***ACCESS MODIFIERS***

**Access Modifiers in Java | Example Program**

***What is Modifier in Java?***

*A****modifier in java****is a keyword that we add to those definitions that we need to change their meaning.*

*In other words, a modifier limits the visibility of classes,*[*fields*](https://www.scientecheasy.com/2020/05/variables-in-java.html/)*,*[*constructors*](https://www.scientecheasy.com/2020/06/constructor-in-java.html/)*, or*[*methods*](https://www.scientecheasy.com/2020/06/java-methods.html/)*in the Java program.*

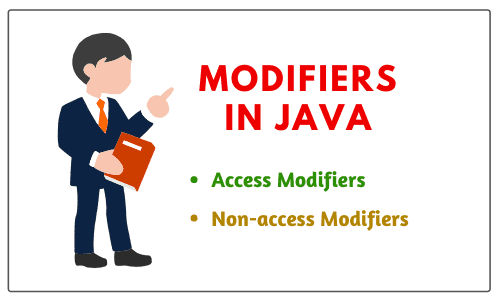
*The functionality of members of a class or a class itself can be protected from other parts of the program by the presence or absence of modifiers.*

*Java language provides a total of 12 modifiers. They are public, private, protected, default, final, synchronized, abstract, native, strictfp, transient, and volatile.*

*Twelve modifiers in java can be divided into two categories:*

* ***Access modifiers***
* ***Non-access modifiers***

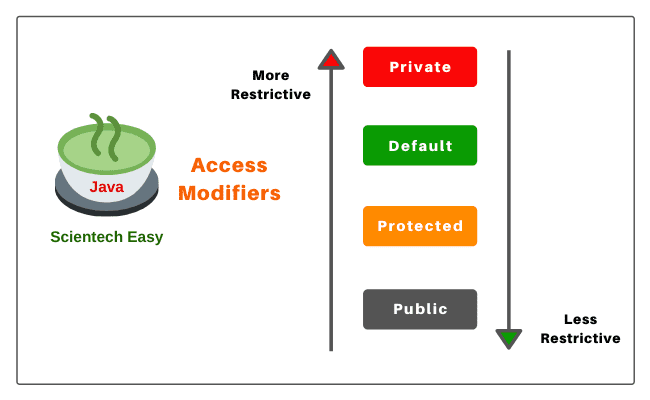
*In this tutorial, we will focus only on access modifiers and in the next tutorial, we will cover non-access modifiers.*

*[](https://www.scientecheasy.com/2020/06/access-modifiers-in-java.html/)*

***Access Modifiers in JavaAccess modifiers/specifiers in java****define the boundary for accessing members of a class and a class itself.*

*In other words, access modifiers are those modifiers that are used to restrict the visibility (accessibility) of classes, fields, methods, or constructors.*

*In short, the accessibility/visibility of data depends on the access modifiers. The access modifiers are also known as visibility modifiers.*

*Java provides four explicit access modifiers in object-oriented programming languages. They are private, default, protected, and public as shown in the below figure.[](https://www.scientecheasy.com/2020/06/access-modifiers-in-java.html/)*

***Private Access Modifier in Java****There are the following points about private access modifiers that need to keep in mind.*

*1. Private access modifier in java can apply to a variable, method, constructor, inner class but not the outer class that is class itself.*

*2. The instance variable can be private but a local variable cannot be private.*

*3. Private members (field, method, or constructor) of a class cannot be accessed from outside the class. They are accessible only within the class.*

*4. Private members of a superclass cannot be inherited to the subclass. Therefore, they are not accessible in subclasses.*

*5. If we make any constructor as private, we cannot create an object of that class from another class and also cannot create the subclass of that class.*

*6. A class cannot be private except for inner classes. Inner classes are members of the outer class. So, members of the class can be private.*

*7. If we declare a method as private, it behaves as a method declared as final. We cannot call the private method from outside the class.*

*Let us create a program where we will create two classes Class A and Class B. In class A, we will declare field and method as private. When we will access this private field and private method from outside the class B, then they will give a compile-time error. Look at the source code to understand better.*

***Program source code 1:***

*package modifiersProgram;*

*class A // Here, Class is default.*

*{*

*// Declaration of Instance Variable.*

*private int data = 30; // Here, instance variable is private.*

*// Declaration of instance method.*

*private void msg() // Here, method is also private.*

*{*

*System.out.println("Hello Java, this is my first java program");*

*}*

*}*

*Now, create another class B and call private field and method of class A.*

*class B*

*{*

*// Main method.*

*public static void main(String[] args)*

*{*

*// Create an object of class A and call members of class A using reference variable 'a'.*

