## 1. Java基础部分

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

### 1、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。

### 2、Java有没有goto?

java中的保留字，现在没有在java中使用。

### 说说&和&&的区别。

### &&：短路 &逻辑

&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。

### 4、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;

}

}

}

### 5、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。

### 6、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。

### 7、char型变量中能不能存贮一个中文汉字?为什么?

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

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

### 8、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。

### 9、请设计一个一百亿的计算器

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

首先，计算机中用固定数量的几个字节来存储的数值，所以计算机中能够表示的数值是有一定的范围的，为了便于讲解和理解，我们先以byte 类型的整数为例，它用1个字节进行存储，表示的最大数值范围为-128到+127。-1在内存中对应的二进制数据为11111111，如果两个-1相加，不考虑Java运算时的类型提升，运算后会产生进位，二进制结果为1,11111110，由于进位后超过了byte类型的存储空间，所以进位部分被舍弃，即最终的结果为11111110，也就是-2，这正好利用溢位的方式实现了负数的运算。-128在内存中对应的二进制数据为10000000，如果两个-128相加，不考虑Java运算时的类型提升，运算后会产生进位，二进制结果为1,00000000，由于进位后超过了byte类型的存储空间，所以进位部分被舍弃，即最终的结果为00000000，也就是0，这样的结果显然不是我们期望的，这说明**计算机中的算术运算是会发生越界情况的，两个数值的运算结果不能超过计算机中的该类型的数值范围。**由于Java中涉及表达式运算时的类型自动提升，我们无法用byte类型来做演示这种问题和现象的实验，大家可以用下面一个使用整数做实验的例子程序体验一下：

int a = Integer.MAX\_VALUE;

int b = Integer.MAX\_VALUE;

int sum = a + b;

System.out.println(“a=”+a+”,b=”+b+”,sum=”+sum);

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

（）这个类内部有两个成员变量，一个表示符号，另一个用字节数组表示数值的二进制数

（）有一个构造方法，把一个包含有多位数值的字符串转换到内部的符号和字节数组中

（）提供加减乘除的功能

public class BigInteger{

int sign;

byte[] val;

public Biginteger(String val) {

sign = ;

val = ;

}

public BigInteger add(BigInteger other) {

}

public BigInteger subtract(BigInteger other) {

}

public BigInteger multiply(BigInteger other){

}

public BigInteger divide(BigInteger other){

}

}

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

### 10、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");

### 11、"=="和equals方法究竟有什么区别？

（单独把一个东西说清楚，然后再说清楚另一个，这样，它们的区别自然就出来了，混在一起说，则很难说清楚）

==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。

### 12、静态变量和实例变量的区别？

在语法定义上的区别：静态变量前要加static关键字，而实例变量前则不加。

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

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

public class VariantTest{

public static int staticVar = 0;

public int instanceVar = 0;

public VariantTest(){

staticVar++;`

instanceVar++;

System.out.println(“staticVar=” + staticVar + ”,instanceVar=” + instanceVar);

}

}

备注：这个解答除了说清楚两者的区别外，最后还用一个具体的应用例子来说明两者的差异，体现了自己有很好的解说问题和设

计案例的能力，思维敏捷，超过一般程序员，有写作能力！

### 13、是否可以从一个static方法内部发出对非static方法的调用？

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

### 14、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。

### 15、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。

### 16、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;

### 17、请说出作用域public，private，protected，以及不写时的区别

这四个作用域的可见范围如下表所示。

说明：如果在修饰的元素上面没有写任何访问修饰符，则表示friendly。

作用域 当前类 同一package 子孙类 其他package

public √ √ √ √

protected √ √ √ ×

friendly √ √ × ×

private √ × × ×

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

### 18、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，JVM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。

### 19、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。

### 20、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。

### 21、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。

### 22、面向对象的特征有哪些方面

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

面向对象的编程语言有封装、继承 、抽象、多态等4个主要的特征。

1封装：

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

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。

继承：

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

多态：

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

UserDao userDao = daofactory.getDao();

userDao.insertUser(user);

