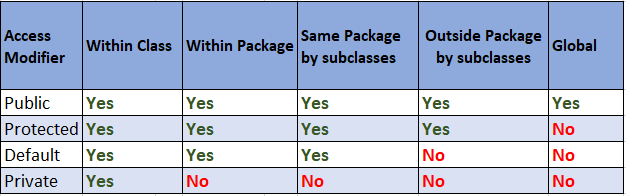
**The four types of access modifiers in java are:**  
  
**1. public**  
**2. protected**  
**3. default**  
**4. private**



**Private:**

A private access modifier offers the highest level of access restriction. The access level of a private access modifier is only within the class. The data members and methods declared as private cannot be accessed from outside the class.  
  
**Private Access Modifier Example:**  
  
The below program results in a compile-time error as we are attempting to access the private data member and method of class A in class B.

**class** **A**

{

**private** **int** totalValue = **50**;

**private** **int** **Multiply**(**int** a,**int** b)

{

**return** a\*b;

}

}

**public** **class** **B**

{

**public** **static** **void** **main**(String args[])

{

A object = **new** A();

System.out.println(object.totalValue); //accessing private data member of class A

System.out.println(object.Multiply(**10**,**15**)); // accessing private method of class A

}

}

**Output:**  
**Compilation error**  
**/B.java:17: error: totalValue has private access in A**  
**System.out.println(object.totalValue);**  
**^**  
**/B.java:18: error: Multiply(int,int) has private access in A**  
**System.out.println(object.Multiply(10,15));**  
**^**  
  
**2 errors**

**Default:**

When no access specifier is given, it will be considered the default access specifier. It is also known as **"package-private"** or **"no modifier"**. The default access modifier allows access only within the package. It does not allow access from outside the package.  
  
**Default Access Modifier Example:**  
  
In the below example program, we have two classes Controller and Multiplication. In class Controller, I am attempting to access the Multiply() method of Multiplication class which has default access. Since class Controller is present in an alternate package i.e B and the scope of default modifier is restricted to a current package level i.e A, this program would result in compilation error.  
  
**Multiplication.java**

**package** A;

**public** **class** **Multiplication** {

/\* As we haven't specified any access modifier here,

\* it will be considered as default.

\*/

**int** **Multiply**(**int** an, **int** b){

**return** a\*b;

}

}

**Controller.java**

**package** B;

**import** **A.\***;

**public** **class** **Controller** {

**public** **static** **void** **main**(String args[])

{

Multiplication object = **new** Multiplication ();

object.Multiply(**10**, **21**);

}

}

**Output:**  
**Exception in thread “main” java.lang.Error : Unresolved compilation problem:The method Multiply(int,int) from the type Multiplication is not visible at B.Controller.main(Controller.java:6)**

**Protected:**

The protected access modifier provides access within the package and outside the package through inheritance only i.e by creating child class (subclass or derived class). It cannot be accessed from outside the package if there is no inheritance.  
  
**Protected Access Modifier Example:**  
  
In this example, the class Controller which is available in another package B can access the Multiply() method, which is declared as protected. This is possible since the class Controller extends class Multiplication. Subclasses of the Multiplication class present in any package can access the protected methods or data members.  
  
**Multiplication.java**

**package** A;

**public** **class** **Multiplication** {

**protected** **int** **multiply**(**int** a, **int** b){

**return** a\*b;

}

}

**Controller.java**

**package** B;

**import** **A.\***;

**class** **Controller** **extends** Multiplication {

/\*

\* Controller is a sub-class of Multiplication

\*/

**public** **static** **void** **main**(String args[]){

Controller object = **new** Controller ();

System.out.println(object.multiply(**11**, **22**));

}

}

**Output:**  
**33**

**Public:**

The public access modifier offers the lowest level of access restriction. Class or variable or method declared as public can be accessed everywhere. Once a variable or method is declared public, it can be accessed from anywhere in the code, i.e within the class, outside the class, within the package and outside the package.  
  
**Public Access Modifier Example:**  
  
The below example is equivalent to the one we have seen for private access modifier however here the method Multiply () has public modifier and class Controller can access it.  
  
**Multiplication.java**

**package** A;

**public** **class** **Multiplication** {

/\*As Multiply is made public it can be accessed

\*anywhere globally

\*/

**public** **int** **Multiply** (**int** an, **int** b){

**return** a\*b;

}

}

**Controller.java**

**package** B;

**import** **A.\***;

**class** **Controller** {

**public** **static** **void** **main**(String args[]){

Multiplication object = **new** Multiplication ();

System.out.println(object.Multiply(**100**, **2**));

}

}

**Output:**  
**200**

**Points to Remember:**

1. Classes in java are allowed to use only the**"public"** or **"default"** access modifier.  
  
2. Class members i.e variables and methods are allowed to use any of the four access modifiers.  
  
3. We are allowed to have only one public class in a source file. public class name must be the same as the source file.  
  
4. The **default** access modifier is also known as **"package-private"** or **"no-modifier"**.  
  
5. Examples of non-access modifiers are static, transient, abstract, synchronized.

**Best Practices To Use Access Modifiers:**

1. Prefer the most restrictive access level that makes sense for a particular member.  
  
2. Use public fields for constants only.

**Access Modifiers in Java with Method Overriding**

The access modifier of a overriding method(declared in a subclass) must not be more restrictive than the access modifier of an overridden method(declared in superclass).  
  
In the below example class Child hello() method has a default access modifier. It is overriding the Parent class hello() method which has a protected access modifier. As we know, the default access modifier is more restrictive than a protected access modifier. Hence, the below code will throw the compile-time error.

**class** **Parent**

{

/\*overridden method has

protected access modifier\*/

**protected** **void** **hello**()

{

System.out.println("Alive is Awesome");

}

}

**public** **class** **Child** **extends** Parent

{

/\*overriding method has

default access modifier \*/

**void** **hello**()

{

System.out.println("Be in present");

}

**public** **static** **void** **main**(String args[]) {

Child obj = **new** Child();

obj.hello();

}

}

**Output:**  
**/Child.java:11: error: hello() in Child cannot override hello() in Parent  
    void hello()  
         ^  
  attempting to assign weaker access privileges; was protected  
1 error**