



Java (programming language)

Java

	
Paradigm	Object-oriented, structured, imperative
Appeared in	1995
Designed by	Sun Microsystems (now owned by Oracle Corporation)
Developer	James Gosling & Sun Microsystems
Stable release	Java Standard Edition 6 (1.6.0_23) (December 8, 2010)
Typing discipline	Static, strong, safe, nominative, manifest
Major implementations	OpenJDK, HotSpot, many others
Dialects	Generic Java, Pizza
Influenced by	Ada 83, C++, C#, ^[1] Delphi Object Pascal, ^[2] Eiffel, ^[3] Generic Java, Mesa, ^[4] Modula-3, ^[5] Objective-C, ^[6] UCSD Pascal, ^[7] ^[8] Smalltalk
Influenced	Ada 2005, BeanShell, C#, Clojure, D, ECMAScript, Groovy, J#, JavaScript, PHP, Python, Scala
OS	Cross-platform (multi-platform)
License	GNU General Public License / Java Community Process
Usual file extensions	.java, .class, .jar
Website	For Java Developers ^[9]
 Java Programming at Wikibooks	

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.^[10] ^[11]

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java, GNU Classpath, and Dalvik.

History

James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991.^[12] Java was originally designed for interactive television, but it was too advanced for the digital cable television industry at the time.^[13] The language was initially called *Oak* after an oak tree that stood outside Gosling's office; it went by the name *Green* later, and was later renamed *Java*, from a list of random words.^[14] Gosling aimed to implement a virtual machine and a language that had a familiar C/C++ style of notation.^[15]

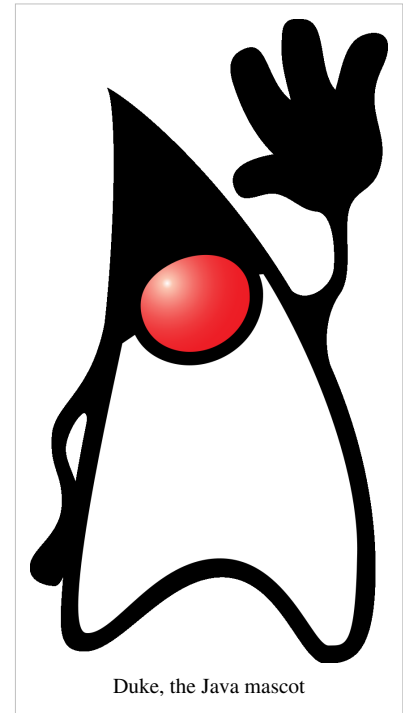
Sun Microsystems released the first public implementation as Java 1.0 in 1995. It promised "Write Once, Run Anywhere" (WORA), providing no-cost run-times on popular platforms. Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to run Java *applets* within web pages, and Java quickly became popular. With the advent of *Java 2* (released initially as J2SE 1.2 in December 1998–1999), new versions had multiple configurations built for different types of platforms. For example, *J2EE* targeted enterprise applications and the greatly stripped-down version *J2ME* for mobile applications (Mobile Java). *J2SE* designated the Standard Edition. In 2006, for marketing purposes, Sun renamed new *J2* versions as *Java EE*, *Java ME*, and *Java SE*, respectively.

In 1997, Sun Microsystems approached the ISO/IEC JTC1 standards body and later the Ecma International to formalize Java, but it soon withdrew from the process.^[16] Java remains a *de facto* standard, controlled through the Java Community Process.^[17] At one time, Sun made most of its Java implementations available without charge, despite their proprietary software status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System. Sun distinguishes between its Software Development Kit (SDK) and Runtime Environment (JRE) (a subset of the SDK); the primary distinction involves the JRE's lack of the compiler, utility programs, and header files.

On November 13, 2006, Sun released much of Java as open source software under the terms of the GNU General Public License (GPL). On May 8, 2007, Sun finished the process, making all of Java's core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright.^[18]

Sun's vice-president Rich Green has said that Sun's ideal role with regards to Java is as an "evangelist."^[19]

Following Oracle Corporation's acquisition of Sun Microsystems in 2009–2010, Oracle has described itself as the "steward of Java technology with a relentless commitment to fostering a community of participation and transparency".^[20]



Duke, the Java mascot

Principles

There were five primary goals in the creation of the Java language:^[21]

1. It should be "simple, object oriented, and familiar".
2. It should be "robust and secure".
3. It should be "architecture neutral and portable".
4. It should execute with "high performance".
5. It should be "interpreted, threaded, and dynamic".

Practices

Java Platform

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any supported hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to platform-specific machine code. Java bytecode instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets.

Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. Just-in-Time compilers were introduced from an early stage that compile bytecodes to machine code during runtime.

Implementations

Sun Microsystems officially licenses the Java Standard Edition platform for Linux,^[22] Mac OS X,^[23] and Solaris. Although in the past Sun has licensed Java to Microsoft, the license has expired and has not been renewed.^[24] Through a network of third-party vendors and licensees,^[25] alternative Java environments are available for these and other platforms.

Sun's trademark license for usage of the Java brand insists that all implementations be "compatible". This resulted in a legal dispute with Microsoft after Sun claimed that the Microsoft implementation did not support RMI or JNI and had added platform-specific features of their own. Sun sued in 1997, and in 2001 won a settlement of US\$20 million, as well as a court order enforcing the terms of the license from Sun.^[26] As a result, Microsoft no longer ships Java with Windows, and in recent versions of Windows, Internet Explorer cannot support Java applets without a third-party plugin. Sun, and others, have made available free Java run-time systems for those and other versions of Windows.

Platform-independent Java is essential to the Java EE strategy, and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications, such as Web services, Java Servlets, and Enterprise JavaBeans, as well as with embedded systems based on OSGi, using Embedded Java environments. Through the new GlassFish project, Sun is working to create a fully functional, unified open source implementation of the Java EE technologies.

Sun also distributes a superset of the JRE called the Java Development Kit (commonly known as the JDK), which includes development tools such as the Java compiler, Javadoc, Jar, and debugger.

Performance

Programs written in Java have a reputation for being slower and requiring more memory than those written in C.^[27] However, Java programs' execution speed improved significantly with the introduction of Just-in-time compilation in 1997/1998 for Java 1.1,^[28] the addition of language features supporting better code analysis (such as inner classes, StringBuffer class, optional assertions, etc.), and optimizations in the Java Virtual Machine itself, such as HotSpot becoming the default for Sun's JVM in 2000. Currently, Java code has approximately half the performance of C code.^[29]

Some platforms offer direct hardware support for Java; there are microcontrollers that can run java in hardware instead of a software JVM, and ARM based processors can have hardware support for executing Java bytecode through its Jazelle option.

Automatic memory management

Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the unreachable memory becomes eligible to be freed automatically by the garbage collector. Something similar to a memory leak may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a nonexistent object are called, a "null pointer exception" is thrown.^[30]^[31]

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the stack, or explicitly allocated and deallocated from the heap. In the latter case the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a memory leak occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable and/or crash. This can be partially remedied by the use of smart pointers, but these add overhead and complexity. Note that garbage collection does not prevent "logical" memory leaks, i.e. those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style pointer arithmetic, where object addresses and unsigned integers (usually long integers) can be used interchangeably. This allows the garbage collector to relocate referenced objects and ensures type safety and security.

