ACCESS MODIFIERS

As in C++ there is no such terminology called as Access specifiers in JAVA. In C++ public, protected, private are called as access specifiers and rest considered to be as access modifiers

But in java we use only terminology called as access modifiers. There are 12 modifiers in JAVA.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MODIFIERS | OUTER CLASS | INNER CLASS | MET  HODS | VAR IABLES | BLOCKS | OUTER INTF | INNER INTF | Outer  Enum | Inner  Enum | Constcts |
| Public | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes |
| Private | No | Yes | Yes | Yes | No | No | Yes | No | Yes | Yes |
| Protected | No | Yes | Yes | Yes | No | No | Yes | No | Yes | Yes |
| <default> | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes |
| Final | Yes | Yes | Yes | Yes | No | No | No | No | No | No |
| Abstract | Yes | Yes | Yes | No | No | Yes | Yes | No | No | No |
| Static | No | Yes | Yes | Yes | Yes | No | Yes | No | Yes | No |
| Synchronized | No | No | Yes | No | Yes | No | No | No | No | No |
| Native | No | No | Yes | No | No | No | No | No | No | No |
| Strictfp | Yes | Yes | Yes | No | No | Yes | Yes | Yes | Yes | No |
| Transient | No | No | No | Yes | No | No | No | No | No | No |
| volatile | No | No | No | Yes | No | No | No | No | No | No |

Notes ::

1. The modifiers which are applicable for inner classes but not for outer classes

Private, protected, static

1. The modifiers which are applicable for classes but not for interface are final
2. The modifiers which are applicable for classes but not for enums are final and abstract
3. The modifiers which are applicable only for methods and which we cant use anywhere else native.
4. The only modifiers which are applicable for constructors are

Public, private, protected ,default

1. The only applicable modifier for local variable is final.

We can define a class inside a class , we can define an interface inside a class, we can define an interface inside an interface, we can define a class inside an interface

Class A

{

Class B

{

}

}

Class A

{

Interface B

{

}

}

Interface A

{

Interface B

{

}

}

Interface A

{

Class B

{

}

}

If we define an interface in a class then the interface will take a default modifier ‘static’

If we define an interface in a interface then the inner interface will take ‘public static’ modifier

If we define an interface inside a class then that inner interface will also take ‘public static’ modifier

Illegal combinations ::

If a method is declared as ‘public’ then the same method cannot be private and protected (illegal combination)

A method if declared with abstract modifier then it cannot be simulteneusly given with

Final,static,synchronized,native,private,strictfp

A variable if public then it cannot be given simultenuosly with private and protected

And if the variable is final then we cannot give it a ‘volatile’

If a class is final then to the same class we cannot give abstract

And if a class is given with public then it cannot be given with private or protected