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。

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 = newObject();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。

代码 if(username.equals(“zxx”){}有什么不妥？

直接调用equals方法可能会抛出NullPointerException异常。为了防止程序抛出异常在调用equals方法之前要先判断一下是否为空。if( username!=null && username.equals(“zxx”){}这样写才完整。

int x = 1; return x==1?true:false;可以直接返回return x==1;本身就是boolean

不推荐使用””.equals(name),如果name==null

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，吗等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：  
  
classPerson{  
  
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(abstract修饰的类为抽象类,抽象类是不能被直接实例化的.  
抽象构造函数?  
头一次听到这个名词,不知道你是在哪里听说的,可能你把抽象类的构造函数认为成抽象构造函数吧.  
构造函数是什么,类实例化时的构造过程.抽象类可以有构造函数.  
抽象静态方法?  
这个词也是头一次听到,这个词也是你创造的吧.  
你充分认识抽象方法,和静态方法之后,就会认为根本不会存在抽象静态方法.  
抽象方法是干什么的?专用于继承类来实现的.  
静态方法是什么?  
大家知道对象有两种,一种是实例对象,一种是静态对象  
一个类可以有多个实例对象,但只有一个静态对象.也就是类本身  
类是可以被继承的.但他的静态对象只有一个,不能被继承,他的静态方法就更不可能被继承.  
所以,抽象方法是需要继承来实现,而静态方法根本不能被继承,两者互相矛盾.  
所以,根本就不存在抽象静态方法. 抽象类是用不到构造方法的，因为你new不出来抽象类的对象。子类要实现，static与子类没关系)。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 classBaseServlet 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 extendsBaseServlet  
  
{  
  
protected voiddoService(HttpServletRequest request, HttpServletResponse response) throwsIOExcetion,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(Stringname) 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。

29、super.getClass()方法调用  
  
下面程序的输出结果是多少？  
  
importjava.util.Date;  
  
public classTestextends Date{  
  
public static voidmain(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。

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(newString(“abc”)的结果为true,而StringBuffer没有实现equals方法，所以，new StringBuffer(“abc”).equals(newStringBuffer(“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()];  
  
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。

37、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。

41、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。

48、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。