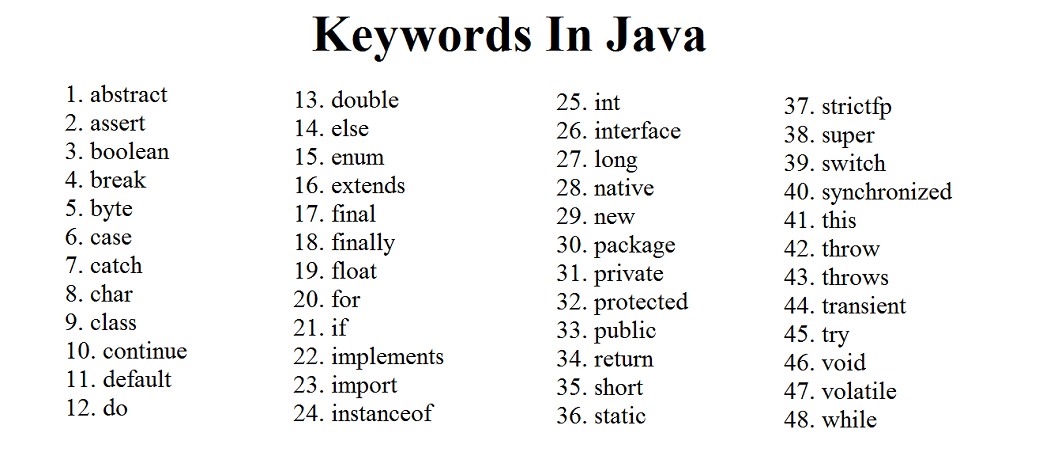
# Keywords

# **Java Language Keywords**

Here is a list of keywords in the Java programming language. You cannot use any of the following as identifiers in your programs. The keywords const and goto are reserved, even though they are not currently used. true, false, and null might seem like keywords, but they are actually literals; you cannot use them as identifiers in your programs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| abstract | continue | for | new | switch |
| assert\*\*\* | default | goto\* | package | synchronized |
| boolean | do | if | private | this |
| break | double | implements | protected | throw |
| byte | else | import | public | throws |
| case | enum\*\*\*\* | instanceof | return | transient |
| catch | extends | int | short | try |
| char | final | interface | static | void |
| class | finally | long | strictfp\*\* | volatile |
| const\* | float | native | super | while |

|  |  |  |
| --- | --- | --- |
| \* |  | not used |
| \*\* |  | added in 1.2 |
| \*\*\* |  | added in 1.4 |
| \*\*\*\* |  | added in 5.0 |



# Java Keywords

Java keywords are also known as reserved words. Keywords are particular words that act as a key to a code. These are predefined words by Java so they cannot be used as a variable or object name or class name.

## List of Java Keywords

A list of Java keywords or reserved words are given below:

