

Selenium

Selenium : -

Selenium is an open-source automation testing tool which is used for automating tests carried out on different web-browsers.

Selenium is basically used to automate the testing across various web browsers. It supports various browsers like Chrome, Mozilla, Firefox, Safari, and IE, and you can very easily [automate browser testing](#) across these browsers using Selenium WebDriver.

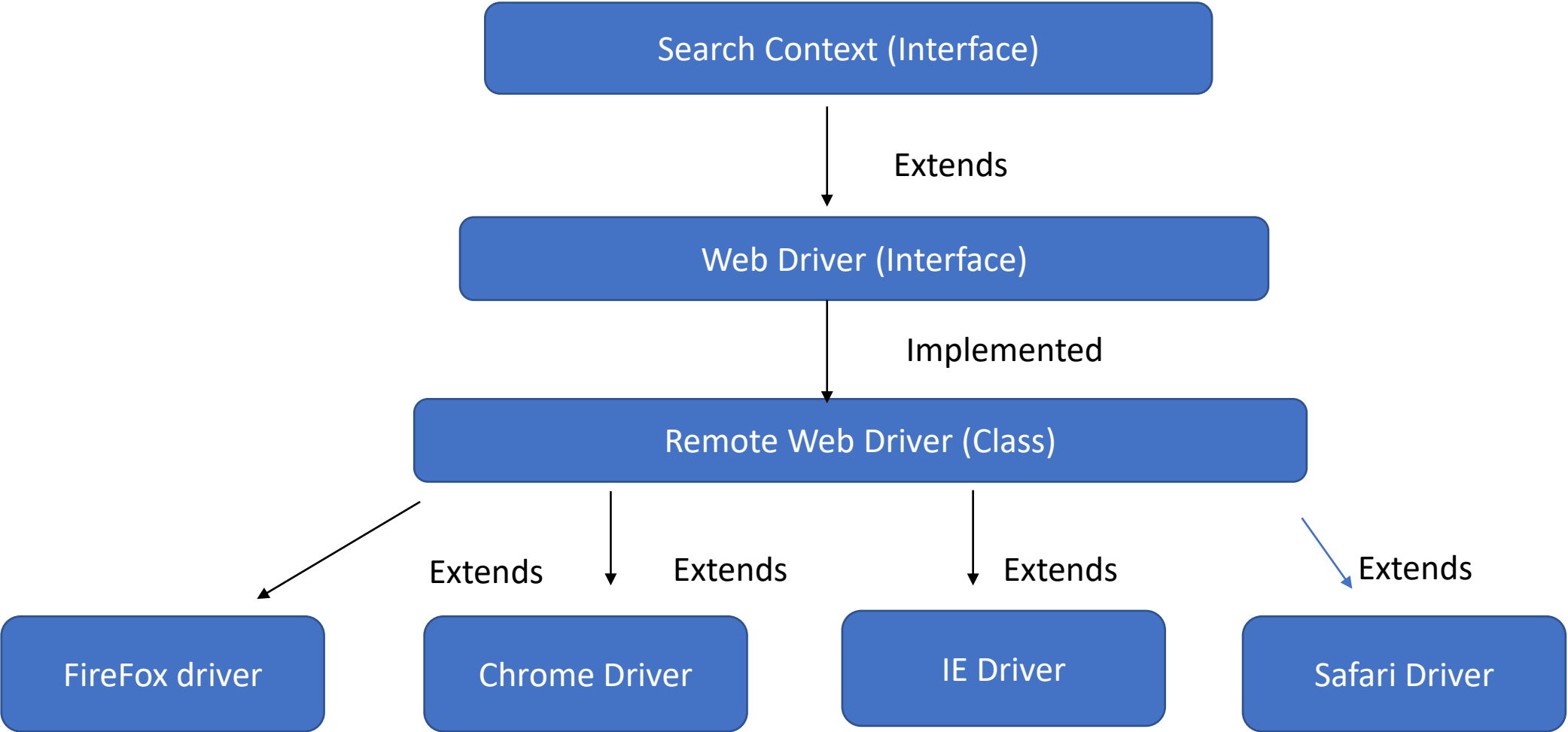
Advantages of Selenium:-

1. Selenium is open source and more than 90% of companies use selenium for automation.
2. Cross browser testing is possible in Selenium.
3. Selenium supports multiple languages like Java, Ruby, Python, C#.
4. Selenium is also platform independent, can be used on various platforms.

