The Java programming language has 50 *keywords*. Each keyword has a specific meaning in the language. You can't use a keyword for anything other than its pre-assigned meaning.

The following table lists Java's keywords.

Keyword	What It Does
`abstract`	Indicates that the details of a class, a method, or an interface are given elsewhere in the code.
`assert`	Tests the truth of a condition that the programmer believes is true.
`boolean`	Indicates that a value is either `true` or `false`.
`break`	Jumps out of a loop or `switch`.
`byte`	Indicates that a value is an 8-bit whole number.
`case`	Introduces one of several possible paths of execution in a `switch` statement.
`catch`	Introduces statements that are executed when something interrupts the flow of execution in a `try` clause.
`char`	Indicates that a value is a character (a single letter, digit, punctuation symbol, and so on) stored in 16 bits of memory.
`class`	Introduces a class — a blueprint for an object.

`const`	You can’t use this word in a Java program. The word has no meaning but, because it’s a keyword, you can’t create a variable named const.
`continue`	Forces the abrupt end of the current loop iteration and begins another iteration.
`default`	Introduces a path of execution to take when no case is a match in a `switch`statement.
`do`	Causes the computer to repeat some statements over and over again (for instance, as long as the computer keeps getting unacceptable results).
`double`	Indicates that a value is a 64-bit number with one or more digits after the decimal point.
`else`	Introduces statements that are executed when the condition in an `if` statement isn’t true.
`enum`	Creates a newly defined type — a group of values that a variable can have.
`extends`	Creates a subclass — a class that reuses functionality from a previously defined class.
`final`	Indicates that a variable’s value cannot be changed, that a class’s functionality cannot be extended, or that a method cannot be overridden.
`finally`	Introduces the last will and testament of the statements in a `try` clause.

`float`	Indicates that a value is a 32-bit number with one or more digits after the decimal point.
`for`	Gets the computer to repeat some statements over and over again (for instance, a certain number of times).
`goto`	You can't use this word in a Java program. The word has no meaning. Because it's a keyword, you can't create a variable named `goto`.
`if`	Tests to see whether a condition is true. If it's true, the computer executes certain statements; otherwise, the computer executes other statements.
`implements`	Indicates that a class provides bodies for methods whose headers are declared in an interface.
`import`	Enables the programmer to abbreviate the names of classes defined in a package.
`instanceof`	Tests to see whether a certain object comes from a certain class.
`int`	Indicates that a value is a 32-bit whole number.
`interface`	Introduces an interface. An interface is like a class but, for the most part, an interface's methods have no bodies.
`long`	Indicates that a value is a 64-bit whole number.

`native`	Enables the programmer to use code that was written in a language other than Java.
`new`	Creates an object from an existing class.
`package`	Puts the code into a package — a collection of logically related definitions.
`private`	Indicates that a variable or method can be used only within a certain class.
`protected`	Indicates that a variable or method can be used in subclasses from another package.
`public`	Indicates that a variable, class, or method can be used by any other Java code.
`return`	Ends execution of a method and possibly returns a value to the calling code.
`short`	Indicates that a value is a 16-bit whole number.
`static`	Indicates that a variable or method belongs to a class, rather than to any object created from the class.
`strictfp`	Limits the computer's ability to represent extra large or extra small numbers when the computer does intermediate calculations on `float` and `double` values.
`super`	Refers to the superclass of the code in which the word `super` appears.

`switch`	Tells the computer to follow one of many possible paths of execution (one of many possible cases), depending on the value of an expression.
`synchronized`	Keeps two threads from interfering with one another.
`this`	A self-reference — refers to the object in which the word `this` appears.
`throw`	Creates a new exception object and indicates that an exceptional situation (usually something unwanted) has occurred.
`throws`	Indicates that a method or constructor may pass the buck when an exception is thrown.
`transient`	Indicates that, if and when an object is serialized, a variable's value doesn't need to be stored.
`try`	Introduces statements that are watched (during runtime) for things that can go wrong.
`void`	Indicates that a method doesn't return a value.
`volatile`	Imposes strict rules on the use of a variable by more than one thread at a time.
`while`	Repeats some statements over and over again (as long as a condition is still true).