Java Non-Access Modifiers

❮ Previous Next ❯

Non-Access Modifiers

Non-access modifiers do not control visibility (like `public` or `private`), but instead add **other features** to classes, methods, and attributes.

The most commonly used non-access modifiers are `final`, `static`, and `abstract`.

Final

If you don't want the ability to override existing attribute values, declare attributes as `final`:

Example

```
public class Main {
  final int x = 10;
  final double PI = 3.14;

  public static void main(String[] args) {
    Main myObj = new Main();
    myObj.x = 50; // will generate an error: cannot assign a value to a final variable
    myObj.PI = 25; // will generate an error: cannot assign a value to a final variable
    System.out.println(myObj.x);
  }
}
```

Try it Yourself »

Static

A `static` method means that it can be accessed without creating an object of the class, unlike `public`:

Example

An example to demonstrate the differences between `static` and `public` methods:

```
public class Main {
  // Static method
  static void myStaticMethod() {
```

```
    System.out.println("Static methods can be called without creating objects");
  }

  // Public method
  public void myPublicMethod() {
    System.out.println("Public methods must be called by creating objects");
  }

  // Main method
  public static void main(String[ ] args) {
    myStaticMethod(); // Call the static method
    // myPublicMethod(); This would output an error

    Main myObj = new Main(); // Create an object of Main
    myObj.myPublicMethod(); // Call the public method
  }
}
```

Try it Yourself »

Abstract

An `abstract` method belongs to an `abstract` class, and it does not have a body. The body is provided by the subclass:

Main.java Second.java

```
// abstract class
abstract class Main {
  public String fname = "John";
  public int age = 24;
  public abstract void study(); // abstract method
}

// Subclass (inherit from Main)
class Student extends Main {
  public int graduationYear = 2018;
  public void study() { // the body of the abstract method is provided here
    System.out.println("Studying all day long");
  }
}
```

Try it Yourself »

Non-Access Modifiers List

For **classes**, you can use either `final` or `abstract`:

Modifier	Description	Try it
`final`	The class cannot be inherited by other classes (You will learn more about inheritance in the Inheritance chapter)	Try it »

`abstract`	The class cannot be used to create objects (To access an abstract class, it must be inherited from another class. You will learn more about inheritance and abstraction in the Inheritance and Abstraction chapters)	Try it »

For **attributes and methods**, you can use the one of the following:

Modifier	Description
`final`	Attributes and methods cannot be overridden/modified
`static`	Attributes and methods belong to the class, not to objects. This means all objects share the same `static` attribute, and `static` methods can be called without creating objects.
`abstract`	Can only be used in an abstract class, and can only be used on methods. The method does not have a body, for example `abstract void run();`. The body is provided by the subclass (inherited from). You will learn more about inheritance and abstraction in the Inheritance and Abstraction chapters
`transient`	Attributes and methods are skipped when serializing the object containing them
`synchronized`	Methods can only be accessed by one thread at a time
`volatile`	The value of an attribute is not cached thread-locally, and is always read from the "main memory"

❮ Previous

Next ❯