What is Access Modifiers?

These are the keywords in Java which set the visibility of a class, interface, variables, data member, method, and constructor. That's why these are also called as **Visibility modifiers**. It helps to achieve encapsulation because by using access modifiers, we can set the control which part of the program can access by the member of a class, and by using access modifiers, we can avoid misuse of data.

Types of Access Modifiers

In Java Programming Language, we have four types of access modifiers, that are:

- Default – Accessible Within the Package
- Private – Accessible within the class
- Public – Accessible anywhere
- Protected – Accessible within the package and all subclasses

Let's discuss one by one in briefly:

Default Java

At the time of writing Program, if you are not mentioned any modifiers for a class, interface, variables, data member, method, or constructor, then Java, by default, assigns all those with the default access modifiers. And the scope of default access modifiers is within the package only, which means you can access the default access modifiers declared variables, classes, method, and constructor within the package alone.

Let’s check a program for better understand:

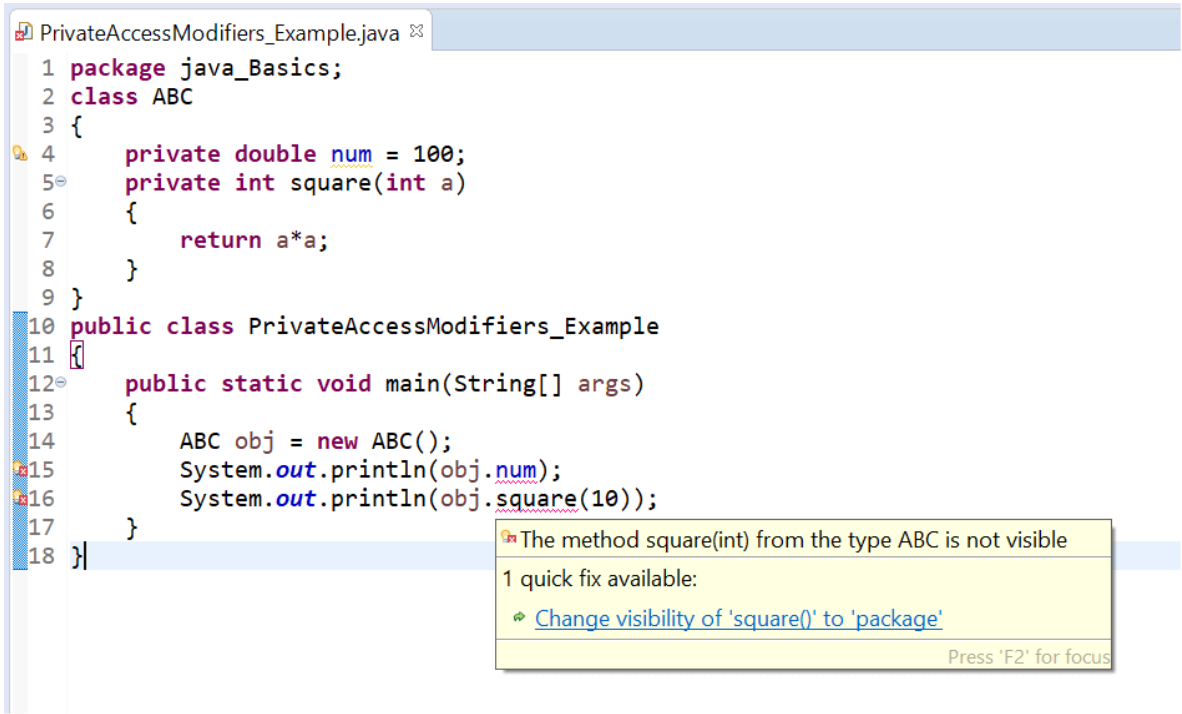
```
package java_Basics;

class DefaultAccessModifier_Example
{
   public static void main(String[] args)
   {
      System.out.println("This DefaultAccessModifier_Example  class");
   }
}
```

Private Java

You Can define any data member or method private using the private access modifiers. And those methods or data members are declared as private that are accessible within that class only in which they are declared.

Let’s go through with a program for better understanding

```
package java_Basics;
class ABC
{
   private double num = 100;
   private int square(int a)
   {
      return a*a;
   }
}
public class PrivateAccessModifiers_Example
{
   public static void main(String[] args)
   {
      ABC obj = new ABC();
      System.out.println(obj.num);
      System.out.println(obj.square(10));
   }
}
```



Private Access Modifiers

Output:

when you compile the program then that time you will get a compile-time error

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
	The field ABC.num is not visible
	The method square(int) from the type ABC is not visible

	at java_Basics.PrivateAccessModifiers_Example.main(PrivateAccessModifiers_Example.java:15)
```

Public Java

Those members or methods declared with public modifiers that are accessible anywhere. That means these modifiers does not put any restriction on the method or data members. Out of all modifiers, the public has the most comprehensive scope among all the modifiers.

Let’s go through with a program for better understanding.

