Encapsulation in Java with example

BY CHAITANYA SINGH | FILED UNDER: OOPS CONCEPT

Encapsulation simply means binding object state(fields) and behaviour(methods) together. If you are creating class, you are doing encapsulation. In this guide we will see how to do encapsulation in java program, if you are looking for a real-life example of encapsulation then refer this guide: OOPs features explained using real-life examples.

> For other OOPs topics such as inheritance and polymorphism, refer OOPs concepts

Lets get back to the topic.

What is encapsulation?

The whole idea behind encapsulation is to hide the implementation details from users. If a data member is private it means it can only be accessed within the same class. No outside class can access private data member (variable) of other class.

However if we setup public getter and setter methods to update (for example `void setSSN(int ssn)`)and read (for example `int getSSN()`) the private data fields then the outside class can access those private data fields via public methods.

This way data can only be accessed by public methods thus making the private fields and their implementation hidden for outside classes. That's why encapsulation is known as **data hiding.** Lets see an example to understand this concept better.

Example of Encapsulation in Java

How to implement encapsulation in java:
1) Make the instance variables private so that they cannot be accessed directly from outside the class. You can only set and get values of these variables through the methods of the class.
2) Have getter and setter methods in the class to set and get the values of the fields.

```
class EncapsulationDemo{
    private int ssn;
    private String empName;
    private int empAge;

    //Getter and Setter methods
    public int getEmpSSN(){
        return ssn;
    }

    public String getEmpName(){
        return empName;
    }

    public int getEmpAge(){
        return empAge;
    }

    public void setEmpAge(int newValue){
        empAge = newValue;
    }

    public void setEmpName(String newValue){
        empName = newValue;
    }

    public void setEmpSSN(int newValue){
        ssn = newValue;
    }
}
public class EncapsTest{
    public static void main(String args[]){
         EncapsulationDemo obj = new EncapsulationDemo();
         obj.setEmpName("Mario");
         obj.setEmpAge(32);
         obj.setEmpSSN(112233);
         System.out.println("Employee Name: " + obj.getEmpName());
         System.out.println("Employee SSN: " + obj.getEmpSSN());
         System.out.println("Employee Age: " + obj.getEmpAge());
    }
}
```

Output:

```
Employee Name: Mario
Employee SSN: 112233
Employee Age: 32
```

In above example all the three data members (or data fields) are private(see: **Access Modifiers in Java**) which cannot be accessed directly. These fields can be accessed via public methods only. Fields `empName`, `ssn` and `empAge` are made hidden data fields using encapsulation technique of OOPs.

Advantages of encapsulation

1. It improves maintainability and flexibility and re-usability: for e.g. In the above code the implementation code of `void setEmpName(String name)` and `String getEmpName()` can be changed at any point of time. Since the implementation is purely hidden for outside classes they would still be accessing the private field empName using the same methods (`setEmpName(String name)` and `getEmpName()`). Hence the code can be maintained at any point of time without breaking the classes that uses the code. This improves the re-usability of the underlying class.
2. The fields can be made read-only (If we don't define setter methods in the class) or write-only (If we don't define the getter methods in the class). For e.g. If we have a field(or variable) that we don't want to be changed so we simply define the variable as private and instead of set and get both we just need to define the get method for that variable. Since the set method is not present there is no way an outside class can modify the value of that field.
3. User would not be knowing what is going on behind the scene. They would only be knowing that to update a field call set method and to read a field call get method but

what these set and get methods are doing is purely hidden from them.

Encapsulation is also known as “**data Hiding**“.

❮ Previous Next ❯

Comments

sowrag says
MAY 6, 2014 AT 6:29 PM

i am confused about when doGet() and doPost() methods are used ,plz could u clarify my doubt

Reply

Ather says
SEPTEMBER 16, 2014 AT 3:53 PM

We use doPost() method when we dont want the data(while submitting a form) to be send through the URL while in doGet() form data is sent though URL

Reply

Asiri says
NOVEMBER 6, 2014 AT 7:57 PM

i am really confused, in java we cannot have two public methods in a same source file. one method should only be public.

Reply

Chaitanya Singh says
NOVEMBER 15, 2014 AT 8:21 AM

@Asiri, No we can have any number of public methods in a class. We cannot have more than one public class in the same source file.

Reply

Anish says
MAY 16, 2016 AT 5:27 AM