As in C++ and some other object-oriented languages, variables of Java's primitive data types are not objects. Values of primitive types are either stored directly in fields (for objects) or on the stack (for methods) rather than on the heap, as commonly true for objects (but see Escape analysis). This was a conscious decision by Java's designers for performance reasons. Because of this, Java was not considered to be a pure object-oriented programming language. However, as of Java 5.0, autoboxing enables programmers to proceed as if primitive types were instances of their wrapper class.

Java contains multiple types of garbage collectors. By default, HotSpot uses the Concurrent Mark Sweep collector, also known as the CMS Garbage Collector. However, there are also several other garbage collectors that can be used to manage the Heap. For 90% of applications in Java, the CMS Garbage Collector is good enough.^[32]

Syntax

The syntax of Java is largely derived from C++. Unlike C++, which combines the syntax for structured, generic, and object-oriented programming, Java was built almost exclusively as an object-oriented language. All code is written inside a class, and everything is an object, with the exception of the primitive data types (integers, floating-point numbers, boolean values, and characters), which are not classes for performance reasons.

Java suppresses several features (such as operator overloading and multiple inheritance) for *classes* in order to simplify the language and to prevent possible errors and anti-pattern design.

Java uses similar commenting methods to C++. There are three different styles of comment: a single line style marked with two slashes (*//*), a multiple line style opened with a slash asterisk (*/**) and closed with an asterisk slash (**/*), and the Javadoc commenting style opened with a slash and two asterisks (*/***) and closed with an asterisk slash (**/*). The Javadoc style of commenting allows the user to run the Javadoc executable to compile documentation for the program.

Example:

```
// This is an example of a single line comment using two slashes

/* This is an example of a multiple line comment using the slash and
   asterisk.
   This type of comment can be used to hold a lot of information or
   deactivate
   code but it is very important to remember to close the comment. */

/**
 * This is an example of a Javadoc comment; Javadoc can compile
   documentation
 * from this text.
 */
```

Examples

Hello world

The traditional Hello world program can be written in Java as:

```
/**
 * @param args Command-line arguments
 * Output "Hello, world!", then exit.
 */
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

To compare this to other programming languages see the list of hello world program examples.

Source files must be named after the public class they contain, appending the suffix `.java`, for example, `HelloWorld.java`. It must first be compiled into bytecode, using a Java compiler, producing a file named `HelloWorld.class`. Only then can it be executed, or 'launched'. The java source file may only contain one public class

but can contain multiple classes with less than public access and any number of public inner classes.

A **class** that is not declared **public** may be stored in any .java file. The compiler will generate a class file for each class defined in the source file. The name of the class file is the name of the class, with *.class* appended. For class file generation, anonymous classes are treated as if their name were the concatenation of the name of their enclosing class, a \$, and an integer.

The keyword **public** denotes that a method can be called from code in other classes, or that a class may be used by classes outside the class hierarchy. The class hierarchy is related to the name of the directory in which the .java file is located.

The keyword **static** in front of a method indicates a static method, which is associated only with the class and not with any specific instance of that class. Only static methods can be invoked without a reference to an object. Static methods cannot access any method variables that are not static.

The keyword **void** indicates that the main method does not return any value to the caller. If a Java program is to exit with an error code, it must call `System.exit()` explicitly.

The method name "main" is not a keyword in the Java language. It is simply the name of the method the Java launcher calls to pass control to the program. Java classes that run in managed environments such as applets and Enterprise JavaBean do not use or need a `main()` method. A java program may contain multiple classes that have main methods, which means that the VM needs to be explicitly told which class to launch from.

The main method must accept an array of **String** objects. By convention, it is referenced as **args** although any other legal identifier name can be used. Since Java 5, the main method can also use variable arguments, in the form of `public static void main(String... args)`, allowing the main method to be invoked with an arbitrary number of String arguments. The effect of this alternate declaration is semantically identical (the `args` parameter is still an array of String objects), but allows an alternative syntax for creating and passing the array.

The Java launcher launches Java by loading a given class (specified on the command line or as an attribute in a JAR) and starting its public static void `main(String[])` method. Stand-alone programs must declare this method explicitly. The `String[] args` parameter is an array of String objects containing any arguments passed to the class. The parameters to `main` are often passed by means of a command line.

Printing is part of a Java standard library: The **System** class defines a public static field called **out**. The `out` object is an instance of the `PrintStream` class and provides many methods for printing data to standard out, including **println(String)** which also appends a new line to the passed string.

The string "Hello, world!" is automatically converted to a String object by the compiler.

A more comprehensive example

```
// OddEven.java
import javax.swing.JOptionPane;

public class OddEven {
    // "input" is the number that the user gives to the computer
    private int input; // a whole number("int" means integer)

    /*
     * This is the constructor method. It gets called when an object of
     the OddEven type
     * is being created.
     */
    public OddEven() {
```

```
/*
 * In most Java programs constructors can initialize objects with
default values, or create
 * other objects that this object might use to perform its
functions. In some Java programs, the
 * constructor may simply be an empty function if nothing needs to
be initialized prior to the
 * functioning of the object. In this program's case, an empty
constructor would suffice, even if
 * it is empty. A constructor must exist, however if the user
doesn't put one in then the compiler
 * will create an empty one.
 */
}

// This is the main method. It gets called when this class is run
through a Java interpreter.
public static void main(String[] args) {
    /*
     * This line of code creates a new instance of this class
called "number" (also known as an
     * Object) and initializes it by calling the constructor. The
next line of code calls
     * the "showDialog()" method, which brings up a prompt to ask
you for a number
     */
    OddEven number = new OddEven();
    number.showDialog();
}

public void showDialog() {
    /*
     * "try" makes sure nothing goes wrong. If something does,
     * the interpreter skips to "catch" to see what it should do.
     */
    try {
        /*
         * The code below brings up a JOptionPane, which is a
dialog box
         * The String returned by the "showInputDialog()" method is
converted into
         * an integer, making the program treat it as a number
instead of a word.
         * After that, this method calls a second method,
calculate() that will
         * display either "Even" or "Odd."
         */
    }
```

```

        input =
Integer.parseInt(JOptionPane.showInputDialog("Please Enter A Number"));
        calculate();
    } catch (NumberFormatException e) {
        /*
         * Getting in the catch block means that there was a
problem with the format of
         * the number. Probably some letters were typed in instead
of a number.
        */
        System.err.println("ERROR: Invalid input. Please type in a
numerical value.");
    }
}

