Lecture 9 and 10

# Access Modifiers in Java

1. [Private access modifier](https://www.javatpoint.com/access-modifiers#accessprivate)
2. [Role of private constructor](https://www.javatpoint.com/access-modifiers#accessprivatecons)
3. [Default access modifier](https://www.javatpoint.com/access-modifiers#accessdefault)
4. [Protected access modifier](https://www.javatpoint.com/access-modifiers#accessprotected)
5. [Public access modifier](https://www.javatpoint.com/access-modifiers#accesspublic)

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

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

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.

**Understanding Java Access Modifiers**

Let's understand the access modifiers in Java 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 |

**1) Private**

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.

1. class A{
2. private int data=40;
3. private void msg(){System.out.println("Hello java");}
4. }
5.
6. public class Simple{
7. public static void main(String args[]){
8. A obj=new A();
9. System.out.println(obj.data);//Compile Time Error
10. obj.msg();//Compile Time Error
11. }
12. }

**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:

1. class A{
2. private A(){}//private constructor
3. void msg(){System.out.println("Hello java");}
4. }
5. public class Simple{
6. public static void main(String args[]){
7. A obj=new A();//Compile Time Error
8. }
9. }

**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.

1. //save by A.java
2. package pack;
3. class A{
4. void msg(){System.out.println("Hello");}
5. }
6. //save by B.java
7. package mypack;
8. import pack.\*;
9. class B{
10. public static void main(String args[]){
11. A obj = new A();//Compile Time Error
12. obj.msg();//Compile Time Error
13. }
14. }

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**

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.

1. //save by A.java
2. package pack;
3. public class A{
4. protected void msg(){System.out.println("Hello");}
5. }
6. //save by B.java
7. package mypack;
8. import pack.\*;
9.
10. class B extends A{
11. public static void main(String args[]){
12. B obj = new B();
13. obj.msg();
14. }
15. }

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**

1. //save by A.java
2.
3. package pack;
4. public class A{
5. public void msg(){System.out.println("Hello");}
6. }
7. //save by B.java
8.
9. package mypack;
10. import pack.\*;
11.
12. class B{
13. public static void main(String args[]){
14. A obj = new A();
15. obj.msg();
16. }
17. }

Output:Hello