# 概念知识

# 1、什么是java虚拟机？为什么Java被称作是“平台无关的编程语言”？

Java虚拟机是可以运行java字节码文件的虚拟进程。Java虚拟机可以将java源程序编译为java字节码文件。

链接：<https://www.nowcoder.com/questionTerminal/a90230b35b5f4a7287f779ecdd88841d>  
来源：牛客网  
  
java源程序先经过javac编译器编译成二进制的.class字节码文件（java的跨平台指的就是.class字节码文件的跨平台，.class字节码文件是与平台无关的），.class文件再运行在jvm上，java解释器（jvm的一部分）会将其解释成对应平台的机器码执行，所以java所谓的跨平台就是在不同平台上安装了不同的jvm，而在不同平台上生成的.class文件都是一样的，而.class文件再由对应平台的jvm解释成对应平台的机器码执行。 最后解释下机器码和字节码的区别: 一，机器码，完全依附硬件而存在～并且不同硬件由于内嵌指令集不同，即使相同的0 1代码 意思也可能是不同的～换句话说，根本不存在跨平台性～比如～不同型号的CPU,你给他个指令10001101，他们可能会解析为不同的结果～ 二，我们知道JAVA是跨平台的，为什么呢？因为他有一个jvm,不论哪种硬件，只要你装有jvm,那么他就认识这个JAVA字节码～～～～至于底层的机器码，咱不用管，有jvm搞定，他会把字节码再翻译成所在机器认识的机器码～～～

# 2、JDK和JRE的区别是什么？

链接：<https://www.nowcoder.com/questionTerminal/36bd480a52d542298a8ca2056021f774>  
来源：牛客网  
JRE： Java Runtime Environment    
JDK：Java Development Kit  
JRE顾名思义是java运行时环境，包含了java虚拟机，java基础类库。是使用java语言编写的程序运行所需要的软件环境，是提供给想运行java程序的用户使用的。  
JDK顾名思义是java开发工具包，是程序员使用java语言编写java程序所需的开发工具包，是提供给程序员使用的。JDK包含了JRE，同时还包含了编译java源码的编译器javac，还包含了很多java程序调试和分析的工具：jconsole，jvisualvm等工具软件，还包含了java程序编写所需的文档和demo例子程序。  
如果你需要运行java程序，只需安装JRE就可以了。如果你需要编写java程序，需要安装JDK。

JRE根据不同操作系统（如：windows，linux等）和不同JRE提供商（IBM,ORACLE等）有很多版本

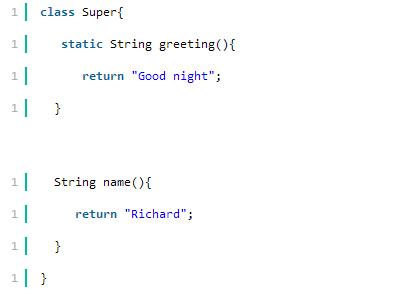
Jre：java runtime envirnment java运行时环境，主要提供java程序的运行。

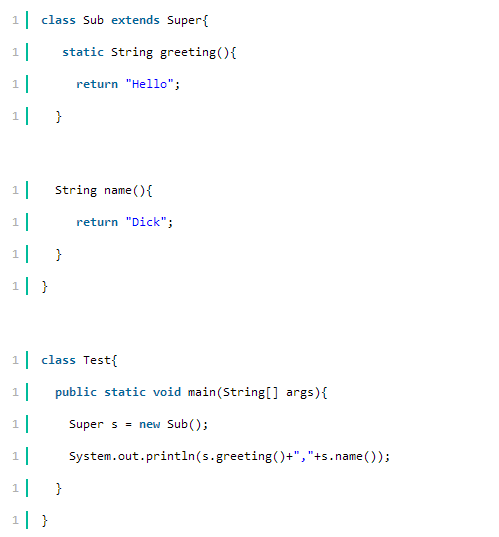
Jdk：java development kit ，java 开发工具包，主要是用于开发java程序的开发工具包。

# 3、”static”关键字是什么意思？Java中是否可以覆盖(override)一个private或者是static的方法？

链接：<https://www.nowcoder.com/questionTerminal/6f66f55a1f844e21b580e73d36223cf7>  
来源：牛客网  
  
Static表示静态的意思，可用于修饰成员变量和成员函数，被静态修饰的成员函数只能访问静态成员，不可以访问非静态成员。静态是随着类的加载而加载的，因此可以直接用类进行访问。 重写是子类中的方法和子类继承的父类中的方法一样（函数名，参数，参数类型，反回值类型），但是子类中的访问权限要不低于父类中的访问权限。重写的前提是必须要继承，private修饰不支持继承，因此被私有的方法不可以被重写。静态方法形式上可以被重写，即子类中可以重写父类中静态的方法。但是实际上从内存的角度上静态方法不可以被重写。

