1.java历史

SUN 是“Stanford University Network”

1982年，Sun Microsystems公司诞生于美国斯坦福大学校园。Sun公司1986年上市，在NASDAQ的标识为SUNW后改为JAVA。

2.J2SE 桌面应用的开发

J2EE 企业级开发

J2ME 手机应用的开发

JVM java virtual machine

SDK: software development kit 软件开发包 函数库或者工具等

JDK java development kit 是面向开发人员的,java开发工具集

JRE java runtime enviroment java运行环境,是面向应用程序使用者的

API application program interface 应用程序编程接口

3.Java中常用的包有：

java.lang 这个包中的类可以直接使用

java.awt/javax.swing/java.awt.event javaGUI的包

java.io 输入输出流

java.net 网络编程的包

java.util 工具类包

4.开发工具：这些开发工具都是可执行程序，主要包括：

javac.exe 编译工具；

java.exe 运行工具；

javadoc.exe 生成JavaDoc文档的工具

jar.exe 打包工具

5.Java语言特征：

Java语言包括语法规则和语法规则的实现。Sun公司对Java语言有自己的一份参考实现：JDK。

a. 编程语言：语言本质即规则：字符集、词法规则、句法规则(条件语句、循环语句等);

桌面应用程序 ：拥有图形化用户接口的软件；

服务器应用程序：通过浏览器访问运行；

b. 开发环境：开发程序，不管是桌面应用还是服务器应用，均需要JDK的支持。

c. 应用环境：运行时需要JDK的支持。

d. 部署环境：服务器应用程序需部署运行在服务器应用程序上，服务器本身也是一个Java程序

用Java语言编写的服务器应用程序部署运行时也需要Java语言的支持；

6.垃圾收集的：

什么叫垃圾回收：

1) 垃圾：无用对象所占据的内存空间；

垃圾回收：将无用对象所占据内存空间进行回收的过程；

为什么要垃圾回收？

1) 对象创建，需要占用内存空间，在一个程序运行过程中要创建无数个对象，如果对象永久占用内存，那么内存很快会消费光，倒致后续对象无法创建或者出现内存不足的错误。因此必须采取一定

措施及时回收那些无用对象的内存，这样保证内存可以被重复利用。

C++等程序由程序员显示进行内存释放，这样有可能：

1) 忘记内存释放，倒致无端的内存占用；

2) 释放核心类库占用内存，倒致系统崩溃；

Java中垃圾回收处理：

1) 由虚拟机通过一个系统级的垃圾回收器线程自动完成，不会忘记也不会释放错，系统更加稳定；

Java中垃圾回收处理特点：

1) 由虚拟机通过垃圾回收器线程自动完成；

2) 只有当对象不再被使用，它的内存才有可能被回收；如果虚拟机认为系统不需要额外的内存，即便对象不再使用，内存也不会回收；

3) 程序无法显示迫使垃圾回收器立即执行垃圾回收，可以通过java.lang.System.gc()/java.lang.Runtime.gc()/java.lang.Runtime.getRuntime().gc()建议(通知)虚拟机回收对象；

4) 垃圾回收器线程在释放无用对象占用内存之前会先行调用该对象的finalize()方法。该方法是否被调用以及被调用的时间极其不可靠。不建议重写。

5) System.runFinalize()/Runtime.getRuntime().runFinalize()强制垃圾回收机制调用对象的finalize()，但有可能不会执行，具有不确定性。

6) 运行命令查看垃圾回收后的提示信息：java -verbose:gc HelloWorld

7.阐述Java平台代码安全实现策略；

Java虚拟机为Java程序提供运行时环境，其中一项重要的任务就是管理类，管理类的加载、连接和初始化；

1) 加载：查找并加载类的二进制文件(class文件),将其置于内存(JVM所管理的内存)的方法区中，然后在堆区中创建一个java.lang.Class对象，用来封装类在方法区内的数据结构。

Java虚拟机可以从多种来源加载类的二进制数据，包括：

a. 从本地文件系统中加载.class文件，最常见；

b. 通过网络下载.class文件；

c. 从zip,jar或其他类型的归档文件中提取.class文件；

d. 从一个专有数据库中提取.class文件；

e. 把一个java源文件动态编译为.class文件；

类加载器最多有五个：启动类加载、扩展类加载、本地类加载、自定义类加载、网络类加载

分别作用是：

启动类加载：指定目录进行加载，返回，交给了扩展类加载器

作用：java –verbose 启动的时候加载的内容，只能加载jar包：主要是rt.jar

扩展类加载：父亲是启动类加载，委托给启动类加载加载执行目录，做不了，返回，交给了本地类加载

作用：只能加载jar包, jdk/jre/lib/ext从网上下载扩展的功能jar包放在这里，然后用扩展类加载

本地类加载：父亲是扩展类加载，委托给扩展类加载加载本地，从$CLASSPATH中获得 .