/*
 * When this gets called, it sends a message to the interpreter.
 * The interpreter usually shows it on the command prompt (For
Windows users)
 * or the terminal (For Linux users). (Assuming it's open)
 */
private void calculate() {
    if (input % 2 == 0) {
        System.out.println("Even");
    } else {
        System.out.println("Odd");
    }
}
}

```

- The **import** statement imports the **JOptionPane** class from the **javax.swing** package.
- The **OddEven** class declares a single **private** field of type **int** named **input**. Every instance of the **OddEven** class has its own copy of the input field. The private declaration means that no other class can access (read or write) the input field.
- **OddEven()** is a **public** constructor. Constructors have the same name as the enclosing class they are declared in, and unlike a method, have no return type. A constructor is used to initialize an object that is a newly created instance of the class.
- The **calculate()** method is declared without the static keyword. This means that the method is invoked using a specific instance of the **OddEven** class. (The reference used to invoke the method is passed as an undeclared parameter of type **OddEven** named **this**.) The method tests the expression `input % 2 == 0` using the **if** keyword to see if the remainder of dividing the input field belonging to the instance of the class by two is zero. If this expression is true, then it prints **Even**; if this expression is false it prints **Odd**. (The input field can be equivalently accessed as `this.input`, which explicitly uses the undeclared `this` parameter.)
- **OddEven number = new OddEven();** declares a local object reference variable in the main method named **number**. This variable can hold a reference to an object of type **OddEven**. The declaration initializes **number** by first creating an instance of the **OddEven** class, using the **new** keyword and the **OddEven()** constructor, and then assigning this instance to the variable.

- The statement **number.showDialog();** calls the calculate method. The instance of OddEven object referenced by the number local variable is used to invoke the method and passed as the undeclared this parameter to the calculate method.
- **input = Integer.parseInt(JOptionPane.showInputDialog("Please Enter A Number"));** is a statement that converts the type of **String** to the primitive data type **int** by using a utility function in the primitive wrapper class **Integer**.

Special classes

Applet

Java applets are programs that are embedded in other applications, typically in a Web page displayed in a Web browser.

```
// Hello.java
import javax.swing.JApplet;
import java.awt.Graphics;

public class Hello extends JApplet {

    @Override
    public void paintComponent(Graphics g) {
        g.drawString("Hello, world!", 65, 95);
    }

}
```

The **import** statements direct the Java compiler to include the **javax.swing.JApplet** and **java.awt.Graphics** classes in the compilation. The import statement allows these classes to be referenced in the source code using the *simple class name* (i.e. JApplet) instead of the *fully qualified class name* (i.e. javax.swing.JApplet).

The Hello class **extends** (subclasses) the **JApplet** (Java Applet) class; the JApplet class provides the framework for the host application to display and control the lifecycle of the applet. The JApplet class is a JComponent (Java Graphical Component) which provides the applet with the capability to display a graphical user interface (GUI) and respond to user events.

The Hello class overrides the **paintComponent(Graphics)** method inherited from the Container superclass to provide the code to display the applet. The paintComponent() method is passed a **Graphics** object that contains the graphic context used to display the applet. The paintComponent() method calls the graphic context **drawString(String, int, int)** method to display the **"Hello, world!"** string at a pixel offset of **(65, 95)** from the upper-left corner in the applet's display.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<!-- Hello.html -->
<html>
  <head>
    <title>Hello World Applet</title>
  </head>
  <body>
    <applet code="Hello" width="200" height="200">
```

```

    </applet>
  </body>
</html>

```

An applet is placed in an HTML document using the **<applet>** HTML element. The applet tag has three attributes set: **code="Hello"** specifies the name of the JApplet class and **width="200" height="200"** sets the pixel width and height of the applet. Applets may also be embedded in HTML using either the object or embed element,^[33] although support for these elements by Web browsers is inconsistent.^[34] However, the applet tag is deprecated, so the object tag is preferred where supported.

The host application, typically a Web browser, instantiates the **Hello** applet and creates an AppletContext for the applet. Once the applet has initialized itself, it is added to the AWT display hierarchy. The paintComponent() method is called by the AWT event dispatching thread whenever the display needs the applet to draw itself.

Servlet

Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. Servlets are server-side Java EE components that generate responses (typically HTML pages) to requests (typically HTTP requests) from clients. A servlet can almost be thought of as an applet that runs on the server side—without a face.

```

// Hello.java
import java.io.*;
import javax.servlet.*;

public class Hello extends GenericServlet {
    public void service(ServletRequest request, ServletResponse
response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        final PrintWriter pw = response.getWriter();
        pw.println("Hello, world!");
        pw.close();
    }
}

```

The **import** statements direct the Java compiler to include all of the public classes and interfaces from the **java.io** and **javax.servlet**^[35] packages in the compilation.

The **Hello** class **extends** the **GenericServlet**^[36] class; the GenericServlet class provides the interface for the server to forward requests to the servlet and control the servlet's lifecycle.

The Hello class overrides the **service(ServletRequest, ServletResponse)**^[37] method defined by the Servlet^[38] interface to provide the code for the service request handler. The service() method is passed a **ServletRequest**^[39] object that contains the request from the client and a **ServletResponse**^[40] object used to create the response returned to the client. The service() method declares that it **throws** the exceptions ServletException^[41] and IOException if a problem prevents it from responding to the request.

The **setContentType(String)**^[42] method in the response object is called to set the MIME content type of the returned data to "text/html". The **getWriter()**^[43] method in the response returns a **PrintWriter** object that is used to write the data that is sent to the client. The **println(String)** method is called to write the "Hello, world!" string to the response and then the **close()** method is called to close the print writer, which causes the data that has been written to the stream to be returned to the client.

JavaServer Pages

JavaServer Pages (JSP) are server-side Java EE components that generate responses, typically HTML pages, to HTTP requests from clients. JSPs embed Java code in an HTML page by using the special delimiters `<%` and `%>`. A JSP is compiled to a Java *servlet*, a Java application in its own right, the first time it is accessed. After that, the generated servlet creates the response.

Swing application

Swing is a graphical user interface library for the Java SE platform. It is possible to specify a different look and feel through the pluggable look and feel system of Swing. Clones of Windows, GTK+ and Motif are supplied by Sun. Apple also provides an Aqua look and feel for Mac OS X. Where prior implementations of these looks and feels may have been considered lacking, Swing in Java SE 6 addresses this problem by using more native GUI widget drawing routines of the underlying platforms.

This example Swing application creates a single window with "Hello, world!" inside:

```
// Hello.java (Java SE 5)
import javax.swing.*;

public class Hello extends JFrame {
    public Hello() {
        super("hello");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        add(new JLabel("Hello, world!"));
        setVisible(true);
        pack();
    }

    public static void main(String[] args) {
        new Hello();
    }
}
```

The first **import** includes all of the public classes and interfaces from the **javax.swing** package.

The **Hello** class **extends** the **JFrame** class; the **JFrame** class implements a window with a title bar and a close control.

The **Hello()** constructor initializes the frame by first calling the superclass constructor, passing the parameter "hello", which is used as the window's title. It then calls the **setDefaultCloseOperation(int)** method inherited from **JFrame** to set the default operation when the close control on the title bar is selected to **WindowConstants.EXIT_ON_CLOSE** — this causes the **JFrame** to be disposed of when the frame is closed (as opposed to merely hidden), which allows the JVM to exit and the program to terminate. Next, a **JLabel** is created for the string "Hello, world!" and the **add(Component)** method inherited from the **Container** superclass is called to add the label to the frame. The **pack()** method inherited from the **Window** superclass is called to size the window and lay out its contents.