*A a = new A(); // a is a reference variable of class A and pointing to the objects of class A.*

*System.out.println(a.data): // Compile time error will occur because we cannot call private members of any class from outside the class.*

*a.msg(); // Compile time error. So we cannot call the private method of any class because this is accessible within the class only.*

*}*

*}*

***Role of Private Constructor***

*If we declare any constructor of a class as private then we cannot create an object of that class from outside the class. In other words, we cannot create the subclass of that class.*

*Let’s take an example program where we will declare a constructor of class as private and try to create an object of that class. But, we will get a compile-time error.****Program source code 2:***

*package modifiersProgram;*

*class A*

*{*

*// Declaration of Constructor.*

*private A() // Here, Constructor is declared as private.*

*{*

*System.out.println("Constructor is private");*

*}*

*// Declaration of instance method.*

*private void msg() // Method is private.*

*{*

*System.out.println("Method is private");*

*}*

*}*

*Now, create another class B and call class A from class B. As you will call class A, it will give compile-time error.*

*class B*

*{*

*// Main method.*

*public static void main(String args[])*

*{*

*// Create the object of class A.*

*A a = new A(); // Compile time error because the constructor is private and we cannot create the object of class A from outside the class.*

*}*

*}*

***Q. Can you find the error in the code?***

*private class Student*

*{*

*private int roll = 4;*

*private class Name*

*{*

*// Inner class*

*}*

*}*

*Ans: Outer class cannot be private but inner class can be private. Instance variable can be private. The outer class is also known as top-level class.*

*The only five modifiers are applicable to a top-level class. They are public, default, final, abstract, and strictfp.*

*By mistake, if you will try to use any other modifiers with top-level class, you will get a compile-time error: “Modifier private not allowed here”.*

*For more detail, go to this tutorial:*[*Private Constructor in Java*](https://www.scientecheasy.com/2021/02/java-private-constructor.html/)

***Default Access Modifier in Java***

*1. When access modifier is not specified to members of a class or a class itself, it is called default access modifier.*

*2. The default can apply to the instance variable, local variable, constructor, methods, inner class, or outer class.*

*3. Default members of a class are visible inside of the class and everywhere within classes in the same package or folder only. Therefore, they can be accessed from outside the classes in the same package but can not be accessed outside the package.*

*4. Default members can be inherited to the subclass within the same package only. It cannot be inherited from outside the package.*

*Let’s understand the default access modifier or specifier better with the help of an example program.*

*In this example program, we have created two packages pack1 and  pack2. We are accessing class A from outside its package since class A is a default, not public. Therefore, it cannot be accessed from outside the package.*

***Program source code 3:***

*package pack1;*

*class Student*

*{*

*// Declaration of Instance variable.*

*int roll = 12; // Here, instance variable is default.*

*// Declare the method.*

*void name() // Here, method has been defined with default access modifier.*

*{*

*System.out.println(" Hello Java");*

*}*

*}*

*package pack2;*

*import pack1.\*;*

*class College*

*{*

*// Main method.*

*public static void main(String[] args)*

*{*

*// Creating an object of class Student from package pack2.*

*Student obj = new Student(); // Compile time error because class Student has been defined with default access modifier and cannot be instantiated from outside the package. But, if it is declared as public then we can instantiate from outside the package.*

*obj.name(); // Compile time error because of the default access modifier.*

*}*

*}*

***Protected Access Modifier in Java***

*1. Protected access modifier can be applied to instance variables, local variables, constructors, methods, inner classes but not the outer class.*

*2. Protected members are accessible inside the class and everywhere within classes in the same package and outside the package but through inheritance only.*

*3. Protected members can be inherited to the subclass.*

*4. If we make constructor as protected then we can create the subclass of that class within the same package but not outside the package*

*Let’s take an example program to understand the concept of default access modifier.*

*In this example, we have created two packages pack1 and pack2. The Student class of pack1 package is public. So, it can be accessed from outside the package.*

*But the name() method of this package is declared as protected. So, it can be accessed from outside the class only through inheritance.*

***Program source code 4:***

*package pack1;*

*public class Student*

*{*

*// Declaration of Instance variable.*

*int roll = 12; // here instance variable is default.*

*protected void name() // Here method has been defined with protected access modifer.*

*{*

*System.out.println("My roll no. is 12");*

*}*

*}*

*package pack2;*

*import pack1.\*;*

*class College extends Student*

*{*

*public static void main(String args[])*

*{*

*// Creating the object of class Student from pack2.*

*Student obj = new Student(); // Calling the method using reference variable obj.*