1. **[abstract](https://www.javatpoint.com/abstract-keyword-in-java)**:**** Java abstract keyword is used to declare an abstract class. An abstract class can provide the implementation of the interface. It can have abstract and non-abstract methods.
2. **[boolean:](https://www.javatpoint.com/boolean-keyword-in-java)** Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.
3. **[break](https://www.javatpoint.com/java-break)**:**** Java break keyword is used to break the loop or switch statement. It breaks the current flow of the program at specified conditions.
4. **[byte](https://www.javatpoint.com/byte-keyword-in-java)**:**** Java byte keyword is used to declare a variable that can hold 8-bit data values.
5. **[case](https://www.javatpoint.com/case-keyword-in-java)**:**** Java case keyword is used with the switch statements to mark blocks of text.
6. **[catch](https://www.javatpoint.com/try-catch-block)**:**** Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.
7. **[char](https://www.javatpoint.com/char-keyword-in-java)**:**** Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters
8. **[class](https://www.javatpoint.com/class-keyword-in-java)**:**** Java class keyword is used to declare a class.
9. **[continue](https://www.javatpoint.com/java-continue)**:**** Java continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
10. **[default](https://www.javatpoint.com/default-keyword-in-java)**:**** Java default keyword is used to specify the default block of code in a switch statement.
11. **[do](https://www.javatpoint.com/java-do-while-loop)**:**** Java do keyword is used in the control statement to declare a loop. It can iterate a part of the program several times.
12. **[double](https://www.javatpoint.com/double-keyword-in-java)**:**** Java double keyword is used to declare a variable that can hold 64-bit floating-point number.
13. **[else](https://www.javatpoint.com/java-if-else)**:**** Java else keyword is used to indicate the alternative branches in an if statement.
14. **[enum](https://www.javatpoint.com/enum-in-java)**:**** Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.
15. **[extends](https://www.javatpoint.com/inheritance-in-java)**:**** Java extends keyword is used to indicate that a class is derived from another class or interface.
16. **[final](https://www.javatpoint.com/final-keyword)**:**** Java final keyword is used to indicate that a variable holds a constant value. It is used with a variable. It is used to restrict the user from updating the value of the variable.
17. **[finally](https://www.javatpoint.com/finally-block-in-exception-handling)**:**** Java finally keyword indicates a block of code in a try-catch structure. This block is always executed whether an exception is handled or not.
18. **[float](https://www.javatpoint.com/float-keyword-in-java)**:**** Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.
19. **[for](https://www.javatpoint.com/java-for-loop)**:**** Java for keyword is used to start a for loop. It is used to execute a set of instructions/functions repeatedly when some condition becomes true. If the number of iteration is fixed, it is recommended to use for loop.
20. **[if](https://www.javatpoint.com/java-if-else)**:**** Java if keyword tests the condition. It executes the if block if the condition is true.
21. **[implements](https://www.javatpoint.com/interface-in-java)**:**** Java implements keyword is used to implement an interface.
22. **[import](https://www.javatpoint.com/package)**:**** Java import keyword makes classes and interfaces available and accessible to the current source code.
23. **[instanceof](https://www.javatpoint.com/downcasting-with-instanceof-operator)**:**** Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.
24. **[int](https://www.javatpoint.com/int-keyword-in-java)**:**** Java int keyword is used to declare a variable that can hold a 32-bit signed integer.
25. **[interface](https://www.javatpoint.com/interface-in-java)**:**** Java interface keyword is used to declare an interface. It can have only abstract methods.
26. **[long](https://www.javatpoint.com/long-keyword-in-java)**:**** Java long keyword is used to declare a variable that can hold a 64-bit integer.
27. ****native:**** Java native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface).
28. **[new](https://www.javatpoint.com/new-keyword-in-java)**:**** Java new keyword is used to create new objects.
29. **[null](https://www.javatpoint.com/null-keyword-in-java)**:**** Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.
30. **[package](https://www.javatpoint.com/package)**:**** Java package keyword is used to declare a Java package that includes the classes.
31. **[private](https://www.javatpoint.com/private-keyword-in-java)**:**** Java private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.
32. **[protected](https://www.javatpoint.com/protected-keyword-in-java)**:**** Java protected keyword is an access modifier. It can be accessible within the package and outside the package but through inheritance only. It can't be applied with the class.
33. **[public](https://www.javatpoint.com/public-keyword-in-java)**:**** Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.
34. **[return](https://www.javatpoint.com/return-keyword-in-java)**:**** Java return keyword is used to return from a method when its execution is complete.
35. **[short](https://www.javatpoint.com/short-keyword-in-java)**:**** Java short keyword is used to declare a variable that can hold a 16-bit integer.
36. **[static](https://www.javatpoint.com/static-keyword-in-java)**:**** Java static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is mainly used for memory management.
37. **[strictfp](https://www.javatpoint.com/strictfp-keyword)**:**** Java strictfp is used to restrict the floating-point calculations to ensure portability.
38. **[super](https://www.javatpoint.com/super-keyword)**:**** Java super keyword is a reference variable that is used to refer to parent class objects. It can be used to invoke the immediate parent class method.
39. **[switch](https://www.javatpoint.com/java-switch)**:**** The Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.
40. **[synchronized](https://www.javatpoint.com/synchronization-in-java)**:**** Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.
41. **[this](https://www.javatpoint.com/this-keyword)**:**** Java this keyword can be used to refer the current object in a method or constructor.
42. **[throw](https://www.javatpoint.com/throw-keyword)**:**** The Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exceptions. It is followed by an instance.
43. **[throws](https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws)**:**** The Java throws keyword is used to declare an exception. Checked exceptions can be propagated with throws.
44. **[transient](https://www.javatpoint.com/transient-keyword)**:**** Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.
45. **[try](https://www.javatpoint.com/try-catch-block)**:**** Java try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.
46. ****void:**** Java void keyword is used to specify that a method does not have a return value.
47. **[volatile](https://www.javatpoint.com/volatile-keyword-in-java)**:**** Java volatile keyword is used to indicate that a variable may change asynchronously.
48. **[while](https://www.javatpoint.com/java-while-loop)**:**** Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iteration is not fixed, it is recommended to use the while loop.



**Why we use static final in Java for constants**

Developers don't use the const keyword in Java to mark fields as constants. Instead, they daisy chain the keywords static final in Java to create an arguably global variable with an unchangeable value.

While the const word is reserved in Java, it's unimplemented and any attempt to use it in code will trigger a compile time error. Developers that come to Java from other languages will find the use of the terms static final in Java instead of the const keyword to be unintuitive. However, there's an object-oriented programming reason why one would implement the language this way.

**The difference between static and final in Java**

A developer creates Java classes such as the Person class or the BankAccount class from which they make instances. Think of it this way, a class is like a cookie cutter while an individual instance is like a cookie.

