# Introduction

In Java, you need access to a class, variable, method, and objects before the current executing code can use these items. The code needs access to both the class definition and its constructor in order to create an instance of the class. Java provides several levels of access through keywords known as access modifiers.

Access control is used as part of encapsulation. This allows the the programmer to decide which code may access properties and methods of a class or its instance.

# Access Modifiers

There are four different access Modifiers; each one provides a different level of access. The access modifiers are: public, protected, private, and *default*. The default access modifier is the only one that does not use keyword. In other words, if no access modifier is provided, the Java compiler treats the item as having the default access level.

## Public

Public access provides global access to the item. This means that any code can access it.

Usually, if a class is to be used outside of its package, it is defined as public with a public constructor. This allows any code that wants to use the class can create its own instance of the class[[1]](#footnote-2).

|  |
| --- |
| public class foobar  {  // Constructor  public foobar() { . . . }  . . .  } |

## Protected

Protected access allows any code within the same package and any subclass to access the item.

This level of access allows data members/properties and methods defined in a class to be used internally to the class, the package, and any derived subclass regardless of its package. In other words, it prevents any classes outside the package that are not subclasses from accessing these methods or variables.

Protected access is often used in cases where the class defines a method that it expects to be overwritten by a derived subclass when each derived subclass has a different way of performing the “something”.

|  |
| --- |
| public class foobar  {  // Public Constructor  public foobar() { . . . }  public int doSomething()  {  return howToDoSomething()  }  protected int howToDoSomething() { . . . }  } |

## Private

Private access limits the access to the item to the object that contains it. In a class or an instance of the class any private property/data member and private methods can only be accessed by code defined within the class.

Private access is the most restrictive level of access. It is normally used to prevent other code from modifying data members, especially when there are rules that limit what values can be assigned or that changing one data member requires another data member to be changed.

|  |
| --- |
| public class foobar  {  private int counter;  private bool flag;  // Public Constructor  public foobar() { counter = 0; flag = true; . . . }  public void doSomething()  {  // do something  incrementCounter();  }  private void incrementCounter()  {  ++counter;  flag = !flag;  }  } |

## Default

Default access provides same package access while denying access by subclasses defined in other packages. Default access is almost the same as protected with the difference being that subclasses derived in a different package cannot access the item.

|  |
| --- |
| public class foobar  {  private int counter;  private bool flag;  // Public Constructor  public foobar() { counter = 0; flag = true; . . . }  public void doSomething()  {  // do something  incrementCounter();  }  void incrementCounter()  {  ++counter;  flag = !flag;  }  } |

# Overview

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modifer | Class | Package | Subclass | Global |
| public | T | T | T | T |
| protected | T | T | T | F |
| private | T | F | F | F |
| *Default* | T | T | F[[2]](#footnote-3) | F |

1. If the constructor has the private access modifier, only the class itself can create instances of itself. In this case, the class needs to provide a static factory method that can create an instance of the class and return that object to the caller. This technique will be described in more detail in another paper. [↑](#footnote-ref-2)
2. If a subclass is defined within the same package as the class, its access is covered under the package access column. [↑](#footnote-ref-3)