Exception in thread "main"
```

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

二、Java 代码查错

```
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 Rollable 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。

三、算法与编程

108. 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？

用插入法进行排序代码如下

```
package test;
import java.util.*;
class InsertSort {
ArrayList al;       //定义一个链表
public InsertSort(int num,int mod)                    //带参数的构造函数{
      al = new ArrayList(num);                                  //实例化链表
      Random rand = new Random();                               //取一个随机数

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);          //定义一个类的对象
is.SortIt();                    //调用类中的方法，执行排序
}
```

}

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

109. 用反射机制声明长度为 10 的 int 型数组的表达式

```
import java.lang.reflect.Array;
public class IntArrayDemo {
    public static void main(String[] args) {
        Class c = int.class;
        Object objArr = Array.newInstance(c, 10);
        for(int i = 0; i < 10; i++) {
            Array.set(objArr, i, i);
        }
        int[] arr=(int[])objArr;
        for(int j = 0; j < arr.length; j++) {
            System.out.println(arr[j]);
        }
    }
}
```

110. 编程题：设有 n 个人依围成一圈，从第 1 个人开始报数，数到第 m 个人出列，然后从出列的下一个人开始报数，数到第 m 个人又出列，…，如此反复到所有的人全部出列为止。设 n 个人的编号分别为 1，2，…，n，打印出出列的顺序；要求用 java 实现。

答：代码如下：

```
package test;
public class CountGame {
    private static boolean same(int[] p, int l, int n) {
        for (int i = 0; i < l; i++) {
            if (p[i] == n) {
                return true;
            }
        }
        return false;
    }

    public static void play(int playerNum, int step) {
        int[] p = new int[playerNum];
        int counter = 1;
        while (true) {
            if (counter > playerNum * step) {
                break;
            }
            for (int i = 1; i < playerNum + 1; i++) {
                while (true) {
                    if (same(p, playerNum, i) == false)
```

```
                        break;
                    else
                        i = i + 1;
                }
                if (i > playerNum)
                    break;
                if (counter % step == 0) {
                    System.out.print(i + " ");
                    p[counter / step - 1] = i;
                }
                counter += 1;
            }
        }
        System.out.println();
    }
    public static void main(String[] args) {
        play(10, 7);
    }
}
```

111. 编写一个程序，将 a.txt 文件中的单词与 b.txt 文件中的单词交替合并到 c.txt 文件中，a.txt 文件中的单词用回车符分隔，b.txt 文件中用回车或空格进行分隔。

```
package cn.itcast;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;

public class MainClass {
    public static void main(String[] args) throws Exception {
        FileManager a = new FileManager("a.txt", new char[] { '\n' });
        FileManager b = new FileManager("b.txt", new char[] { '\n', ' ' });
        FileWriter c = new FileWriter("c.txt");
        String aWord = null;
        String bWord = null;
        while ((aWord = a.nextWord()) != null) {
            c.write(aWord + "\n");
            bWord = b.nextWord();
            if (bWord != null)
                c.write(bWord + "\n");
        }
        while ((bWord = b.nextWord()) != null) {
            c.write(bWord + "\n");
        }
        c.close();
```

```
        }
    }

    class FileManager {
        String[] words = null;
        int pos = 0;
        public FileManager(String filename, char[] seperators) throws Exception {
            File f = new File(filename);
            FileReader reader = new FileReader(f);
            char[] buf = new char[(int) f.length()];
            int len = reader.read(buf);
            String results = new String(buf, 0, len);
            String regex = null;
            if (seperators.length > 1) {
                regex = "" + seperators[0] + "|" + seperators[1];
            } else {
                regex = "" + seperators[0];
            }
            words = results.split(regex);
        }
        public String nextWord() {
            if (pos == words.length)
                return null;
            return words[pos++];
        }
    }
```

112. 写一个程序，将 d:\java 目录下所有.java 文件复制到 d:\jad 目录，并将原来文件的扩展名.java 改为.jad。

答：listFiles 方法接受一个 FileFilter 对象，这个 FileFilter 对象就是过虑的策略对象，不同的人提供不同的 FileFilter 实现，即提供了不同的过滤策略。

```
package test;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FilenameFilter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class Jad2Java {

    public static void main(String[] args) throws Exception {
```

```
            File srcDir = new File("java");
            if (!(srcDir.exists() && srcDir.isDirectory()))
                throw new Exception("目录不存在");
            File[] files = srcDir.listFiles(new FilenameFilter() {

                public boolean accept(File dir, String name) {
                    return name.endsWith(".java");
                }
            });

            System.out.println(files.length);
            File destDir = new File("jad");
            if (!destDir.exists())
                destDir.mkdir();
            for (File f : files) {
                FileInputStream fis = new FileInputStream(f);
                String destFileName = f.getName().replaceAll("\\.java$", ".jad");
                FileOutputStream fos = new FileOutputStream(new File(destDir,
                        destFileName));
                copy(fis, fos);
                fis.close();
                fos.close();
            }
        }

        private static void copy(InputStream ips, OutputStream ops)
                throws Exception {
            int len = 0;
            byte[] buf = new byte[1024];
            while ((len = ips.read(buf)) != -1) {
                ops.write(buf, 0, len);
            }
        }
    }
```

由本题总结的思想及策略模式的解析：

class jad2java{

1. 得到某个目录下的所有的 java 文件集合

1.1 得到目录 File srcDir = new File("d:\\java");

1.2 得到目录下的所有 java 文件：File[] files = srcDir.listFiles(new MyFileFilter());

1.3 只想得到.java 的文件： class MyFileFilter implememyts FileFilter{

```
public boolean accept(File pathname){
    return pathname.getName().endsWith(".java")
}
```

```
        }
```

2.将每个文件复制到另外一个目录，并改扩展名

2.1 得到目标目录，如果目标目录不存在，则创建之

2.2 根据源文件名得到目标文件名，注意要用正则表达式，注意.的转义。

2.3 根据表示目录的 File 和目标文件名的字符串，得到表示目标文件的 File。

//要在硬盘中准确地创建出一个文件，需要知道文件名和文件的目录。

2.4 将源文件的流拷贝成目标文件流，拷贝方法独立成为一个方法，方法的参数采用抽象流的形式。

//方法接受的参数类型尽量面向父类，越抽象越好，这样适应面更宽广。

```
}
```

分析 listFiles 方法内部的策略模式实现原理

```
File[] listFiles(FileFilter filter){
    File[] files = listFiles();
    //Arraylist acceptedFilesList = new ArrayList();
    File[] acceptedFiles = new File[files.length];
    int pos = 0;
    for(File file: files){
        boolean accepted = filter.accept(file);
        if(accepted){
            //acceptedFilesList.add(file);
            acceptedFiles[pos++] = file;
        }
    }
    Arrays.copyOf(acceptedFiles,pos);
    //return (File[])accpetedFilesList.toArray();
}
```

113. 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，但要保证汉字不被截取半个，如“我 ABC”，4，应该截取“我 AB”，输入“我 ABC 汉 DEF”，6，应该输出“我 ABC”，而不是“我 ABC+汉的半个”。

答：首先要了解中文字符有多种编码及各种编码的特征。

假设 n 为要截取的字节数。

```
    public static void main(String[] args) throws Exception{
        String str = "我 a 爱中华 abc 我爱中国 def';
        String str = "我 ABC 汉";
        int num = trimGBK(str.getBytes("GBK"),5);
        System.out.println(str.substring(0,num) );
    }

    public static int    trimGBK(byte[] buf,int n){
        int num = 0;
        boolean bChineseFirstHalf = false;
        for(int i=0;i<n;i++){
```

```
                if(buf[i]<0 && !bChineseFirstHalf){
                    bChineseFirstHalf = true;
                }else{
                    num++;
                    bChineseFirstHalf = false;
                }
            }
            return num;
        }
```

114. 有一个字符串，其中包含中文字符、英文字符和数字字符，请统计和打印出各个字符的个数。

答：哈哈，其实包含中文字符、英文字符、数字字符原来是出题者放的烟雾弹。

```
String content = “中国 aadf 的 111 萨 bbb 菲的 zz 萨菲”;
HashMap map = new HashMap();
for(int i=0;i<content.length;i++){
    char c = content.charAt(i);
    Integer num = map.get(c);
    if(num == null)
        num = 1;
    else
        num = num + 1;
    map.put(c,num);
}
for(Map.EntrySet entry : map){
    system.out.println(entry.getkey() + “:” + entry.getValue());
}
```

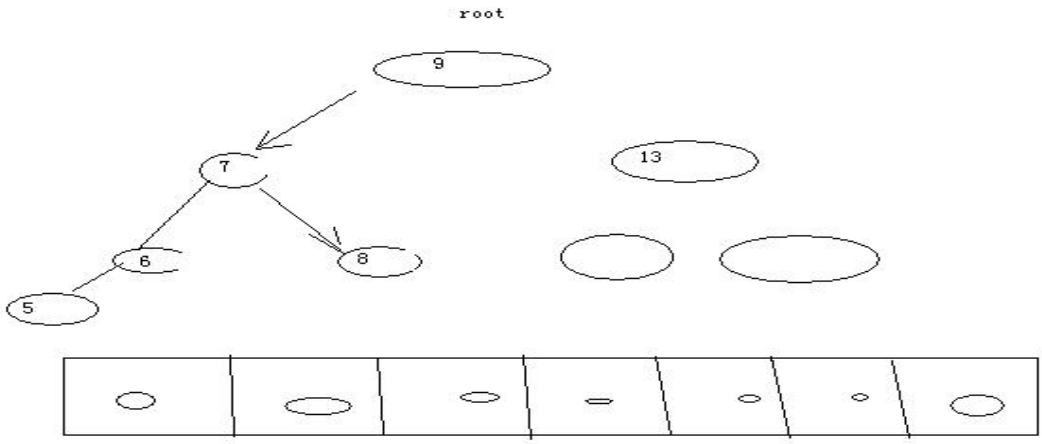
估计是当初面试的那个人表述不清楚，问题很可能是：

如果一串字符如"aaaabbc 中国 1512"要分别统计英文字符的数量，中文字符的数量，和数字字符的数量，假设字符中没有中文字符、英文字符、数字字符之外的其他特殊字符。

```
int engishCount;
int chineseCount;
int digitCount;
for(int i=0;i<str.length;i++){
    char ch = str.charAt(i);
    if(ch>=’0’ && ch<=’9’){
        digitCount++
    }
    else if((ch>=’a’ && ch<=’z’) || (ch>=’A’ && ch<=’Z’)){
        engishCount++;
    }else{
        chineseCount++;
    }
}
System.out.println(……);
```

115. 说明生活中遇到的二叉树，用 java 实现二叉树

这是组合设计模式。
我有很多个(假设 10 万个)数据要保存起来，以后还需要从保存的这些数据中检索是否存在某个数据，（我想说出二叉树的好处，该怎么说呢？那就是说别人的缺点），假如存在数组中，那么，碰巧要找的数字位于 99999 那个地方，那查找的速度将很慢，因为要从第 1 个依次往后取，取出来后进行比较。平衡二叉树（构建平衡二叉树需要先排序，我们这里就不作考虑了）可以很好地解决这个问题，但二叉树的遍历（前序，中序，后序）效率要比数组低很多，原理如下图：



代码如下：

```
package com.huawei.interview;

public class Node {
    public int value;
    public Node left;
    public Node right;

    public void store(int value){
        if(value<this.value){
            if(left == null){
                left = new Node();
                left.value=value;
            }
            else{
                left.store(value);
            }
        }
        else if(value>this.value){
            if(right == null){
                right = new Node();
                right.value=value;
            }
```

```
            else{
                right.store(value);
            }
        }
    }

    public boolean find(int value){
        System.out.println("happen " + this.value);
        if(value == this.value){
            return true;
        }
        else if(value>this.value){
            if(right == null) return false;
            return right.find(value);
        }else{
            if(left == null) return false;
            return left.find(value);
        }
    }

    public    void preList(){
        System.out.print(this.value + ",");
        if(left!=null) left.preList();
        if(right!=null) right.preList();
    }

    public void middleList(){
        if(left!=null) left.preList();
        System.out.print(this.value + ",");
        if(right!=null) right.preList();
    }
    public void afterList(){
        if(left!=null) left.preList();
        if(right!=null) right.preList();
        System.out.print(this.value + ",");
    }
    public static void main(String [] args){
        int [] data = new int[20];
        for(int i=0;i<data.length;i++){
            data[i] = (int)(Math.random()*100) + 1;
            System.out.print(data[i] + ",");
        }
        System.out.println();
```

```
        Node root = new Node();
        root.value = data[0];
        for(int i=1;i<data.length;i++){
            root.store(data[i]);
        }

        root.find(data[19]);

        root.preList();
        System.out.println();
        root.middleList();
        System.out.println();
        root.afterList();
    }
}
----------又一次临场写的代码----------
import java.util.Arrays;
import java.util.Iterator;

public class Node {
    private Node left;
    private Node right;
    private int value;
    //private int num;

    public Node(int value){
        this.value = value;
    }
    public void add(int value){

        if(value > this.value){
            if(right != null)
                right.add(value);
            else{
                Node node = new Node(value);
                right = node;
            }
        }
        else{
            if(left != null)
                left.add(value);
            else{
                Node node = new Node(value);
                left = node;
```

```
            }
        }
    }

    public boolean find(int value){
        if(value == this.value) return true;
        else if(value > this.value){
            if(right == null) return false;
            else return right.find(value);
        }else{
            if(left == null) return false;
            else return left.find(value);
        }
    }

    public void display(){
        System.out.println(value);
        if(left != null) left.display();
        if(right != null) right.display();
    }

    /*public Iterator iterator(){

    }*/

    public static void main(String[] args){
        int[] values = new int[8];
        for(int i=0;i<8;i++){
            int num = (int)(Math.random() * 15);
            //System.out.println(num);
            //if(Arrays.binarySearch(values, num)<0)
            if(!contains(values,num))
                values[i] = num;
            else
                i--;
        }

        System.out.println(Arrays.toString(values));

        Node root    = new Node(values[0]);
        for(int i=1;i<values.length;i++){
            root.add(values[i]);
        }
        System.out.println(root.find(13));
```

```
            root.display();
        }

        public static boolean contains(int [] arr, int value){
            int i = 0;
            for(;i<arr.length;i++){
                if(arr[i] == value) return true;
            }
            return false;
        }
}
```

