Website

https://www.curiousinfotechsolutions.com

Access Modifiers in Java

Java Programming

Day-15

Email id:
info@curiousinfotechsolutions.com

Syllabus Day-15

- Visibility Control
- Types of Modifiers in Java
- Types of Java Access Modifiers
- Understanding Java Access Modifiers
- Private Access Modifier
- Default Access Modifier
- Protected Access Modifier
- Public Access Modifier

Visibility Control

- The extends keyword is used to inherit all the members of a class by a subclass.
- The variables and methods of a class are ***visible everywhere in the program***.
- However, it may be necessary in **some situations** to restrict the access to certain *variable and methods* from outside the class.
- **For example**, we may not like the objects of the class directly alter the value of a variable or access a method.

Types of Modifiers in Java

- 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 many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc.

Visibility Control - Access Modifiers in Java

- There are four types of ***Java access modifiers*** are used to change the access level of fields, constructors, methods, and class :
  - private
  - public
  - protected
  - default

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.

Understanding Java Access Modifiers

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

Private Access Modifier

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

```
class A{
   int data=40;
   void msg()
   {
      System.out.println("Hello java");
   }
}
public class Main{
 public static void main(String args[]){
  A obj=new A();
  System.out.println(obj.data);
  obj.msg();
  }
}
```

```
class A{
   private int data=40;
   private void msg()
   {
      System.out.println("Hello java");
   }
}
public class Main{
 public static void main(String args[]){
  A obj=new A();

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

Role of Private Constructor


```
class A{
private A()
   {
      System.out.println("Hello java");
   }
}
public class Main{
 public static void main(String args[]){
  //Compile Time Error
  A obj=new A();
 }
}
```

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.
- It cannot be accessed from outside the package.
- It provides more accessibility than private. ***But, it is more restrictive than protected, and public***.

Example: Default Access Modifier


```
1.//save by A.java
2.package pack;
3.class A{
4.  void msg()
5.{System.out.println("Hello");}
6.}
```

```
1.//save by B.java
2.package mypack;
3.import pack.*;
4.class B{
5.  public static void main(String args[]){
6.   A obj = new A();//Compile Time Error
7.   obj.msg();//Compile Time Error
8.  }
9.}
```

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

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.
- It provides more accessibility than the default modifier.

Example: Protected Access Modifier


```
class B
{
protected void msg1()
    {
        System.out.println("Hello");
    }
}
```

```
class A extends B
{
    protected void msg2()
    {
        System.out.println("Java");
    }
}
```

```
class Main extends A
{
  public static void main(String args[]){
   A obj = new A();
   obj.msg1();//Hello
   obj.msg2();//Java
  }
}
```

Public Access Modifier

The **public access modifier** is accessible everywhere. It has the widest scope among all other modifiers (private, default and protected).

```
1./save by A.java
2.
3.package pack;
4.public class A{
5.public void msg()
6.{System.out.println("Hello");}
7.}
```

```
1.//save by B.java
2.
3.package mypack;
4.import pack.*;
5.
6.class B{
7.  public static void main(String args[]){
8.   A obj = new A();
9.   obj.msg();
10.  }
11.}
```

Java Access Modifiers with Method Overriding


```
class A{
void msg()
  {
  System.out.println("Hello java through class A");
  }
}

class Main extends A{
void msg(){System.out.println("Hello java through class  Main");}
 public static void main(String args[]){
   Simple obj=new Simple();
   obj.msg();
   }
}
```

Case-1
Both msg() , if protected, then no issues.

Caae-2
Both msg() , if public, then no issues.

Case-3
If msg() of A is protected, then there is Compiler Time Error