Encapsulation in OOPS | Selenium

Table of content

Encapsulation in Java/selenium

Encapsulation is the process of binding/encapsulating the data and code together. It is used to hide the implementation details. The encapsulation can be achieved by the use of 3 keywords in java.

private : The variables or methods that have been declared as private will not be accessible outside the class, But the same class member can access private methods and variables in the class.

protected : The variable or methods that have been declared as protected will be accessible inside the class in which they have been defined and in the classes which extend this class. Protected Getter and Protected setter can be used to give access to only their family members.

public : The variable or methods that have been declared as public will be accessible anywhere. Also, a class can be made public.

Mostly we use private, and public methods for encapsulation, all the important details which we donot want to disclose to other members will be part of the private method.

But somehow we want to let the user to know the result of the private method, so we will create a public method in a way that the method will show only the result of the private method to the user but not the implementation of the private method.

In case if we want to give more access to the same family member, the inherited class then we create some protected methods which will provide more details of the private method result which outside members of the family cannot see.

In my Language : Consider an organization as a private method, employees as inherited class methods, General public or press as Outside members.

For every Quarter, the organization releases a news letter stating the Profits/Losses to the press and employees, but the same organization gives details about which project made what profit to the employees through the internal newsletter.

In below example we allow the user to set and get the name of the employee using getter and setter, but we donot reveal any other details like how the name is processed and what is the operation happening to make up the emp name

```
package newtest;
class EncapsulationExample{
    private String empName;

    private void modifyEmpName(String newName) {
      empName = newName + " : He acts in BigBang Theory";
    }
    public String getEmpName(){
        return empName;
    }
    public void setEmpName(String newName){
      modifyEmpName(newName);
    }
}
public class EncapsulationTest{
    public static void main(String args[]){
         EncapsulationExample obj = new EncapsulationExample();
         obj.setEmpName("Sheldon Cooper");
         System.out.println(obj.getEmpName());
    }
}
```

Result

```
$javac EncapsulationTest.java
$java -Xmx128M -Xms16M EncapsulationTest
Sheldon Cooper : He acts in BigBang Theory
```

Recommended Readings

- Wrapper Classes in Java
- Collections in Selenium
- Arraylist in Selenium
- HashMap in Java
- Polymorphism in OOPS | Selenium
- Inheritance in OOPS | Selenium
- Classes & Constructors in OOPS | Selenium
- Benefits of Java