**Modifier**

**Introduction**

In the exercise above, we often see the keywords of public and static etc. Such tags used to modify or indicate a class or the attributes of a variable are called **modifier**. Modifiers are often placed at the very beginning of a statement and can be grouped into two categories in terms of function:

* **Access control modifiers:** The modifiers used to control the access permissions of class, methods and variables.
* **Non Access Modifiers:**The modifiers apart from access control modifiers used to modify variables and methods.

Now we will explain in detail:

**Access control modifiers**

**default:**

Default can modify class, interface, variable and method, which is visible in the same package. Default doesn't use any modifying tags. As its name indicates, it can be left unwritten.

**public:**

Public can modify class,interface, variable and method, which is visible to all classes. All the variables in an interface is declared implicitly of the public static final type. The default access permission status for the methods in the interface are public.

**private：**

Private can modify variable and method, which is visible within one class. As private is of the most exclusive access level, the methods, variables and constructors modified declared **private** can only be accessed by its own class. But class and interface cannot be declared **private**.The variable declared private can be accessed by external classes but only through the public getter method within the class. The access modifier of private is mianly used to hide the implementation details of class and protect the data of class.

**protected：**

Protected can modify variable and method. But please be noted it cannot modify class. It is visible to all the classes and subclasses within one package.

* **Subclass and base class are inside one package**: The variable, method and constructor declared protected can be accessed by any other classes within one package:
* **Subclass and base class are in different packages**: In a subclass, subclass instances can access the protected methods inherited from the base class, but cannot access the protected methods owned by the base class.

For more on their differences, please visit here:[The detailed explanation of the keyword protected](http://www.runoob.com/w3cnote/java-protected-keyword-detailed-explanation.html)

**Differences and relationship**

| **modifier** | **current class** | **within one package** | **descendant class (within one package)** | **descendant class (different packages)** | **other packages** |
| --- | --- | --- | --- | --- | --- |
| public | Y | Y | Y | Y | Y |
| protected | Y | Y | Y | Y/N（[Introduction](http://www.runoob.com/java/java-modifier-types.html#protected-desc)） | N |
| default | Y | Y | Y | N | N |
| private | Y | N | N | N | N |

**Non Access Modifiers**

**static**

When modifying a variable, static is called static variable; when modifying a method, it is called static method.

* **Static variable:** Static variable belongs to the class itself. In other words, no matter how many objects a class instantiates, the class only has one copy of static variable. Static variable is also called class variable. Local variable cannot be declared static variable.
* **Static method:** Static variable also belongs to the class itself. But you should note that static method cannot use non-static variables of the class.

To access class variable and method, you can directly use **classname.variablename** and **classname.methodname**.

**final**

As final has the meaning of "at last and in the end", it is usually used to declare the common constants which remain unchanged. There are a few points for your attention:

* A variable cannot be reassigned once it's assigned a value.
* The instance variable modified by final must be assigned an initial value explicitly.
* The modifier of final is often used together with static to create class constants. Usually variables are written in capital letters.
* Final can also modify methods. The final methods in class can be inherited by subclasses, but cannot be revised by subclasses.
* Final can also modify classes. The final classes cannot be inherited. No class can inherit any features of final classes.
* **public class Test{**
* **public static final int BOXWIDTH = 6;**
* **static final String TITLE = "Manager";**

**}**

**Other**

* **abstract**
* **synchronized**
* **transient**
* **volatile**