The **main()** method is called by the JVM when the program starts. It instantiates a new **Hello** frame and causes it to be displayed by calling the **setVisible(boolean)** method inherited from the **Component** superclass with the boolean parameter **true**. Once the frame is displayed, exiting the main method does not cause the program to terminate because the AWT event dispatching thread remains active until all of the Swing top-level windows have been disposed.

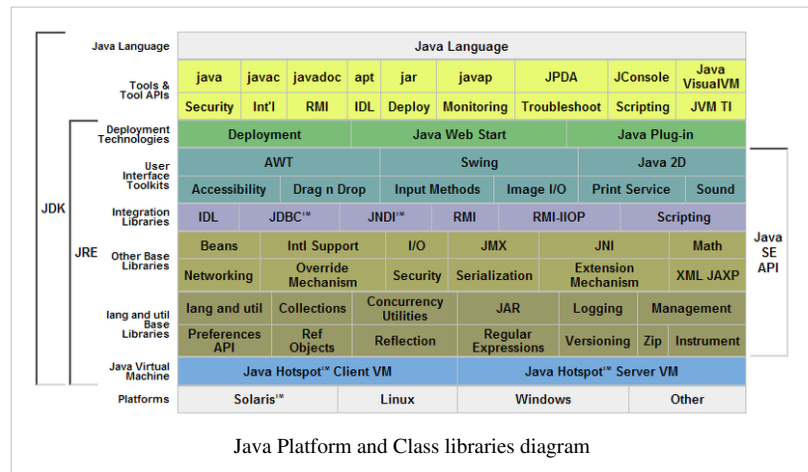
Generics

In 2004, generics were added to the Java language, as part of J2SE 5.0. Prior to the introduction of generics, each variable declaration had to be of a specific type. For container classes, for example, this is a problem because there is no easy way to create a container that accepts only specific types of objects. Either the container operates on all subtypes of a class or interface, usually `Object`, or a different container class has to be created for each contained class. Generics allow compile-time type checking without having to create a large number of container classes, each containing almost identical code.

Class libraries

- Java libraries are the compiled bytecodes of source code developed by the JRE implementor to support application development in Java. Examples of these libraries are:

- The core libraries, which include:
 - Collection libraries that implement data structures such as lists, dictionaries, trees, sets, queues and double-ended queue, or stacks
 - XML Processing (Parsing, Transforming, Validating) libraries
 - Security
 - Internationalization and localization libraries
- The integration libraries, which allow the application writer to communicate with external systems. These libraries include:
 - The Java Database Connectivity (JDBC) API for database access
 - Java Naming and Directory Interface (JNDI) for lookup and discovery
 - RMI and CORBA for distributed application development
 - JMX for managing and monitoring applications
- User interface libraries, which include:
 - The (heavyweight, or native) Abstract Window Toolkit (AWT), which provides GUI components, the means for laying out those components and the means for handling events from those components
 - The (lightweight) Swing libraries, which are built on AWT but provide (non-native) implementations of the AWT widgetry
 - APIs for audio capture, processing, and playback
- A platform dependent implementation of Java Virtual Machine (JVM) that is the means by which the byte codes of the Java libraries and third party applications are executed
- Plugins, which enable applets to be run in Web browsers
- Java Web Start, which allows Java applications to be efficiently distributed to end-users across the Internet
- Licensing and documentation.




Documentation

Javadoc is a comprehensive documentation system, created by Sun Microsystems, used by many Java developers. It provides developers with an organized system for documenting their code. Javadoc comments have an extra asterisk at the beginning, ie the tags are `/**` and `*/`, whereas the normal multi-line comment tags comments in Java and C are set off with `/*` and `*/`.

Editions

Java editions


Java Card
Micro Edition (ME)
Standard Edition (SE)
Enterprise Edition (EE)
PersonalJava (discontinued)

Sun has defined and supports four editions of Java targeting different application environments and segmented many of its APIs so that they belong to one of the platforms. The platforms are:

- Java Card for smartcards.
- Java Platform, Micro Edition (Java ME) — targeting environments with limited resources.
- Java Platform, Standard Edition (Java SE) — targeting workstation environments.
- Java Platform, Enterprise Edition (Java EE) — targeting large distributed enterprise or Internet environments.

The classes in the Java APIs are organized into separate groups called packages. Each package contains a set of related interfaces, classes and exceptions. Refer to the separate platforms for a description of the packages available.

The set of APIs is controlled by Sun Microsystems in cooperation with others through the Java Community Process program. Companies or individuals participating in this process can influence the design and development of the APIs. This process has been a subject of controversy.

Sun also provided an edition called PersonalJava that has been superseded by later, standards-based Java ME configuration-profile pairings.

Notes

- [1] Java 5.0 added several new language features (the enhanced for loop, autoboxing, varargs and annotations), after they were introduced in the similar (and competing) C# language (<http://www.barrycornelius.com/papers/java5/>) (<http://www.levenez.com/lang/>)
- [2] "About Microsoft's "Delegates"" (<http://java.sun.com/docs/white/delegates.html>). . Retrieved 2010-01-11. "We looked very carefully at Delphi Object Pascal and built a working prototype of bound method references in order to understand their interaction with the Java programming language and its APIs. [...] Our conclusion was that bound method references are unnecessary and detrimental to the language. This decision was made in consultation with Borland International, who had previous experience with bound method references in Delphi Object Pascal."
- [3] Gosling and McGilton (May 1996). "The Java Language Environment" (<http://java.sun.com/docs/white/langenv/Intro.doc1.html#943>). .
- [4] J. Gosling, B. Joy, G. Steele, G. Brachda. "The Java Language Specification, 2nd Edition" (http://java.sun.com/docs/books/jls/second_edition/html/intro.doc.html#237601). .

