Access Modifiers in Java

- A modifier limits the visibility of classes, fields, constructors, or methods in the Java program.

- The functionality of members of a class or a class itself can be protected from other parts of the program by the presence or absence of modifiers.

Access Modifiers in Java

Java language provides a total of 12 modifiers. They are as:

1. public
2. private
3. protected
4. default
5. final
6. synchronized
7. abstract
8. native
9. strictfp
10. transient
11. volatile

Access Modifiers in Java

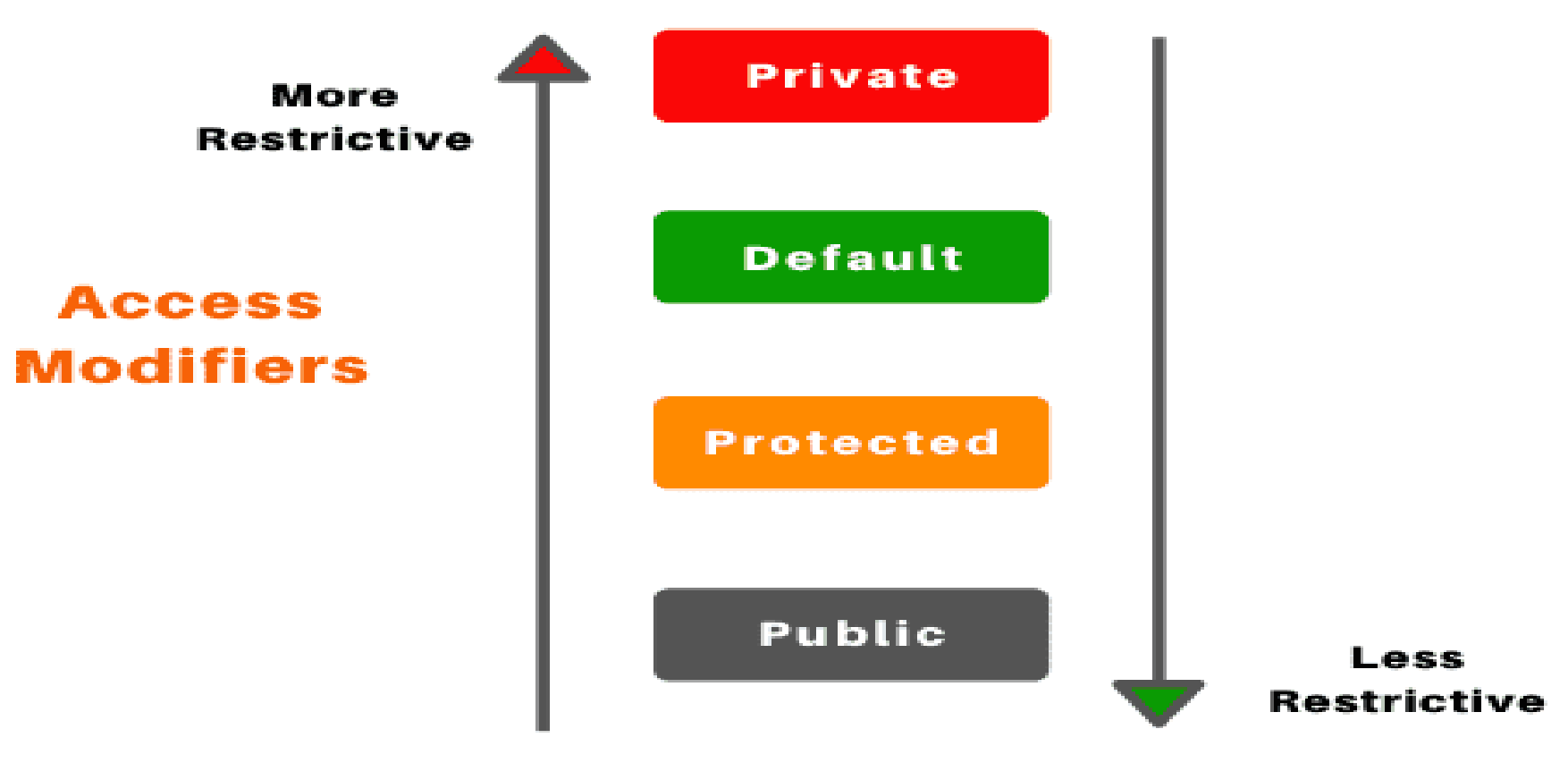
Twelve modifiers in Java can be divided into two categories:

1. Access modifiers
2. Non-access modifiers

Access Modifiers in Java

Java provides four explicit access modifiers in object-oriented programming languages.

They are private, default, protected, and public



Private Access Modifier in Java

1. Private access modifier in java can apply to a variable, method, constructor, inner class but not the outer class that is class itself.
2. The instance variable can be private but a local variable cannot be private.
3. Private members (field, method, or constructor) of a class cannot be accessed from outside the class. They are accessible only within the class.
4. Private members of a superclass cannot be inherited to the subclass. Therefore, they are not accessible in subclasses.
5. If we make any constructor as private, we cannot create an object of that class from another class and also cannot create the subclass of that class.
6. A class cannot be private except for inner classes. Inner classes are members of the outer class. So, members of the class can be private.
7. If we declare a method as private, it behaves as a method declared as final. We cannot call the private method from outside the class.

