下面有关JVM内存：

1）程序计数器是一个比较小的内存区域，用于指示当前线程所执行的字节码执行到了第几行，是线程隔离的

2）错误：方法区用于存储JVM加载的类信息、常量、静态变量、即使编译器编译后的代码等数据，是线程隔离的

3）原则上讲，所有的对象都在堆区上分配内存，是线程之间共享的

4）虚拟机栈描述的是Java方法执行的内存模型，用于存储局部变量，操作数栈，动态链接，方法出口等信息，是线程隔离的

JVM将内存区划分5个：

方法区(Method Area)—线性共享

堆(Heap)—线性共享

程序计数器(Propram Counter Register)—非线性共享

虚拟机栈/java方法栈(VM Stack)—非线性共享

本地方法栈(Native Method Stack)—非线性共享

方法区域存放了所加载的类的信息（名称、修饰符等）、类中的静态变量、类中定义为final类型的常量、类中的Field信息、类中的方法信息

有关servlet中init,service,destroy方法描述:

1）init()方法是servlet生命的起点。一旦加载了某个servlet，服务器将立即调用它的init()方法

2）service()方法处理客户机发出的所有请求

3）destroy()方法标志servlet生命周期的结束

4）错误：servlet在多线程下使用了同步机制，因此，在并发编程下servlet是线程安全的

servlet在多线程下其本身并不是线程安全的。

如果在类中定义成员变量，而在service中根据不同的线程对该成员变量进行更改，那么在并发的时候就会引起错误。

由于方法中的局部变量是在栈中，彼此各自都拥有独立的运行空间而不会互相干扰，因此才做到线程安全。

init方法： 是在servlet实例创建时调用的方法，用于创建或打开任何与servlet相的资源和初始 化servlet的状态，Servlet规范保证调用init方法前不会处理任何请求

 service方法：是servlet真正处理客户端传过来的请求的方法，由web容器调用， 根据HTTP请求方法（GET、POST等），将请求分发到doGet、doPost等方法

destory方法：是在servlet实例被销毁时由web容器调用。Servlet规范确保在destroy方法调用之 前所有请求的处理均完成，需要覆盖destroy方法的情况：释放任何在init方法中 打开的与servlet相关的资源存储servlet的状态

Servlet是线程不安全的，在Servlet类中可能会定义共享的类变量，这样在并发的多线程访问的情况下，不同的线程对成员变量的修改会引发错误。

Struts工作原理   
MVC即Model-View-Controller的缩写，是一种常用的设计模式。MVC   减弱了业务逻辑接口和数据接口之间的耦合，以及让视图层更富于变化。   
Struts   是MVC的一种实现，它将   Servlet和   JSP   标记（属于   J2EE   规范）用作实现的一部分。Struts继承了MVC的各项特性，并根据J2EE的特点，做了相应的变化与扩展。   
控 制：有一个XML文件Struts-config.xml，与之相关联的是Controller，在Struts中，承担MVC中Controller角 色的是一个Servlet，叫ActionServlet。ActionServlet是一个通用的控制组件。这个控制组件提供了处理所有发送到 Struts的HTTP请求的入口点。它截取和分发这些请求到相应的动作类（这些动作类都是Action类的子类）。另外控制组件也负责用相应的请求参数 填充   Action   From（通常称之为FromBean）,并传给动作类（通常称之为ActionBean）。动作类实现核心商业逻辑，它可以访问java   bean   或调用EJB。最后动作类把控制权传给后续的JSP   文件，后者生成视图。所有这些控制逻辑利用Struts-config.xml文件来配置。   
视图：主要由JSP生成页面完成视图，Struts提供丰富的JSP   标签库：   Html，Bean，Logic，Template等，这有利于分开表现逻辑和程序逻辑。   
模 型：模型以一个或多个java   bean的形式存在。这些bean分为三类：Action   Form、Action、JavaBean   or   EJB。Action   Form通常称之为FormBean，封装了来自于Client的用户请求信息，如表单信息。Action通常称之为ActionBean，获取从 ActionSevlet传来的FormBean，取出FormBean中的相关信息，并做出相关的处理，一般是调用Java   Bean或EJB等。   
流程：在Struts中，用户的请求一般以\*.do作为请求服务名，所有的\*.do请求均被指向 ActionSevlet，ActionSevlet根据Struts-config.xml中的配置信息，将用户请求封装成一个指定名称的 FormBean，并将此FormBean传至指定名称的ActionBean，由ActionBean完成相应的业务操作，如文件操作，数据库操作等。 每一个\*.do均有对应的FormBean名称和ActionBean名称，这些在Struts-config.xml中配置。   
核心：Struts的核心是ActionSevlet，ActionSevlet的核心是Struts-config.xml。