链接：<https://www.nowcoder.com/questionTerminal/6f66f55a1f844e21b580e73d36223cf7>  
来源：牛客网





运行结果：//~~Good night,Dick   
这个例子说明“实例方法被覆盖，静态方法被隐藏”

Static 是静态的意思，可以用于修饰成员变量和成员方法。被修饰的成员变量和成员方法，可以在类实例未创建的情况下访问。

对于覆盖（override）也就是重写，发生在子类继承父类的情况下，对于private和static 都不能被重写，因为重写是运行时绑定，static是静态绑定的，而且重新要求子类中的访问权限不能低于父类中的访问权限，所以private 修饰的方法也不能被重新，private修饰的方法只能在本类中访问。

# 4、是否可以在static环境中访问非static变量？

因为静态的成员属于类，随着类的加载而加载到静态方法区内存，当类加载时，此时不一定有实例创建，没有实例，就不可以访问非静态的成员。

但是可以不直接调用啊，我在static方法中创建实例不就可以调用了？比如说main函数就是static的，肯定可以new实例调方法啊

Static不能访问非静态的变量，因为静态成员属于类，随着类加载而加载到静态方法区内存，当类加载时，此时不一定有实例被创建，没有实例，就不能访问非静态的成员。但是可以在静态方法中创建实例对象，进行对非静态成员的访问。

# 5、Java支持的数据类型有哪些？什么是自动拆装箱？

链接：<https://www.nowcoder.com/questionTerminal/a5eb239111024d1b90cf00f24cfee36f>  
来源：牛客网

基本数据类型：

整数值型：byte,short,int,long,

字符型：char

浮点类型：float,double

布尔型：boolean

整数默认int型，小数默认是double型。Float和long类型的必须加后缀。

首先知道String是引用类型不是基本类型，引用类型声明的变量是指该变量在内存中实际存储的是一个引用地址，实体在堆中。引用类型包括类、接口、数组等。String类还是final修饰的。

 而包装类就属于引用类型，自动装箱和拆箱就是基本类型和引用类型之间的转换，至于为什么要转换，因为基本类型转换为引用类型后，就可以new对象，从而调用包装类中封装好的方法进行基本类型之间的转换或者toString（当然用类名直接调用也可以，便于一眼看出该方法是静态的），还有就是如果集合中想存放基本类型，泛型的限定类型只能是对应的包装类型。

基本数据类型有八种:

整数类型 : byte short int long

浮点类型：float double

布尔类型: boolean

字符类型：char

自动装箱和拆箱,装箱是将基本数据类型转化为引用类型，将基本数据转化成对象。转化为对象可以方便的操作数据。利用面向对象的方法方便的实现对数据的操作。拆箱就是将引用类型转化为基本数据类型。

# 6、J2EE中常用的名词解释

J2EE中常用的名词解释

1.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直接和容器中的环境变量接接口互，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。

2.Web container：实现J2EE体系结构中Web组件协议的容器。这个协议规定了一个Web组件运行时的环境，包括安全，一致性，生命周期管理，事务，配置和其它的服务。一个提供和JSP和J2EE平台APIs界面相同服务的容器。一个Web container 由Web服务器或者J2EE服务器提供。

3.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。一个实现了J2EE体系结构中EJB组件规范的容器。 这个规范指定了一个Enterprise bean的运行时环境，包括安全，一致性，生命周期，事务， 配置，和其他的服务。

4.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。

5.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。

6.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。

7.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。

8.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。RMI-IIOP出现以前，只有RMI和CORBA两种选择来进行分布式程序设计。RMI-IIOP综合了RMI和CORBA的优点，克服了他们的缺点，使得程序员能更方便的编写分布式程序设计，实现分布式计算。首先，RMI-IIOP综合了RMI的简单性和CORBA的多语言性（兼容性），其次RMI-IIOP克服了RMI只能用于Java的缺点和CORBA的复杂性（可以不用掌握IDL）。

# 7、有关java类加载器

jvm classLoader architecture :

a、Bootstrap ClassLoader/启动类加载器  
主要负责jdk\_home/lib目录下的核心 api 或 -Xbootclasspath 选项指定的jar包装入工作.

B、Extension ClassLoader/扩展类加载器  
主要负责jdk\_home/lib/ext目录下的jar包或 -Djava.ext.dirs 指定目录下的jar包装入工作

C、System ClassLoader/系统类加载器  
主要负责java -classpath/-Djava.class.path所指的目录下的类与jar包装入工作.