```
package java_Basics;
class PQR
{
   public void display()
   {
      System.out.println("Public Method Of Other Class Executed");
   }
}
public class PublicAccessModifiers_Example
{
   public static void main(String[] args)
   {
      PQR obj=new PQR();
      obj.display();
   }
}
```

Output:

```
Public Method Of Other Class Executed
```

Important Points:

- If any other programmer is using your class, then it's better to give the most restrict access level for particular data member.
- Avoid Public access modifiers public field excepts for constants.

Protected Java

Protected modifiers are specified by the keyword protected. Suppose you assign a data member or methods with protected. In that case, that is accessible by the classes of the same package and the subclasses of the different packages but through inheritance only. The

protected and default access modifiers are almost the same, with only one exception: its visibility in subclasses of a separate package and more accessibility than the default access modifier.

We can access the protected access modifier with the data member method and constructor, but we cannot apply it to a class and interface. In the below example, we are trying to explain how the protected access modifier or specifier is working. So for this, we have created two packages that are package 1 and package 2.

In package 1, we have created one class ABC with public access modifier so that it can be accessed outside of the package. Still, inside the class abc we have declared one method testmethod with access modifier protected.

```
package com.SoftwareTestingO.Java.Package1;
public class Abc
{
   protected void testmethod()
   {
      System.out.println("From Parent Message");
   }
}
```

In package 2, do we have to declare another class pqr, which extends the class of package 1(abc class)? Here we are trying to call the protected method of package 1 class inside package 2 class.

```
package com.SoftwareTestingO.Java.Package2;
import com.SoftwareTestingO.Java.Package1.Abc;
public class Pqr extends Abc
{
   public static void main(String args[])
   {
      Pqr obj=new Pqr();
      obj.testmethod();
   }
}
```

Output:

```
From Parent Message
```

If you try to access the test method of package 1 abc class n package 2 pqr class then that time you will get the below error:

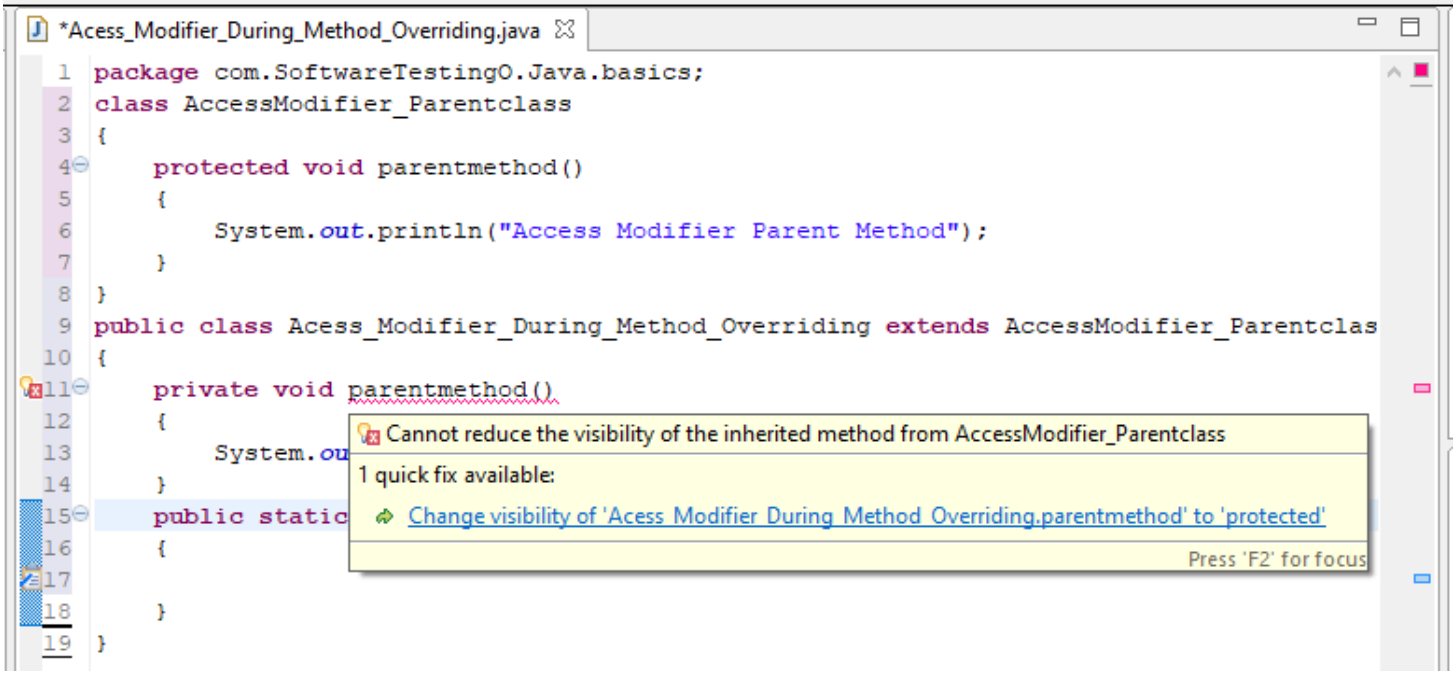
Protected Method In Child Class With Out Extends

Java Acess Modifier During Method Overriding

If you want to override a method of a subclass, then the subclass method must not be restrictive, which means you can not decrease the scope of the child class method.

```
package com.SoftwareTestingO.Java.basics;
class AccessModifier_Parentclass
{
   protected void parentmethod()
   {
      System.out.println("Access Modifier Parent Method");
   }
}
public class Acess_Modifier_During_Method_Overriding extends
AccessModifier_Parentclass
{
   private void parentmethod()
   {
      System.out.println("Access Modifier Child Method");
   }
   public static void main(String[] args)
   {


   }
}
```

As you have seen, Java suggest that you can not reduce the visibility of an inheritated method and also you will get compile time error.



Access Modifier Java Overriding Scope Method