比喻：人吃饭，你看到的是左手，还是右手？

### 23、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。

### 24、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。如果的子类没有实现抽象父类中的所有抽象方法，那么子类也必须定义为abstract类型。

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

下面比较一下两者的语法区别：

1.抽象类可以有构造方法，接口中不能有构造方法。

2.抽象类中可以有普通成员变量，接口中没有普通成员变量

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

4. 抽象类中的抽象方法的访问类型可以是public，protected和（默认类型,虽然

eclipse下不报错，但应该也不行），但接口中的抽象方法只能是public类型的，并且默认即为public abstract类型。

5. 抽象类中可以包含静态方法，接口中不能包含静态方法

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

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

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

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

private native void open(String name) throws FileNotFoundException;

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。

### 26、什么是内部类？Static Nested Class 和 Inner Class的不同。

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();

public class Inner2 //在方法体内部定义的内部类

{

public method()

{

out\_x = 3;

}

}

Inner2 inner2 = new Inner2();

}

public class Inner1 //在方法体外面定义的内部类

{

}

}

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

Outer outer = new Outer();

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

在方法内部定义的内部类前面不能有访问类型修饰符，就好像方法中定义的局部变量一样，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 Runable(){

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

### 27、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。

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

可以继承其他类或实现其他接口。不仅是可以，而是必须!

### 29、super.getClass()方法调用

下面程序的输出结果是多少？

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();

### 30、String是最基本的数据类型吗?

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

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

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

没有。因为String被设计成不可变(immutable)类，所以它的所有对象都是不可变对象。在这段代码中，s原先指向一个String对象，内容是 "Hello"，然后我们对s进行了+操作，那么s所指向的那个对象是否发生了改变呢？答案是没有。这时，s不指向原来那个对象了，而指向了另一个 String对象，内容为"Hello world!"，原来那个对象还存在于内存之中，只是s这个引用变量不再指向它了。

通过上面的说明，我们很容易导出另一个结论，如果经常对字符串进行各种各样的修改，或者说，不可预见的修改，那么使用String来代表字符串的话会引起很大的内存开销。因为 String对象建立之后不能再改变，所以对于每一个不同的字符串，都需要一个String对象来表示。这时，应该考虑使用StringBuffer类，它允许修改，而不是每个不同的字符串都要生成一个新的对象。并且，这两种类的对象转换十分容易。  
同时，我们还可以知道，如果要使用内容相同的字符串，不必每次都new一个String。例如我们要在构造器中对一个名叫s的String引用变量进行初始化，把它设置为初始值，应当这样做：  
public class Demo {  
private String s;  
...  
public Demo {  
s = "Initial Value";  
}  
...  
}  
而非  
s = new String("Initial Value");  
后者每次都会调用构造器，生成新对象，性能低下且内存开销大，并且没有意义，因为String对象不可改变，所以对于内容相同的字符串，只要一个String对象来表示就可以了。也就说，多次调用上面的构造器创建多个对象，他们的String类型属性s都指向同一个对象。  
上面的结论还基于这样一个事实：对于字符串常量，如果内容相同，Java认为它们代表同一个String对象。而用关键字new调用构造器，总是会创建一个新的对象，无论内容是否相同。  
至于为什么要把String类设计成不可变类，是它的用途决定的。其实不只String，很多Java标准类库中的类都是不可变的。在开发一个系统的时候，我们有时候也需要设计不可变类，来传递一组相关的值，这也是面向对象思想的体现。不可变类有一些优点，比如因为它的对象是只读的，所以多线程并发访问也不会有任何问题。当然也有一些缺点，比如每个不同的状态都要一个对象来代表，可能会造成性能上的问题。所以Java标准类库还提供了一个可变版本，即 StringBuffer。

### 32、是否可以继承String类?

String类是final类故不可以继承。

### 33、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。

### \*\*34、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。

### 35、如何把一段逗号分割的字符串转换成一个数组?

如果不查jdk api，我很难写出来！我可以说说我的思路：

1. 用正则表达式，代码大概为：String [] result = orgStr.split(“,”);
2. 用 StingTokenizer ,代码为：StringTokenizer tokener = StringTokenizer(orgStr,”,”);

String [] result = new String[tokener .[countTokens](mk:@MSITStore:C:\\Documents%20and%20Settings\\Administrator\\桌面\\jdk150.chm::/jdk150/api/java/util/StringTokenizer.html" \l "countTokens())()];

Int i=0;

while(tokener.hasNext(){result[i++]=toker.nextToken();}

### 36、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。