Disadvantages of Selenium:-

1. Selenium can automate only web based application.
2. Standalone application cannot be automated by Selenium, like calculator etc.
3. File upload and download is not possible in Selenium to do that we have to use third party tool like Autoit.

Architecture of Selenium:-



Scanner Class :-

Scanner is a predefined class in JAVA which is used to user input from console.

So inorder to take user input we have to create the object of scanner class and pass system.in in its argument. And by using its object reference we can use its method to take different inputs from user.

Few methods are:

nextLine() – to take String as an input

nextInt() – to take integer as an input....and so on

Example program : -

```
public class ChildClass extends Parent {  
    public static void main(String[] args) {  
        System.out.println("Enter the amount");  
        Scanner scan = new Scanner(System.in);  
        double amount = scan.nextDouble();  
        String currency = scan.nextLine(); // to take string as an input  
        System.out.println("Amount entered is" +amount + currency);  
    }  
}
```

Selenium Class 2

Browser initialization in selenium:-

In order to initialize a browser in selenium we have to first the set property of browser for which we will use the below syntax:-

Step 1:-`System.setProperty("webdriver.chrome.driver", "exe_path");`

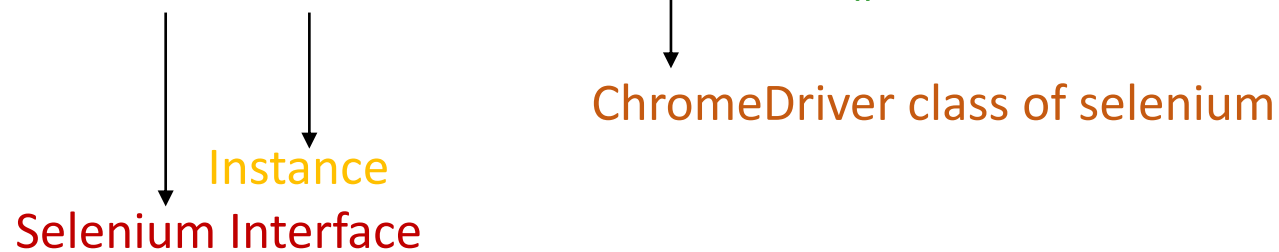
System is class in java, we use the method setproperty of System class. This method include two parameters key and value.

key the name of the system property.

value the value of the system property.

Step 2:- **Instantiate a ChromeDriver class.**

`WebDriver driver = new ChromeDriver();`



This statement create a object reference of WebDriver Interface with with the constructor of browser class. This is also known as upcasting i.e sending back the complete methods back to interface. Here we are sending completed methods back to WebDriver from ChromeDriver class.

WebDriver :-

WebDriver is a selenium interface which provides us methods or say function which are used to perform action on a web browser.

To call any url we use a predefined method of WebDriver called as get() method.

syntax:-

```
driver.get(" url which we want to access")
```

Methods of WebDriver:-

navigate() method:-

navigate method is also used to call or navigate to an url but above that it also perform different functions like Forward, backward or refresh on browser.

Syntax:-

```
driver.navigate().to("url")
```

```
driver.navigate().back(); // to go back to previous page
```

```
driver.navigate().forward(); // to go to the next page
```

```
driver.navigate().refresh(); // to refresh the page
```

Note :- Navigate() is present inside the webdriver, navigate method open the navigation interface and post that we can access the method of navigation interface.

- getTitle() :- This method helps us to get the title of the current page on selenium focus is. This methods returns a String

Syntax:- driver.getTitle();

- getCurrentUrl:- This method of webdriver is used to get the current url of the webpage. This methods returns a String

syntax: - driver.getCurrentUrl();

- Maximize:- To maximize a browser we have method of webdriver called as maximize();

syntax:- `driver.manage().window().maximize();`

- Minimize:- to minimize a browser in selenium we have inbuild method of webdriver.

syntax : - `driver.manage().window().minimize();`

- Note :- manage is method of WebDriver which returns Options Class, then we call the method window() of Options class which returns Window interface, then we use the method maximize/ minimize of Window interface.