- [5] "The A-Z of Programming Languages: Modula-3" (<http://www.computerworld.com.au/index.php/id;1422447371;pp;3;fp;4194304;fpid;1>). Computerworld.com.au. . Retrieved 2010-06-09.
- [6] Patrick Naughton cites Objective-C as a strong influence on the design of the Java programming language, stating that notable direct derivatives include Java interfaces (derived from Objective-C's protocol) and primitive wrapper classes. (<http://cs.gmu.edu/~sean/stuff/java-objc.html>)
- [7] TechMetrix Research (1999). "History of Java" (<http://www.fscript.org/prof/javapassport.pdf>). *Java Application Servers Report*. . "The project went ahead under the name "green" and the language was based on an old model of UCSD Pascal, which makes it possible to generate interpretive code"
- [8] "A Conversation with James Gosling – ACM Queue" (<http://queue.acm.org/detail.cfm?id=1017013>). Queue.acm.org. 2004-08-31. . Retrieved 2010-06-09.
- [9] <http://www.oracle.com/technetwork/java/>
- [10] "Programming Language Popularity" (<http://www.langpop.com/>). 2009. . Retrieved 2009-01-16.
- [11] "TIOBE Programming Community Index" (<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>). 2009. . Retrieved 2009-05-06.
- [12] Byous, Jon (ca. 1998). "Java technology: The early years" (<http://web.archive.org/web/20050420081440/http://java.sun.com/features/1998/05/birthday.html>). *Sun Developer Network*. Sun Microsystems. Archived from the original (<http://java.sun.com/features/1998/05/birthday.html>) on April 20, 2005. . Retrieved 2005-04-22.
- [13] "The History of Java Technology" (<http://www.java.com/en/javahistory/>). *Sun Developer Network*. ca. 1995. . Retrieved 2010-04-30.
- [14] "Jonathan Schwartz's Blog: Different Isn't Always Better, But Better's Always Different" (http://blogs.sun.com/jonathan/entry/better_is_always_different). Blogs.sun.com. . Retrieved 2010-06-09.
- [15] Heinz Kabutz, *Once Upon an Oak* (<http://www.artima.com/weblogs/viewpost.jsp?thread=7555>). Artima. Retrieved April 29, 2007.
- [16] Java Study Group (<http://www.open-std.org/JTC1/SC22/JSG/>); Why Java Was – Not – Standardized Twice (<http://csdl2.computer.org/comp/proceedings/hicss/2001/0981/05/09815015.pdf>); What is ECMA—and why Microsoft cares (<http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2832719,00.html>)
- [17] "Java Community Process website" (<http://www.jcp.org/en/home/index>). Jcp.org. 2010-05-24. . Retrieved 2010-06-09.
- [18] "JAVAONE: Sun – The bulk of Java is open sourced" (http://open.itworld.com/4915/070508opsjava/page_1.html). open.itworld.com. . Retrieved 2010-06-09.
- [19] "Sun's Evolving Role as Java Evangelist" (<http://onjava.com/pub/a/onjava/2002/04/17/evangelism.html>). O'Reilly Media. .
- [20] "Oracle and Java" (<http://www.oracle.com/us/technologies/java/index.html>). *oracle.com*. Oracle Corporation. . Retrieved 2010-08-23.
"Oracle has been a leading and substantive supporter of Java since its emergence in 1995 and takes on the new role as steward of Java technology with a relentless commitment to fostering a community of participation and transparency."
- [21] "1.2 Design Goals of the Java™ Programming Language" (<http://java.sun.com/docs/white/langenv/Intro.doc2.html>). Java.sun.com. 1999-01-01. . Retrieved 2010-06-09.
- [22] Andy Patrizio (2006). "Sun Embraces Linux With New Java License" (<http://www.internetnews.com/dev-news/article.php/3606656>). *Internet News*. Web Media Brands. . Retrieved 2009-09-29.
- [23] "Java for Mac OS X" (<http://developer.apple.com/java/>). *Apple Developer Connection*. Apple. . Retrieved 2009-09-29.
- [24] "Microsoft Java Virtual Machine Support" (<http://www.microsoft.com/mscorp/java/default.mspx>). Microsoft.com. . Retrieved 2010-06-09.
- [25] "Java SE – Licensees" (<http://java.sun.com/javase/licensees.jsp>). Java.sun.com. 2008-08-12. . Retrieved 2010-06-09.
- [26] James Niccolai (January 23, 2001). "Sun, Microsoft settle Java lawsuit" (<http://www.javaworld.com/javaworld/jw-01-2001/jw-0124-iw-mssuncourt.html>). *JavaWorld* (IDG). . Retrieved 2008-07-09.
- [27] Jelovic, Dejan. "Why Java will always be slower than C++" (http://www.jelovic.com/articles/why_java_is_slow.htm). . Retrieved 2008-02-15.
- [28] "Symantec's Just-In-Time Java Compiler To Be Integrated Into Sun JDK 1.1" (http://www.symantec.com/about/news/release/article.jsp?prid=19970407_03). .
- [29] <http://shootout.alioth.debian.org/u32q/which-programming-languages-are-fastest.php?gcc=on&gpp=on&javasteady=on&java=on&csharp=on&javaxint=on&python3=on&python=on&jruby=on&php=on&perl=on&calc=chart>
- [30] "NullPointerException" (<http://java.sun.com/j2se/1.4.2/docs/api/java/lang/NullPointerException.html>). Java.sun.com. . Retrieved 2010-06-09.
- [31] "Exceptions in Java" (<http://www.artima.com/designtechniques/exceptions.html>). Artima.com. . Retrieved 2010-08-10.
- [32] "Using applet, object and embed Tags" (http://download.oracle.com/javase/1.5.0/docs/guide/plugin/developer_guide/using_tags.html). oracle.com. . Retrieved 2010-10-14.
- [33] "Deploying Applets in a Mixed-Browser Environment" (http://download.oracle.com/javase/1.5.0/docs/guide/plugin/developer_guide/using_tags.html#mixed). oracle.com. . Retrieved 2010-10-14.
- [34] <http://java.sun.com/javaee/6/docs/api/javax/servlet/package-summary.html>
- [35] <http://java.sun.com/javaee/6/docs/api/javax/servlet/GenericServlet.html>
- [36] [http://java.sun.com/javaee/6/docs/api/javax/servlet/Servlet.html#service\(javax.servlet.ServletRequest,javax.servlet.ServletResponse\)](http://java.sun.com/javaee/6/docs/api/javax/servlet/Servlet.html#service(javax.servlet.ServletRequest,javax.servlet.ServletResponse))
- [37] <http://java.sun.com/javaee/6/docs/api/javax/servlet/Servlet.html>

- [39] <http://java.sun.com/jvase/6/docs/api/javax/servlet/ServletRequest.html>
- [40] <http://java.sun.com/jvase/6/docs/api/javax/servlet/ServletResponse.html>
- [41] <http://java.sun.com/jvase/6/docs/api/javax/servlet/ServletException.html>
- [42] [http://java.sun.com/jvase/6/docs/api/javax/servlet/ServletResponse.html#setContentType\(java.lang.String\)](http://java.sun.com/jvase/6/docs/api/javax/servlet/ServletResponse.html#setContentType(java.lang.String))
- [43] [http://java.sun.com/jvase/6/docs/api/javax/servlet/ServletResponse.html#getWriter\(\)](http://java.sun.com/jvase/6/docs/api/javax/servlet/ServletResponse.html#getWriter())

References

- James Gosling, *A brief history of the Green project* (<https://duke.dev.java.net/green/>). Java.net, no date [ca. Q1/1998]. Retrieved April 29, 2007.
- James Gosling, Bill Joy, Guy Steele, and Gilad Bracha, *The Java language specification*, third edition. Addison-Wesley, 2005. ISBN 0-321-24678-0 (see also online edition of the specification (<http://java.sun.com/docs/books/jls/index.html>)).
- Tim Lindholm and Frank Yellin. *The Java Virtual Machine specification*, second edition. Addison-Wesley, 1999. ISBN 0-201-43294-3 (see also online edition of the specification (<http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html>)).