116. 从类似如下的文本文件中读取出所有的姓名，并打印出重复的姓名和重复的次数，并按重复次数排序：

1,张三,28
2,李四,35
3,张三,28
4,王五,35
5,张三,28
6,李四,35
7,赵六,28
8,田七,35

程序代码如下（答题要博得用人单位的喜欢，包名用该公司，面试前就提前查好该公司的网址，如果查不到，现场问也是可以的。还要加上实现思路的注释）：

```
package com.huawei.interview;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.TreeSet;


public class GetNameTest {
    public static void main(String[] args) {
        //InputStream                                    ips                                    =
GetNameTest.class.getResourceAsStream("/com/huawei/interview/info.txt");
        //用上一行注释的代码和下一行的代码都可以，因为 info.txt 与 GetNameTest 类在同一包下面，所以，可以用下面的相对路径形式
```

```
        Map results = new HashMap();
        InputStream ips = GetNameTest.class.getResourceAsStream("info.txt");
        BufferedReader in = new BufferedReader(new InputStreamReader(ips));
        String line = null;
        try {
            while((line=in.readLine())!=null){
                dealLine(line,results);
            }
            sortResults(results);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    static class User{
        public   String name;
        public Integer value;
        public User(String name,Integer value){
            this.name = name;
            this.value = value;
        }
        public boolean equals(Object obj) {
            //下面的代码没有执行，说明往 treeset 中增加数据时，不会使用到 equals 方法。
            boolean result = super.equals(obj);
            System.out.println(result);
            return result;
        }
    }

    private static void sortResults(Map results) {
        TreeSet sortedResults = new TreeSet(
                new Comparator(){
                    public int compare(Object o1, Object o2) {
                        User user1 = (User)o1;
                        User user2 = (User)o2;
                        /*如果 compareTo 返回结果 0，则认为两个对象相等，新的对象
不会增加到集合中去
                         * 所以，不能直接用下面的代码，否则，那些个数相同的其他
姓名就打印不出来。
                         * */

                        //return user1.value-user2.value;
                        //return user1.value<user2.value?-1:user1.value==user2.value?0:1;
                        if(user1.value<user2.value)    {
```

```
                    return -1;
                }else if(user1.value>user2.value){
                    return 1;
                }else{
                    return user1.name.compareTo(user2.name);
                }
            }
        }
    );
    Iterator iterator = results.keySet().iterator();
    while(iterator.hasNext()){
        String name = (String)iterator.next();
        Integer value = (Integer)results.get(name);
        if(value > 1){
            sortedResults.add(new User(name,value));
        }
    }

    printResults(sortedResults);
}
private static void printResults(TreeSet sortedResults) {
    Iterator iterator    = sortedResults.iterator();
    while(iterator.hasNext()){
        User user = (User)iterator.next();
        System.out.println(user.name + ":" + user.value);
    }
}
public static void dealLine(String line,Map map){
    if(!"".equals(line.trim())){
        String [] results = line.split(",");
        if(results.length == 3){
            String name = results[1];
            Integer value = (Integer)map.get(name);
            if(value == null) value = 0;
            map.put(name,value + 1);
        }
    }
}
}
```

117. 写一个 Singleton 出来。

第一种：饱汉模式

```
public class SingleTon {
    private SingleTon(){
    }
```

```
    //实例化放在静态代码块里可提高程序的执行效率，但也可能更占用空间
    private final static SingleTon instance = new SingleTon();
    public static SingleTon getInstance(){
        return instance;
    }
}
```

第二种：饥汉模式

```
public class SingleTon {
    private SingleTon(){}
    private static instance = null;//new SingleTon();
    public static synchronized SingleTon getInstance(){
        if(instance == null)
            instance = new SingleTon();
        return instance;
    }
}
```

第三种：用枚举

```
    public enum SingleTon{
        ONE;
    }
```

第三：更实际的应用（在什么情况用单例）

```
public class SequenceGenerator{
    //下面是该类自身的业务功能代码
    private int count = 0;
    public synchronized int getSequence(){
        ++count;
    }

    //下面是把该类变成单例的代码
    private SequenceGenerator(){}
    private final static instance = new SequenceGenerator();
    public static SingleTon getInstance(){
        return instance;
    }

}
```

第四：

```
public class MemoryDao{
    private HashMap map = new HashMap();

    public void add(Student stu1){
```

```
                map.put(SequenceGenerator.getInstance().getSequence(),stu1);
        }
      //把 MemoryDao 变成单例
    }
```

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;
    }
}
```

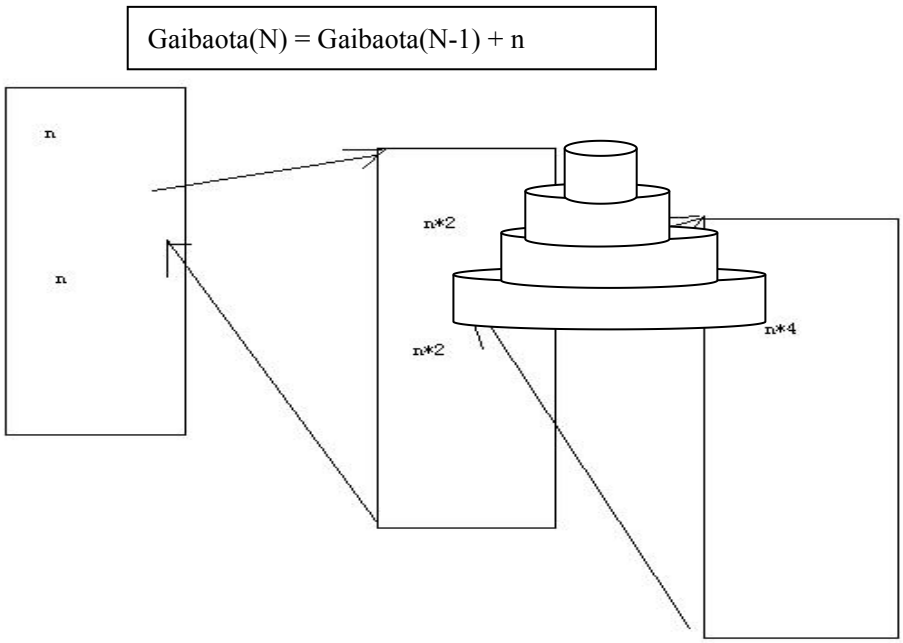
其他形式:
定义一个类，它的构造函数为 private 的，所有方法为 static 的。
一般认为第一种形式要更加安全些

118. 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，
提示：写程序时，先致谢按递增方式的代码，写好递增的以后，再增加考虑递减部分。

```
public static void doubleNum(int n){
        System.out.println(n);
        if(n<=5000)
            doubleNum(n*2);
        System.out.println(n);
}
```



119. 递归算法题 2

第 1 个人 10 岁，第 2 个比第 1 个人大 2 岁，依次递推，请用递归方式计算出第 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);
    }
}
```

120. 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 {
    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);
        }
    }

    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] + " ");
        }
    }
```

```
}
```

121. 有数组 a[n]，用 java 代码将数组元素顺序颠倒

```
//用下面的也可以
//for(int i=0,int j=a.length-1;i<j;i++,j--)  是否等效于  for(int i=0;i<a.length/2;i++)呢？

import java.util.Arrays;

public class SwapDemo{

	public static void main(String[] args){
		int [] a = new int[]{
					(int)(Math.random() * 1000),
					(int)(Math.random() * 1000),
					(int)(Math.random() * 1000),
					(int)(Math.random() * 1000),
					(int)(Math.random() * 1000)

		};

		System.out.println(a);
		System.out.println(Arrays.toString(a));
		swap(a);
		System.out.println(Arrays.toString(a));
	}

	public static void swap(int a[]){
		int len = a.length;
		for(int i=0;i<len/2;i++){
			int tmp = a[i];
			a[i] = a[len-1-i];
			a[len-1-i] = tmp;
		}
	}
}
```

122. 金额转换，阿拉伯数字的金额转换成中国传统形式如：(￥1011)－>(一千零一拾一元整) 输出。

去零的代码：

```
return  sb.reverse().toString().replaceAll(" 零 [ 拾 佰 仟 ]"," 零 ").replaceAll(" 零 + 万 "," 万 ").replaceAll("零+元","元").replaceAll("零+","零");

public class RenMingBi {
	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();
    }
}
```

四、html&JavaScript&ajax 部分

123. 判断第二个日期比第一个日期大

如何用脚本判断用户输入的的字符串是下面的时间格式 2004-11-21 必须要保证用户的输入是此格式，并且是时间，比如说月份不大于 12 等等，另外我需要用户输入两个，并且后一个要比前一个晚，只允许用 JAVASCRIPT，请详细帮助作答，，

```
//这里可用正则表达式判断提前判断一下格式，然后按下提取各时间字段内容
<script type="text/javascript">
    window.onload = function(){
        //这么写是为了实现 js 代码与 html 代码的分离，当我修改 js 时，不能影响 html 代码。
        document.getElementById("frm1").onsubmit =
            function(){
                var d1 = this.d1.value;
                var d2 = this.d2.value;
                if(!verifyDate (d1)) {alert("第一个日期格式不对");return false;}
                if(!verifyDate (d2)) {alert("第二个日期格式不对");return false;}
                if(!compareDate(d1,d2)) {alert("第二个日期比第一日期小");return false;}

            };
        }
```

```
        function compareDate(d1,d2){
            var arrayD1 = d1.split("-");
            var date1 = new Date(arrayD1[0],arrayD1[1],arrayD1[2]);
            var arrayD2 = d2.split("-");
            var date2 = new Date(arrayD2[0],arrayD2[1],arrayD2[2]);
            if(date1 > date2) return false;
            return true;
        }

        function verifyDate(d){
            var datePattern = /^\d{4}-(0?[1-9]|1[0-2])-(0?[1-9]|[1-2]\d|3[0-1])$/;
            return datePattern.test(d);
        }
</script>

<form id="frm1" action="xxx.html">
<input type="text" name="d1" />
<input type="text" name="d2" />
<input type="submit"/>
</form>
```

124. 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。

```
<body>
<table id="tbl">
    <tr><td>1</td></tr>
    <tr><td>2</td></tr>
    <tr><td>3</td></tr>
    <tr><td>4</td></tr>
    <tr><td>5</td></tr>
    <tr><td>6</td></tr>
    <tr><td>7</td></tr>
    <tr><td>8</td></tr>
    <tr><td>9</td></tr>
    <tr><td>10</td></tr>
</table>
</body>
<script type="text/javascript">
    window.onload=function(){
            var tbl = document.getElementById("tbl");
            rows = tbl.getElementsByTagName("tr");
            for(i=0;i<rows.length;i++){
                var j = parseInt(i/3);
                if(j%2==0) rows[i].style.backgroundColor="#f00";
```

```
                else    rows[i].style.backgroundColor="#0f0";
            }
        }
</script>
```

125. HTML 的 form 提交之前如何验证数值文本框的内容全部为数字？否则的话提示用户并终止提交?

```
<form onsubmit=’return chkForm(this)’>
<input type="text" name="d1"/>
<input type="submit"/>
</form>
<script type=”text/javascript” />
function chkForm(this){
        var value = thist.d1.value;
        var len = value.length;
        for(var i=0;i<len;i++){
            if(value.charAt(i)>"9" || value.charAt(i)<"0"){
                alert("含有非数字字符");
                return false;
            }
        }
        return true;
    }
</script>
```

126. 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

```
<input type="text" id="d1" onblur=" chkNumber (this)"/>
<script type=”text/javascript” />
function chkNumber(eleText){
        var value = eleText.value;
        var len = value.length;
        for(var i=0;i<len;i++){
            if(value.charAt(i)>"9" || value.charAt(i)<"0"){
                alert("含有非数字字符");
                eleText.focus();
                break;
            }
        }
    }
</script>
```

除了写完代码，还应该在网页上写出实验步骤和在代码中加入实现思路，让面试官一看就明白你的意图和检查你的结果。

127. 说说你用过那些 ajax 技术和框架，说说它们的区别

答:去掉对 web.xml 的监视，把 jsp 提前编辑成 Servlet。

有富余物理内存的情况，加大 tomcat 使用的 jvm 的内存

五、Java web 部分

128. Tomcat 的优化经验

答:去掉对 web.xml 的监视，把 jsp 提前编辑成 Servlet。
有富余物理内存的情况，加大 tomcat 使用的 jvm 的内存

129. HTTP 的请求过程？

当点击一个链接时,浏览器首先找到站点的 IP 地址,这是通过 DNS 来实现的,在找到 IP 地址后就可以建立 TCP 连接了,连接建立后我们就可以发送请求了.但这个请求是什么样子的呢 ？我们现在假设点击了一个从 www.webmonkey.com/HTML/96/47/Index2A , HTML 点击了 WWW.GRIPY.ORG/MATTARG/ 这时浏览器会发出下面的请求:
Get/MATTARG/HTML/1.0
User-Agent: Mozilla/2.0(macitosh;1;PPC)
Accept: text/html: */*
Cookie: name = value
Refetet: http://www.webmonkey.com/html/96/47/index2a.html
Host: www.gtippy.org
第一行称为请求,它告诉服务器从 MATTMARG 取得文件,这是的目录一般是要加 / 的,下面几行通知服务器你所使用的浏览器是什么类型,你所接收的数据是什么类型,如果你以前访问过这个站点,站点可能向你发送了 Cookie ,如果你已经有了一个这样的 Cookie ,浏览器会将这个 Cookie 返回给服务器, referer 行通知服务器用户从哪一页到达此页的.
下面服务器就要返回文件了,每次服务器返回文件时,都要返回一个 Http/1.0 响应,同进带有状态码,在此之后是述内部的头信息,下面就是一个响应:
HTTP/1.0 200 Pound
Data: Mon 10 Feb 1997 23:48:22 GMT
Server: Apache/1.1 1 Hot&ired/1.0
Content-type: text/html
Last-Moditied: Tues,11 Feb 1997 22:45:55 GMT
不同的数据可能返回不同的 Content-type ，因此不同的内容需要不同的 Content-type ，因此有时候这个过程是很慢的。

130. HTTP 请求的 GET 与 POST 方式的区别

答：Form 中的 get 和 post 方法，在数据传输过程中分别对应了 HTTP 协议中的 GET 和 POST 方法。二者主要区别如下：

1）Get 是用来从服务器上获得数据，而 Post 是用来向服务器上传递数据；

2）Get 将表单中数据按照 variable=value 的形式，添加到 action 所指向的 URL 后面，并且两者使用“?”连接，而各个变量之间使用“&”连接；Post 是将表单中的数据放在 form 的数据体中，按照变量和值相对应的方式，传递到 action 所指向 URL；

3）Get 是不安全的，因为在传输过程，数据被放在请求的 URL 中；Post 的所有操作对用户来说都是不可见的；

4）Get 传输的数据量小，这主要是因为受 URL 长度限制；而 Post 可以传输大量的数据，所以在上传文件只能使用 Post；

5）Get 限制 Form 表单的数据集必须为 ASCII 字符，而 Post 支持整个 ISO10646 字符集；

6）Get 是 Form 的默认方法。

131. 解释一下什么是 servlet?

Servlet 是一种独立于平台和协议的服务器端的 Java 技术，可以用来生成动态的 Web 页面。与传统的 CGI（计算机图形接口）和许多其他类似 CGI 技术相比，Servlet 具有更好的可移植性、更强大的功能，更少的投资，更高的效率，更好的安全性等特点。

Servlet 是使用 Java Servlet 应用程序接口（API）及相关类和方法的 Java 程序。Java 语言能够实现的功能，Servlet 基本上都能实现（除了图形界面之外）。Servlet 主要用于处理客户端传来的 Http 请求，并返回一个响应。通常所说的 Servlet 就是指 HttpServlet，用于处理 Http 请求，其能够处理的请求有 doGet()、doPost()、service()等方法。在开发 Servlet 时，可以直接继承 javax.servlet.http.HttpServlet。

Servlet 需要在 web.xml 中进行描述，例如：映射执行 Servlet 的名字，配置 Servlet 类、初始化参数，进行安全配置、URL 映射和设置启动的优先权等。Servlet 不仅可以生成 HTML 脚本输出，也可以生成二进制表单输出。

Servlet 应用范围很广泛，我们现在用的很多流行的框架技术，其最基本的代码离不开 Servelt 的支持。比如我所熟悉的 SSH 框架，Spring 容器启动时，要在 web.xml 中装载 Spring 容器的 ActionContext 类来初始化 Spring 的一些参数，如进行依赖注入、数据库表的映射、初始化系统的安全配置设置 read 等属性等一些相关操作。

132. 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。

133. Servlet 的基本架构