If a developer wants a field never to change after it's been assigned a value, mark that field as final. For example, my bank account number will never change after it's been created, and neither will yours. Obviously, your bank account number is different from mine, so they're not the same for every instance of the BankAccount class. But for each individual instance, the account ID will never change.

**Java's static vs. final keywords**

Now let's look at the Java keyword static. A field marked static is the same for every instance of a class. A bank account might have a static variable that represents the interest rate, for example. When the interest rate changes, it changes for every bank account instance. If tomorrow the interest rate goes from 1% to 2%, it will affect everyone. It's like a global variable that every instance of the class can access. But a static variable is not constant because it can be changed at any time.

Now imagine that a developer wants a value that can't be changed and is the same for every instance of the class. A developer needs to combine the keywords static final to achieve this in Java. **The static keyword means the value is the same for every instance of the class. The final keyword means once the variable is assigned a value it can never be changed. The combination of static final in Java is how to create a constant value.**

Example of static final in Java

Here's a static final example to further demonstrate the point.

The following Account class has three properties:

* accountId of type int marked as final;
* interestRate of type double marked as static; and
* a final static variable named odLimit of type int for the overdraft limit.

class Account {

public Account(int id) { accountId = id; }

//const int dne;

final int accountId;

static double rate = 1.5;

static final int odLimit = 1000;

}

Even though the accountId is marked final, each individual account can have its own unique value. The only rule is that once the value is assigned, it can never be changed.

Account first = new Account(123);

Account second = new Account(456);

// The following prints 123

System.out.println(first.accountId);

// The following prints 456

System.out.println(second.accountId);

// Next line triggers a compile error

// first.accountId = 789;

With the non-final static variable, any instance can change it, but if the value changes, it affects every instance.

first.rate=2.5;

second.interestRate=3.5;

// Both lines print out 3.5

System.out.println(first.interestRate);

System.out.println(second.interestRate);

Finally, with the static final variable, it's both the same for each class and it can't be changed after it's initialized.

// Both lines print out 1000

System.out.println(first.odLimit);

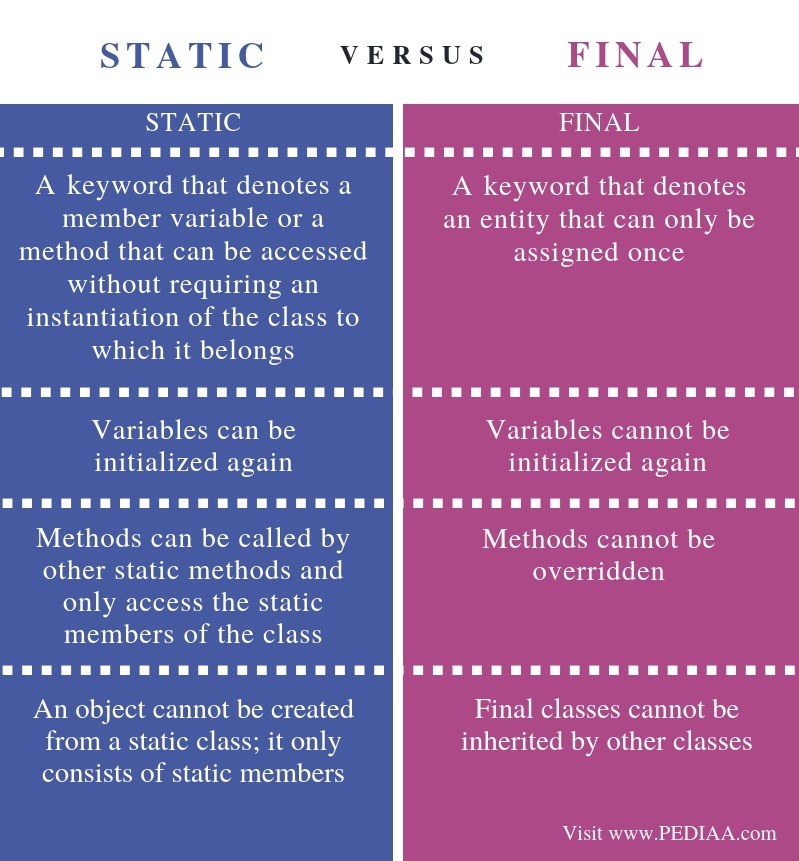
System.out.println(second.odLimit);

// Next line triggers a compile error

// first.odLimit = 500;

**No need for a Java const implementation**

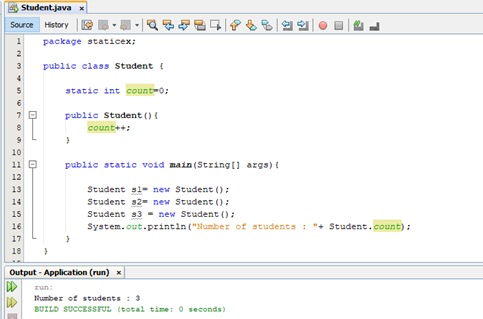
The terms static and final in Java have distinct meanings. The final keyword implies something cannot be changed. The static keyword implies class-level scope. When you combine static final in Java, you create a variable that is global to the class and impossible to change. Developers from other platforms would consider this creation to be equivalent to a global, constant variable. The static final keywords used together eliminate the need for an implementation of Java's const keyword.



