|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Modifiers | Outer classes | Inner classes | Methods | Variable | blocks | Outer interface | Inner interface | Outer enum | Inner enum | constructor |
| public | Y | Y | Y | Y |  | Y | Y | Y | Y | Y |
| private |  | Y | Y | Y |  | N | Y |  | Y | Y |
| protected |  | Y | Y | Y |  | N | Y |  | Y | Y |
| <default> | Y | Y | Y | Y |  | Y | Y | Y | Y | Y |
| final | Y | Y | Y | Y |  | N | N | N |  |  |
| abstract | Y | Y | Y | N |  | Y | Y | N |  |  |
| Static |  | Y | Y | Y | Y |  | Y |  | Y |  |
| Synchronized |  |  | Y | N | Y |  |  |  |  |  |
| Native |  |  | Y | N |  |  |  |  |  |  |
| strictfp | Y | Y | Y | N |  | Y |  | Y | Y |  |
| Transient |  |  |  | Y |  |  |  |  |  |  |
| Volatile |  |  |  | Y |  |  |  |  |  |  |

Every enum is always final implicitly. We can’t declare enum as final.

Important Conclusions :

1. The modifiers which are applicable for inner classes but not for outer classes :- private, protected and static.
2. The modifiers which are applicable for classes but not for interface are final.
3. The modifiers which are applicable for classes but not for enums are final and abstract.
4. The modifiers which are applicable only for methods and which we can’t use anywhere else : native.
5. The only modifiers which are applicable for constructors are : public, private, protected and default.
6. The only applicable modifier for local variable is final.

Loopholes :

Case 1:

Class A

{

Class B{}

}

Case 2:

Class A

{

Interface B{}

}

Every interface inside class is by default static.

Case 3:

Interface A

{

Interface B{}

}

Every inner interface is by default public and static.

Case 4:

Interface A

{

Class B{}

}

Every class inside interface is public and static.

Conclusions:

1. The interface which is declared inside a class is always static whether we are declaring or not.
2. The interface which is declared inside interface is always public and static whether we are declaring or not.
3. The class which is declared inside interface is always public and static whether we are declaring or not.

Illegal combinations( method level) :

If you declare abstract method then you can not use following modifiers :

-final, static, synchronized, native, private,strictfp

If you declare public you can not declare private or protected.

Illegal combinations( variable level) :

If a variable is public we can’t declare private or protected.

If a variable is final(can’t change) we can’t declare volatile(keep changing).

Illegal combinations( class level) :

If a class is final, can’t declare as abstract.

If a inner class is public, can’t declare private or protected.