jvm中垃圾回收分为scanvenge gc和full GC，其中full GC触发的条件可能有哪些：

答：老年代满、持久代满、System.gc()

1，新生代：（1）所有对象创建在新生代的Eden区，当Eden区满后触发新生代的Minor GC，将Eden区和非空闲Survivor区存活的对象复制到另外一个空闲的Survivor区中。（2）保证一个Survivor区是空的，新生代Minor GC就是在两个Survivor区之间相互复制存活对象，直到Survivor区满为止。  
2，老年代：当Survivor区也满了之后就通过Minor GC将对象复制到老年代。老年代也满了的话，就将触发Full GC，针对整个堆（包括新生代、老年代、持久代）进行垃圾回收。  
3，持久代：持久代如果满了，将触发Full GC。

public class Test {

public static void main(String[] args) {

    StringBuffer a = new StringBuffer("A");

    StringBuffer b = new StringBuffer("B");

    operator(a, b);

    System.out.println(a + "," + b);

}

public static void operator(StringBuffer x, StringBuffer y) {

    x.append(y); y = x;

}

}

输出：AB, B

a,b是对象的引用，指向堆内存，将a，b两个引用传给x，y，执行x.append(y)，改变了x引用指向的堆内存的存储内容，变为AB， y = x，表示引用y,指向引用x指向的存储区域，没有改变引用b，指向的存储空间的内容。

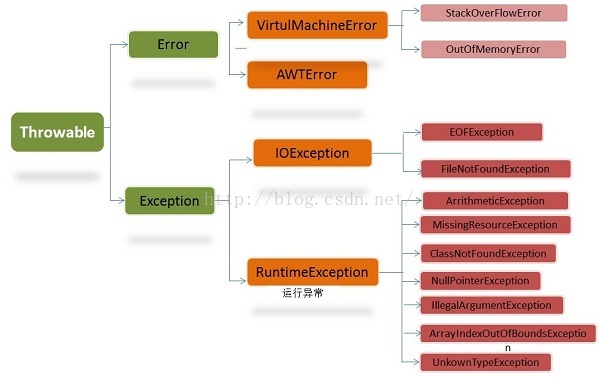
a.实例方法可以调用超类**公有**实例方法

b.实例方法可以直接调用超类的**公有**类方法

c. 实例方法可直接调用本类的类方法

d.实例方法可以通过类名调用其他类的类方法

哪个类来把基本类型数据转换为对象? 答：包装



可查异常（编译器要求必须处置的异常）：除了RuntimeException及其子类以外，其他的Exception类及其子类都属于可查异常。这种异常的特点是Java编译器会检查它，也就是说，当程序中可能出现这类异常，要么用try-catch语句捕获它，要么用throws子句声明抛出它，否则编译不会通过。