自定义类加载：模板已经写好，继承下，自定义类加载器就写好了。WHY：进行些特殊处理，比如，加载过程中对字节码进行加密（不经常使用）

网络类加载：通过网络把远程的字节码加载到本地运行（有一定的风险，不太推荐使用）

试验：

jar –cvf hello.jar HelloWorld.class

cp hello.jar /opt/jdk/jre/lib/ext

java HelloWord

2) 连接：

a. 验证：确保被加载类的正确性(有正确的内部结构，并且与其他类协调一致)；

为什么要验证：Java虚拟机不知道某个特定的.class文件到底是由正常的Java编译器生成的，还是黑客特制的。类的验证能提

高程序的健壮性，确保程序被安全地执行。

字节码校验内容：. 代码和jvm规范一致

. 代码不能破坏系统的完整性

. 没有堆栈的上溢和下溢

. 参数类型是正确的

. 类型转换的正确的

b. 准备：为类的静态变量分配内存，并将其初始化为默认值；

c. 解析：把类中的符号引用转换为直接引用(一个指向所调用方法在方法区内的内存位置的指针)；

3) 初始化：给类的静态变量赋予正确的初始值；

即时编译器（JIT compiler，just-in-time compiler）是一个把Java的字节码（包括需要被解释的指令的程序）转换成可以直接发送给处理器的指令的程序。

8.Java的工作方式：

源代码: 编写Java源代码文件

|

编译器: 用编译器编译源代码。编译器会检查错误，如果有错就要改正才能产生正确的输出。

|

输出: 编译器会产生字节码。任何支持Java的装置都能够把它转译成可执行的内容。编译后的字节码与平台无关。

|

Java虚拟机: 可以读取与执行字节码。加载指定类，开始执行它的main方法，然后一直运行到main的所有程序代码结束为止。

9.public: 公开给其它类存取；

class: 类声明的关键字；

FirstJavaProgram: 类的名称；

void: 方法的返回值；

main: 方法的名称；

String[]: 数组类型；

args: 参数名称；

System.out.println: 打印到标准输出上(默认为命令行)

10.重要的java命令

1编译java源文件

Javac <options> <source file>

2,执行类

Java [-options] class [args1 args2 args3 ...]

3,创建java文档

Javadoc [option] [packagenames] [sourcefiles]

4,生成jar包

jar {ctxu} [jar-file] [-C directory] files ...

7. 注释

类型：1) 单行注释：

//text——从"//"到本行结束的所有字符均作为注释而被编译器忽略

2) 多行注释：

/\*text\*/——从"/\*"到"\*/"间的所有字符会被编译器忽略

/\*\*/多行注释，可以出现在任意位置(包括[]的内部),

但是不能出现在关键字的中间 e.g main—>ma/\*test\*/in 这是不允许的

3) 文档注释：

/\*\* text \*/——从"/\*\*"到"\*/"间的所有字符会被编译器忽略。当这类注释出现在任何声明(如类的声明、类的成员变量的声明或者类的成员方法的声明)之前时，会作为JavaDoc文档的内容；

8. 分号、块和, 标识符

1) 每个语句短语以;结束 ;

2) 代码段以{}结束;所有的逻辑语句必须出现在{}中间

3) 空白处（空格、tab键、新行和回车（几个语句短语连接在一起））是无关紧要的。

4) 类、方法和变量的名字

java标识符以字母、\_和$开头，不能以数字开头，后跟字母、数字、“\_”和“$”的组合;

大小写敏感

没有长度限制。

举例： 合法标识符 非法标识符

try try#

GROUP\_7 7GROUP

openDoor open-door

boolean1 boolean

11..关键字

Java语言的关键字是程序代码中的特殊字符。包括：

. 类和接口的声明——class, extends, implements(实现), interface

. 包引入和包声明——import, package

. 数据类型——boolean布尔, byte, char, double, float, int, long, short

. 某些数据类型的可选值——false, true, null

. 流程控制——break, case, continue, default, do, else, for, if, return, switch, while

. 异常处理——catch, finally, throw, throws, try

. 修饰符——abstract, final, native, private, protected, public, static,

synchronized(同步锁), transient

. 操作符——instanceof 比较类型是否一致

. 创建对象——new

. 引用——this引用当前对象, super引用父类对象

. 方法返回类型——void 没有返回类型