```
package test;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

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 {
        }
}
```

134. 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。

135. 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()。

136. 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

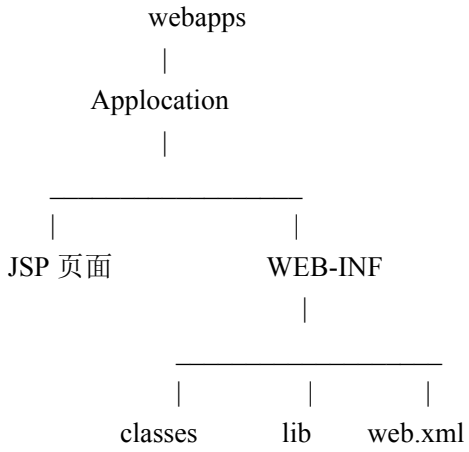
137. 简述 HttpSession 的作用、使用方法，可用代码说明。

HttpSession 中可以跟踪并储存用户信息，把值设置到属性中，有 2 个方法：setAttribute(),getAttrribute()；

例如：在一个方法中用 session.setAttribute("student",student);在 session 中设置一个属性名为 student,值为一个名为 student 的对象。而后可在同一 session 范围内用 getAttribute("student") 取出该属性，得到 student 对象。

138. 请画出 Servlet 2.2 以上 Web Application 的基本目录结构

答：目录结构如下图所示：



139. cookie 和 session 的区别？

注意 cookie 有时候禁不掉

Cookie	session
存储在客户端	存储在服务器端
两种类型	两种实现方式
有声明周期	依赖于 cookie
无声明周期	url 重写
父路径不能访问子路径的 cookie	同一个 session 的窗口共享一个 session
典型应用：	典型应用：
3 个月不用再登陆	用户登陆
购物车（http://www.china-pub.com/）	购物车也可以用 session 实现。 •
不可靠	可靠

140. 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。

141. request.getAttribute() 和 request.getParameter() 有何区别?

当你要传递普通的数据类型给下一个页面时，你在下一个页面中就可以用 getParameter()方法来获得上一个页面传递过来的数据了！（普通的数据类型是指 int,float,double,string 等在 Java 中常用的基本类型，但是在下一个页面中你用 getParameter()方法获得的值永远只能时 String 类型的，你可以把 String 类型转换为你所需要的类型！）

当你要传递一个对象给下一个页面时，你就要使用 getAttribut()方法了！如：你要把一个 List

或 Map 传递到下一个页面，这时你就必须要用 setAttribut()和 getAttribut()方法传递数据了！从更深层次的考虑，getParameter()方法传递数据，只会从 WEB 客户端传递到 WEB 服务器，代表 HTTP 请求数据，getParameter()方法返回 String 类型的数据！setAttribut()和 getAttribut()方法传递的数据只会在 WEB 服务器内部，在具有转发关系的 WEB 组件之间传递，这两个方法能设置 Object 类型的共享数据！

142. 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

143. 介绍在 JSP 中如何使用 JavaBeans？

答：在 JSP 中使用 JavaBean 常用的动作有：
1）<jsp:useBean />：用来创建和查找 bean 对象；
2）<jsp:setProperty />：用来设置 bean 的属性，即调用其 setXxx()方法；
3）<jsp:getProperty />：用来获得 bean 的属性，即调用其 getXxx()方法。

144. 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

145. JSP 的常用指令

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

146. jsp 的四种范围？

答：a.page 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

b.request 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）

c.session 是代表与用于某个 Web 客户机的一个用户体验相关对象和属性。一个 Web 会话可以也经常会跨越多个客户机请求

d.application 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。

147. 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 %>

148. 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.

149. 页面间对象传递的方法

request，session，application，cookie 等

150. 过滤器有哪些作用？

答：可以验证客户是否来自可信的网络，可以对客户提交的数据进行重新编码，可以从系统里获得配置的信息，可以过滤掉客户的某些不应该出现的词汇，可以验证用户是否登录，可以验证客户的浏览器是否支持当前的应用，可以记录系统的日志等等。

151. 过滤器的用法？（对客户端的请求统一编码和对客户端进行认证）

答：首先要实现（implements）Filter 接口，同时覆盖 Filter 接口的三个方法：

init(FilterConfig config) //用于获得 FilterConfig 对象；

doFilter(ServletRequest request, ServletResponse response, FilterChain chain) //进行过滤处理一些业务；

destroy() //销毁 Filter。

152. JSP 和 Servlet 中的请求转发分别如何实现？

答：JSP 中的请求转发可利用 forward 动作实现：<jsp:forward />；

Serlvet 中实现请求转发的方式为：

getServletContext().getRequestDispatcher(path).forward(req,res)。

153. 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。

154. 详细描述 MVC。

答：基于 Java 的 Web 应用系统采用 MVC 架构模式，即 model（模型）、view（视图）、control（控制）分离设计；这是目前 WEB 应用服务系统的主流设计方向。

Model：即处理业务逻辑的模块，每一种处理一个模块；

View：负责页面显示，显示 MODEL 处理结果给用户，主要实现数据到页面转换过程；

Control：负责每个请求的分发，把 FORM 数据传递给 MODEL 处理，把处理结果的数据传递给 VIEW 显示。

155. 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。

156. 在 web 应用开发过程中经常遇到输出某种编码字符，如 iso8859-1 等，如何输出一个某种编码字符串？

```
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;
}
```

157. 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.(答案有点疑惑,问题有点不清不楚)

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

158. 说出数据连接池的工作机制是什么？

答：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。

159. javascript 的优缺点和内置对象。

答：1）优点：简单易用，与 Java 有类似的语法，可以使用任何文本编辑工具编写，只需要浏览器就可执行程序，并且事先不用编译，逐行执行，无需进行严格的变量声明，而且内置大量现成对象，编写少量程序可以完成目标；

2）缺点：不适合开发大型应用程序；

3）Javascript 有 11 种内置对象： Array、String、Date、Math、Boolean、Number、Function、Global、Error、RegExp、Object。

六、数据库部分

160. 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。

161. 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。

162. 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。

163. 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。

164. 存储过程，触发器，范式，事务的概念及作用？

存储过程：是数据库管理系统里的一个很重要的对象。用它可以封装一些功能。把多个 SQL 语句封装到存储过程里面。起到封装功能的作用。类似面向对象里，封装对象的一个功能一样。几乎任何可写成批处理的 Transact-SQL 代码都可用于创建存储过程。
触发器：触发器是在用户进行某项操作的时候，会触发触发器的执行。它类似于 JAVA 中图形截面编程里的事件操作一样，是触发执行。和存储过程的主要区别在于：存储过程类似 JAVA 里面的对象一样，进行功能的封装（方法）。在调用的时候才会执行。而触发器只能在别的操作执行的时候才会触发触发器的执行。
事务：类似于 JAVA 里面线程的同步一样，作为一个单元执行。它有四大特性：原子性，隔离性，一致性，持久性。在 SQL SERVER 2000 里面还支持存储点的用法。大家都知道，事务是做为一个单元运行，要么全部执行，要么全部不执行。但是有时候我们可以保证事务的一部分可能正确执行，并且这些执行可以直接刷新到数据库里面。那么我们就可以在这个事务的中间部分设置一个或者多个存储点。这样在这个事务大单元里就分成了几个小部分。如果上面的部分执行正确，下面的部分执行错误，那么就没必要回滚整个事务，只需要回滚到存储点的地方就可以了
范式：目地：规范化目的是使结构更合理，消除存储异常，使数据冗余尽量小，便于插入、删除和更新
原则：遵从概念单一化 "一事一地"原则，即一个关系模式描述一个实体或实体间的一种联系。规范的实质就是概念的单一化。
方法：将关系模式投影分解成两个或两个以上的关系模式。
要求：分解后的关系模式集合应当与原关系模式"等价"，即经过自然联接可以恢复原关系而不丢失信息，并保持属性间合理联系。

165. 写一个 SQL Server 中的存储过程:

以下为一个带有一个输入参数 Vdeptno ,返回部门为 Vdeptnor 的所有职员的信息.

```
create procedure Emp_dept
    @Vdeptno number(2) AS
begin
    select * from emp where deptno=@Vdeptno
end
```

166. 写一个 Oracle 中的存储过程:

带 IN 参数的过程

```
create or replace procedure addnew(dno IN number,
                                   name IN varchar2,
                                   location IN varchar2)IS
begin
insert into dept values(dno,name,location);
dbms_output.put_line('1 record inserted');
        end;
```

带 OUT 参数的过程

```
        create or replace procedure getsal(name IN varchar2,
                                           salary OUT number)AS
        begin
select sal into salary from emp where ename=name;
        end;
```

167. 数据库 SQL 语句题

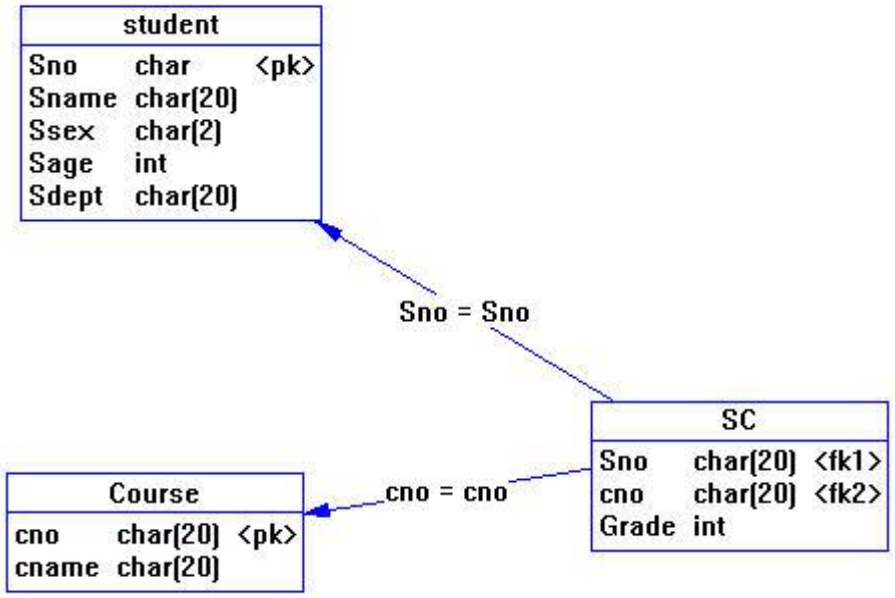
106、有 3 个表（15 分钟）：【基础】

Student 学生表 (学号，姓名，性别，年龄，组织部门)

Course 课程表 (编号，课程名称)

Sc 选课表 (学号，课程编号，成绩)

表结构如下：



1）写一个 SQL 语句，查询选修了’计算机原理’的学生学号和姓名（3 分钟）
2）写一个 SQL 语句，查询’周星驰’同学选修了的课程名字（3 分钟）
3）写一个 SQL 语句，查询选修了 5 门课程的学生学号和姓名（9 分钟）
答：1）SQL 语句如下：

```
select stu.sno, stu.sname from Student stu
where (select count(*) from sc where sno=stu.sno and cno =
        (select cno from Course where cname='计算机原理')) != 0;
```

2）SQL 语句如下：

```
select cname from Course
where cno in ( select cno from sc where sno =
        (select sno from Student where sname='周星驰'));
```

3）SQL 语句如下：

```
select stu.sno, stu.sname from student stu
where (select count(*) from sc where sno=stu.sno) = 5;
```

107、有三张表,学生表 S,课程 C,学生课程表 SC,学生可以选修多门课程,一门课程可以被多个学生选修,通过 SC 表关联。【基础】
1）写出建表语句；
2）写出 SQL 语句,查询选修了所有选修课程的学生；
3）写出 SQL 语句,查询选修了至少 5 门以上的课程的学生。
答：1）建表语句如下（mysql 数据库）：

```
create table s(id integer primary key, name varchar(20));
create table c(id integer primary key, name varchar(20));
create table sc(
    sid integer references s(id),
    cid integer references c(id),
    primary key(sid,cid)
 );
```

2）SQL 语句如下：

```
select stu.id, stu.name from s stu
```

```
        where (select count(*) from sc where sid=stu.id)
            = (select count(*) from c);
```

3）SQL 语句如下：

