Access Modifiers in java

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

The access modifiers in java specifies accessibility(scope) of a data member, method, constructor or class.

There are 4 types of java access modifiers:

1. private
2. default
3. protected
4. public

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

}

}

Role of Private Constructor

|  |
| --- |
| If you make any class constructor private, you cannot create the instance of that class  from outside the class. For example: |

**class** A

{

**private** A()

{}//private constructor

**void** msg(){System.out.println("Hello java");}

}

**public** **class** Simple{

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

{

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

 }

}

2) default access modifier

|  |
| --- |
| If you don't use any modifier, it is treated as **default** bydefault. 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

{

**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

Understanding all java access modifiers

Let's understand the access modifiers by a simple table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

Java access modifiers with method overriding

If you are overriding any method, overridden method (i.e. declared in subclass) must not be more restrictive.

1. **class** A{
2. **protected** **void** msg(){System.out.println("Hello java");}
3. }
5. **public** **class** Simple **extends** A{
6. **void** msg(){System.out.println("Hello java");}//C.T.Error
7. **public** **static** **void** main(String args[]){
8. Simple obj=**new** Simple();
9. obj.msg();
10. }
11. }

|  |
| --- |
| The default modifier is more restrictive than protected. That is why there is compile time error |