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` (only accessible within the same class)
- provide public **setter** and **getter** 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 have no access to it). However, it is possible to access them if we provide public **getter** and **setter** 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
  public String getName() {
    return name;
  }
```

```
  // Setter
  public void setName(String newName) {
    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 MyClass {
  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 acccess and update the variable:

Example

```
public class MyClass {
  public static void main(String[] args) {
    Person myObj = new Person();
    myObj.setName("John"); // Set the value of the name variable to
"John"
    System.out.println(myObj.getName());
  }
}

// Outputs "John"
```

Run example »

Why Encapsulation?

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

‹ Previous Next ›