```
select stu.id, stu.name from s stu
        where (select count(*) from sc where sid=stu.id)>=5;
```

108、数据库表(Test)结构如下：【基础】

ID	NAME	AGE	MANAGER(所属主管人 ID)
106	A	30	104
109	B	19	104
104	C	20	111
107	D	35	109
112	E	25	120
119	F	45	NULL

要求:列出所有年龄比所属主管年龄大的人的 ID 和名字?

答：SQL 语句如下：

```
select employee.name from test employee
where employee.age > (select manager.age from test manager
                        where manager.id=employee.manager);
```

109、有如下两张表：【中等难度】

表 city：

CityNo	CityName	StateNo
BJ	北京	（Null）
SH	上海	（Null）
GZ	广州	GD
DL	大连	LN

表 state：

State No	State Name
GD	广东
LN	辽宁
SD	山东
NMG	内蒙古

欲得到如下结果：

City No	City Name	State No	State Name
BJ	北京	（Null）	（Null）
DL	大连	LN	辽宁
GZ	广州	GD	广东
SH	上海	（Null）	（Null）

写相应的 SQL 语句。

答：SQL 语句为：

```
SELECT C.CITYNO, C.CITYNAME, C.STATENO, S.STATENAME
FROM CITY C, STATE S
WHERE C.STATENO=S.STATENO(+)
ORDER BY(C.CITYNO);
```

168. 用两种方式根据部门号从高到低，工资从低到高列出每个员工的信息。

employee:

```
    eid,ename,salary,deptid;
select * from employee order by deptid desc,salary
```

169. 列出各个部门中工资高于本部门的平均工资的员工数和部门号，并按部门号排序

创建表：

```
mysql> create table employee921(id int primary key auto_increment,name varchar(5
0),salary bigint,deptid int);
```

插入实验数据：

```
mysql> insert into employee921 values(null,'zs',1000,1),(null,'ls',1100,1),(null
,'ww',1100,1),(null,'zl',900,1) ,(null,'zl',1000,2), (null,'zl',900,2) ,(null,'z
l',1000,2) , (null,'zl',1100,2);
```

编写 sql 语句：

（）select avg(salary) from employee921 group by deptid;

（）mysql> select employee921.id,employee921.name,employee921.salary,employee921.dep
tid tid from employee921 where salary > (select avg(salary) from employee921 where deptid =
tid);

效率低的一个语句，仅供学习参考使用（在 group by 之后不能使用 where，只能使用 having，在 group by 之前可以使用 where，即表示对过滤后的结果分组）：

```
mysql> select employee921.id,employee921.name,employee921.salary,employee921.dep
tid tid from employee921 where salary > (select avg(salary) from employee921 group by deptid
having deptid = tid);
```

（）select count(*) ,tid

```
from (
    select employee921.id,employee921.name,employee921.salary,employee921.deptid tid
    from employee921
    where salary >
        (select avg(salary) from employee921 where deptid = tid)
) as t
group by tid ;
```

另外一种方式：关联查询

```
select a.ename,a.salary,a.deptid
 from emp a,
    (select deptd,avg(salary) avgsal from emp group by deptid ) b
 where a.deptid=b.deptid and a.salary>b.avgsal;
```

170. 存储过程与触发器必须讲，经常被面试到?

```
create procedure insert_Student (_name varchar(50),_age int ,out _id int)
begin
    insert into student value(null,_name,_age);
    select max(stuId) into _id from student;
end;

call insert_Student('wfz',23,@id);
select @id;
```

```
mysql> create trigger update_Student BEFORE update on student FOR EACH ROW
-> select * from student;
```
触发器不允许返回结果

```
create trigger update_Student BEFORE update on student FOR EACH ROW
insert into    student value(null,'zxx',28);
```
mysql 的触发器目前不能对当前表进行操作

```
create trigger update_Student BEFORE update on student FOR EACH ROW
delete from articles    where id=8;
```
这个例子不是很好，最好是用删除一个用户时，顺带删除该用户的所有帖子
这里要注意使用 OLD.id

触发器用处还是很多的，比如校内网、开心网、Facebook，你发一个日志，自动通知好友，其实就是在增加日志时做一个后触发，再向通知表中写入条目。因为触发器效率高。而 UCH 没有用触发器，效率和数据处理能力都很低。
存储过程的实验步骤：

```
mysql> delimiter |
mysql> create procedure insertArticle_Procedure (pTitle varchar(50),pBid int,out
  pId int)
      -> begin
      -> insert into article1 value(null,pTitle,pBid);
      -> select max(id) into pId from article1;
      -> end;
      -> |
Query OK, 0 rows affected (0.05 sec)

mysql> call insertArticle_Procedure('中国北京',1,@pid);
      -> |
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
mysql> select @pid;
+------+
| @pid |
+------+
| 3    |
+------+
1 row in set (0.00 sec)

mysql> select * from article1;
+----+--------------+------+
| id | title        | bid  |
+----+--------------+------+
```

```
| 1   | test              | 1      |
| 2   | chuanzhiboke | 1      |
| 3   | 中国北京          | 1      |
+----+--------------+------+
3 rows in set (0.00 sec)
```

触发器的实验步骤：

```
create table board1(id int primary key auto_increment,name varchar(50),ar
ticleCount int);

create table article1(id int primary key auto_increment,title varchar(50)
,bid int references board1(id));

delimiter |

create trigger insertArticle_Trigger after insert on article1 for each ro
w begin
    -> update board1 set articleCount=articleCount+1 where id= NEW.bid;
    -> end;
    -> |

delimiter ;

insert into board1 value (null,'test',0);

insert into article1 value(null,'test',1);
```

还有，每插入一个帖子，都希望将版面表中的最后发帖时间，帖子总数字段进行同步更新，用触发器做效率就很高。下次课设计这样一个案例，写触发器时，对于最后发帖时间可能需要用 declare 方式声明一个变量，或者是用 NEW.posttime 来生成。

171. 数据库三范式是什么?

第一范式（1NF）：字段具有原子性,不可再分。所有关系型数据库系统都满足第一范式）数据库表中的字段都是单一属性的，不可再分。例如，姓名字段，其中的姓和名必须作为一个整体，无法区分哪部分是姓，哪部分是名，如果要区分出姓和名，必须设计成两个独立的字段。

第二范式（2NF）：第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。要求数据库表中的每个实例或行必须可以被惟一地区分。通常需要为表加上一个列，以存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键。第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是非主属性非部分依赖于主关键字。

第三范式的要求如下： 满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，

第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。所以第三范式具有如下特征：

1，每一列只有一个值 。2，每一行都能区分。3，每一个表都不包含其他表已经包含的非主关键字信息。

例如，帖子表中只能出现发帖人的 id，而不能出现发帖人的 id，还同时出现发帖人姓名，否则，只要出现同一发帖人 id 的所有记录，它们中的姓名部分都必须严格保持一致，这就是数据冗余。

172. 说出一些数据库优化方面的经验?

用 PreparedStatement 一般来说比 Statement 性能高：一个 sql 发给服务器执行，步骤：语法检查、语义分析， 编译，缓存

“inert into user values(1,1,1)”-→二进制

“inert into user values(2,2,2)”-→二进制

“inert into user values(?,?,?)”-→二进制

有外键约束会影响插入和删除性能，如果程序能够保证数据的完整性，那在设计数据库时就去掉外键。（比喻：就好比免检产品，就是为了提高效率，充分相信产品的制造商）

（对于 hibernate 来说，就应该有一个变化：empleyee->Deptment 对象，现在设计时就成了 employee→deptid）

看 mysql 帮助文档子查询章节的最后部分，例如，根据扫描的原理，下面的子查询语句要比第二条关联查询的效率高：

1. select e.name,e.salary where e.managerid=(select id from employee where name='zxx');
2. select e.name,e.salary,m.name,m.salary from employees e,employees m where e.managerid = m.id and m.name='zxx';

表中允许适当冗余，譬如，主题帖的回复数量和最后回复时间等

将姓名和密码单独从用户表中独立出来。这可以是非常好的一对一的案例哟！

sql 语句全部大写，特别是列名和表名都大写。特别是 sql 命令的缓存功能，更加需要统一大小写，sql 语句→发给 oracle 服务器→语法检查和编译成为内部指令→缓存和执行指令。

根据缓存的特点，不要拼凑条件，而是用?和 PreparedStatment

还有索引对查询性能的改进也是值得关注的。

备注：下面是关于性能的讨论举例

4 航班 3 个城市

m*n

select * from flight,city where flight.startcityid=city.cityid and city.name='beijing';

m + n

select * from flight where startcityid = (select cityid from city where cityname='beijing');

select flight.id,'beijing',flight.flightTime from flight where startcityid = (select cityid from city where cityname='beijing')

173. union 和 union all 有什么不同?

假设我们有一个表 Student，包括以下字段与数据：

drop table student;

create table student

```
(
id int primary key,
name nvarchar2(50) not null,
score number not null
);
insert into student values(1,'Aaron',78);
insert into student values(2,'Bill',76);
insert into student values(3,'Cindy',89);
insert into student values(4,'Damon',90);
insert into student values(5,'Ella',73);
insert into student values(6,'Frado',61);
insert into student values(7,'Gill',99);
insert into student values(8,'Hellen',56);
insert into student values(9,'Ivan',93);
insert into student values(10,'Jay',90);
commit;
```

Union 和 Union All 的区别。

```
select *from studentwhere id < 4union
select *from studentwhere id > 2 and id < 6
```

结果将是

```
1    Aaron    78
2    Bill    76
3    Cindy    89
4    Damon    90
5    Ella    73
```

如果换成 Union All 连接两个结果集，则返回结果是：

```
1    Aaron    78
2    Bill    76
3    Cindy    89
3    Cindy    89
4    Damon    90
5    Ella    73
```

可以看到，Union 和 Union All 的区别之一在于对重复结果的处理。

UNION 在进行表链接后会筛选掉重复的记录，所以在表链接后会对所产生的结果集进行排序运算，删除重复的记录再返回结果。实际大部分应用中是不会产生重复的记录，最常见的是过程表与历史表 UNION。如：

```
select * from gc_dfys
union
select * from ls_jg_dfys
```

这个 SQL 在运行时先取出两个表的结果，再用排序空间进行排序删除重复的记录，最后返回结果集，如果表数据量大的话可能会导致用磁盘进行排序。

而 UNION ALL 只是简单的将两个结果合并后就返回。这样，如果返回的两个结果集中有重复的数据，那么返回的结果集就会包含重复的数据了。

从效率上说，UNION ALL 要比 UNION 快很多，所以，如果可以确认合并的两个结果集中不包含重复的数据的话，那么就使用 UNION ALL，

174. 分页语句

取出 sql 表中第 31 到 40 的记录（以自动增长 ID 为主键）
sql server 方案 1：

```
select top 10 * from t where id not in (select top 30 id from t order by id ) orde by id
```

sql server 方案 2：

```
select top 10 * from t where id in (select top 40 id from t order by id) order by id desc
```

mysql 方案：select * from t order by id limit 30,10

oracle 方案：select * from (select rownum r,* from t where r<=40) where r>30

----------待整理进去的内容----------
pageSize=20;
pageNo = 5;

1.分页技术 1（直接利用 sql 语句进行分页，效率最高和最推荐的）
mysql:sql = "select * from articles limit " + (pageNo-1)*pageSize + "," + pageSize;
oracle: sql = "select * from " +"(select rownum r,* from " +"(select * from articles order by postime desc)" +"where rownum<= " + pageNo*pageSize +") tmp " +"where r>" + (pageNo-1)*pageSize;
注释：第 7 行保证 rownum 的顺序是确定的，因为 oracle 的索引会造成 rownum 返回不同的值
简洋提示：没有 order by 时，rownum 按顺序输出，一旦有了 order by，rownum 不按顺序输出了，这说明 rownum 是排序前的编号。如果对 order by 从句中的字段建立了索引，那么，rownum 也是按顺序输出的，因为这时候生成原始的查询结果集时会参照索引表的顺序来构建。

sqlserver:sql = "select top 10 * from id not id(select top " + (pageNo-1)*pageSize + "id from articles)"

```
DataSource ds = new InitialContext().lookup(jndiurl);
Connection cn = ds.getConnection();
//"select * from user where id=?"    --->binary directive
PreparedStatement pstmt = cn.prepareSatement(sql);
ResultSet rs = pstmt.executeQuery()
while(rs.next()){
    out.println(rs.getString(1));
}
```

2.不可滚动的游标
pageSize=20;
pageNo = 5;

```
cn = null
stmt = null;
rs = null;
try{
sqlserver:sql = "select    * from articles";

DataSource ds = new InitialContext().lookup(jndiurl);
Connection cn = ds.getConnection();
//"select * from user where id=?"    --->binary directive
PreparedStatement pstmt = cn.prepareSatement(sql);
ResultSet rs = pstmt.executeQuery()
for(int j=0;j<(pageNo-1)*pageSize;j++){
      rs.next();
}
int i=0;
while(rs.next() && i<10){
      i++;
      out.println(rs.getString(1));
}
}
cacth(){}
finnaly{
      if(rs!=null)try{rs.close();}catch(Exception e){}
      if(stm.....
      if(cn.....
}
```

3.可滚动的游标

```
pageSize=20;
pageNo = 5;
cn = null
stmt = null;
rs = null;
try{
sqlserver:sql = "select    * from articles";

DataSource ds = new InitialContext().lookup(jndiurl);
Connection cn = ds.getConnection();
//"select * from user where id=?"    --->binary directive
PreparedStatement                                    pstmt                                    =
cn.prepareSatement(sql,ResultSet.TYPE_SCROLL_INSENSITIVE,...);
//根据上面这行代码的异常 SQLFeatureNotSupportedException，就可判断驱动是否支持可滚动游标
ResultSet rs = pstmt.executeQuery()
```

```
rs.absolute((pageNo-1)*pageSize)
int i=0;
while(rs.next() && i<10){
    i++;
    out.println(rs.getString(1));
}
}
cacth(){}
finnaly{
    if(rs!=null)try{rs.close();}catch(Exception e){}
    if(stm.....
    if(cn.....
}
```

175. 用一条 SQL 语句 查询出每门课都大于 80 分的学生姓名

name	kecheng	fenshu
张三	语文	81
张三	数学	75
李四	语文	76
李四	数学	90
王五	语文	81
王五	数学	100
王五	英语	90

准备数据的 sql 代码：