External links

- Java: Java for End-users (<http://www.java.com/>)
- Oracle: Developer Resources for Java Technology (<http://www.oracle.com/technetwork/java/>).
- Sun Microsystems: Java Language Specification 3rd Edition (http://java.sun.com/docs/books/jls/third_edition/html/j3TOC.html).
- Java SE 6 API Javadocs
- A Brief History of the Green Project (<https://duke.dev.java.net/green/>)
- Michael O'Connell: Java: The Inside Story (<http://sunsite.uakom.sk/sunworldonline/swol-07-1995/swol-07-java.html>), SunWorld, July 1995.
- Patrick Naughton: Java Was Strongly Influenced by Objective-C (<http://cs.gmu.edu/~sean/stuff/java-objc.html>) (no date).
- David Bank: The Java Saga (<http://www.wired.com/wired/archive/3.12/java.saga.html>), *Wired* Issue 3.12 (December 1995).
- Shahrooz Feizabadi: A history of Java (http://ei.cs.vt.edu/~wwwbtb/book/chap1/java_hist.html) in: Marc Abrams, ed., *World Wide Web – Beyond the Basics*, Prentice Hall, 1998.
- Patrick Naughton: The Long Strange Trip to Java (<http://www.blinkenlights.com/classiccmp/javaorigin.html>), March 18, 1996.
- Open University (UK): M254 Java Everywhere (<http://computing.open.ac.uk/m254/>) (free open content documents).
- is-research GmbH: List of programming languages for a Java Virtual Machine (<http://www.is-research.de/info/vmlanguages/>).
- How Java's Floating-Point Hurts Everyone Everywhere (<http://www.eecs.berkeley.edu/~wkahan/JAVAhurt.pdf>), by W. Kahan and Joseph D. Darcy, University of California, Berkeley.

