**[Java中Native关键字的作用](https://www.cnblogs.com/Qian123/p/5702574.html)**

**阅读目录**

* [一、认识 native 即 JNI,Java Native Interface](https://www.cnblogs.com/Qian123/p/5702574.html#_label0)
* [二、用 Java 调用 C 的“Hello，JNI”](https://www.cnblogs.com/Qian123/p/5702574.html#_label1)
* [三、JNI 调用 C 流程图](https://www.cnblogs.com/Qian123/p/5702574.html#_label2)
* [四、其他介绍](https://www.cnblogs.com/Qian123/p/5702574.html#_label3)

初次遇见 native是在 java.lang.Object 源码中的一个hashCode方法：

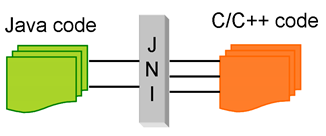
|  |  |
| --- | --- |
| 1 | public native int hashCode(); |

为什么有个native呢？这是我所要学习的地方。所以下面想要总结下native。

[回到顶部](https://www.cnblogs.com/Qian123/p/5702574.html#_labelTop)

**一、认识 native 即 JNI,Java Native Interface**

凡是一种语言，都希望是纯。比如解决某一个方案都喜欢就单单这个语言来写即可。Java平台有个用户和本地C代码进行互操作的API，称为Java Native Interface (Java本地接口)。

[](http://www.bysocket.com/wp-content/uploads/2015/04/image1.png)

[回到顶部](https://www.cnblogs.com/Qian123/p/5702574.html#_labelTop)

**二、用 Java 调用 C 的“Hello，JNI”**

我们需要按照下班方便的步骤进行：

**1、创建一个Java类**，里面包含着一个 native 的方法和加载库的方法 loadLibrary。HelloNative.java 代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public class HelloNative  {      static      {          System.loadLibrary("HelloNative");      }        public static native void sayHello();        @SuppressWarnings("static-access")      public static void main(String[] args)      {          new HelloNative().sayHello();      }  } |

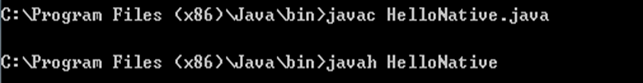
首先让大家注意的是native方法，那个加载库的到后面也起作用。native 关键字告诉编译器（其实是JVM）调用的是该方法在外部定义，这里指的是C。如果大家直接运行这个代码，  JVM会告之：“A Java Exception has occurred.”控制台输出如下：

|  |  |
| --- | --- |
| 1  2  3  4  5 | Exception in thread "main" java.lang.UnsatisfiedLinkError: no HelloNative in java.library.path      at java.lang.ClassLoader.loadLibrary(Unknown Source)      at java.lang.Runtime.loadLibrary0(Unknown Source)      at java.lang.System.loadLibrary(Unknown Source)      at HelloNative.<clinit>(HelloNative.java:5) |

这是程序使用它的时候，虚拟机说不知道如何找到sayHello。下面既可以手动写，自然泥瓦匠是用

**2、运行javah**，得到包含该方法的C声明**头文件.h**

将HelloNative.java ，简单地 javac javah，如图

[](http://www.bysocket.com/wp-content/uploads/2015/04/image2.png)

就得到了下面的 **HelloNative.h文件**：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | /\* DO NOT EDIT THIS FILE - it is machine generated \*/  #include <jni.h>  /\* Header for class HelloNative \*/    #ifndef \_Included\_HelloNative  #define \_Included\_HelloNative  #ifdef \_\_cplusplus  extern "C" {  #endif  /\*   \* Class:     HelloNative   \* Method:    sayHello   \* Signature: ()V   \*/  JNIEXPORT void JNICALL Java\_HelloNative\_sayHello    (JNIEnv \*, jclass);    #ifdef \_\_cplusplus  }  #endif  #endif |

jni.h 这个文件，在/%JAVA\_HOME%include

3、根据头文件，写**C实现本地方法**。

这里我们简单地实现这个sayHello方法如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #include "HelloNative.h"  #include <stdio.h>    JNIEXPORT void JNICALL Java\_HelloNative\_sayHello  {      printf("Hello，JNI");  } |

**4、生成dll共享库**，然后Java程序load库，**调用**即可。

在Windows上，MinGW GCC 运行如下

|  |  |
| --- | --- |
| 1 | gcc -m64  -Wl,--add-stdcall-alias -I"C:\Program Files\Java\jdk1.7.0\_71\include" -I"C:\Program Files\Java\jdk1.7.0\_71\include\include\win32" -shared -o HelloNative.dll HelloNative.c |

