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:

1-variable

2-method

3-class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only. We will have detailed learning of these. Let's first learn the basics of final keyword.

Java Final Keyword

- ⇨ Stop Value Change
- ⇨ Stop Method Overridding
- ⇨ Stop Inheritance

- **1) Java final variable**
- **If you make any variable as final, you cannot change the value of final variable(It will be constant).**

- **Example of final variable**
- **There is a final variable speedlimit, we are going to change the value of this variable, but It can't be changed because final variable once assigned a value can never be changed.**

```
class Bike9{
 final int speedlimit=90;//final variable
 void run(){
  speedlimit=400;
 }
 public static void main(String args[]){
 Bike9 obj=new  Bike9();
 obj.run();
 }
}//end of class
```

2) Java final method

If you make any method as final, you cannot override it.

Example of final method

```
class Bike{
  final void run(){System.out.println("running");}
}

class Honda extends Bike{
   void run(){System.out.println("running safely with 100kmph");}

   public static void main(String args[]){
   Honda honda= new Honda();
   honda.run();
   }
}
```

3) Java final class

If you make any class as final, you cannot extend it.

Example of final class

```
final class Bike{}
  class Honda1 extends Bike{
  void run(){System.out.println("running safely with 100kmph");}
      public static void main(String args[]){
 Honda1 honda= new Honda1();
  honda.run();
  }
}
```

- Output:Compile Time Error

Q) Is final method inherited?

Ans) Yes, final method is inherited but you cannot override it. For Example:

```
class Bike{
  final void run(){System.out.println("running...");}
}
class Honda2 extends Bike{
   public static void main(String args[]){
    new Honda2().run();
   }
}
```

- Output:running...

Q) What is blank or uninitialized final variable?

A final variable that is not initialized at the time of declaration is known as blank final variable.

If you want to create a variable that is initialized at the time of creating object and once initialized may not be changed, it is useful. For example PAN CARD number of an employee.

It can be initialized only in constructor.

Example of blank final variable

```
class Student{
int id;
String name;
final String PAN_CARD_NUMBER;
...
}
```

Que) Can we initialize blank final variable?

Yes, but only in constructor. For example:

```
class Bike10{
  final int speedlimit;//blank final variable

  Bike10(){
  speedlimit=70;
  System.out.println(speedlimit);
  }

  public static void main(String args[]){
    new Bike10();
 }
}
```

Output: 70

- Q) What is final parameter?
- If you declare any parameter as final, you cannot change the value of it.

```
1.class Bike11{
2.  int cube(final int n){
3.   n=n+2;//can't be changed as n is final
4.   n*n*n;
5.  }
6.  public static void main(String args[]){
7.    Bike11 b=new Bike11();
8.    b.cube(5);
9. }
10.}
```

Static Access Modifier

- **Static access modifier is an access modifier that is applicable for methods and variables but not for classes. We can declare a class static by using the static keyword. A class can be declared static only if it is a nested class. It does not require any reference of the outer class. The property of the static class is that it does not allows us to access the non-static members of the outer class.**

```
// Java Program to Illustrate Static Access Modifier
// Importing required classes
import java.io.*;
import java.util.*;

// Main class
class GFG{

        // Creating a static variable and
        // initializing a custom value
        static int x = 10;
        // Creating a instance variable and
        // initializing a custom value
        int y = 20;

        // Main driver method
        public static void main(String[] args)
        {
                // Creating an object of class inside main() method
                GFG t1 = new GFG();
                // Accessing and re-initializing the
```

```
        // static and instance variable
        // using t1 reference
        t1.x = 88;
        t1.y = 99;

        // Creating an object of class inside main() method
        // again
        GFG t2 = new GFG();

        // Accessing the static and instance variable using
        // t2 reference as we know that for each object
        // there is a separate copy of instance variable
        // created. While a same copy of static variable will
        // be shared between the objects
        // Displaying the value of static and instance
        // variable using t2 object reference
        System.out.println(
                "Value of Static variable x = " + t2.x + "\n"
                + "Value of Instance variable y = " + t2.y);
    }
}
```

Final Access Modifier	Static Access Modifier
This modifier is applicable to both outer and inner classes, variables, methods, and blocks.	This modifier is only applicable to inner classes, methods, and variables.
It is not necessary to initialize the final variable at the time of its declaration.	It is necessary to initialize the static variable at the time of its declaration.
Final variable cannot be reinitialized.	Static variables can be reinitialized.
Final method can be inherited.	Static methods can only access the static members of the class and can only be called by other static methods.
Final class can’t be inherited by any class.	The static class object can’t be created and it only contains static members only.
Final keyword doesn’t support any block for initialization of final variables.	Static block is used to initialize the static variables.
Final local variables are allowed.	Unlike C/C++, static local variables are not allowed in Java