```
create table score(id int primary key auto_increment,name varchar(20),subject varchar(20),score int);
insert into score values
(null,'张三','语文',81),
(null,'张三','数学',75),
(null,'李四','语文',76),
(null,'李四','数学',90),
(null,'王五','语文',81),
(null,'王五','数学',100),
(null,'王五 ','英语',90);
```

提示：当百思不得其解时，请理想思维，把小变成大做，把大变成小做，

答案：

A: select distinct name from score where name not in (select distinct name from score where score<=80)

B:select distince name t1 from score where 80< all (select score from score where name=t1);

176. 所有部门之间的比赛组合

一个叫 department 的表，里面只有一个字段 name,一共有 4 条纪录，分别是 a,b,c,d,对应四个球对，现在四个球对进行比赛，用一条 sql 语句显示所有可能的比赛组合.

答：select a.name, b.name from team a, team b where a.name < b.name

177. 每个月份的发生额都比 101 科目多的科目

请用 SQL 语句实现：从 TestDB 数据表中查询出所有月份的发生额都比 101 科目相应月份的发生额高的科目。请注意：TestDB 中有很多科目，都有 1—12 月份的发生额。
AccID：科目代码，Occmonth：发生额月份，DebitOccur：发生额。
数据库名：JcyAudit，数据集：Select * from TestDB
准备数据的 sql 代码：

```
drop table if exists TestDB;
create table TestDB(id int primary key auto_increment,AccID varchar(20), Occmonth date, DebitOccur bigint);
insert into TestDB values
(null,'101','1988-1-1',100),
(null,'101','1988-2-1',110),
(null,'101','1988-3-1',120),
(null,'101','1988-4-1',100),
(null,'101','1988-5-1',100),
(null,'101','1988-6-1',100),
(null,'101','1988-7-1',100),
(null,'101','1988-8-1',100);
--复制上面的数据，故意把第一个月份的发生额数字改小一点
insert into TestDB values
(null,'102','1988-1-1',90),
(null,'102','1988-2-1',110),
(null,'102','1988-3-1',120),
(null,'102','1988-4-1',100),
(null,'102','1988-5-1',100),
(null,'102','1988-6-1',100),
(null,'102','1988-7-1',100),
(null,'102','1988-8-1',100);
--复制最上面的数据，故意把所有发生额数字改大一点
insert into TestDB values
(null,'103','1988-1-1',150),
(null,'103','1988-2-1',160),
(null,'103','1988-3-1',180),
(null,'103','1988-4-1',120),
(null,'103','1988-5-1',120),
(null,'103','1988-6-1',120),
(null,'103','1988-7-1',120),
(null,'103','1988-8-1',120);
--复制最上面的数据，故意把所有发生额数字改大一点
insert into TestDB values
(null,'104','1988-1-1',130),
(null,'104','1988-2-1',130),
(null,'104','1988-3-1',140),
```

```
(null,'104','1988-4-1',150),
(null,'104','1988-5-1',160),
(null,'104','1988-6-1',170),
(null,'104','1988-7-1',180),
(null,'104','1988-8-1',140);
--复制最上面的数据，故意把第二个月份的发生额数字改小一点
insert into TestDB values
(null,'105','1988-1-1',100),
(null,'105','1988-2-1',80),
(null,'105','1988-3-1',120),
(null,'105','1988-4-1',100),
(null,'105','1988-5-1',100),
(null,'105','1988-6-1',100),
(null,'105','1988-7-1',100),
(null,'105','1988-8-1',100);
```

答案：

```
select distinct AccID from TestDB
where AccID not in
    (select TestDB.AccIDfrom TestDB,
        (select * from TestDB where AccID='101') as db101
    where TestDB.Occmonth=db101.Occmonth and TestDB.DebitOccur<=db101.DebitOccur
    );
```

178. 统计每年每月的信息

```
year  month amount
1991    1      1.1
1991    2      1.2
1991    3      1.3
1991    4      1.4
1992    1      2.1
1992    2      2.2
1992    3      2.3
1992    4      2.4
```

查成这样一个结果

```
year m1  m2  m3  m4
1991 1.1 1.2 1.3 1.4
1992 2.1 2.2 2.3 2.4
```

提示：这个与工资条非常类似，与学生的科目成绩也很相似。

准备 sql 语句：

```
drop table if exists sales;
create table sales(id int auto_increment primary key,year varchar(10), month varchar(10), amount
float(2,1));
insert into sales values
(null,'1991','1',1.1),
```

(null,'1991','2',1.2),
(null,'1991','3',1.3),
(null,'1991','4',1.4),
(null,'1992','1',2.1),
(null,'1992','2',2.2),
(null,'1992','3',2.3),
(null,'1992','4',2.4);

答案一、select sales.year ,
(select t.amount from sales t where t.month='1' and t.year= sales.year) '1',
(select t.amount from sales t where t.month='1' and t.year= sales.year) '2',
(select t.amount from sales t where t.month='1' and t.year= sales.year) '3',
(select t.amount from sales t where t.month='1' and t.year= sales.year) as '4'
from sales group by year;

179. 显示文章标题，发帖人、最后回复时间

表：id,title,postuser,postdate,parentid
准备 sql 语句：
drop table if exists articles;
create table articles(id int auto_increment primary key,title varchar(50), postuser varchar(10), postdate datetime,parentid int references articles(id));
insert into articles values
(null,'第一条','张三','1998-10-10 12:32:32',null),
(null,'第二条','张三','1998-10-10 12:34:32',null),
(null,'第一条回复 1','李四','1998-10-10 12:35:32',1),
(null,'第二条回复 1','李四','1998-10-10 12:36:32',2),
(null,'第一条回复 2','王五','1998-10-10 12:37:32',1),
(null,'第一条回复 3','李四','1998-10-10 12:38:32',1),
(null,'第二条回复 2','李四','1998-10-10 12:39:32',2),
(null,'第一条回复 4','王五','1998-10-10 12:39:40',1);

答案：
select a.title,a.postuser,
(select max(postdate) from articles where parentid=a.id) reply
from articles a where a.parentid is null;
注释：子查询可以用在选择列中，也可用于 where 的比较条件中，还可以用于 from 从句中。

180. 删除除了 id 号不同,其他都相同的学生冗余信息

2.学生表 如下:
id 号 学号 姓名 课程编号 课程名称 分数
1 2005001 张三 0001 数学 69
2 2005002 李四 0001 数学 89
3 2005001 张三 0001 数学 69
A: delete from tablename where id 号 not in(select min(id 号) from tablename group by 学号,姓名,课程编号,课程名称,分数)

实验：

```
create table student2(id int auto_increment primary key,code varchar(20),name varchar(20));
insert into student2 values(null,'2005001','张三'),(null,'2005002','李四'),(null,'2005001','张三');
```

//如下语句，mysql 报告错误，可能删除依赖后面统计语句，而删除又导致统计语句结果不一致。

```
delete from student2 where id not in(select min(id) from student2 group by name);
```

//但是，如下语句没有问题：

```
select *  from student2 where id not in(select min(id) from student2 group by name);
```

//于是，我想先把分组的结果做成虚表，然后从虚表中选出结果，最后再将结果作为删除的条件数据。

```
delete from student2 where id not in(select mid from (select min(id) mid
from student2 group by name) as t);
```

或者：

```
delete from student2 where id not in(select min(id) from (select * from s
tudent2) as t group by t.name);
```

181. 航空网的几个航班查询题：

表结构如下：

flight{flightID,StartCityID ,endCityID,StartTime}

city{cityID, CityName)

实验环境：

```
create table city(cityID int auto_increment primary key,cityName varchar(20));
create table flight (flightID int auto_increment primary key,
      StartCityID int references city(cityID),
      endCityID   int references city(cityID),
      StartTime timestamp);
```

//航班本来应该没有日期部分才好，但是下面的题目当中涉及到了日期

```
insert into city values(null,'北京'),(null,'上海'),(null,'广州');
insert into flight values
      (null,1,2,'9:37:23'),(null,1,3,'9:37:23'),(null,1,2,'10:37:23'),(null,2,3,'10:37:23');
```

1、查询起飞城市是北京的所有航班，按到达城市的名字排序

参与运算的列是我起码能够显示出来的那些列，但最终我不一定把它们显示出来。各个表组合出来的中间结果字段中必须包含所有运算的字段。

```
  select   * from flight f,city c
     where f.endcityid = c.cityid and startcityid =
     (select c1.cityid from city c1 where c1.cityname = "北京")
     order by c.cityname asc;
```

```
mysql> select flight.flightid,'北京' startcity, e.cityname from flight,city e wh
ere flight.endcityid=e.cityid and flight.startcityid=(select cityid from city wh
ere cityname='北京');
```

```
mysql> select flight.flightid,s.cityname,e.cityname from flight,city s,city e wh
ere flight.startcityid=s.cityid and s.cityname='北京' and flight.endCityId=e.cit
yID order by e.cityName desc;
```

2、查询北京到上海的所有航班纪录（起飞城市，到达城市，起飞时间，航班号）

```
select c1.CityName,c2.CityName,f.StartTime,f.flightID
from city c1,city c2,flight f
where f.StartCityID=c1.cityID
and f.endCityID=c2.cityID
and c1.cityName='北京'
and c2.cityName='上海'
```

3、查询具体某一天（2005-5-8）的北京到上海的的航班次数

```
select count(*) from
(select c1.CityName,c2.CityName,f.StartTime,f.flightID
from city c1,city c2,flight f
where f.StartCityID=c1.cityID
and f.endCityID=c2.cityID
and c1.cityName='北京'
and c2.cityName='上海'
and  查帮助获得的某个日期处理函数(startTime) like '2005-5-8%'
```

mysql 中提取日期部分进行比较的示例代码如下：

```
select * from flight where date_format(starttime,'%Y-%m-%d')='1998-01-02'
```

182. 查出比经理薪水还高的员工信息：

```
Drop table if not exists employees;
create table employees(id int primary key auto_increment,name varchar(50)
,salary int,managerid int references employees(id));
insert into employees values (null,' lhm',10000,null), (null,' zxx',15000,1
),(null,'flx',9000,1),(null,'tg',10000,2),(null,'wzg',10000,3);
```

Wzg 大于 flx,lhm 大于 zxx

解题思路：

根据 sql 语句的查询特点，是逐行进行运算，不可能两行同时参与运算。

涉及了员工薪水和经理薪水，所有，一行记录要同时包含两个薪水，所有想到要把这个表自关联组合一下。

首先要组合出一个包含有各个员工及该员工的经理信息的长记录，譬如，左半部分是员工，右半部分是经理。而迪卡尔积会组合出很多垃圾信息，先去除这些垃圾信息。

```
select e.* from employees e,employees m where e.managerid=m.id and e.sala
ry>m.salary;
```

183. 求出小于 45 岁的各个老师所带的大于 12 岁的学生人数

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
预备知识： 1.sql 语句是对每一条记录依次处理，条件为真则执行动作（select,insert,delete,update）
2.只要是迪卡尔积，就会产生“垃圾”信息，所以，只要迪卡尔积了，我们首先就要想到清除“垃圾”信息
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，统计信息放在中间表中：
select teaid,count(*) from tea_stu group by teaid;
2.接着其实应该是筛除掉小于 12 岁的学生，然后再进行统计，中间表必须与 student 关联才能得到 12 岁以下学生和把该学生记录从中间表中剔除，代码是：
select tea_stu.teaid,count(*) total from student,tea_stu
where student.stuid=tea_stu.stuid and student.age>12 group by tea_stu.teaid
3.接着把上面的结果做成虚表与 teacher 进行关联，并筛除大于 45 的老师
select teacher.teaid,teacher.name,total from teacher ,***(select tea_stu.tea***
id,count(*) total from student,tea_stu where student.stuid=tea_stu.stuid and stu
dent.age>12 group by tea_stu.teaid) as tea_stu2 where teacher.teaid=tea_stu2.tea
id and teacher.age<45;

184. 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);
select t.authorid,max(t.total) from（select authorid,count(*) total from articles ）as t

这条语句不行，因为 max 只有一列，不能与其他列混淆。
select authorid,count(*) total from articles
group by authorid having total=max(total)也不行。

185. 一个用户表中有一个积分字段，假如数据库中有 100 多万个用户，若要在每年第一天凌晨将积分清零，你将考虑什么，你将想什么办法解决?

alter table drop column score;
alter table add colunm score int;
可能会很快，但是需要试验，试验不能拿真实的环境来操刀，并且要注意，
这样的操作时无法回滚的，在我的印象中，只有 inert update delete 等 DML 语句才能回滚，
对于 create table,drop table ,alter table 等 DDL 语句是不能回滚。

解决方案一，update user set score=0;
解决方案二，假设上面的代码要执行好长时间，超出我们的容忍范围，那我就 alter table user drop column score;alter table user add column score int。

下面代码实现每年的那个凌晨时刻进行清零。

```
Runnable runnable =
      new Runnable(){
            public void run(){
                  clearDb();
                  schedule(this,new Date(new Date().getYear()+1,0,0));
                  }
            };
schedule(runnable,
      new Date(new Date().getYear()+1,0,1));
```

186. 一个用户具有多个角色，请查询出该表中具有该用户的所有角色的其他用户。

```
select count(*) as num,tb.id from tb,
  (select role from tb where id=xxx) as t1
where tb.role = t1.role and tb.id != t1.id
group by tb.id
having num = select count(role) from tb where id=xxx;
```

187. xxx 公司的 sql 面试

Table **EMPLOYEES** Structure:
EMPLOYEE_ID NUMBER Primary Key,
FIRST_NAME VARCHAR2(25),
LAST_NAME VARCHAR2(25),
Salary number(8,2),
HiredDate DATE,
Departmentid number(2)
Table **Departments** Structure:
Departmentid number(2) Primary Key,
DepartmentName VARCHAR2(25).

（2）基于上述 EMPLOYEES 表写出查询：写出雇用日期在今年的，或者工资在[1000,2000]之间的，或者员工姓名（last_name）以'Obama'打头的所有员工，列出这些员工的全部个人信息。（4 分）

```
select * from employees
where Year(hiredDate) = Year(date())
      or (salary between 1000 and 200)
      or left(last_name,3)='abc';
```

(3) 基于上述 EMPLOYEES 表写出查询：查出部门平均工资大于 1800 元的部门的所有员工，列出这些员工的全部个人信息。

