Java Encapsulation

❮ Previous Next ❯

Encapsulation

The meaning of **Encapsulation**, is to make sure that "sensitive" data is hidden from users. To achieve this, you must:

- declare class variables/attributes as `private`
- provide public **get** and **set** methods to access and update the value of a `private` variable

Get and Set

You learned from the previous chapter that `private` variables can only be accessed within the same class (an outside class has no access to it). However, it is possible to access them if we provide public **get** and **set** methods.

The `get` method returns the variable value, and the `set` method sets the value.

Syntax for both is that they start with either `get` or `set`, followed by the name of the variable, with the first letter in upper case:

Example

```
public class Person {
  private String name; // private = restricted access

  // Getter

    return name;
  }

  // Setter

    this.name = newName;
  }
}
```

Example explained

The `get` method returns the value of the variable `name` .

The `set` method takes a parameter (`newName`) and assigns it to the `name` variable. The `this` keyword is used to refer to the current object.

However, as the `name` variable is declared as `private` , we **cannot** access it from outside this class:

Example

```
public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.name = "John";  // error
    System.out.println(myObj.name); // error
  }
}
```

Run Example »

If the variable was declared as `public` , we would expect the following output:

```
John
```

However, as we try to access a `private` variable, we get an error:

```
MyClass.java:4: error: name has private access in Person
    myObj.name = "John";
         ^
MyClass.java:5: error: name has private access in Person
    System.out.println(myObj.name);
                  ^
2 errors
```

Instead, we use the `getName()` and `setName()` methods to access and update the variable:

Example

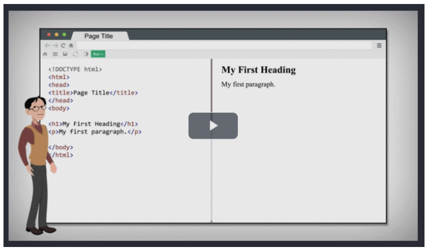
```
public class Main {
  public static void main(String[] args) {
    Person myObj = new Person();



  }
}

// Outputs "John"
```

Try it Yourself »

Why Encapsulation?

- Better control of class attributes and methods
- Class attributes can be made **read-only** (if you only use the `get` method), or **write-only** (if you only use the `set` method)
- Flexible: the programmer can change one part of the code without affecting other parts
- Increased security of data

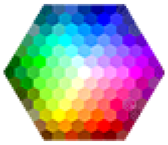
❮ Previous Next ❯



```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

My First Heading

My first paragraph.



CODE GAME

Play Game

Report Error

Spaces

Pro

Buy Certificate

Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
Angular Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples

Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate

FORUM | ABOUT

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using W3Schools, you agree to have read and accepted our terms of use, cookie and privacy policy.

W3Schools is Powered by W3.CSS.