Where you have applied OOPS in Automation Framework

Concepts of OOPS in Selenium Automation Framework

Click on > Button

#1. ABSTRACTION

Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users.

Let's see an example of data abstraction in Selenium Automation Framework.

In Page Object Model design pattern, we write locators (such as id, name, Xpath etc.,) and the methods in a Page Class. We utilize these locators in tests but we can't see the implementation of the methods. Literally we hide the implementations of the locators from the tests.

In Java, abstraction is achieved by interfaces and abstract classes. Using interfaces, we can achieve 100% abstraction.

Let's see interface concept below.

#2. INTERFACE

Basic statement we all know in Selenium is **WebDriver driver = new FirefoxDriver();**

WebDriver itself is an Interface. So based on the above statement **WebDriver driver = new FirefoxDriver();** we are initializing Firefox browser using Selenium WebDriver. It means we are creating a *reference variable (driver)* of the *interface (WebDriver)* and creating

an *Object*. Here *WebDriver* is an *Interface* as mentioned earlier and *FirefoxDriver* is a *class*.

An interface in Java looks similar to a class but both the interface and class are two different concepts. An interface can have methods and variables just like the class but the methods declared in interface are by default abstract. We can achieve 100% abstraction and multiple inheritance in Java with Interface.

#3. INHERITANCE

The mechanism in Java by which one class acquires the properties (instance variables) and functionalities of another class is known as Inheritance.

We create a Base Class in the Automation Framework to initialize WebDriver interface, WebDriver waits, Property files, Excels, etc., in the Base Class.

We extend the Base Class in other classes such as Tests and Utility Class.

Here we extend one class (Base Class like WebDriver Interface) into other class (like Tests, Utility Class) is known as Inheritance.

#4. POLYMORPHISM

Polymorphism allows us to perform a task in multiple ways.

Combination of overloading and overriding is known as Polymorphism. We will see both overloading and overriding below.

#1. METHOD OVERLOADING

We use **Implicit wait** in Selenium. Implicit wait is an example of overloading. In Implicit wait we use different time stamps such as SECONDS, MINUTES, HOURS etc.,

Action class in TestNG is also an example of overloading.

Assert class in TestNG is also an example of overloading.

A class having multiple methods with same name but different parameters is called Method Overloading

#2. METHOD OVERRIDING

We use a method which was already implemented in another class by changing its parameters. To understand this, you need to understand Overriding in Java.

Declaring a method in child class which is already present in the parent class is called Method Overriding. Examples are **get** and **navigate** methods of different drivers in Selenium.

#5. ENCAPSULATION

All the classes in a framework are an example of Encapsulation. In POM classes, we declare the data members using **@FindBy** and initialization of data members will be done using Constructor to utilize those in methods.

Encapsulation is a mechanism of binding code and data (variables) together in a single unit.

Other Selenium Automation Framework Concepts

I would like to discuss some other topics which we use in Automation Framework.

#1. WEB ELEMENT

Web element is an interface used to identify the elements in a web page.

#2. WEBDRIVER

WebDriver is an interface used to launch different browsers such as Firefox, Chrome, Internet Explorer, Safari etc.,

#3. FIND BY

Find By is an annotation used in Page Object Model design pattern to identify the elements.

#4. FIND ELEMENT

Find Element is a method in POM to identify the elements in a web page