```
mysql> select id,name,salary,deptid did from employee1 where (select avg(salary)
 from employee1 where deptid = did) > 1800;
```

(4) 基于上述 EMPLOYEES 表写出查询：查出个人工资高于其所在部门平均工资的员工，列出这些员工的全部个人信息及该员工工资高出部门平均工资百分比。（5 分）

```
select employee1.*,(employee1.salary-t.avgSalary)*100/employee1.salary
from employee1,
      (select deptid,avg(salary) avgSalary from employee1 group by deptid) as t
where employee1.deptid = t.deptid and employee1.salary>t.avgSalary;
```

188. 注册 Jdbc 驱动程序的三种方式?

如下:

```
1.          Class.forName("com.mysql.jdbc.Driver");
2.          System.setProperty("jdbc.drivers", "com.mysql.jdbc.Driver");
3.          DriverManager.registerDriver(new com.mysql.jdbc.Driver());
```

1 中，不需要.newInstance()
2 中，通过系统的属性设置即可
3 中，是看起来比较直观的一种方式，注册相应的 db 的 jdbc 驱动，

总结：推荐 1，和 2 两种方式。
原因：3 在编译时需要导入对应的 lib。1，2 不需要。

补充：2 的方式的话，可以同时导入多个 jdbc 驱动，中间用冒号“：”分开

189. 用 JDBC 如何调用存储过程

代码如下：

```
package com.huawei.interview.lym;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Types;
```

```
public class JdbcTest {
    public static void main(String[] args) {
        Connection cn = null;
        CallableStatement cstmt = null;
        try {
            //这里最好不要这么干，因为驱动名写死在程序中了
            Class.forName("com.mysql.jdbc.Driver");
            //实际项目中，这里应用 DataSource 数据，如果用框架，
            //这个数据源不需要我们编码创建，我们只需 Datasource ds = context.lookup()
            //cn = ds.getConnection();
            cn = DriverManager.getConnection("jdbc:mysql:///test","root","root");
            cstmt = cn.prepareCall("{call insert_Student(?,?,?)}");
            cstmt.registerOutParameter(3,Types.INTEGER);
            cstmt.setString(1, "wangwu");
            cstmt.setInt(2, 25);
            cstmt.execute();
            //get 第几个，不同的数据库不一样，建议不写
            System.out.println(cstmt.getString(3));
        } catch (Exception e) {
            e.printStackTrace();
        }
        finally{
            /*try{cstmt.close();}catch(Exception e){}
            try{cn.close();}catch(Exception e){}*/
            try {
                if(cstmt != null)
                    cstmt.close();
                if(cn != null)
                    cn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

190. JDBC 中的 PreparedStatement 相比 Statement 的好处

答：一个 sql 命令发给服务器去执行的步骤为：语法检查，语义分析，编译成内部指令，缓存指令，执行指令等过程。

select * from student where id =3----缓存--→xxxxx 二进制命令

select * from student where id =3----直接取-→xxxxx 二进制命令

select * from student where id =4--- -→会怎么干？

如果当初是 select * from student where id =?--- -→又会怎么干？

上面说的是性能提高

可以防止 sql 注入。

191. 写一个用 jdbc 连接并访问 oracle 数据的程序代码，能够完成修改和查询工作。

```
public void testJdbc(){
        Connection con = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        try{
            //step1：注册驱动；
            Class.forName("oracle.jdbc.driver.OracleDriver");
            //step 2：获取数据库连接；
            con=DriverManager.getConnection(
                    "jdbc:oracle:thin:@192.168.0.39:1521:TARENADB",
                    "sd0605","sd0605");
            /************************查 询************************/
            //step 3：创建 Statement；
            String sql = "SELECT id, fname, lname, age, FROM Person_Tbl";
            ps = con.prepareStatement(sql);
            //step 4 ：执行查询语句，获取结果集；
            rs = ps.executeQuery();
            //step 5：处理结果集—输出结果集中保存的查询结果；
            while (rs.next()){
                System.out.print("id = " + rs.getLong("id"));
                System.out.print(" , fname = " + rs.getString("fname"));
                System.out.print(" , lname = " + rs.getString("lname"));
                System.out.print(" , age = " + rs.getInt("age"));
            }
            /************************JDBC 修 改********************/
            sql = "UPDATE Person_Tbl SET age=23 WHERE id = ?";
            ps = con.prepareStatement(sql);
            ps.setLong(1, 88);
            int rows = ps.executeUpdate();
            System.out.println(rows + " rows affected.");
        } catch (Exception e){
            e.printStackTrace();
        } finally{
            try{
                con.close(); //关闭数据库连接，以释放资源。
            } catch (Exception e1) {
            }
        }
    }
```

192. Class.forName 的作用?为什么要用?

答：按参数中指定的字符串形式的类名去搜索并加载相应的类，如果该类字节码已经被加载过，则返回代表该字节码的 Class 实例对象，否则，按类加载器的委托机制去搜索和加载该

类，如果所有的类加载器都无法加载到该类，则抛出 ClassNotFoundException。加载完这个 Class 字节码后，接着就可以使用 Class 字节码的 newInstance 方法去创建该类的实例对象了。有时候，我们程序中所有使用的具体类名在设计时（即开发时）无法确定，只有程序运行时才能确定，这时候就需要使用 Class.forName 去动态加载该类，这个类名通常是在配置文件中配置的，例如，spring 的 ioc 中每次依赖注入的具体类就是这样配置的，jdbc 的驱动类名通常也是通过配置文件来配置的，以便在产品交付使用后不用修改源程序就可以更换驱动类名。

193. 大数据量下的分页解决方法。

答：最好的办法是利用 sql 语句进行分页，这样每次查询出的结果集中就只包含某页的数据内容。再 sql 语句无法实现分页的情况下，可以考虑对大的结果集通过游标定位方式来获取某页的数据。

sql 语句分页，不同的数据库下的分页方案各不一样，下面是主流的三种数据库的分页 sql：

sql server:

```
String sql = "select top " + pageSize + " * from students where id not in" +
  "(select top " + pageSize * (pageNumber-1) + " id from students order by id)" +
  "order by id";
```

mysql:String sql = "select * from students order by id limit " + pageSize*(pageNumber-1) + "," + pageSize;

oracle:String sql = "select * from " + (select *,rownum rid from (select * from students order by postime desc) where rid<=" + pagesize*pagenumber + ") as t" + "where t>" + pageSize*(pageNumber-1);

194. 用 JDBC 查询学生成绩单，把主要代码写出来（考试概率极大）.

```
Connection cn = null;
PreparedStatement pstmt =null;
Resultset rs = null;
try{
      Class.forname(driveClassName);
      cn =  DriverManager.getConnection(url,username,password);
      pstmt = cn.prepareStatement("select  score.* from score ,student " +
            "where score.stuId = student.id and student.name = ?");
      pstmt.setString(1,studentName);
      Resultset rs = pstmt.executeQuery();
      while(rs.next()){
            system.out.println(rs.getInt("subject")  +  "      " + rs.getFloat("score") );
      }
}catch(Exception e){e.printStackTrace();}
finally{
      if(rs != null) try{ rs.close() }catch(exception e){}
      if(pstmt != null) try{pstmt.close()}catch(exception e){}
      if(cn != null) try{ cn.close() }catch(exception e){}
}
```

195. 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) {
}
```

答：没有 finally 语句来关闭各个对象，另外，使用 finally 之后，要把变量的定义放在 try 语句块的外面，以便在 try 语句块之外的 finally 块中仍可以访问这些变量。

196. 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。
实现方式，返回的 Connection 是原始 Connection 的代理，代理 Connection 的 close 方法不是真正关连接，而是把它代理的 Connection 对象还回到连接池中。

197. 为什么要用 ORM? 和 JDBC 有何不一样?

orm 是一种思想，就是把 object 转变成数据库中的记录，或者把数据库中的记录转变成 objecdt，我们可以用 jdbc 来实现这种思想，其实，如果我们的项目是严格按照 oop 方式编写的话，我们的 jdbc 程序不管是有意还是无意，就已经在实现 orm 的工作了。
现在有许多 orm 工具，它们底层调用 jdbc 来实现了 orm 工作，我们直接使用这些工具，就省去了直接使用 jdbc 的繁琐细节，提高了开发效率，现在用的较多的 orm 工具是 hibernate。
也听说一些其他 orm 工具，如 toplink,ojb 等。

198. 121、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%>页
<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>
```

199. JDBC，Hibernate 分页怎样实现？

答：方法分别为：

1) Hibernate 的分页：

```
Query query = session.createQuery("from Student");
query.setFirstResult(firstResult);//设置每页开始的记录号
query.setMaxResults(resultNumber);//设置每页显示的记录数
Collection students = query.list();
```

2) JDBC 的分页：根据不同的数据库采用不同的 sql 分页语句

例如：Oracle 中的 sql 语句为："SELECT * FROM (SELECT a.*, rownum r FROM TB_STUDENT) WHERE r between 2 and 10" 查询从记录号 2 到记录号 10 之间的所有记录

七、 XML 部分

200. 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)

201. 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。

202. 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());
        }
```

```
    }
}
```

203. 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>
```

204. 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)

八、流行的框架与新技术

205. 谈谈你对 SSH 的理解

典型的 JavaEE 三层结构，分为表现层、中间层（业务逻辑层）和数据服务层。三层体系将业务规则、数据访问及合法性校验等工作放在中间层处理。客户端不直接与数据库交互，而是通过组件与中间层建立连接，再由中间层与数据库交互。
表现层是传统的 JSP 技术，自 1999 年问世以来，经过多年的发展，其广泛的应用和稳定的表现，为其作为表现层技术打下了坚实的基础。
中间层采用的是流行的 Spring+Hibernate，为了将控制层与业务逻辑层分离，又细分为以下几种。
Web 层，就是 MVC 模式里面的“C”（controller），负责控制业务逻辑层与表现层的交互，调用业务逻辑层，并将业务数据返回给表现层作组织表现，该系统的 MVC 框架采用 Struts。
Service 层（就是业务逻辑层），负责实现业务逻辑。业务逻辑层以 DAO 层为基础，通过对 DAO 组件的正面模式包装，完成系统所要求的业务逻辑。
DAO 层，负责与持久化对象交互。该层封装了数据的增、删、查、改的操作。
PO，持久化对象。通过实体关系映射工具将关系型数据库的数据映射成对象，很方便地实现以面向对象方式操作数据库，该系统采用 Hibernate 作为 ORM 框架。
Spring 的作用贯穿了整个中间层，将 Web 层、Service 层、DAO 层及 PO 无缝整合，其数据服务层用来存放数据。

206. 谈谈你对 Struts 的理解。

1. struts 是一个按 MVC 模式设计的 Web 层框架，其实它就是一个大大的 servlet，这个 Servlet 名为 ActionServlet，或是 ActionServlet 的子类。我们可以在 web.xml 文件中将符合某种特征的所有请求交给这个 Servlet 处理，这个 Servlet 再参照一个配置文件（通常为 /WEB-INF/struts-config.xml）将各个请求分别分配给不同的 action 去处理。
一个扩展知识点：struts 的配置文件可以有多个，可以按模块配置各自的配置文件，这样可以防止配置文件的过度膨胀；
2. ActionServlet 把请求交给 action 去处理之前，会将请求参数封装成一个 formbean 对象（就是一个 java 类，这个类中的每个属性对应一个请求参数），封装成一个什么样的 formbean 对象呢？看配置文件。
3.要说明的是， ActionServlet 把 formbean 对象传递给 action 的 execute 方法之前，可能会调用 formbean 的 validate 方法进行校验，只有校验通过后才将这个 formbean 对象传递给 action 的 execute 方法，否则，它将返回一个错误页面，这个错误页面由 input 属性指定，（看配置文件）作者为什么将这里命名为 input 属性，而不是 error 属性，我们后面结合实际的运行效果进行分析。
4.action 执行完后要返回显示的结果视图，这个结果视图是用一个 ActionForward 对象来表示的，actionforward 对象通过 struts-config.xml 配置文件中的配置关联到某个 jsp 页面，因为程序中使用的是在 struts-config.xml 配置文件为 jsp 页面设置的逻辑名，这样可以实现 action 程序代码与返回的 jsp 页面名称的解耦。
你对 struts 可能还有自己的应用方面的经验，那也要一并说出来。

207. 谈谈你对 Hibernate 的理解。

1. 面向对象设计的软件内部运行过程可以理解成就是在不断创建各种新对象、建立对象之间的关系，调用对象的方法来改变各个对象的状态和对象消亡的过程，不管程序运行的过程

和操作怎么样，本质上都是要得到一个结果，程序上一个时刻和下一个时刻的运行结果的差异就表现在内存中的对象状态发生了变化。

2.为了在关机和内存空间不够的状况下，保持程序的运行状态，需要将内存中的对象状态保存到持久化设备和从持久化设备中恢复出对象的状态，通常都是保存到关系数据库来保存大量对象信息。从 Java 程序的运行功能上来讲，保存对象状态的功能相比系统运行的其他功能来说，应该是一个很不起眼的附属功能，java 采用 jdbc 来实现这个功能，这个不起眼的功能却要编写大量的代码，而做的事情仅仅是保存对象和恢复对象，并且那些大量的 jdbc 代码并没有什么技术含量，基本上是采用一套例行公事的标准代码模板来编写，是一种苦活和重复性的工作。

3.通过数据库保存 java 程序运行时产生的对象和恢复对象，其实就是实现了 java 对象与关系数据库记录的映射关系，称为 ORM（即 Object Relation Mapping），人们可以通过封装 JDBC 代码来实现了这种功能，封装出来的产品称之为 ORM 框架，Hibernate 就是其中的一种流行 ORM 框架。使用 Hibernate 框架，不用写 JDBC 代码，仅仅是调用一个 save 方法，就可以将对象保存到关系数据库中，仅仅是调用一个 get 方法，就可以从数据库中加载出一个对象。

4.使用 Hibernate 的基本流程是：配置 Configuration 对象、产生 SessionFactory、创建 session 对象，启动事务，完成 CRUD 操作，提交事务，关闭 session。

5.使用 Hibernate 时，先要配置 hibernate.cfg.xml 文件，其中配置数据库连接信息和方言等，还要为每个实体配置相应的 hbm.xml 文件，hibernate.cfg.xml 文件中需要登记每个 hbm.xml 文件。