-m64表示生成dll库是64位的。然后运行 **HelloNative：**

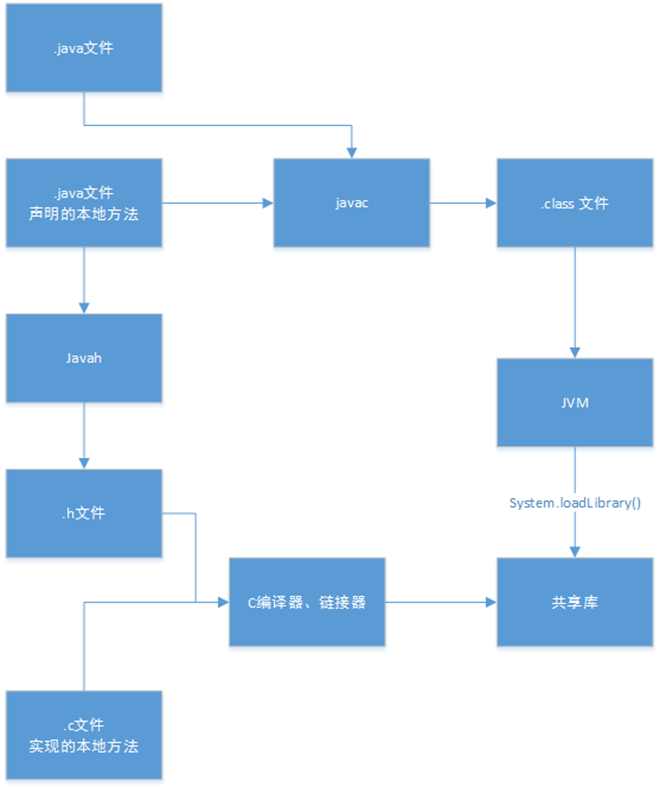
|  |  |
| --- | --- |
| 1 | java HelloNative |

终于成功地可以看到控制台打印如下：

|  |  |
| --- | --- |
| 1 | Hello，JNI |

[回到顶部](https://www.cnblogs.com/Qian123/p/5702574.html#_labelTop)

**三、JNI 调用 C 流程图**



[回到顶部](https://www.cnblogs.com/Qian123/p/5702574.html#_labelTop)

**四、其他介绍**

native是与C++联合开发的时候用的！java自己开发不用的！

使用native关键字说明这个方法是原生函数，也就是这个方法是用C/C++语言实现的，并且被编译成了DLL，由java去调用。  
这些函数的实现体在DLL中，JDK的源代码中并不包含，你应该是看不到的。对于不同的平台它们也是不同的。这也是java的底层机制，实际上java就是在不同的平台上调用不同的native方法实现对操作系统的访问的。

1。native 是用做java 和其他语言（如c++）进行协作时用的  
也就是native 后的函数的实现不是用java写的  
2。既然都不是java，那就别管它的源代码了，呵呵

native的意思就是通知操作系统，  
这个函数你必须给我实现，因为我要使用。  
所以native关键字的函数都是操作系统实现的，  
java只能调用。

java是跨平台的语言，既然是跨了平台，所付出的代价就是牺牲一些对底层的控制，而java要实现对底层的控制，就需要一些其他语言的帮助，这个就是native的作用了

Java不是完美的，Java的不足除了体现在运行速度上要比传统的C++慢许多之外，Java无法直接访问到操作系统底层（如系统硬件等)，为此Java使用native方法来扩展Java程序的功能。  
　　可以将native方法比作Java程序同Ｃ程序的接口，其实现步骤：  
　　１、在Java中声明native()方法，然后编译；  
　　２、用javah产生一个.h文件；  
　　３、写一个.cpp文件实现native导出方法，其中需要包含第二步产生的.h文件（注意其中又包含了JDK带的jni.h文件）；  
　　４、将第三步的.cpp文件编译成动态链接库文件；  
　　５、在Java中用System.loadLibrary()方法加载第四步产生的动态链接库文件，这个native()方法就可以在Java中被访问了。

JAVA本地方法适用的情况   
1.为了使用底层的主机平台的某个特性，而这个特性不能通过JAVA API访问

2.为了访问一个老的系统或者使用一个已有的库，而这个系统或这个库不是用JAVA编写的

3.为了加快程序的性能，而将一段时间敏感的代码作为本地方法实现。

-------------我是低调的分割线--------------------------

如果对你有帮助，可以点击“推荐”哦`(\*∩\_∩\*)′

分类: [①Java基础学习笔记](https://www.cnblogs.com/Qian123/category/782995.html)

标签: [java基础](https://www.cnblogs.com/Qian123/tag/java%E5%9F%BA%E7%A1%80/)