Access Modifiers in Java

In Java, access modifiers are used to set the accessibility (visibility) of classes, interfaces, variables, methods, constructors, data members, and the setter methods.

There are two types of modifiers in Java: **access modifiers** and **non-access modifiers**.

There are four types of Java access modifiers:

|  |  |
| --- | --- |
| **Modifier** | **Description** |
| Default | declarations are visible only within the package (package private) |
| Private | declarations are visible within the class only |
| Protected | declarations are visible within the package or all subclasses |
| Public | declarations are visible everywhere |

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc. Here, we are going to learn the access modifiers only.

### 1) Privates

The private access modifier is accessible only within the 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 a 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  A     }  } |

Note: A class cannot be private or protected except nested class.

### 2) Default

If you don't use any modifier, it is treated as **default** by default. The default modifier is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than private. But, it is more restrictive than protected, and public.

**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

|  |
| --- |
| **package** pack;  **class** A{  **void** msg(){System.out.println("Hello");}  } |

|  |
| --- |
| **package** mypack;  **import** pack.\*;  **class** B{  **public** **static** **void** main(String args[]){     A obj = **new** A();//Compile Time Error     obj.msg();//Compile Time Error    }  } |

### 3) Protected

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.

It provides more accessibility than the default modifer.

**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.

|  |  |
| --- | --- |
| **package** pack;  **public** **class** A{  **protected** **void** msg(){System.out.println("Hello");}  } | |
| **package** mypack;  **import** pack.\*;    **class** B **extends** A{  **public** **static** **void** main(String args[]){     B obj = **new** B();     obj.msg();    }  }  **Output**:Hello | |

### 4) Public

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

**Example of public access modifier**

|  |
| --- |
| **package** pack;  **public** **class** A{  **public** **void** msg(){System.out.println("Hello");}  } |

|  |
| --- |
| **package** mypack;  **import** pack.\*;    **class** B{  **public** **static** **void** main(String args[]){     A obj = **new** A();     obj.msg();    }  }  **Output**:Hello |

Non-access modifiers

**Non-access modifiers :**In java, we have 7 non-access modifiers. They are used with classes, methods, variables, constructors etc to provide information about their behavior to JVM.They are

* [static](https://www.geeksforgeeks.org/static-keyword-java/)
* [final](https://www.geeksforgeeks.org/final-keyword-java/)
* abstract
* [synchronized](https://www.geeksforgeeks.org/synchronized-in-java/)
* [transient](https://www.geeksforgeeks.org/transient-keyword-java/)
* [volatile](https://www.geeksforgeeks.org/volatile-keyword-in-java/)
* native

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

* The *static* modifier for creating class methods and variables.
* The *final* modifier for finalizing the implementations of classes, methods, and variables.
* The *abstract* modifier for creating abstract classes and methods.
* The *synchronized* and *volatile* modifiers, which are used for threads.
* **The transient** is a variables modifier used in [serialization](http://quiz.geeksforgeeks.org/serialization-in-java/).
* The **native keyword** is applied to a **method** to indicate that the **method** is implemented in **native** code using JNI (Java **Native** Interface). **native** is a **modifier** applicable only for methods and we can't apply it anywhere else