6.在应用 Hibernate 时，重点要了解 Session 的缓存原理，级联，延迟加载和 hql 查询。

208. AOP 的作用。

面向切面编程（AOP）提供另外一种角度来思考程序结构，通过这种方式弥补了面向对象编程（OOP）的不足

除了类（classes）以外，AOP 提供了切面。切面对关注点进行模块化，例如横切多个类型和对象的事务管理

Spring 的一个关键的组件就是 AOP 框架，可以自由选择是否使用 AOP 提供声明式企业服务，特别是为了替代 EJB 声明式服务。最重要的服务是声明性事务管理，这个服务建立在 Spring 的抽象事物管理之上

允许用户实现自定义切面，用 AOP 来完善 OOP 的使用.可以把 Spring AOP 看作是对 Spring 的一种增强

209. 你对 Spring 的理解。

1.Spring 实现了工厂模式的工厂类（在这里有必要解释清楚什么是工厂模式），这个类名为 BeanFactory（实际上是一个接口），在程序中通常 BeanFactory 的子类 ApplicationContext。Spring 相当于一个大的工厂类，在其配置文件中通过<bean>元素配置用于创建实例对象的类名和实例对象的属性。

2. Spring 提供了对 IOC 良好支持，IOC 是一种编程思想，是一种架构艺术，利用这种思想可以很好地实现模块之间的解耦。IOC 也称为 DI（Depency Injection），什么叫依赖注入呢？

譬如，Class Programmer{

```
    Computer computer = null;
    public void code(){
        //Computer computer = new IBMComputer();
        //Computer computer = beanfacotry.getComputer();
```

```
            computer.write();
        }
        public void setComputer(Computer computer){
            this.computer = computer;
        }
}
```

另外两种方式都由依赖，第一个直接依赖于目标类，第二个把依赖转移到工厂上，第三个彻底与目标和工厂解耦了。在 spring 的配置文件中配置片段如下：

```
<bean id=”computer” class=”cn.itcast.interview.Computer”>
</bean>
<bean id=”programmer” class=”cn.itcast.interview.Programmer”>
    <property name=”computer”    ref=”computer”></property>
</bean>
```

3. Spring 提供了对 AOP 技术的良好封装， AOP 称为面向切面编程，就是系统中有很多各不相干的类的方法，在这些众多方法中要加入某种系统功能的代码，例如，加入日志，加入权限判断，加入异常处理，这种应用称为 AOP。实现 AOP 功能采用的是代理技术，客户端程序不再调用目标，而调用代理类，代理类与目标类对外具有相同的方法声明，有两种方式可以实现相同的方法声明，一是实现相同的接口，二是作为目标的子类在，JDK 中采用 Proxy 类产生动态代理的方式为某个接口生成实现类，如果要为某个类生成子类，则可以用 CGLIB。在生成的代理类的方法中加入系统功能和调用目标类的相应方法，系统功能的代理以 Advice 对象进行提供，显然要创建出代理对象，至少需要目标类和 Advice 类。spring 提供了这种支持，只需要在 spring 配置文件中配置这两个元素即可实现代理和 aop 功能，例如，

```
<bean id=”proxy” type=”org.spring.framework.aop.ProxyBeanFactory”>
    <property name=”target” ref=””></property>
    <property name=”advisor” ref=””></property>
</bean>
```

210. 谈谈 Struts 中的 Action servlet。

211. 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，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:
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
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。
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。

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。

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。

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。

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。

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。

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.

212. 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。

213. 说说 struts1 与 struts2 的区别。

1.都是 MVC 的 WEB 框架,

2.struts1 的老牌框架，应用很广泛，有很好的群众基础，使用它开发风险很小，成本更低！struts2 虽然基于这个框架，但是应用群众并多，相对不成熟，未知的风险和变化很多，开发人员相对不好招，使用它开发项目的风险系数更大，用人成本更高！

3.struts2 毕竟是站在前辈的基础设计出来，它会改善和完善 struts1 中的一些缺陷，struts1 中一些悬而未决问题在 struts2 得到了解决。

4.struts1 的前端控制器是一个 Servlet，名称为 ActionServlet，struts2 的前端控制器是一个 filter，在 struts2.0 中叫 FilterDispatcher，在 struts2.1 中叫 StrutsPrepareAndExecuteFilter。

5.struts1 的 action 需要继承 Action 类，struts2 的 action 可以不继承任何类；struts1 对同一个路径的所有请求共享一个 Action 实例，struts2 对同一个路径的每个请求分别使用一个独立 Action 实例对象，所有对于 struts2 的 Action 不用考虑线程安全问题。

6.在 struts1 中使用 formbean 封装请求参数，在 struts2 中直接使用 action 的属性来封装请求参数。

7.struts1 中的多个业务方法放在一个 Action 中时（即继承 DispatchAction 时），要么都校验，要么都不校验；对于 struts2，可以指定只对某个方法进行校验，当一个 Action 继承了 ActionSupport 且在这个类中只编写了 validateXxx()方法，那么则只对 Xxx()方法进行校验。（一个请求来了的执行流程进行分析，struts2 是自动支持分模块开发，并可以不同模块设置不同的 url 前缀，这是通过 package 的 namespace 来实现的；struts2 是支持多种类型的视图；struts2 的视图地址可以是动态的，即视图的名称是支持变量方式的，举例，论坛发帖失败后回来还要传递 boardid。视图内容显示方面：它的标签用 ognl，要 el 强大很多，在国际化方

面支持分模块管理，两个模块用到同样的 key，对应不同的消息；）
与 Struts1 不同，Struts2 对用户的每一次请求都会创建一个 Action，所以 Struts2 中的 Action 是线程安全的。
我印象最深刻的是：struts 配置文件中的 redirect 视图的 url 不能接受参数，而 struts2 配置文件中的 redirect 视图可以接受参数。

214. hibernate 中的 update()和 saveOrUpdate()的区别，session 的 load()和 get()的区别。

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

216. Hibernate 的应用（Hibernate 的结构）？

答：//首先获得 SessionFactory 的对象
SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
//然后获得 session 的对象
Session session = sessionFactory.openSession();
//其次获得 Transaction 的对象
Transaction tx = session.beginTransaction();
//执行相关的数据库操作:增,删,改,查
session.save(user); //增加, user 是 User 类的对象
session.delete(user); //删除
session.update(user); //更新
Query query = session.createQuery(“from User”); //查询
List list = query.list();
//提交事务
tx.commit();
//如果有异常,我们还要作事务的回滚,恢复到操作之前
tx.rollback();
//最后还要关闭 session,释放资源
session.close()

217. 112、Hibernate 有哪 5 个核心接口？

答：Configuration 接口：配置 Hibernate，根据其启动 hibernate，创建 SessionFactory 对象；
SessionFactory 接口：初始化 Hibernate，充当数据存储源的代理，创建 session 对象，sessionFactory 是线程安全的，意味着它的同一个实例可以被应用的多个线程共享，是重量级、二级缓存；
Session 接口：负责保存、更新、删除、加载和查询对象，是线程不安全的，避免多个线程共享同一个 session，是轻量级、一级缓存；
Transaction 接口：管理事务；
Query 和 Criteria 接口：执行数据库的查询。

218. iBatis 与 Hibernate 有什么不同?

相同点：屏蔽 jdbc api 的底层访问细节，使用我们不用与 jdbc api 打交道，就可以访问数据。jdbc api 编程流程固定，还将 sql 语句与 java 代码混杂在了一起，经常需要拼凑 sql 语句，细节很繁琐。
ibatis 的好处：屏蔽 jdbc api 的底层访问细节；将 sql 语句与 java 代码进行分离;提供了将结

果集自动封装称为实体对象和对象的集合的功能，queryForList 返回对象集合，用 queryForObject 返回单个对象；提供了自动将实体对象的属性传递给 sql 语句的参数。
Hibernate 是一个全自动的 orm 映射工具，它可以自动生成 sql 语句,ibatis 需要我们自己在 xml 配置文件中写 sql 语句，hibernate 要比 ibatis 功能负责和强大很多。因为 hibernate 自动生成 sql 语句，我们无法控制该语句，我们就无法去写特定的高效率的 sql。对于一些不太复杂的 sql 查询，hibernate 可以很好帮我们完成，但是，对于特别复杂的查询，hibernate 就很难适应了，这时候用 ibatis 就是不错的选择，因为 ibatis 还是由我们自己写 sql 语句。

219. 写 Hibernate 的一对多和多对一双向关联的 orm 配置?

220. 什么是 ORM?

答：对象关系映射（Object—Relational Mapping，简称 ORM）是一种为了解决面向对象与面向关系数据库存在的互不匹配的现象的技术；简单的说，ORM 是通过使用描述对象和数据库之间映射的元数据，将 java 程序中的对象自动持久化到关系数据库中；本质上就是将数据从一种形式转换到另外一种形式。

221. hibernate 的 inverse 属性的作用?

解决方案一，按照 Object[]数据取出数据，然后自己组 bean
解决方案二，对每个表的 bean 写构造函数，比如表一要查出 field1,field2 两个字段，那么有一个构造函数就是 Bean(type1 filed1,type2
field2) ，然后在 hql 里面就可以直接生成这个 bean 了。

222. 在 DAO 中如何体现 DAO 设计模式?

解决方案一，按照 Object[]数据取出数据，然后自己组 bean
解决方案二，对每个表的 bean 写构造函数，比如表一要查出 field1,field2 两个字段，那么有一个构造函数就是 Bean(type1 filed1,type2
field2) ，然后在 hql 里面就可以直接生成这个 bean 了。

223. spring+Hibernate 中委托方案怎么配置?

解决方案一，按照 Object[]数据取出数据，然后自己组 bean
解决方案二，对每个表的 bean 写构造函数，比如表一要查出 field1,field2 两个字段，那么有一个构造函数就是 Bean(type1 filed1,type2
field2) ，然后在 hql 里面就可以直接生成这个 bean 了。

224. spring+Hibernate 中委托方案怎么配置?

解决方案一，按照 Object[]数据取出数据，然后自己组 bean
解决方案二，对每个表的 bean 写构造函数，比如表一要查出 field1,field2 两个字段，那么有一个构造函数就是 Bean(type1 filed1,type2
field2) ，然后在 hql 里面就可以直接生成这个 bean 了。

225. hibernate 进行多表查询每表中各取几个字段，就是说查询出的结果集没有一个实体类与之对应如何解决

解决方案一，按照 Object[]数据取出数据，然后自己组 bean
解决方案二，对每个表的 bean 写构造函数，比如表一要查出 field1,field2 两个字段，那么有一个构造函数就是 Bean(type1 filed1,type2
field2) ，然后在 hql 里面就可以直接生成这个 bean 了。

226. 介绍一下 Hibernate 的二级缓存

按照以下思路来回答：（1）首先说清楚什么是缓存，（2）再说有了 hibernate 的 Session 就是一级缓存，即有了一级缓存，为什么还要有二级缓存，（3）最后再说如何配置 Hibernate 的二级缓存。

（1)缓存就是把以前从数据库中查询出来和使用过的对象保存在内存中(一个数据结构中)，这个数据结构通常是类似于 Hashmap，当以后要使用某个对象时，先查询缓存中是否有这个对象，如果有则使用缓存中的对象，如果没有则去查询数据库，并将查询出来的对象保存在缓存中，以便下次使用。下面是缓存的伪代码：

引出 hibernate 的第二级缓存，用下面的伪代码分析了 Cache 的实现原理

```
Dao{
    hashmap map = new map();
    User getUser(integer id){
        User user = map.get(id)
        if(user == null){
            user = session.get(id);
            map.put(id,user);
        }
        return user;
    }
}

Dao{
    Cache cache = null
    setCache(Cache cache){
        this.cache = cache
    }

    User getUser(int id){
        if(cache!=null){
            User user = cache.get(id);
            if(user ==null){
                user = session.get(id);
                cache.put(id,user);
            }
            return user;
        }
        return session.get(id);
    }
}
```

（2）Hibernate 的 Session 就是一种缓存，我们通常将之称为 Hibernate 的一级缓存，当想使用 session 从数据库中查询出一个对象时，Session 也是先从自己内部查看是否存在这个对象，存在则直接返回，不存在才去访问数据库，并将查询的结果保存在自己内部。由于 Session 代表一次会话过程，一个 Session 与一个数据库连接相关连，所以 Session 最好不要长时间保持打开，通常仅用于一个事务当中，在事务结束时就应关闭。并且 Session 是线程不安全

的，被多个线程共享时容易出现问题。通常只有那种全局意义上的缓存才是真正的缓存应用，才有较大的缓存价值，因此，Hibernate 的 Session 这一级缓存的缓存作用并不明显，应用价值不大。Hibernate 的二级缓存就是要为 Hibernate 配置一种全局缓存，让多个线程和多个事务都可以共享这个缓存。我们希望的是一个人使用过，其他人也可以使用，session 没有这种效果。

（3）二级缓存是独立于 Hibernate 的软件部件，属于第三方的产品，多个厂商和组织都提供有缓存产品，例如，EHCache 和 OSCache 等等。在 Hibernate 中使用二级缓存，首先就要在 hibernate.cfg.xml 配置文件中配置使用哪个厂家的缓存产品，接着需要配置该缓存产品自己的配置文件，最后要配置 Hibernate 中的哪些实体对象要纳入到二级缓存的管理中。明白了二级缓存原理和有了这个思路后，很容易配置起 Hibernate 的二级缓存。扩展知识：一个 SessionFactory 可以关联一个二级缓存，也即一个二级缓存只能负责缓存一个数据库中的数据，当使用 Hibernate 的二级缓存后，注意不要有其他的应用或 SessionFactory 来更改当前数据库中的数据，这样缓存的数据就会与数据库中的实际数据不一致。

227. 解释 Spring 的依赖注入？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？

228. 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。

229. STRUTS 的工作流程！

第一：

在 web 应用启动时就会加载初始化 ActionServlet,ActionServlet 从 struts-config.xml 文件中读取配置信息,把它们存放到各种配置对象,例如:Action 的映射信息存放在 ActionMapping 对象中.当 ActionServlet 接收到一个客户请求时,将执行如下流程.