*obj.name();*

*}*

*}*

*Output:*

*My roll no. is 12.*

***Public Access Modifier in Java***

*1. Public access modifier can apply to instance variables, constructors, inner classes, outer class, methods but not with local variables.*

*2. Public members of a class can be used anywhere.*

*3. Public members of a class can be inherited to any subclass.*

*Let’s understand public access modifier with the help of an example program to understand better.*

***Program source code 4:***

*package pack1;*

*public class Student*

*{*

*int roll = 12; // Here, instance variable is default.*

*public void name() // Here, method has been defined with protected access modifier.*

*{*

*System.out.println(" My roll no. is 12");*

*}*

*}*

*package pack2;*

*import pack1.\*;*

*class College extends Student*

*{*

*public static void main(String args[])*

*{*

*// Creating the object of class Student from pack2.*

*Student obj = new Student();*

*obj.name();*

*}*

*}*

*Output:*

*My roll no. is 12*

***Private Protected Access in Java***

*A variable or field can be declared with two keywords as private and protected together like this:*

*private protected int age;*

*It provides the visibility level in between the “protected access” and “private access”. This modifier makes the variable visible in all the subclasses regardless of what package they are in but the field cannot be accessed by other classes in the same package.*

***Visibility of Access Modifier in Java***

*The below table summarizes the visibility of various access modifiers in java.*

***Key of Access modifiers:***

*Private  >  Default  >  Protected  >  Public*

*More restrictive ----------------> Less restrictive.*

*Decreasing*

***Applicable Modifiers with Classes, Methods, Variables, Interfaces***

***Outer Class (Top-level Class):***

*1. The only five modifiers are applicable to the outer class. They are public, default, final, abstract, and strictfp.*

*2. Final and abstract both cannot be applied simultaneously with a class. It is an illegal combination.*

*3. If a class is declared as public, private, or protected cannot be applied simultaneously with public class. It is also an illegal combination.*

***Inner Class:***

*1. The applicable modifiers for the inner class are public, private, protected, default, final, abstract, static, and strictfp.*

*2. The non-applicable modifiers for inner class are synchronized, native, transient, and volatile.*

*3. The modifiers which are not applicable for inner classes but not applicable for outer classes are private, protected, and static.*

***Methods:***

*1. The only two modifiers out of 12 that are not applicable for methods are transient and volatile.*

*2. If a method is declared as public, we cannot declare simultaneously private or protected with a public method.*

*3. If a method is defined as abstract, we cannot apply final, static, synchronized, native, private, or strictfp.*

***Variables:***

*1. The modifiers applicable for variables are public, protected, private, default, final, static, transient, and volatile.*

*2. The non-access modifiers such as abstract, synchronized, native, and strictfp are not applicable for variables.*

*3. The only applicable modifier with local variable is final. If a variable is declared as final, we cannot declare as volatile.*

***Constructors:***

*1. For a constructor, only access modifiers are applicable. The access modifiers like public, protected, default, and private can be applied with constructors.*

*2. Non-access modifiers cannot be applied with constructors. By mistake, if you apply any other modifiers with constructor except these four access modifiers, you will get a compile-time error.*

***Blocks:***

*1. Static and synchronized modifiers are only applicable for blocks.*

***Outer Interface:***

*1. Public, default, abstract, and strictfp are modifiers that are applicable for outer interface. Except for these modifiers, we cannot apply other modifiers with outer or top-level interface.*

*2. The non-access modifier that is applicable for classes but not applicable for an interface, is final.*

***Inner Interface:***

*1. For inner interface, we can apply modifiers like private, default, protected, public, abstract, static, and strictfp.*

*2. The non-access modifiers such as final, synchronized, native, transient, and volatile are not applicable for inner interface.*

***Outer Enum:***

*1. The only three modifiers like public, default, and strictfp are applicable for outer enum.*

*2. The modifiers which are applicable for classes but not applicable for enum are final and abstract.*

***Inner Enum:***

*1. For enum, we can apply all four access modifiers and two non-access modifiers like static, and strictfp.*

