Modifiers are keywords that you add to those definitions to change their meanings. The Java language has a wide variety of modifiers, including the following:

* [Java Access Modifiers](http://www.tutorialspoint.com/java/java_access_modifiers.htm)
* [Non Access Modifiers](http://www.tutorialspoint.com/java/java_nonaccess_modifiers.htm)

To use a modifier, you include its keyword in the definition of a class, method, or variable. The modifier precedes the rest of the statement, as in the following examples (Italic ones):

*public* class className {

// ...

}

*private* boolean myFlag;

*static final* double weeks = 9.5;

*protected static final* int BOXWIDTH = 42;

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

// body of method

}

Access Control Modifiers:

Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are:

* Visible to the package, the default. No modifiers are needed.
* Visible to the class only (private).
* Visible to the world (public).
* Visible to the package and all subclasses (protected).

Non Access Modifiers:

Java provides a number of non-access modifiers to achieve many other functionality.

* The *static* modifier for creating class methods and variables
* The *final* modifier for finalizing the implementations of classes, methods, and variables.

### 1) private access modifier

|  |
| --- |
| The private access modifier is accessible only within class. |

### Simple example of private access modifier

|  |
| --- |
| In this example, we have created two classes A and Simple. A class contains private data member and private method. We are accessing these private members from outside the class, so there is compile time error. |

**class** A

{

**private** **int** data=40;

**private** **void** msg()

{

System.out.println("Hello java");

}

}

**public** **class** Simple

{

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

A obj=**new** A();

System.out.println(obj.data);//Compile Time Error

obj.msg();//Compile Time Error

   }

}

### 2) default access modifier

|  |
| --- |
| If you don't use any modifier, it is treated as **default** by default. The default modifier is accessible only within package. |

### Example of default access modifier

|  |
| --- |
| In this example, we have created two packages pack and mypack. We are accessing the A class from outside its package, since A class is not public, so it cannot be accessed from outside the package. |

//save by A.java

**package** pack;

**class** A{

**void** msg(){

System.out.println("Hello");

}

}

//save by B.java

**package** mypack;

**import** pack.\*;

**class** B{

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

{

   A obj = **new** A();//Compile Time Error

   obj.msg();//Compile Time Error

  }

}

In the above example, the scope of class A and its method msg() is default so it cannot be accessed from outside the package.

### 3) protected access modifier

The **protected access modifier** is accessible within package and outside the package but through inheritance only.

The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.

### Example of protected access modifier

In this example, we have created the two packages pack and mypack. The A class of pack package is public, so can be accessed from outside the package. But msg method of this package is declared as protected, so it can be accessed from outside the class only through inheritance.

//save by A.java

**package** pack;

**public** **class** A{

**protected** **void** msg(){

System.out.println("Hello");

}

}

//save by B.java

**package** mypack;

**import** pack.\*;

**class** B **extends** A{  //inheritance

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

   B obj = **new** B();

   obj.msg();

  }

}

Output:Hello

### 4) public access modifier

|  |
| --- |
| The **public access modifier** is accessible everywhere. It has the widest scope among all other modifiers. |

### Example of public access modifier

//save by A.java

**package** pack;

**public** **class** A

{

**public** **void** msg()

{

System.out.println("Hello");

}

}

//save by B.java

**package** mypack;

**import** pack.\*;

**class** B{

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

   A obj = **new** A();

   obj.msg();

  }

}

Output:Hello