NON ACCESS MODIFIERS IN JAVA

By : karan Jagota

In JAVA there are special non-access modifiers that does not change the accessibility of variable and member functions like ACCESS MODIFIERS (PUBLIC,PRIVATE ,MODERATE ,DEFAULT) but they do provide some special properties. In this article, we will discuss all these modifiers and their properties when used with CLASS, VARIABLES, and METHODS.

They are :

1. Final
2. Static
3. Scynchronized
4. volatile

FINAL

PROPERTIES WHEN USE WITH :

CLASS : Class declared final cannot be inherited . eg String class

VARIABLE : It is used to prevent the content/field from being MODIFIED BUT it must be initialized when it is declared .

METHOD : Method declared final can be inherited but cannot be override (REDEFINE).

EXAMPLE :

```
final class Year // cannot be inherited

{

 final int year= 2017; // cannot be changed

final void show() // cannot be override / redefine

{

System.out.println(“Welcome to” + year);

} )
```

STATIC

PROPERTIES WHEN USE WITH :

CLASS : JAVA has static nested classes where outer class can never be static ,only inner(nested) classes can be static .

```
public class Outer_class  // cannot be static

{ public static Inner_class{}   } // inner class can be static.
```

VARIABLE : They are mostly used to represent common property of class static variable has only one single storage . They are initialized only once

Example :

```
public class Outer_class

{ public static Inner_class{}   }

Example :

Class Gfg_student

{

String name;

static String website_name = “GEEKS FOR GEEKS ”; // common property

public void show()

{

System.out.println(“ ”+name + “likes” + website_name);

}

Public static void main (string[],args)

{

Gfg_student g1= new Gfg_student();

g1.name = “karan”;

g1.show();

Gfg_student g2= new Gfg_student();

g2.name=”ram”;

g2.show();

}}
```

Output :

karan likes GEEKS FOR GEEKS

ram likes GEEKS FOR GEEKS

METHOD : Static methods do not need any instance of its class for being access. They are called before the object is created. Eg : main method is a static method in java . Also Non-static variable cant be accessed directly by static method.

EXAMPLE:

```
class Add

{

Public static void addition(int a, int b)  // static method

{

System.out.println("addition is "+ a+b);

}

Public static void main(string[],args)

{

Addition(6,4);   // no need tp create object

}

}
```

Output :

Addition is 10

EXAMPLE 2:

```
Class A{

int a;

Public static void main(string[],args)

{

A obj = new A() ;

Obj.a =10;  // way to access non static variable inside static method ie. (using object ).

}

}
```

SYNCHRONISED

PROPERTIES WHEN USED WITH:

CLASS : A class cannot be synchronized

VARIABLE : A variable cannot be synchronized

METHOD : Synchronization refers to multithreading and synchronized can be used only with methods and blocks inside the class. When a method is synchronized it can only be used by one thread at a time. It can be used with both static and non static methods.

Example:

```
Public class Example

{

Public synchronized static void eg_method() { }

}
```

Or

```
Public class Example

{

Public void eg_method()

{

Synchronized(Example.class)

{

// thread safe code can be executed here.

}}}
```

In above example only one eg_method() can be executed while other will not execute by same thread .

VOLATILE

PROPERTIES WHEN USE WITH :

CLASS : class cannot be volatile

FUNCTION : cannot be volatile.

VARIABLE : A volatile variable always reads value from the main memory. it can be used as an alternate way of achieving Synchronization in JAVA.It is genrally used when we want to perform READ and WRITE operation on the same variable. It also guarantees that a reader thread will see updated value of volatile variable after write thread performs execution .