The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases and might use technology no longer available.
See Java Language Changes for a summary of updated language features in Java SE 9 and subsequent releases.
See JDK Release Notes for information about new features, enhancements, and removed or deprecated options for all JDK releases.

Controlling Access to Members of a Class

Access level modifiers determine whether other classes can use a particular field or invoke a particular method. There are two levels of access control:

- At the top level—`public`, or *package-private* (no explicit modifier).
- At the member level—`public`, `private`, `protected`, or *package-private* (no explicit modifier).

A class may be declared with the modifier `public`, in which case that class is visible to all classes everywhere. If a class has no modifier (the default, also known as *package-private*), it is visible only within its own package (packages are named groups of related classes — you will learn about them in a later lesson.)

At the member level, you can also use the `public` modifier or no modifier (*package-private*) just as with top-level classes, and with the same meaning. For members, there are two additional access modifiers: `private` and `protected`. The `private` modifier specifies that the member can only be accessed in its own class. The `protected` modifier specifies that the member can only be accessed within its own package (as with *package-private*) and, in addition, by a subclass of its class in another package.

The following table shows the access to members permitted by each modifier.

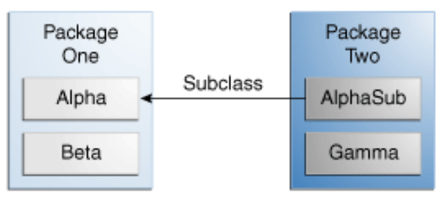
Access Levels

Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

The first data column indicates whether the class itself has access to the member defined by the access level. As you can see, a class always has access to its own members. The second column indicates whether classes in the same package as the class (regardless of their parentage) have access to the member. The third column indicates whether subclasses of the class declared outside this package have access to the member. The fourth column indicates whether all classes have access to the member.

Access levels affect you in two ways. First, when you use classes that come from another source, such as the classes in the Java platform, access levels determine which members of those classes your own classes can use. Second, when you write a class, you need to decide what access level every member variable and every method in your class should have.

Let's look at a collection of classes and see how access levels affect visibility. The following figure shows the four classes in this example and how they are related.



Classes and Packages of the Example Used to Illustrate Access Levels

The following table shows where the members of the Alpha class are visible for each of the access modifiers that can be applied to them.

Visibility

Modifier	Alpha	Beta	Alphasub	Gamma
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

Tips on Choosing an Access Level:

If other programmers use your class, you want to ensure that errors from misuse cannot happen. Access levels can help you do this.

- Use the most restrictive access level that makes sense for a particular member. Use `private` unless you have a good reason not to.
- Avoid `public` fields except for constants. (Many of the examples in the tutorial use public fields. This may help to illustrate some points concisely, but is not recommended for production code.) Public fields tend to link you to a particular implementation and limit your flexibility in changing your code.

About Oracle | Contact Us | Legal Notices | Terms of Use | Your Privacy Rights

Copyright © 1995, 2022 Oracle and/or its affiliates. All rights reserved.

Cookie Preferences | Ad Choices