-(1)检索和用户请求匹配的 ActionMapping 实例,如果不存在,就返回请求路径无效信息;

-(2)如果 ActionForm 实例不存在,就创建一个 ActionForm 对象,把客户提交的

表单数据保存到 ActionForm 对象中;

-(3)根据配置信息决定是否需要表单验证.如果需要验证,就调用 ActionForm 的 validate()方法;

-(4)如果 ActionForm 的 validate()方法返回 null 或返回一个不包含 ActionMessage 的 ActuibErrors 对象,就表示表单验证成功;

-(5)ActionServlet 根据 ActionMapping 所包含的映射信息决定将请求转发给哪个 Action,如果相应的 Action 实例不存在,就先创建这个实例,然后调用 Action 的 execute()方法;

-(6)Action 的 execute()方法返回一个 ActionForward 对象,ActionServlet 再把客户请求转发给 ActionForward 对象指向的 JSP 组件;

-(7)ActionForward 对象指向 JSP 组件生成动态网页,返回给客户;

第二：

在 web 服务器启动的时候就会自动加载 ActionServlet，ActionServlet 从 struts-config.xml 文件中读取配置信息，把他们存放到各种配置对象.当 ActionServlet 接收到一个客户请求的时

候，将执行如下流程：

（1）检索和用户请求匹配的 ActionMapping 实例，如果不存在，就返回请求路径无效

（2）如果 ActionForm 实例不存在，就创建一个 ActionForm 对象，把客户提交的表单数据保存到 ActionForm 中

（3）根据配置信息决定是否需要表单验证，如果需要验证，就调用 ActionForm 的 validate()方法；

（4）在 ActionForm（）的方法返回一个 null 或者一个不包含 ActionMeassage 的 ActionError 对象，就表示表单验证通过

（5）ActionServlet 根据 ActionMapping 所包含的映射信息决定将请求转发个哪个 Action，如果相应的 Action 不存在，就创建这个实例，再调用其 execute（）方法

（6）Action 的 execute()方法返回一个 ActionForward 对象，ActionServlet 再把客户请求转发给 ActionForward 对象所指的 jsp 组建

（7）ActionForward 对象指向 jsp 组件生成的动态网页，返回给客户。

Struts 使用 Model 2 架构。Struts 的 ActionServlet 控制导航流。其他 Struts 类，比如 Action, 用来访问业务逻辑类。当 ActionServlet 从容器接收到一个请求，它使用 URI (或者路径“path”) 来决定那个 Action 将用来处理请求。一个 Action 可以校验输入，并且访问业务层以从数据库或其他数据服务中检索信息。为校验输入或者使用输入来更新数据库，Action 需要知道什么指被提交上来。并不是强制每个 Action 从请求中抓取这些值，而是由 ActionServlet 将输入绑定到 JavaBean 中。

输入 bean 是 Struts ActionForm c 类的子类。ActionServlet 通过查找请求的路径可以决定使用哪个 ActionForm，Action 也是通过同样的方法选取的。ActionForm 扩展 org.apache.struts.action.ActionForm 类。每个都必须以 HTTP 响应进行应答。通常, Struts Action 并不自行加工响应信息，而是将请求转发到其他资源，比如 JSP 页面。Struts 提供一个 ActionForward 类，用来将一个页面的路径存储为逻辑名称。当完成业务逻辑后，Action 选择并向 Servlet 返回一个 ActionForward。Servlet 然后使用存储在 ActionForward 对象中的路径来调用页面完成响应。

Struts 将这些细节都绑定在一个 ActionMapping 对象中。每个 ActionMapping 相对于一个特定的路径。当某个路径被请求时，Servlet 就查询 ActionMapping 对象。ActionMapping 对象告诉 servlet，哪个 Actions, ActionForms, 和 ActionForwards 将被使用。所有这些细节，关于 Action， ActionForm， ActionForward， ActionMapping，以及其他一些东西，都在 struts-config.xml 文件中定义。 ActionServlet 在启动时读取这个配置文件,并创建一个配置对象数据库。在运行时，Struts 应用根据文件创建的配置对象，而不是文件本身。

第三：

1. ActionServlet：Struts 的 ActionServlet 控制导航流。当 ActionServlet 从容器接到一个请求，它使用 URI（或者“path”）也决定哪个 Action 来处理请求。< Control Layout >

2. Action：用来访问业务逻辑类。一个 Action 可以校验输入，并且访问业务层以从数据库检索信息。为校验输入或者使用输入来更新数据库，Action 需要知道什么值被提交上来。它并不是强制每个 Action 都要从请求中抓取这些值，而是由 ActionServlet 将输入绑定到 JavaBean 中。< Model Layout >

3.ActionForm：输入 bean 是 Struts ActionServlet 类的子类。ActionServlet 通过查找请求的路径可以决定使用哪个 ActionForm（输入 Javabean ），Action 也是通过同样的方法选取的。ActionForm 扩展了 org.apache.struts.action.ActionForm 类。< Data >

4.ActionMapping：Struts 将这些细节绑定在一个 ActionMapping 对象中。每个 ActionMapping 相对于一个特定的路径。当某个路径被请求时，Servlet 就查询

ActionMapping 对象。ActionMapping 对象告诉 servlet，哪些个 Action，ActionForm，和 ActionForward 将被本次请求使用。

每个请求都必须以 HTTP 响应进行应答。通常，Struts Action 并不自行渲染响应信息，而是将请求转发到其他资源，比如 JSP 页面。Struts 提供一个 ActionForward 类，用于将一个页面的路径保存为逻辑名称。当完成业务逻辑后，Action 选择并向 Servlet 返回一个 ActionForward 。Servlet 然后使用保存在 ActionForward 对象中的路径来调用页面完成响应。

所有这些细节，关于 Action，ActionForm，ActionForward，ActionMapping，以及其它一些东西，都在 struts-config.xml 文件中定义。ActionServlet 在启动时读取这个配置文件，并创建一个配置对象数据库。在运行时，Struts 应用根据文件创建的配置对象，而不是文件本身。

230. spring 与 EJB 的区别！

九、软件工程与设计模式

231. 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。

232. 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。

233. 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。

十、j2ee 部分

234. 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。

235. 应用服务器与 WEB SERVER 的区别？

236. 应用服务器有那些？

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

237. 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。

238. 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。

239. 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。

240. 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

241. 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。

242. 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.

243. 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。

244. 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。

245. 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

246. 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。

247. 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

十一、 ejb 部分

248. EJB 基于哪些技术实现的？说出 SessionBean 和 EntityBean 区别，StatefulBean 和 StatelessBean 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， Stateful 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。

249. 简要讲一下 EJB 的 7 个 Transaction Level?

250. 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。

251. 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。

252. 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。

253. 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。

254. 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

255. 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。

十二、 webservice 部分

256. 名词解释：①WEB SERVICE ②JAXP ③JAXM ④SOAP ⑤UDDI ⑥WSDL。JSWDL 开发包的介绍。

Web Service：Web 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。
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。
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）。

257. 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。

十三、 Linux

258. 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。

十四、 问得稀里糊涂的题

259. 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

260. 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

十五、 其他

261. 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。

262. 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.

263. 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?
Administrator 10:34:20
1，堆和栈的区别，有一个 64k 的字符串，是放到堆上，还是放到栈上，为什么？
2，什么时候用到接口，什么时候用到抽象类，二者区别
3，有一个 100 万的数组，里边有两个市重复的，如何设计算法找到。
4，设计数据库时，n 维，如何设计。
例如[省份][城市][网吧]，这是三维关系，它的表也应该有三个，网吧有外键引用城市，城市有外键应用省份，这个规律就是下层的要有一外键去引用上层。

264. Java1.5 的新特性

一：Java 基础部分新特性：
自动拆箱
可变参数与 for each 循环
静态导入
二：IO 流部分：
printf 格式输出
Scanner
三：其他
泛型
枚举
Annotation

265. Java 部分试题

试题 1：编写一个程序，这个程序把一个整数数组中的每个元素用逗号连接成一个字符串，例如，根据内容为[1][2][3]的数组形成内容为"1,2,3"的字符串。
试题 2：请在一个类中编写一个方法，这个方法搜索一个字符数组中是否存在某个字符，如果存在，则返回这个字符在字符数组中第一次出现的位置（序号从 0 开始计算），否则，返回-1。要搜索的字符数组和字符都以参数形式传递给该方法，如果传入的数组为 null，应抛出 IllegalArgumentException 异常。在类的 main 方法中以各种可能出现的情况测试验证该方法编写得是否正确，例如，字符不存在，字符存在，传入的数组为 null 等。
试题 3：编写一个程序，它先将键盘上输入的一个字符串转换成十进制整数，然后打印出这个十进制整数对应的二进制形式。这个程序要考虑输入的字符串不能转换成一个十进制整数的情况，并对转换失败的原因要区分出是数字太大，还是其中包含有非数字字符的情况。

提示：*十进制数转二进制数的方式是用这个数除以 2，余数就是二进制数的最低位，接着再用得到的商作为被除数去除以 2，这次得到的余数就是次低位，如此循环，直到被除数为 0 为止。其实，只要明白了打印出一个十进制数的每一位的方式（不断除以 10，得到的余数就分别是个位，十位，百位），就很容易理解十进制数转二进制数的这种方式。*

试题 4：请用移位的方式打印出一个十进制整数的十六进制形式。提示：按每 4 个二进制位对整数进行移位和去高位处理，得到的结果就是十六进制数的一位，然后按下面三种方式之一（作为作业，要求每种方式都用到）计算出一个十六进制数值对应的十六进制形式：

（1）0-9 之间的数值直接加上字符'0'，9 以上的数值减去 10 以后再加上字符'A'

（2）定义一个数组，其中包含 0-F 这些字符，然后用要计算的数值作为数组的索引号，即可获得其对应的十六进制数据。

（3）Character.forDigit 静态方法可以将一个十六进制的数字转变成其对应的字符表示形式，例如，根据数值 15 返回字符'F'。

试题 5：编写一个程序，用于实现文件的备份，程序运行时的命令语法为：

java MyCopy (sourcefile) (destfile)

试题 6：请编写一个字符输入流的包装类，通过这个包装类对底层字符输入流进行包装，让程序通过这个包装类读取某个文本文件（例如，一个 java 源文件）时，能够在读取的每行前面都加上有行号和冒号。

试题 7：请参照《Java 就业培训教程》书第八章编写一个窗口程序，用户单击窗口上的“×”按纽时，能关闭该窗口。

试题 8：请结合我们的《javascript 网页开发》一书中介绍的正则表达式与 String.split 方法，从"http://www.it315.org/get.jsp?user=zxx&pass=123"这样的 URL 地址中提取出每个参数的名称和值。这里要注意在正则表达式中要对?进行转义处理。

试题 9：利用 Socket 套接字进行面向连接通信的编程。客户端读取本地文件并发送；服务器接收文件并保存到本地文件系统中。

试题 10：在 javascript 视频的第七讲的第一个片断，讲到了将一个保存有 ip 地址与地区对照关系的文本文件导入到数据库时，应该将其中的某些空格替换成逗号（，），即对于如下格式的文本文件内容：

起始 IP	结束 IP	地区
61.54.231.245	61.54.231.245	河南省安阳市 新世纪网吧
61.54.231.246	61.54.231.246	河南省安阳市 未知地区
61.54.231.9	61.54.231.247	河南省安阳市 红日网吧
61.54.231.248	61.54.231.248	河南省安阳市 安阳师范学院
61.54.231.249	61.54.231.249	河南省安阳市 黑蜘蛛网吧(师范学院附近)

应转换成下面的这种格式：

61.54.231.245，61.54.231.245 ，河南省安阳市 新世纪网吧

61.54.231.246，61.54.231.246，河南省安阳市 未知地区

61.54.231.9，61.54.231.247 ，河南省安阳市 红日网吧

61.54.231.248，61.54.231.248，河南省安阳市 安阳师范学院

61.54.231.249，61.54.231.249，河南省安阳市 黑蜘蛛网吧(师范学院附近)

在视频教程中，讲解了使用 UltraEdit 的正则表达式替换功能来完成上面的转换。从 jdk1.4 开始，java 语言中提供用于处理正则表达式的相关 API 类和方法，在 jdk 帮助文档中，查看 String 类的 replaceAll 方法，可以看到该方法就支持类似 UltraEdit 的正则表达式替换功能。

任务 1：阅读 String.replaceAll 方法的帮助，以及它提供的相关超链接，了解该方法的用法后，编写一个 java 程序来自动实现上面的正则表达式替换，将 **a.txt（下载）** 中的 IP 地址数

字后的空格替换成“,”号后，将替换结果保存到 b.txt 文件中。

任务 2：我们在实现 www.it315.org 网站中的 ip 地区查询系统时，使用的是类似如下的 sql 语法：

select 地区 from ip 表 where 用户 ip>起始 IP and 用户 ip<结束 ip

通过这条 sql 语句就可以查询出用户 ip 所对应的地区结果。由于用户 ip 与起始 ip 和结束 ip 的比较属于字符串比较，如果用户 ip 为 9.1.1.1，那么它与 61.54.231.245 比较的结果就是前者大于后者，因为用户 ip 的第一个字符“9”大于 61.54.231.245 中的第一个字符“6”。现在请你想出一种解决办法，让上面的 sql 语句能够返回正确结果。提示:将 9.1.1.1 变化成 009.001.001.001 后与 061.054.231.245 进行比较就可以了。

请按这种思路用正则表达式改进你的程序，即程序在把 a.txt 文件中的 IP 地址转换后保存到 b.txt 文件中时，能在每个不足 3 位的 IP 地址前补 0，以补齐 3 位。例如 61.5.23.115，这个 IP 地址保存到 b.txt 文件中的形式应为 061.005.023.115。

在源程序中，要对程序代码的功能进行注释说明，提交你编写的程序给我们时,请附带该程序的使用说明。

二、Javasript 试题部分

试题 1：请编写一个类似于如下形式的表单页面：

试题 2：请按下面内容编写一个页面，点页面里的“全选”时，能选中或清除上面的所有水果。

选择你喜欢的水果：

□苹果
□桔子
□香蕉
□葡萄
□桃子
□全选

试题 3：请登陆访问 http://www.it315.org/bbs/页面，这个页面左侧导航栏部分可以收缩、显示，请参照此页面编写一个也能把导航栏收缩、显示的页面。

三、JavaWEB 试题部分

试题 1：请设计一个 XML 格式的文件，该文件能表达出一个国家中的每个省及省长名字，每个省下面的每个市及市长名字，每个市下面的每个镇及镇长名字。

试题 2：请简述 HTTP1.0 和 HTTP1.1 的区别，GET 提交和 POST 提交方式的区别。

试题 3：请编写一个 Servlet 程序，这个程序能打印出 Javascript 试题部分试题 1 表单提交的数据。

Java、android、大数据、python、前端、iOS、PHP

更多开发、编程资源、源码、笔记加 QQ:208695827

播妞

扫一扫二维码，加我QQ。