

See also: Getting Started with the Annotation Processing Tool (apt) ^[12]

The *appletviewer* tool

Java applets require a particular environment to execute. Typically, this environment is provided by a browser with a Java plug-in, and a web server serving the applet. However, during development and testing of an applet it might be more convenient to start an applet without the need to fiddle with a browser and a web server. In such a case, Sun's *appletviewer* from the JDK can be used to run an applet.

The *javah* tool

A Java class can call *native*, or non-Java, code that has been prepared to be called from Java. The details and procedures are specified in the JNI (*Java Native Interface*). Commonly, native code is written in C (or C++). The JDK tool *javah* helps to write the necessary C code, by generating C header files and C stub code.

The *extcheck* tool

extcheck also appeared first with Java 1.5. It can be used prior to the installation of a Java extension into the JDK or JRE environment. It checks if a particular Jar file conflicts with an already installed extension.

Security Tools

The JDK comes with a large number of tools related to the security features of Java. Usage of these tools first requires study of the particular security mechanisms.

The tools are:

keytool

To manage keys and certificates

jarsigner

To generate and verify digital signatures of JARs (Java ARchives)

policytool

To edit policy files

kinit

To obtain Kerberos v5 tickets

klist

To manage Kerberos credential cache and key table

ktab

To manage entries in a key table

The *native2ascii* tool

native2ascii is an important, though underappreciated, tool for writing *properties files* -- files containing configuration data -- or *resource bundles* -- files containing language translations of text.

Such files can contain only ASCII and Latin-1 characters, but international programmers need a full range of character sets. Text using these characters can appear in properties files and resource bundles only if the non-ASCII and non-Latin-1 characters are converted into Unicode escape sequences (\uXXXX notation).

The task of writing such escape sequences is handled by *native2ascii*. You can write the international text in an editor using the appropriate character encoding, then use *native2ascii* to generate the necessary ASCII text with embedded Unicode escape sequences. Despite the name, *native2ascii* can also convert from ASCII to native, so it is

useful for converting an existing properties file or resource bundle back to some other encoding.

native2ascii makes most sense when integrated into a build system to automate the conversion.

Monitoring and Management Tools / Troubleshooting Tools

With Java 1.5 a set of *monitoring and management tools* have been added to the JDK, in addition to a set of troubleshooting tools.

The monitoring and management tools are intended for monitoring and managing the virtual machine and the execution environment. They allow, for example, monitoring memory usage during the execution of a Java program.

The troubleshooting tools provide rather esoteric insight into aspects of the virtual machine. (Interestingly, the Java debugger is not categorized as a troubleshooting tool.)

All the monitoring and management and troubleshooting tools are currently marked as "experimental" (which does not affect jdb). So they might disappear in future JDKs.

The Jar tool

Jar is short for *Java archive*. It is a tool for creating Java archives or *jar files* - a file with .jar as the extension. A Java archive is a collection of compiled Java classes and other resources which those classes may require (such as text files, configuration files, images) at runtime. Internally, a jar file is really a .zip file.

The jdb tool

Jdb is short for *Java debugger*. The Java debugger is a command-line console that provides a debugging environment for Java programs. Although you can use this command line console, IDE's normally provide easier to use debugging environments.

The Javadoc tool

As programs grow large and complex, programmers need ways to track changes and to understand the code better at each step of its evolution. For decades, programmer have been employing the use of special programming constructs called comments - regions that help declare user definitions for a code snippet within the source code. But comments are prone to be verbose and incomprehensible, let alone be difficult to read in applications having hundreds of lines of code.

Java provides the user with a way to easily publish documentation about the code using a special commenting system and the javadoc tool. The javadoc tool generates documentation about the w:application programming interface (API) of a set of user-created Java classes. javadoc reads source file comments from the .java source files and generates HTML documents that are easier to read and understand without looking at the code itself.

The javap tool

Where Javadoc provide a detailed view into the API and documentation of a Java class, the javap tool prints information regarding members (constructors, methods and variables) in a class. In other words, it lists the class' API and/or the compiled instructions of the class. javap is a formatting disassembler for Java bytecode.

See other tools ^[13] for all the tools that are bundled with Sun's JDK.

Installation

To formally begin programming in Java, you would need to obtain the Java software necessary and install it on your system. This section will help you set up any system with the latest Java software.

Obtaining Java software

For now, a text editor should be the least of your worries. What matters the most is to get your hands on the two main components required for Java programming: the compiler and the JVM. The compiler comes bundled with other useful programs for you to use. This bundle of programs is called the **Java Development Kit** (or **JDK**). On the other hand, the JVM also comes bundled with exciting up-to-date features called the **Java Runtime Environment** (or **JRE**). For more in-depth detail, read the section on the Java Platform. Unlike most other computer programming environments, the complete Java platform (both JDK and JRE) is free to download from Sun Microsystem's website - <http://java.sun.com/javase/downloads/index.jsp> ^[14]

Clicking on website link above, you would be taken to a download page giving you an option to download the various components needed for Java programming. If you feel you don't have enough space for both the software to be installed on your system (which is highly unlikely) you should, in that case, opt only for the JDK as that is the only component necessary to build a complete Java program. The JDK comes with its very own JRE. With that said, once the software is downloaded and installed on the system, we can begin with the rest.

Setting up the environment

Once you have Java installed on your system, we just need to worry about writing the actual code. This is done using a text editor - Notepad or WordPad would do. But these editors can only provide you with a limited ability to write text and save files. You need to make your editor do lots of other stuff as well, like compiling the code for you, showing you how the code works and telling you off when you write wrong code. A text editor that can do all that is called an **integrated development environment** (in short **IDE**). There are a dozen IDEs to choose from and the good news is - most of them are free to download and install. Some IDEs worth mentioning are:

1. Eclipse: This is a powerful IDE that supports Java and many other programming languages like C/C++, PHP and Python. It is open source and free to download, install and use. Since Eclipse is open source, it has a strong user base and you can find a lot of people online who can tell you how to work the IDE.
2. NetBeans: Like Eclipse, NetBeans is also an integrated development environment that support various languages other than Java but the plus point for this IDE is that it is built by the people who made the Java language. And it is free, too!