Default Access Modifier in Java

1. When access modifier is not specified to members of a class or a class itself, it is called default access modifier.

2. The default can apply to the instance variable, local variable, constructor, methods, inner class, or outer class.

3. Default members of a class are visible inside of the class and everywhere within classes in the same package or folder only.

4. Default members can be inherited to the subclass within the same package only.

5. It cannot be inherited from outside the package.

Protected Access Modifier in Java

1. Protected access modifier can be applied to instance variables, local variables, constructors, methods, inner classes **but not the outer class.**

2. Protected members are accessible inside the class and everywhere within classes in the same package and outside the package but through inheritance only.

3. Protected members can be inherited to the subclass.

4. If we make constructor as protected then we can create the subclass of that class within the same package but not outside the package

Public Access Modifier in Java

1. Public access modifier can apply to instance variables, constructors, inner classes, outer class, methods but not with local variables.

2. Public members of a class can be used anywhere.

3. Public members of a class can be inherited to any subclass.

Summary of Access Modifiers Visibility in Java

Access Modifier	Within Class	Within Package	Same Package by subclasses	Outside Package by subclasses	Global
Public	Yes	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	Yes	No
Default	Yes	Yes	Yes	No	No
Private	Yes	No	No	No	No

Key of Access modifiers:

Private > Default > Protected > Public

More restrictive ----------> Less restrictive.

Decreasing ----->

Applicable Modifiers with Classes, Methods, Variables, Interfaces

Outer Class (Top-level Class):

1. The only five modifiers are applicable to the outer class. They are **public**, **default**, **final**, **abstract**, and **strictfp**.

2. Final and abstract both cannot be applied simultaneously with a class. It is an illegal combination.

3. If a class is declared as public, private, or protected cannot be applied simultaneously with public class. It is also an illegal combination.

Applicable Modifiers with Classes, Methods, Variables, Interfaces

Inner Class:

1. The applicable modifiers for the inner class are **public**, **private**, **protected**, **default**, **final**, **abstract**, **static**, and **strictfp**.

2. The non-applicable modifiers for inner class are **synchronized, native, transient, and volatile.**

Applicable Modifiers with Classes, Methods, Variables, Interfaces

Methods:

1. The only two modifiers out of 12 that are not applicable for methods are **transient** and **volatile**.

2. If a method is declared as **public**, we cannot declare simultaneously **private or protected** with a **public** method.

3. If a method is defined as **abstract**, we cannot apply **final**, **static**, **synchronized**, **native**, **private**, or **strictfp**.

Applicable Modifiers with Classes, Methods, Variables, Interfaces

Variables:

1. The modifiers applicable for variables are **public, protected, private, default, final, static, transient, and volatile.**

2. The non-access modifiers such as **abstract, synchronized, native, and strictfp** are not applicable for variables.

3. The only applicable modifier with local variable is **final**. If a variable is declared as **final**, we cannot declare as **volatile**.

Applicable Modifiers with Classes, Methods, Variables, Interfaces

Constructors:

1. For a constructor, only access modifiers are applicable. The access modifiers like public, protected, default, and private can be applied with constructors.

2. Non-access modifiers cannot be applied with constructors. By mistake, if you apply any other modifiers with constructor except these four access modifiers, you will get a compile-time error.

Applicable Modifiers with Classes, Methods, Variables, Interfaces

Blocks:

1. **static** and **synchronized** modifiers are only applicable for blocks.

Example:

```
void printTable(int n){
   synchronized(this){  //synchronized block
     for(int i=1;i<=5;i++){
      System.out.println(n*i);
      try{
       Thread.sleep(400);
      }catch(Exception e){System.out.println(e);}
     }
   }
 }//end of the method
```

Applicable Modifiers with Classes, Methods, Variables, Interfaces

Outer Interface:

1. **public, default, abstract, and strictfp** are modifiers that are applicable for outer interface. Except for these modifiers, we cannot apply other modifiers with outer or top-level interface.

2. The non-access modifier that is applicable for classes but not applicable for an interface, is final.

Inner Interface:

1. For inner interface, we can apply modifiers like **private, default, protected, public, abstract, static, and strictfp.**

2. The non-access modifiers such as **final, synchronized, native, transient, and volatile** are not applicable for inner interface.

Applicable Modifiers with Classes, Methods, Variables, Interfaces

Outer Enum:

The only three modifiers like **public, default, and strictfp** are applicable for outer enum.

The modifiers which are applicable for classes but not applicable for enum are **final and abstract.**

Inner Enum:

For **enum**, we can apply all four access modifiers and two non-access modifiers **like static, and strictfp.**