**Design pattern**

A Design pattern is a generic solution to a common software design/architecture problem. Implementation of these design patterns leads to inclusion of best practices and best solution, evolved over the time by others while working with similar problems.

**Page Object Model in Selenium**

A Page Object Model is a design pattern that can be implemented using selenium webdriver. It essentially models the pages/screen of the application as objects called Page Objects, all the functions that can be performed in the specific page are encapsulated in the page object of that screen. In this way any change made in the UI will only affect that screens page object class thus abstracting the changes from the test classes.

**Page Factory:**

Page Factory is an inbuilt Page Object Model concept for Selenium WebDriver but it is very optimized.

Here as well, we follow the concept of separation of Page Object Repository and Test Methods. Additionally, with the help of PageFactory class, we use annotations **@FindBy** to find WebElement. We use initElements method to initialize web elements

**@FindBy** can accept **tagName, partialLinkText, name, linkText, id, css, className, xpath**as attributes.

**Advantages of using Page Object Model**

* Increases code reusability - code to work with events of a page is written only once and used in different test cases
* Improves code maintainability - any UI change leads to updating the code in page object classes only leaving the test classes unaffected
* Makes code more readable and less brittle
* Creating a Page Object Model in Java

**About TestNg:**

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionality that make it more powerful and easier to use.

It is an open source automated testing framework; where NG of TestNG means Next Generation. TestNG is similar to JUnit but it is much more powerful than JUnit but still it’s inspired by JUnit. It is designed to be better than JUnit, especially when testing integrated classes. Pay special thanks to Cedric Beust who is the creator of TestNG.

TestNG eliminates most of the limitations of the older framework and gives the developer the ability to write more flexible and powerful tests with help of easy annotations, grouping, sequencing & parametrizing.

**Benefits of TestNG**

* There are number of benefits but from Selenium perspective, major advantages of TestNG are :
* It gives the ability to produce HTML Reports of execution
* Annotations made testers life easy
* Test cases can be Grouped & Prioritized more easily
* Parallel testing is possible
* Generates Logs
* Data Parameterization is possible

**TestNG Listeners in Selenium WebDriver**

There are two main listeners.

1. WebDriver Listeners
2. TestNG Listeners

## Listeners in Selenium WebDriver

Listener is defined as interface that modifes the default TestNG's behavior. As the name suggests Listeners "listen" to the event defined in the selenium script and behave accordingly. It is used in selenium by implementing Listeners Interface. It allows customizing TestNG reports or logs. There are many types of TestNG listeners available.

There are many types of listeners which allows you to change the TestNG's behavior.

Below are the few TestNG listeners:

1. IAnnotationTransformer ,
2. IAnnotationTransformer2 ,
3. IConfigurable ,
4. IConfigurationListener ,
5. IExecutionListener,
6. IHookable ,
7. IInvokedMethodListener ,
8. IInvokedMethodListener2 ,
9. IMethodInterceptor ,
10. IReporter,
11. ISuiteListener,
12. ITestListener .

Above Interface are called TestNG Listeners. These interfaces are used in selenium to generate logs or customize the[Testing](https://www.guru99.com/software-testing.html)reports.

In this tutorial, we will implement the ITestListener.

ITestListener has following methods

* **OnStart-** OnStart method is called when any Test starts.
* **onTestSuccess-** onTestSuccess method is called on the success of any Test.
* **onTestFailure-** onTestFailure method is called on the failure of any Test.
* **onTestSkipped-**onTestSkippedmethod is called on skipped of any Test.
* **onTestFailedButWithinSuccessPercentage-**method is called each time Test fails but is within success percentage.
* **onFinish-**onFinish method is called after all Tests are executed.

**Steps to create TestNG Listener**

**Step 1)** Create class "Listener\_Demo" and implements 'ITestListener '. Move the mouse over redline text, and Eclipse will suggest you 2 quick fixes as shown in below screen:

Just click on "Add unimplemented methods". Multiple unimplemented methods (without a body) is added to the code. Check below-

**Step 2)** Create another class "TestCases" for the login process automation. Selenium will execute this 'TestCases' to login automatically.

**Step 3)** Next, implement this listener in our regular project class i.e. "TestCases". There are two different ways to connect to the class and interface.

The first way is to use Listeners annotation (@Listeners) as shown below:

**Step 4):** Execute the "TestCases " class. Methods in class "ListenerTest " are called automatically according to the behavior of methods annotated as @Test.

**Step 5):** Verify the Output