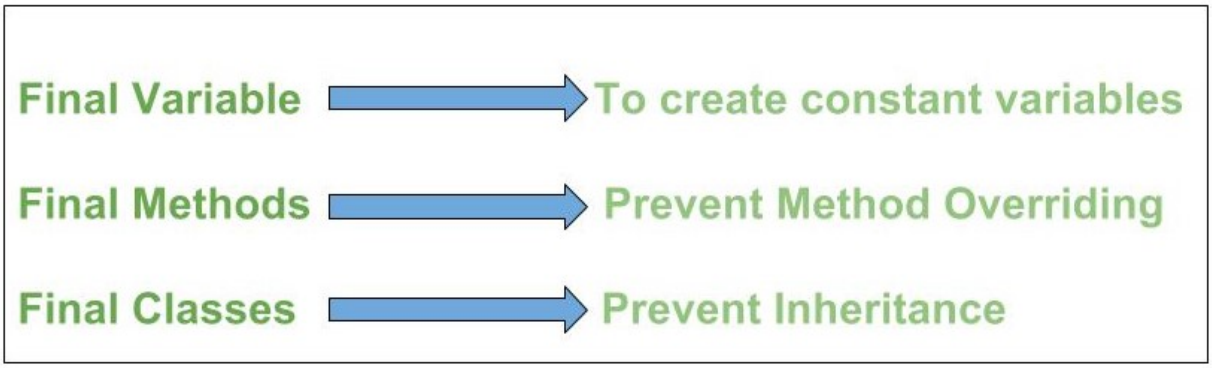
final keyword in java

final keyword is used in different contexts. First of all, *final* is a non-access modifier applicable **only to a variable, a method or a class**.Following are different contexts where final is used.



Final variables

When a variable is declared with *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 internal state of the object pointed by that reference variable can be changed i.e. you can add or remove elements from final array or final collection. It is good practice to represent final variables in all uppercase, using underscore to separate words.

Examples :

```
// a final variable
final int THRESHOLD = 5;
// a blank final variable
final int THRESHOLD;
// a final static variable PI
static final double PI = 3.141592653589793;
// a  blank final static  variable
static final double PI;
```

Initializing a final variable :

We must initialize a final variable, otherwise compiler will throw compile-time error.A final variable can only be initialized once, either via an initializer or an assignment statement. There are three ways to initialize a final variable :

1. You can initialize a final variable when it is declared.This approach is the most common. A final variable is called **blank final variable**,if it is **not** initialized while declaration. Below are the two ways to initialize a blank final variable.
2. A blank final variable can be initialized inside instance-initializer block or inside constructor. If you have more than one constructor in your class then it must be initialized in all of them, otherwise compile time error will be thrown.
3. A blank final static variable can be initialized inside static block.

Let us see above different ways of initializing a final variable through an example.

```
//Java program to demonstrate different
// ways of initializing a final variable

class Gfg
{
    // a final variable
    // direct initialize
    final int THRESHOLD = 5;

    // a blank final variable
    final int CAPACITY;

    // another blank final variable
    final int  MINIMUM;

    // a final static variable PI
    // direct initialize
    static final double PI = 3.141592653589793;

    // a  blank final static  variable
    static final double EULERCONSTANT;

    // instance initializer block for
    // initializing CAPACITY
    {
```

```
        CAPACITY = 25;
    }

    // static initializer block for
    // initializing EULERCONSTANT
    static{
        EULERCONSTANT = 2.3;
    }

    // constructor for initializing MINIMUM
    // Note that if there are more than one
    // constructor, you must initialize MINIMUM
    // in them also
    public GFG()
    {
        MINIMUM = -1;
    }

}
```

When to use a final variable :

The only difference between a normal variable and a final variable is that we can re-assign 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 program.

Reference final variable :
When a final variable is a reference to an object, then this final variable is called reference final variable. For example, a final StringBuffer variable looks like

```
final StringBuffer sb;
```

As you know that a final variable cannot be re-assign. But in case of a reference final variable, 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 mean by internal state of the object, see below example :

```
// Java program to demonstrate
// reference final variable

class Gfg
{
    public static void main(String[] args)
    {
        // a final reference variable sb
        final StringBuilder sb = new StringBuilder("Geeks");

        System.out.println(sb);
```

```
        // changing internal state of object
        // reference by final reference variable sb
        sb.append("ForGeeks");

        System.out.println(sb);
    }
}
```

Output:

```
Geeks
GeeksForGeeks
```

The *non-transitivity* property also applies to arrays, because arrays are objects in java. Arrays with final keyword are also called final arrays.

Note :

1. As discussed above, a final variable cannot be reassign, doing it will throw compile-time error.

```
// Java program to demonstrate re-assigning
// final variable will throw compile-time error

class Gfg
{
    static final int CAPACITY = 4;

    public static void main(String args[])
    {
        // re-assigning final variable
        // will throw compile-time error
        CAPACITY = 5;
    }
}
```

Output


```
Compiler Error: cannot assign a value to final variable CAPACITY
```

2. 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

```
// Java program to demonstrate
// local final variable

// The following program compiles and runs fine

class Gfg
{
    public static void main(String args[])
    {
        // local final variable
        final int i;
        i = 20;
        System.out.println(i);
    }
}
```

Output:

```
20
```

3. Note the difference between C++ *const* variables and Java *final* variables. const variables in C++ must be assigned a value when declared. For final variables in Java, it is not necessary as we see in above examples. A final variable can be assigned value later, but only once.
4. *final* with foreach loop : final with for-each statement is a legal statement.

```
// Java program to demonstrate final
// with for-each statement

class Gfg
{
    public static void main(String[] args)
    {
        int arr[] = {1, 2, 3};

        // final with for-each statement
        // legal statement
        for (final int i : arr)
            System.out.print(i + " ");
    }
}
```

Output:

```
1 2 3
```

Explanation : Since the i variable goes out of scope with each iteration of the loop, it is actually re-declaration each iteration, allowing the same token (i.e. i) t be used to represent multiple variables.

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 :

1. One is definitely to prevent inheritance, as final classes cannot be extended. For example, all Wrapper Classes like Integer,Float etc. are final classes. We can not extend them.

```
final class A
{
     // methods and fields
}
// The following class is illegal.
class B extends A
{
    // COMPILE-ERROR! Can't subclass A
}
```

2. The other use of final with classes is to create an immutable class like the predefined String class.You 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. The Object class does this—a number of its methods are final.We must declare methods with final keyword for which we required to follow the same implementation throughout all the derived classes. The following fragment illustrates final keyword with a method:

```
class A
{
    final void m1()
    {
        System.out.println("This is a final method.");
    }
}

class B extends A
{
    void m1()
    {
        // COMPILE-ERROR! Can't override.
        System.out.println("Illegal!");
```

```
    }
}
```

For more examples and behavior of final methods and final classes, please see Using final with inheritance

final vs abstract

Please see abstract in java article for differences between final and abstract.

Related Interview Question(Important) : Difference between final, finally and finalize in Java

This article is contributed by **Gaurav Miglani**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

Unreachable statement using final and non-final variable in Java
final vs Immutability in Java
Using final with Inheritance in Java
Final arrays in Java
Blank Final in Java
final variables in Java
final, finally and finalize in Java
Final local variables in Java
Final static variable in Java
Instance variable as final in Java
Private and final methods in Java
Assigning values to static final variables in Java
volatile keyword in Java
Super Keyword in Java
static keyword in java