# **Non-Access Modifiers**

Java provides some other modifiers to provide the functionalities other than the visibility. These modifiers are called **Non-Access Modifiers**. There are many non-access modifiers available in java. Each modifier have their own functionality. Some of the most used non-access modifiers are listed below.

**static**: The members which are declared as static are common to all instances of a class. Static members are class level members which are stored in the class memory.

**final**: This modifier is used to restrict the further modification of a variable or a method or a class. The value of a variable which is declared as final can’t be modified once it gets a value. A final method can’t be overridden in the sub class and you can’t create a sub class to a final class. See [this](https://javaconceptoftheday.com/final-keyword-in-java/) post for more info on final keyword in java.

**abstract**: This modifier can be used either with a class or with a method. You can’t apply this modifier to variable and constructor. A method which is declared as abstract must be modified in the sub class. You can’t instantiate a class which is declared as abstract.

## Static:

Static keyword can be used with class, variable, method and block. Static members belong to the class instead of a specific instance, this means if you make a member static, you can access it without object.

**Static Variables:**

**package** StaticPackage;

**public** **class** Static1 {

**int** count = 0;

**public** **void** count() {

count++;

}

**public** **int** returncount() {

**return** count;

}

**public** **static** **void** main(String[] args) {

System.***out***.println("Hello");

Static1 s1 = **new** Static1();

Static1 s2 = **new** Static1();

s1.count();

**int** c = s1.returncount();

**int** d = s2.returncount();

System.***out***.println("Count value is "+c);

System.***out***.println("Count value is "+d);

}

}

Remove the static keyword before the variable and observe the results.

Change the static keyword to final and observe the results.

**Static Methods:**

We can call a static method without any object because when we make a member static it becomes class level. If we remove the static keyword and make it non-static then we must need to create an object of the class in order to call it.

**package** FPPackage;

**class** StaticMethod

{

// This is a static method

//**void** myMethod()

static **void** myMethod()

{

System.***out***.println("myMethod");

}

**public** **static** **void** main(String[] args)

{

/\* You can see that we are calling this

\* method without creating any object.

\*/

//StaticMethod s = **new** StaticMethod();

//s.myMethod();

myMethod();

}

}

In the above example, we have created an object to access the method “myMethod()”. If we declare this method as static, we don’t have to create an object.

**Static Block:**

Static block is used for initializing the static variables. This block gets executed when the class is loaded in the memory. A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program.

**package** FPPackage;

**public** **class** StaticBlock {

**static** **int** *num*;

**static** String *text*;

**static** {

System.***out***.println("Initiating variables in block 1");

*num* = 10;

*text* = "hello";

}

**static** {

System.***out***.println("Initiating variables in block 2");

*num* = 11;

*text* = "hello2";

}

**public** **static** **void** main(String[] args) {

System.***out***.println("Value of num is "+*num*);

System.***out***.println("Value of text is "+*text*);

}

}

**Static Class:**

A class can be made static only if it is a nested class.

Nested static class doesn’t need reference of Outer class

A static class cannot access non-static members of the Outer class.

**package** FPPackage;

**class** StaticClass{

**private** **static** String *str* = "Static Class Example";

//Static class

**static** **class** MyNestedClass{

//non-static method

**public** **void** disp() {

/\* If you make the str variable of outer class

\* non-static then you will get compilation error

\* because: a nested static class cannot access non-

\* static members of the outer class.

\*/

System.***out***.println(*str*);

}

}

**public** **static** **void** main(String args[])

{

/\* To create instance of nested class we didn't need the outer

\* class instance but for a regular nested class you would need

\* to create an instance of outer class first

\*/

StaticClass.MyNestedClass obj = **new** StaticClass.MyNestedClass();

obj.disp();

}

}