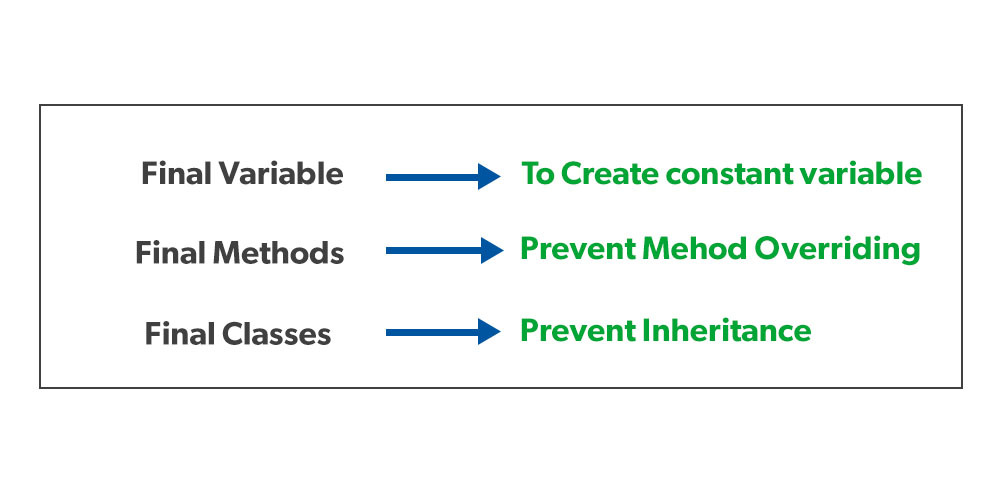
Final Keyword In Java

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:



When a variable is declared with the ***final keyword***, its value can’t be modified, essentially, a constant. This also means that you must initialize a final variable. If the final variable is a reference, this means that the variable cannot be re-bound to reference another object, but the internal state of the object pointed by that reference variable can be changed i.e. you can add or remove elements from the [final array](https://www.geeksforgeeks.org/final-arrays-in-java/) or final collection. It is good practice to represent final variables in all uppercase, using underscore to separate words.

**Observation 1:**When to use a final variable?

The only difference between a normal variable and a final variable is that we can re-assign the value to a normal variable but we cannot change the value of a final variable once assigned. Hence final variables must be used only for the values that we want to remain constant throughout the execution of the program.

**Observation 2:**Reference final variable?

When a final variable is a reference to an object, then this final variable is called the reference final variable. For example, a final StringBuffer variable looks defined below as follows:

final StringBuffer sb;

As we all know that a final variable cannot be re-assign. But in the case of a reference final variable, the internal state of the object pointed by that reference variable can be changed. Note that this is not re-assigning. This property of *final* is called *non-transitivity*. To understand what is meant by the internal state of the object as shown in the below example as follows:

|  |
| --- |
| // Java Program to demonstrate  // Reference of Final Variable    // Main class  **class** GFG {        // Main driver method  **public** **static** **void** main(String[] args)      {          // Creating sn object of StringBuilder class          // Final reference variable  **final** StringBuilder sb = **new** StringBuilder("Geeks");            // Printing the element in StringBuilder object          System.out.println(sb);            // changing internal state of object reference by          //  final reference variable sb          sb.append("ForGeeks");            // Again printing the element in StringBuilder          // object after appending above element in it          System.out.println(sb);      }  } |

***Remember:****When a final variable is created inside a method/constructor/block, it is called local final variable, and it must initialize once where it is created. See below program for local final variable.*

### Final classes

When a class is declared with *final* keyword, it is called a final class. A final class cannot be extended(inherited).

**There are two uses of a final class:**

**Usage 1:** One is definitely to prevent [inheritance](https://www.geeksforgeeks.org/inheritance-in-java/), as final classes cannot be extended. For example, all [Wrapper Classes](https://www.geeksforgeeks.org/wrapper-classes-java/) like [Integer](https://www.geeksforgeeks.org/java-lang-integer-class-java/), [Float](https://www.geeksforgeeks.org/java-lang-float-class-in-java/), etc. are final classes. We can not extend them.

**Usage 2:** The other use of final with classes is to [create an immutable class](https://www.geeksforgeeks.org/create-immutable-class-java/) like the predefined [String](https://www.geeksforgeeks.org/string-class-in-java/) class. One can not make a class immutable without making it final.

### **Final Methods**

When a method is declared with *final* keyword, it is called a final method. A final method cannot be [overridden](https://www.geeksforgeeks.org/overriding-in-java/). The [Object](https://www.geeksforgeeks.org/object-class-in-java/) class does this—a number of its methods are final. We must declare methods with the final keyword for which we are required to follow the same implementation throughout all the derived classes.

# Java instanceof

The **java instanceof operator** is used to test whether the object is an instance of the specified type (class or subclass or interface).

The instanceof in java is also known as type *comparison operator* because it compares the instance with type. It returns either true or false. If we apply the instanceof operator with any variable that has null value, it returns false.

<https://www.scientecheasy.com/2021/11/java-final-interview-questions.html/>