Selenium Class 3

How to change size of the browser in Selenium:-

By using setSize method we can change the size of the browser.

The setSize() methods belong to window interface and it expects object the Dimension class as an argument.

syntax :-

```
Dimension d =new Dimension(200, 400);  
driver.manage().window().setSize(d);
```

How to change position of browser in Selenium:-

By using setPosition() method we can change the browser position on window.

The setPosition() methods belong to window interface and it expects object the Point class as an argument.

syatax :-

```
Point p = new Point(500,600);  
driver.manage().window().setPosition(p);
```

Html Page:-

All webpage are made of HTML, and we use this HTML page to perform automation.

```
<html>  
<body>  
<input type="text" class="inputtext" >
```

Tagname

Attribute

attribute value

```
<font style="vertical allign"> forget password</font>
```

Text

TagName :-

Anything that is present in front of "<" the given symbol is known as HTML Tagname

eg:- <html, <body, <input, <a.

Attribute :-

Anything that is present after TagName with "=" equals symbols is known as Attribute.

eg :- text = , ID = , class= .

Attribute value :-

Anything present after Attribute and written after equals symbol in quotes is known as the value of that attribute is called as attribute value.

eg:- type = " text", id = "email ".

Text :-

Anything that is present in between the ">" ---- "<" the given symbol is called as text on an HTML page.

eg:- >forgot Password< ,

Locators:-

Locators are used to find elements on webpage and perform actions on them such as providing input, click, check uncheck etc.

Locators are the static method of BY class. In selenium we find the elements present on the web page by using findElements() method. FindElement() method accepts the BY class in its argument and we use the methods of By class to locate the elements on a webpage.

There are 8 locators in By Class as:-

1)Tagname

2)ID

3)Name

4)ClassName

5)CSS

6)Xpath

7)Linked Text

8)Partial Linked Text

Syntax :-

```
WebElement w = driver.findElement(By.xpath("Xpath expression"));
```

findElement() method :-

findElement is method webdriver which uses the static method of BY class and help us to locate element on webpage.

The findElement() methods return an interface called WebElement.

WebElement :-

WebElement is an interface which provides us different method to perform action on an webpage.

Xpath :-

Xpath is type of locators which help us to locate element on a webpage, xpath provide us multiple option to locate elements on web page.

Type Of Xpaths are:-

1. Xpath by attribute.
2. Xpath by text().
3. Xpath by contains.
4. Xpath by Index.
5. Absolute Xpath.
6. Relative Xpath.

1) Xpath by attribute :-

Here we use attribute and attribute value to locate the element on a webpage.

syntax:

```
//tagname[@attribute="attributeValue"]
```

2) Xpath by text :-

Locating an element on webpage with the help of text.

syntax:-

```
//tagname[text()='textValue']
```

 (Note:- we can use single inverted comas or double inverted comma for mentioning a text.)

3) Xpath by contains:-

Used when we have long attribute value then we can use a part of text to locate element.

syntax:-

```
//tagname[contains(@attributename,partialAttributeValue)]
```

4) Xpath by Index:-

When we have multiple matching of locators at that time to distinguish we use the index value.along with locator.

syntax:-

```
(Any Xpath Expression)[index]
```


5). Absolute Xpath:-

An absolute xpath navigates from root of the parent to its immediate child, to identify a element is called as absolute Xpath, its is the absolute path to the element.

we use '/' to find absolute Xpath.

DrawBack of absolute Xpath:-

- To find absolute xpath user must must draw the HTML tree diagram/structure, which is very difficult and time consuming.
- Absolute xpath are always so lengthy.
- To store absolute xpath it requires large memory.

6) Relative xpath :-

A relative xpath navigates from root of the parent to any child is called as relative xpath.

identify relative xpath. “//” is used to

Html
Head
Body*

div
div
div *

a
a
a
a
a
a
div
div*

a*
html/body/div[3]/div[2]/a[1]

Selenium Class 4

Methods of WebElement:-

Methods of WebElement helps us to perform action on the element on webpage.