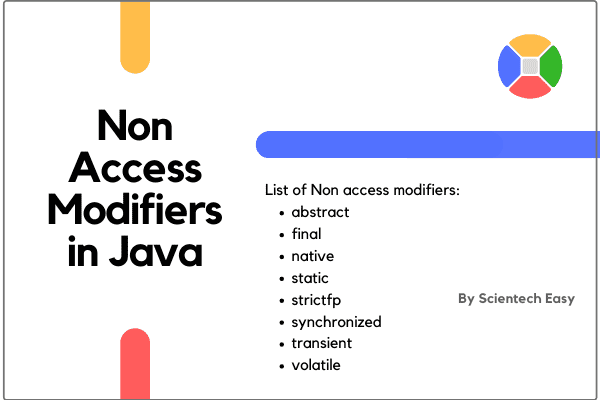
***Non Access Modifiers in Java with Example***

***Non-access Modifiers in Java |****In the previous chapter, we have known*[*access modifiers*](https://www.scientecheasy.com/2020/06/access-modifiers-in-java.html/)*such as public, protected, default, and private.*

*We have also learned the visibility/accessibility of these modifiers with some example programs.*

*Except for these modifiers, all the remaining modifiers available in java are called non-access modifiers. They are abstract, final, native, static, strictfp, synchronized, transient, and volatile.*

*Let’s understand one by one in brief about java non-access modifiers and their usage.*

*[](https://www.scientecheasy.com/2020/06/non-access-modifiers-in-java.html/)*

***Abstract Modifier in Java***

*1. Abstract is a keyword that can be applied with outer class,*[*inner class*](https://www.scientecheasy.com/2020/06/inner-class-in-java.html/)*, method, outer interface, and inner interface. It cannot be applied with variable, constructor, block, and enum.*

*2. When a class is declared with a keyword abstract, it is called abstract class in java.*

*3. An abstract keyword cannot be simultaneously declared with the final keyword.*

*4. An*[*interface in java*](https://www.scientecheasy.com/2019/05/interface-in-java.html/)*is by default abstract and does not need to be declared abstract.*

*5. When a method is declared with an abstract keyword, it is known as abstract method. It contains only a signature and no body. If you declare a method as abstract in a class, the class must be declared as an abstract class.*

*6. Abstract keyword cannot be declared simultaneously with final, private, native, static, or synchronized.*

*For best practice example programs based on abstract keyword, go to this tutorial:*[*Abstraction in Java | Realtime Example*](https://www.scientecheasy.com/2020/05/java-abstraction.html/)

***Final Modifier in Java***

*1. A final is a keyword that can be applied with outer class, inner class, variable, and method. It cannot be applied with interface, constructor, block, and enum.*

*2. A final class cannot be extended.*

*3. A final method cannot be overridden in java.*

*4. Once a final variable is declared and initialized, it cannot be changed.*

*For practice, example programs step by step, go to this tutorial:*[*Final Keyword in Java*](https://www.scientecheasy.com/2020/07/final-keyword-in-java.html/)

***Native Modifier in Java***

*1. A native modifier can be used only with a method. It contains only a signature but not a body.*

*2. A native method is generally used to merge other programming languages like C and C++  code into Java programming.*

*3. A native keyword cannot be used with strictfp simultaneously.*

***Static Modifier in Java***

*1. A*[*static modifier*](https://www.scientecheasy.com/2020/06/java-static-variable.html/)*can be applied with inner class, variable, method, block, and inner interface. It cannot be declared with top-level class, outer interface.*

*2. A method declared with static keyword can be accessed through the class name. It is generally used to access the static variable.*

*3. A static variable or static data member is also accessed through class name.*

***Strictfp Modifier in Java***

*1. A strictfp modifier can be applied with outer class, inner class, method, and outer interface. It cannot be applied with variable, block, inner interface, constructor, and enum.*

*2. A strictfp class uses the IEEE 754-1985 floating-point specification for all of its floating-point operations.*

*3. The method defined within the interface cannot be declared with strictfp modifier.*

*4. Stricfp modifiers cannot be used with native modifiers simultaneously.*

***Synchronized Modifier in Java***

*1. A synchronized modifier in java can be applied only with method and block.*

*2. In the synchronized method block, only one thread is allowed to execute at a time. It makes thread-safe. Therefore, the synchronized method or synchronized block is mainly used for thread safety.*

*3. Statements can also be synchronized in java.*

***Transient Modifier in Java***

*1. The transient modifier can be applied only with variables or data members.*

*2. When a class is serialized then transient data member is not serialized.*

***Volatile Modifier in Java***

*1. A volatile modifier can be applied only with variables. It is a keyword.*

*2. If a data member or variable is declared as volatile, all threads can see consistent the value of variable in java memory.*

*3. A volatile keyword cannot be applied with a method declaration.*

*4. Volatile does not create any kind of lock on the variable in the program.*