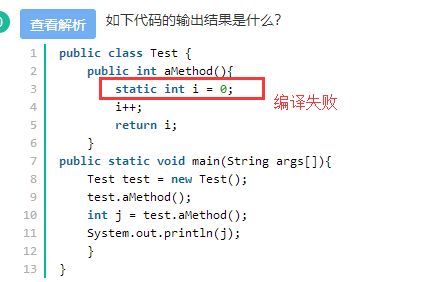
B、 User Custom ClassLoader/用户自定义类加载器(java.lang.ClassLoader的子类)

# 8、时间复杂度

算法的时间复杂度，用来度量算法的运行时间，记作: T(n) = O(f(n))。它表示随着 输入大小n 的增大，算法执行需要的时间的增长速度可以用 f(n) 来描述。

<https://www.jianshu.com/p/f4cca5ce055a>

# 9、静态变量只能在类主体中定义，不能在方法中定义



# 题目收集

# 第一题：



注意：

1、**java语言参数之间只有值传递，包括按值调用和按引用调用。** **一个方法可以修改传递引用所对应的变量值，而不能修改传递值调用所对应的变量值。**

按值调用：包括八大基本数据类型都是按值调用。传值的时候，也就是说方法得到的是所有参数值的一个**拷贝。**

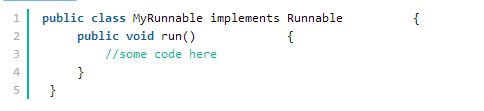
按引用调用：数组、对象。传值时候，传递的是引用地址的拷贝，但是都是指向同一个对象。

2、**String是不可变类（final and Immutable）**,这里只是把副本的指向修改成指向“test ok”，原地址str的指向的值没有发生改变。

# 第二题：

<https://www.nowcoder.com/profile/6612790/test/15979371/25818#summary>

java中下面哪个能创建并启动线程（）



new Thread(new MyRunnable()).start()

**首先：**创建并启动线程的过程为：定义线程—》实例化线程—》启动线程。

**一 、定义线程：** 1、扩展java.lang.Thread类。 2、实现java.lang.Runnable接口。

**二、实例化线程：** 1、如果是扩展java.lang.Thread类的线程，则直接new即可。

                 2、如果是实现了java.lang.Runnable接口的类，则用Thread的构造方法：

        Thread(Runnable target)   
        Thread(Runnable target, String name)   
        Thread(ThreadGroup group, Runnable target)   
        Thread(ThreadGroup group, Runnable target, String name)   
        Thread(ThreadGroup group, Runnable target, String name, long stackSize)

**所以A、D的实例化线程错误。**

**三、启动线程：** 在线程的Thread对象上调用start()方法，而不是run()或者别的方法。

**所以B的启动线程方法错误。**

# 第三题：

<https://www.nowcoder.com/profile/6612790/test/15979371/22470#summary>

下面程序的运行结果是

String str1 = "hello";

String str2 = "he" + new String("llo");

System.err.println(str1 == str2);

这个问题我试着回答一下，同时也是相互学习。String str1= "hello", String str2="he"+"llo";之所以str1==str2返回true是因为两者都是在字符串常量池中（由于初始化就会在此区域分布内存）而常量池中的有个与栈区类似的特性，就是当str2指向的常量在常量区已存在时，他不会创建新的内存空间来存此常量，而是指向已有常量的内存（应该是以此节约空间），此时str1与str2这两个引用变量的值都是存"hello"的内存空间地址，但是String str3= "he"+a;String a="llo";时str1==str3返回的为false，是因为：str1指向的hello在编译期一如既往的还是分配在常量区内，a指向的llo也在常量区，虽然str3也是初始化但是编译器无法判断a这货到底是什么个情况，进而不会将str3的等号右侧声明在常量区内，而是在通过构造时在堆区中的非常量池外的内存中声明，至此str3与str1不止是分配内存的时期不同（一个在编译期，一个在运行期）而且在内存空间的区域也不同，上面最高票答案只区分了时间没区分空间。

String str1 = "hello";这里的str1指的是方法区的字符串常量池中的“hello”，编译时期就知道的； String str2 = "he" + new String("llo");这里的str2必须在运行时才知道str2是什么，所以它是指向的是堆里定义的字符串“hello”，所以这两个引用是不一样的。

如果用str1.equal(str2)，那么返回的是True；因为两个字符串的内容一样。

# 第四题：

<https://www.nowcoder.com/profile/6612790/test/15979371/36313#summary>

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

Java方法执行内存模型，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

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

运行时数据区包括：虚拟机栈区，堆区，方法区，本地方法栈，程序计数器

**虚拟机栈区**：也就是我们常说的栈区，**线程私有**，存放基本类型，对象的引用和 returnAddress ，在编译期间完成分配。

**堆区**， JAVA 堆，也称 GC 堆，所有**线程共享**，存放对象的实例和数组， JAVA 堆是垃圾收集器管理的主要区域。

**方法区**：所有**线程共享**，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。这个区域的内存回收目标主要是针对常量池的对象的回收和对类型的卸载。

**程序计数器**：**线程私有**，每个线程都有自己独立的程序计数器，用来指示下一条指令的地址。