Article Sources and Contributors

Java (programming language) *Source:* <http://en.wikipedia.org/w/index.php?oldid=413468184> *Contributors:* -Barry-, 16@r, 172.176.26.xxx, 1exec1, 1w0lfblake, 4twenty42o, 7, =JaCyX=-, @modi, ABCD, AJim, Aaron Bowen, Aaron Rotenberg, Abarnea 2000, Abelson, Adam1213, AdamH, Adamacious, Adashiel, Addaintstopmme, Addshore, Aditya, AdjustShift, Adw2000, Ae-a, Aeons, Aesopos, Agrawawa, Ahoerstemeier, Air pacquiaio, Aitias, Aiyizo, Aka042, Akamad, Akersmc, Aksi great, Aktsu, Alai, Alainr345, Alan Rockefeller, Alansohn, AlbertCahalan, Alerante, Alex LE, Alex.atkins, Alex.muller, Alexdethier, Alexius08, Alfio, Alhoori, Alicam8, AlistairMcMillan, Allan McInnes, Alterego, Altmany, Alvin-cs, Am088, Amareto2, Amazon911, Ambarishmohan, Amicon, Armei80, Amitchaudhary, Ammubhave, Ancheta Wis, Andonic, Andre Engels, Andrea Parri, Andres, Andrewferrier, Andrewlp1991, Andy Dingley, Andrewpandy.UK, Anger22, Angusmclellan, Anirudhsshastry, Anirudhvyas010, Anizzah, Anon lynx, Anoopan, Antaeus Feldspar, Antandrus, Antidrugue, Anwar saatad, Aphonik, Arabic Pilot, Arbitrarily0, ArchMageZeratuL, ArchNemesis, Archenzo, Ardonik, Ardrick47, Aremith, Arloz, Armando82, Arnehans, Arrenlex, AscendantOat, Asuhaspatil, Austin Hair, Averell23, B3t, BBUCommander, Baa, Babarcash, Babeledcas, Babomb, Bacchus87, Bachmann1234, Bact, Badgernet, Banatic, Baricom, Baronet, BarretBonden, Basitj, Batneil, Bboy123, Berowell, Beao, Bedel23, Belem tower, Ben Arnold, BenAveling, Benhocking, Beno1000, Betacommand, Bevo, Biblbroks, BigGuyC, Bijee, Biker Biker, Billy On Bannana Peels, Biot, Bissingler, Blablablob, Blankfrack, Bluemoose, Blurpeace, Bobblewik, Bobo192, Boing! said Zebedee, Booyabazooka, Borislav, Bovlb, Bravegag, Brick Thrower, Brion VIBBER, Brossow, BruceMagnus, Bubba-, Buchanan-Hermit, Bulatych, BurmSky, Burzmali, Byronknoll, Bytbox, CAKira, Cal 1234, Calvin 1998, Calvinaustin, Canadafreakazoid, CanadianLinuxUser, Cananian, Candear, CanisRufus, Cap'n Refsmaat, CapitalR, Capricorn42, CaribDigita, Casperl, Cat-five, Catamorphism, Catgut, Cburnett, Centrx, CesarB, Cgs, Chandrasekar78, Chaotic cultist, Charles Matthews, CharlesC, ChasingsoL, Chazwatson, Chealer, Cheesy mike, Chei, Cheung1303, Chiaweihuang, Chinhtn2k3, ChioK, Chip Zero, Chiprunner, Chocolateboy, Chowbok, Chrisk102, Chrissyboi, ChristianEdwardGruber, ChristopheS, Christopher Mahan, Chu Jetcheng, Chuayw2000, Chuq, Chzz, ClancCP, Classical Esther, Cleared as filed, ClockworkLunch, Closedmouth, Clydedoris, Coffee, Coley s, Cometsyles, Connelly, Conor H., Conversion script, Coolshit, CorpX, Corti, Cp98ak, Cpl Syx, Crazz bug 5, Creidieki, CuningLinguist, Cureden, Curmudgeon99, Curps, Cwfrei, Cwllsheep, Cyan, Cybercobra, CyborgTosser, Cynic783, Cyp, Cyrius, D'Agosta, D6, DARTH SIDIOUS 2, DGG, DJ Clayworth, DMacks, DNewhall, Daf, Daimengrui, Damian Yerrick, Dan aka jack, Danakil, Dancraggs, Danhash, Daniel Brockman, DanielCardenas, DanielTroX, Danrah, Darklliac, Dave Runger, David Gerard, Davidjk, Davidweiner23, Dawnsseeker2000, Dbiagioli, Dcoetzee, Deprice555, DeadEyeArrow, Debajit, Debashish, Debeo Morium, Decagon, Delian31, Delirium, Delldot, Delta G, DennisWithem, Derek farn, DetlevSchm, Dgies, Dgwarwick, Dhoom, Diberri, Dibujon, Diderot's dreams, DigiPen92, Dillard421, DineshMungra, Dingskes, Dipet343, Discospinster, Djsuess, DII99, Dmyersturnbul, Dominic7848, Don-vip, Donama, Donaulio, Doradus, Doug Bell, Dpark, Dr-unifex, Dr. Vicodine, Drldiot, Drakkos, Drappel, Dreadstar, Dream of Goats, DreamGuy, Drmies, DropDeadGorgias, Dterei, Dumitru Gherea, Dysprosia, Dystopianrj, Eagleal, Easyas12c, EbulaJonez, Ed Poor, Edcolins, Edknol, Edward, EdwardMcBride, Edwin.wei, Efriedman, Egomaniac, Eik Korell, Ekashp, ElBenevolente, Elf, Ellenaz, Ellmist, Eloquence, Elvarg, Ems2, Enchanter, EngineerScotty, Enough2000, EpiQ SkLl, Er Komandante, ErKURITA, Errandir, Ervinn, Essjay, Ethridgela, Etz Haim, Euchiasmus, Everyking, Evice, Evil Monkey, Evil saltine, Evildeathmath, Excirial, Fabricidosanjossilva, FactChecker1199, Fagstein, Fakahi, Falcon300000, Falcon8765, Fang Aili, Fasca, Fashionslide, Fast.ch, Fasten, Favonian, Fennec, Ferdinand Pienaar, Feureau, Ffi1959, Fieldday-sunday, Finlay McWalter, Flamingantichimp, Flash200, FlavioMattos, FlavrSavr, Flemnira, Florentyna, Fogger, Fotinakis, Frap, Frecklefoot, Fredrik, Freedom to share, FreplySpang, Ftiercel, Fuchsias, Funandtrvl, Furrykef, Fx2, Gail, Gaius Cornelius, Galaxy001, Galwhaa, Gamma, Gary King, Garyateaux, Garyzz, Gazwim, Gdavidp, Ged UK, Geekler, Generalguy11, Georgia guy, Giftlife, Gilgamesh, Gimme danger, Gimmekat, Givings, Glassher, Glenn Maddox, Glezos, Gmoore19, GnuDoyng, Gogo Dodo, Gortsack, Gracenotes, GraemeL, Graffiti, Grandscribe, Graue, Great Cthulhu, Green caterpillar, Green meklar, Greenrd, Gronky, Grunt, Gscshoyru, Gudeldar, Gurch, Gurmklur, Gutworth, Guusbosman, Guy Peters, Guyjohnston, Gwern, Gzkn, Gökhan, H-b-g, H4xx0r, Hadal, Haham hanuka, HalfShadow, Hao2lian, HappyInGeneral, Harai100, Hariva, Harm.frielink, HarmonicSphere, HarryAlffa, Harshadoak, Haseo9999, Havarhen, Hayabusa future, Hdante, Hedoluna, Henning Makholm, Henrygb, HereToHelp, Herevegrid, Hfstedge, Hgferman, Hirudo, Hirzel, Hmains, HoganLong, Hooperbloob, Hosterweis, Hrshtkumar, Hu12, Husond, IBlender, Ideogram, Imaginationac, Immunize, Int19h, Ioakar, Iodine, Iridescent, Irish Souffle, Irishguy, Ishanthasiribaddana, J Di, J.J.Sagnella, J.delanoy, J.I.barthel, J7, JEBrown87544, JHeinonen, JIP, JLaTondre, JOptionPane, JTN, JWaide, Jabbercock, Jaglnx, JamesBWatson, Jamesday, Jarchitect, Java jack jan, Javageek212, Javawizard, Jay, Jay Gatsby, Jaybee, Jaydeki, Jcwe69, Jdforrester, Jeff G., Jeffq, Jeffrey Mall, Jeffuit, Jeltz, Jemijohn, JenniferHeartsU, Jeronimo, Jesse Viviano, JesseHogan, Jglick, Jibijibij, Jijithp, JimWae, Jimguot, Jiy, Jj137, Jjaazz, Jleedev, Jmendez, Joelr31, Joerite, John Hendrixx, John Vandenberg, JohnCongerton, Johnuniq, Jojit fb, Jonabbey, Jonathanischoice, Jondel, Jonik, Jopo sf, JorgePeixoto, Joseph Solis in Australia, JoshHolloway, Joshiaschin79, Jsavit, Juansempere, Julian Mendez, Junes, Jusdafax, Justforasecond, KUSam, Kaldari, Kamalraja, Kamasutra, Kariteh, Karl-Henner, Kbolino, Kcirb, Keshi1181, Keith Azzopardi, KelleyCook, Kevin Saff, KevinMalachowski, Keycard, Khakipuce, Khalid hassani, Killick, Kingturtle, Kkm010, Kks krishna, Klausness, Klingpl0x, KnowledgeOfSelf, Knyf, Kona1611, Koyaanis Qatsi, Kozuch, Kprobst, Kula85, Kungfuadam, Kurrgo master of planet x, Kuru, Kvdveer, Kwaku, Kwalsh5, Kyle Baggs, LFaraone, Landryhc, Lavellem, Leafyplant, LeaveSleaves, Lee Daniel Crocker, Lee J Haywood, LeeHunter, LeilaniLad, Lerdthenerd, LestatdeLioncourt, Levin, Lights, Little