不可查异常(编译器不要求强制处置的异常):包括运行时异常（RuntimeException与其子类）和错误（Error）。

java虚拟机功能：

通过 ClassLoader 寻找和装载 class 文件

解释字节码成为指令并执行，提供 class 文件的运行环境

进行运行期间垃圾回收

提供与硬件交互的平台

byte i=128 //错，byte是1字节

long i=0xfffL //对

double i=0.9239d //对

java 中的 wait()方法和 sleep()方法的区别：

wait()方法属于Object类，二sleep()属于Thread类

调用wait()方法的时候，线程会放弃对象锁

调用sleep()方法的过程中，线程不会释放对象锁

sleep()方法导致了程序暂停执行指定的时间，让出cpu给其他线程 //错，是线程暂停执行指定时间而不是程序暂停执行指定的时间。

在java的多态调用中，new的是哪一个类就是调用的哪个类的方法。 //错，子类继承父类，但是没有实现其中的方法，当进行向上转型赋值时，调用的方法就只是父类的。

java 的字符类型采用的是 Unicode 编码方案，每个 Unicode 码占用（16）个比特位。

Java运行时的数据区包括：（其中前两个是线程共享的）  
1.方法区（Method Area） 存储已被虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等数据  
2.堆（Heap） 存放对象实例，几乎所有对象实例都在这里分配内存  
3.虚拟机栈（VM Stack） 描述的是Java方法执行的内存模型：每个方法在执行的同时会创建一个Stack Frame（方法运行时的基础数据结构）用于存储局部变量表、操作数栈、动态连接、方法出口等信息  
4.本地方法栈（Native Method Stack）  与虚拟机栈了类似，不过则为虚拟机使用的到的Native方法服务。（有的虚拟机譬如Sun HotSpot虚拟机直接把本地方法栈和虚拟机栈合二为一）  
5.程序计数器（Program Counter Register） 可看作当前线程所执行的字节码的行号的标识器

方法通常存储在进程中的（代码区）

关于ThreadLocal类 描述：

ThreadLocal是采用哈希表的方式来为每个线程都提供一个变量的副本

ThreadLocal保证各个线程间数据安全，每个线程的数据不会被另外线程访问和破坏

ThreadLocal继承Object，相当于没继承任何特殊的。ThreadLocal没有实现任何接口。

ThreadLocal并不是一个Thread，而是Thread的局部变量。

Synchronized用于线程间的数据共享，而ThreadLocal则用于线程间的数据隔离。

String str1 = “abc”，“abc”分配在内存哪个区域？ 答：字符串常量区

用new创建的对象在堆区

函数中的临时变量在栈去

java中的字符串在字符串常量区

栈：存放基本类型的数据和对象的引用，但对象本身不存放在栈中，而是存放在堆中

堆：存放用new产生的数据

静态域：存放在对象中用static定义的静态成员

常量池：存放常量

非RAM存储：硬盘等永久存储空间

instanceof运算符能够用来判断一个对象是否为: 一个类的实例、一个实现指定接口的类的实例、一个子类的实例

**instance是java的二元运算符，用来判断他左边的对象是否为右面类（接口，抽象类，父类）的实例**

byte b1=1,b2=2,b3,b6,b8;

final byte b4=4,b5=6,b7;

b3=(b1+b2);  /\*语句1\*/

b6=b4+b5;    /\*语句2\*/

b8=(b1+b4);  /\*语句3\*/

b7=(b2+b5);  /\*语句4\*/

System.out.println(b3+b6);

错误的语句：语句1、语句3、语句4

Java表达式转型规则**由低到高转换**：

1、所有的byte,short,char型的值将被提升为int型；

2、如果有一个操作数是long型，计算结果是long型；

3、如果有一个操作数是float型，计算结果是float型；

4、如果有一个操作数是double型，计算结果是double型；

5、被fianl修饰的变量不会自动改变类型，当2个final修饰相操作时，结果会根据左边变量的类型而转化。

--------------解析--------------

语句1错误：b3=(b1+b2);自动转为int，所以正确写法为b3=(byte)(b1+b2);或者将b3定义为int；

语句2正确：b6=b4+b5;b4、b5为final类型，不会自动提升，所以和的类型视左边变量类型而定，即b6可以是任意数值类型；

语句3错误：b8=(b1+b4);虽然b4不会自动提升，但b1仍会自动提升，所以结果需要强转，b8=(byte)(b1+b4);

语句4错误：b7=(b2+b5); 同上。同时注意b7是**final修饰，即只可赋值一次，便不可再改变**。