NAME

javah − Generates C header and source files from a Java class.

SYNOPSIS

javah [*options*] f*ully−qualified−class−name ...*

options The command-line options. See Options.

fully-qualified-class-name
The fully qualified location of the classes to be converted to C header and source files.

DESCRIPTION

The **javah** command generates C header and source files that are needed to implement native methods. The generated header and source files are used by C programs to reference an object's instance variables from native source code. The **.h** file contains a **struct** definition with a layout that parallels the layout of the corresponding class. The fields in the **struct** correspond to instance variables in the class.

The name of the header file and the structure declared within it are derived from the name of the class. When the class passed to the **javah** command is inside a package, the package name is added to the beginning of both the header file name and the structure name. Underscores (_) are used as name delimiters.

By default the **javah** command creates a header file for each class listed on the command line and puts the files in the current directory. Use the **-stubs** option to create source files. Use the **-o** option to concatenate the results for all listed classes into a single file.

The Java Native Interface (JNI) does not require header information or stub files. The **javah** command can still be used to generate native method function prototypes needed for JNI-style native methods. The **javah** command produces JNI-style output by default and places the result in the **.h** file.

OPTIONS

-o *outputfile*
Concatenates the resulting header or source files for all the classes listed on the command line into an output file. Only one of **-o** or **-d** can be used.

-d *directory*
Sets the directory where the **javah** command saves the header files or the stub files. Only one of **-d** or **-o** can be used.

-stubs
Causes the **javah** command to generate C declarations from the Java object file.

-verbose
Indicates verbose output and causes the **javah** command to print a message to **stdout** about the status of the generated files.

-help
Prints a help message for **javah** usage.

-version
Prints **javah** command release information.

-jni
Causes the **javah** command to create an output file containing JNI-style native method function prototypes. This is the default output; use of **-jni** is optional.

-classpath *path*
Specifies the path the **javah** command uses to look up classes. Overrides the default or the **CLASSPATH** environment variable when it is set. Directories are separated by colons on Oracle Solaris and semicolons on Windows. The general format for path is:

Oracle Solaris:

.:*your-path*

Example: **.:/home/avh/classes:/usr/local/java/classes**

Windows:

.;*your-path*

Example: **.;C:\users\dac\classes;C:\tools\java\classes**

As a special convenience, a class path element that contains a base name of * is considered equivalent to specifying a list of all the files in the directory with the extension **.jar** or **.JAR**.

For example, if directory **mydir** contains **a.jar** and **b.JAR**, then the class path element **mydir/*** is expanded to a **A.jar:b.JAR**, except that the order of jar files is unspecified. All JAR files in the specified directory, including hidden ones, are included in the list. A class path entry that consists of * expands to a list of all the JAR files in the current directory. The **CLASSPATH** environment variable, where defined, is similarly expanded. Any class path wild card expansion occurs before the Java Virtual Machine (JVM) is started. A Java program will never see unexpanded wild cards except by querying the environment. For example, by calling **System.getenv("CLASSPATH")**.

-bootclasspath *path*
: Specifies the path from which to load bootstrap classes. By default, the bootstrap classes are the classes that implement the core Java platform located in **jre\lib\rt.jar** and several other JAR files.

-old
: Specifies that old JDK 1.0-style header files should be generated.

-force
: Specifies that output files should always be written.

-J*option*
: Passes **option** to the Java Virtual Machine, where **option** is one of the options described on the reference page for the Java application launcher. For example, **-J-Xms48m** sets the startup memory to 48 MB. See java(1).

SEE ALSO

- javah(1)
- java(1)
- jdb(1)
- javap(1)
- javadoc(1)