Mountain 5, LittleDana, Loadmaster, Logariasmo, Loizbec, Lpetraziska, Lucas Malor, Luk, Lukeja, Lumingz, Luna Santin, Lupin, Lysander89, M1chu, M4gnum0n, MER-C, MIT Trekkie, MYC36, MaBe, Magicfox13, Magioladitis, Magister Mathematicae, Magnus Manske, Mahanga, MaikSchreiber, Mailer diablo, Mainepenguin, Majorly, Malcohol, Mani1, Maoj-wsu-MC, MarSch, MarXidat, Maraist, Marcelbutucea, Marcoandre, Mark Harrison, Mark Renier, MarkSweep, Marky1981, Marlow4, Martarius, Marteau, Marty360, Marty01912, Massyett, Matherson, MattGiuca, Matthew Yeager, Matthewpun, Matthieu fr, Mav, Max Schwarz, MaxEnt, Maxim, McClain, Meara, Mecanismio, MeekMark, Meekohi, Melnaakeeb, Melsaran, Menchi, Mentifisto, Metamorf, Mfb52, Mhoribec, Michaelneale, Mikademos, Mike0001, Mikelo.Arbaro, Mikm, Minesweeper, Minghong, Minipie8, Minkythecat, Minute Lake, Mipadi, Mirror Vax, Misantropo, Misza13, Mitchellfx, Mjroots, Modster, Modulatum, Mohamedloey, Mohsens, Molteanu, Momet, Monkbel, MoraSique, Morte, Morwen, Mr link, MrCoder, MrJones, Mritunjai, Mrstonyk, Mussumangani, Mleslie, MuZemiece, Mucus, Mxn, Myanw, Mzajac, N4te, NJM, Naddy, Nanshu, NapoliRoma, NathanBeach, Ncmathsadist, Neilc, Nepheлин, Nephtes, Neurolysis, NevilleDNZ, NewEnglandYankee, Newsmaestro, Nick R Hill, Nick125, NickBush24, Nigelj, Nikai, Nikhil.bandekar, Nikhilgupta1270, Nil Einne, Nimur, Nixeagle, Nmrdr, Nnp, NoirNoir, Nopetro, Npovmachine, Ntsimp, NuclearWarfare, Nwbeeson, Oatmealcookiemon, Oblivious, Odinjobs, Ohoitsjamie, OIEnglish, Oldadamml, Oleg Alexandrov, Oli Filth, Onefortytwotwo, Onehundreddandtwo, Open Source Guy, Opt 05, Orderud, Orisino, OscarTheCat3, OwenBlacker, OwenX, Oysterguitarist, P-unit, Polyglut, PGibbons, PMJain, Pagingmrherman, Pascaltv, Patman g, Patrick987, Paul Murray, Paul Richter, Paul99, Paules nl, Pavel Vozenilek, Pawan shanku, Peepeedia, Pengo, PerLundberg, Perfecto, Peter Delmonte, Peter Griffin, Peter lawrey, Peterl, Pkg, PhageRules1, Phantomsteve, Pharos, Phgaio, Phil websurfer@yahoo.com, PhilKnight, Philipwhiuk, Phoenix-forgotten, Piano non troppo, Piet Delpot, Piggpicking, Pinktulp, Pipedreamgrey, Pizza Puzzle, Plugwash, PlusMinus, Pmronchi, Pne, Poccil, Polluxian, Polydour, Pontillo, Poor Yorick, Potatoj316, Premvnc, Priyankgokani, Pro Grape, Professor Calculus, Proficient, ProvingReliability, Public Menace, Pundeerd, PuzzletChung, Pwooster, Quadell, Quantumelfmage, Qwertyus, Qwyrxian, R3m0t, Rab8613, Raistlin11325, RandalSchwartz, Ranjith16, Rao crazy, Raul654, Ravchit, Raveendra Lakpriya, Raven4x4x, Raynoism, Rccarman, Rcs, Rdsmith4, Real-Life Sock Puppet, RedWolf, RedZombie125, Redrocket, Redvers, Reisio, Rettetast, RexNL, Rgeorgy, Rhobite, RichF, Rick Jelliffe, Risk one, Rjmfernandes, Rjwimsi, Robert Merkel, Robert Skyhawk, RobertG, Robo.mind, Rodhullandemu, Rogerd, Ron mmi, Ronz, RossPatterson, Rossmann, RoyBoy, Rror, Rrtanz, Rufous, Ruiz, Rursus, Ruud Koot, Rzwitserloot, S-n-ushakov, SAE1962, SF007, SGNDave, SNIyer12, ST47, Saltlakejohn, Sam Hocevar, Sam Korn, Sam jervis, Samohyl Jan, San25872, Santiago Roza (Kq), Saravask, Sasha Slutsker, Sass24, Satanjaya, Scadian, Schapel, Schissel, SchnitzelMannGreek, Scientus, Scjessey, Scm83x, Scovetta, Sdfisher, Sdorman, SeanJA, Seanhan, Search4Lancer, Sebastiangarth, Seraphimblade, Sfmontyo, Shadowjams, Shenme, Siddhartha.kaja, Sigma 7, SimonP, Sinn, Sir Anon, SirWoland, Siroro, Sirworthyduck, Sj, Sjarosz, Sjc, SkyWalker, Sleske, Slike, SlitherM, Smyth, Snowolf, Socablg2, SocratesJedi, Sohil ic, Some P. Erson, Soptep, Spalding, Sparker046, Speedogoo, Spencer, Spiff, SpikeToronto, Spookfish, Spoon!, Spug, Squingynaut, Starblind, Starwiz, Steffen Felbinger, Stephen, Stephen B Streater, Stereo, Stevenj, Stevenasnisk, Stevethemart, Stormie, Styrofoam1994, Subanark, Suffusion of Yellow, Sumoanand, Sunderland06, Super Quinn, SuperU, Superfly Jn, Superm401, Supernova17, Supertouch, Susvolans, SvGeloven, Swintrob, Sylent, Synthiac, Szajd, TOReilly, TRauMa, TakuyaMurata, Talandor, Tar5047, Tariqabjotu, Tasc, TastyPoutine, Tcgunner90, Tcnvc, Teabear, Technopat, Tedickey, Tellyaddict, Template namespace initialisation script, TerokNor, Terrible Tim, Testmod1, The Anome, The Elves Of Dunsimore, The Hokkaido Crow, The Thing That Should Not Be, The wub, TheIphysicist, The2ndflood, TheDarkArchon, TheSpook, Thehefulone, Theresa knott, Theuser, Thingg, Thornelytaylor, Thue, Thumperward, Thunderbolt116, Tillmo, Tim1988, TimTay, Timwhit, Titoxd, Tjansen, Tkgd2007, Tlim7882, Tmaher, Tmbg37, Tom 99, Tommy2010, Tompsci, Torc2, Tothwolf, Toussaint, Toytoy, Tresiden, Troels Arvin, Tthorley, Ttwaring, Turing, Tushar.kute, Tustin2121, Tut21, Tvynr, Twister nt, Tyraios, Tyrol5, Tyw7, Ubuntu2, Ultimux, Uncle G, UnitedStatesian, Unixer, Unixxx, Urod, Uselesswarrior, Utcursch, UtherSRG, VX, Valerio81, Vanished user 39948282, VantagePoint, Vchimpanzee, VeLoCiTy89, Vera.tetrix, Viciodk, Viebel, Viento, Violetriga, Visor, Vizspring, Vocaro, Volt4ire, Vvidetta, WJBscribe, Wasimbargujar, Wayne Slam, Wbrameld, Wdyoung, Web-Crawling Stickler, Weregerbil, Wesley crossman, Wgw2024, Wgw4, White Cat, Whosyourjudas, Wiggin15, Wik, WikiLeon, Wikianon, Wikipelli, Wikipendant, Wikiwerks, Wikiwikiwif, Will Beback Auto, William Allen Simpson, William987, Wimt, Windchaser, Winterst, Wknight94, Wlievens, Wmahan, Wonko, Wootery, Wowus, Wrp103, Wsaryada, WurmWoode, Ww, Wyredchris, Wzww, X deadly Snipez, Xan 213, Xaosflux, Xelgen, Xezbeth, Xinconnu, Xpodmaniac, Xproject, Yamaguchi先生, Yamamoto Ichiro, Yamla, Yozh, Yulraco, Yurik, Yurivict, ZacBowling, Zawersh, Zapot, Zekeo, Zenohockey, Zijian, Zik-Zak, Ztothefith, Zxcvbnm, Zyphrix, Ævar Arnfrjörð Bjarmason, 六脉飞天牛, 2569 anonymous edits

Image Sources, Licenses and Contributors

File:Java logo.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Java_logo.svg *License:* unknown *Contributors:* Eastmain, MBisanz, Open Source Guy, Sophus Bie, The wub, TimTay, Tkgd2007, 2 anonymous edits

File:Wikibooks-logo-en.svg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Wikibooks-logo-en.svg> *License:* logo *Contributors:* User:Bastique, User:Ramac

File:Wave.svg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Wave.svg> *License:* unknown *Contributors:* sbmehta converted to SVG from Sun Microsystems AI version.

Image:JavaPlatform.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:JavaPlatform.png> *License:* unknown *Contributors:* Sun Microsystems, Inc.

Image:Wave.svg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Wave.svg> *License:* unknown *Contributors:* sbmehta converted to SVG from Sun Microsystems AI version.

License

Creative Commons Attribution-Share Alike 3.0 Unported
<http://creativecommons.org/licenses/by-sa/3.0/>