## What is Static

A class consists of variables and methods. An object is created from a class; this object can be used to call variable and methods. When a class member is declared with static, it is not required to create an object to call methods and variables. Instead, the class name can be used to call it. In other words, static is a keyword that belongs to the class rather than to the object.

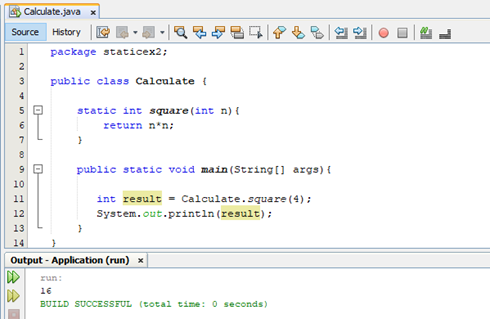
A variable with the static keyword is called a static variable. They are used to refer a common property of a collection of objects. These variables get memory at the time of loading the class. The main advantage of a static variable is that it helps to save memory.



*****Figure 1: Static Variable*****

In the above program, there is a static variable called count. In the constructor, the count is incremented by 1. In the main program, three student objects are created. Printing the count will give the result 3 as there are three objects. The count variable is shared by all objects. Each time when an object is created, the count is incremented by one. When displaying the count, it should be written with the class name (e.g. – Student.count).

A method with the static keyword is known as a static method. An example is as follows.

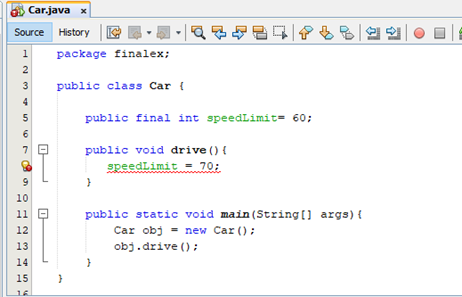


*****Figure 2: Static Method*****

In the above program, square is a static method. It receives an integer value. In the main method, the static method is called, and the value 4 is passed. The answer from the method is stored in the variable result and is finally printed. Here, the class name is used to access the static method. (e.g – Calculate.square(4)).  On the other hand, the static methods cannot use non-static data members or call non-static methods directly. Assessing a non-static variable by a static method will give a compile time error.

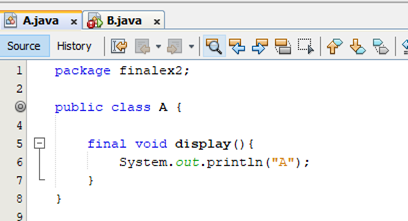
## What is Final

Final is a keyword to restrict the user. It can be used in variables, methods, and classes. A value of a final variable cannot be changed.

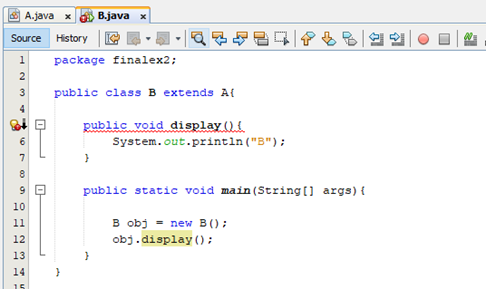


*****Figure 3: Final Variable*****

In the above program, the variable speedLimit is declared as final. Therefore, its value cannot be changed in the drive method. So, it displays an error.



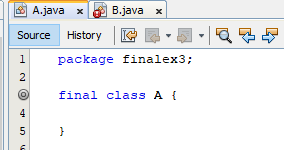
*****Figure 4: Class A*****



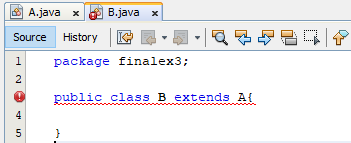
*****Figure 5: Class B*****

In the above program, class A has a final method called display. Class B extends A. So, class B can inherit all the variables and methods of class A.  Class B also has the method display. As the class A display method is final, it does not allow to override that method in class B.

Moreover, a final class cannot be extended as follows.



*****Figure 6: final class*****



*****Figure 7: Class B*****

As the class A is final, it cannot be extended or inherited by class B.