- `sendKeys()` :- By using `sendKeys` you can send values to any input field . It accepts string as an argument.
- `click()` :- We can perform click action any input field radio button.
- `isSelected()` :- `isSelected` is used to check whether the element is selected or not. Returns Boolean value.
- `isEnabled()` :- `isEnabled` is used to verify whether the element is in enabled or disabled condition. Returns Boolean value.
- `isDisplayed()` :- `isDisplayed` is used to check whether the element is displayed or not. Returns Boolean value.
- `getText()` :- `getText` is used to get the text from webpage, Returns a String Value.
- `clear()` :- `clear()` is used to clear some existing text from a field.

Practise Website:- <http://demo.guru99.com/test/radio.html>

How to Select value from a select box/ dropdown:-

To select value form list or dropdown :-

1. Identify the element by using findElement method.

```
WebElement product = driver.findElement(By.xpath("xpathexpression"));
```

2. Create the object of Select class and pass the element as an argument of constructor.

```
Select a = new Select(product);
```

3. Then choose the appropriate method from select class to select the value from dropdown.

Methods of select class:-

- selectByIndex(index) ;
- selectByValue("value");
- selectByVisibleText("text");

Note:- By using select class we can only select value form dropdown if the tag name for the element is select.

Practise website :- <https://chercher.tech/practice/practice-dropdowns-selenium-webdriver>

How to take screenshot in selenium:-

In order to take screen shot in selenium we use getScreenShot() method

TakesScreenShot interface.

- So as TakesScreenShot is an interface we need to complete its methods before using it so we will upcast the driver reference of webdriver to Takes ScreenShot interface.

```
File source = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
```

- The getScreenshotAs() method of TakesScreenShot interface takes the screenshot of the webpage and its argument "OutputType.File" set the screenshot output as File which saved in reference of File class.
- Then we have to save the File to a destination in our memory location which is specified in constructor of File class.

```
File destination = new File("FilePath//filename.extension");
```

- Now we have to copy the file from source that is some random memory of java to our required destination so for that we use the copy() method of file handler class which accepts the source and destination in its argument.

```
FileHandler.copy(source, destination);
```

Selenium Class

Popup's :-

Popups are small windows which are open when user perform any action on main page.

There are several types of Popups :-

- Hidden division popup.
- Alert Popup
- Child Browser Popup
- Authentication Pop
- File Upload Popup
- File Download Popup
- Windows Popups.

1. Hidden Division Popup:-

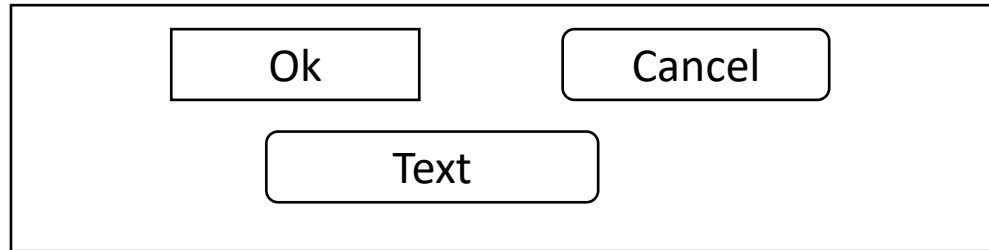
These popups are used for properties or features, We can inspect the elements present on Hidden Division popup.

Hidden division popups are colourful.

2. Alert Popup :-

We cannot inspect element present on Alert popup.

We can drag and drop Alert popups.



How to Handle Alert popup:-

1. To handle Alert popup we need to use method of Alert interface. Which provides us different methods to perform action on a alert popup.
2. When an alert popup is appears, the focus of selenium is not automatically switched to alert popup, tto switch the selenium focus to alert popup we have to use switchTo() method. switchTo() method helps to switch the selenium focus from main page to required area.

```
Alert a = driver.switchTo().alert();
```

3. Now with the help of Alert interface reference we can call the methods of alert interface to perform action on the alert popup. Methods to Alert interface are.
1. Accept() :- To select OK or accept the alert.
 2. Dismiss() :- To select Cancel or dismiss the alert.
 3. getText() :- We can get the text present on alert popup.
 4. sendKeys() :- We can send the text in the input field of alertpopup.

Child Browser Popup:-

This are basically multiple windows of browser that are opened when we perform some action a webpage. We can inspect the element on child browser popups. This are just like any other webpage but occur when we perform some action.

How to handle child browser popups:-

- To handle child browser popup we use getWindowHandle() method of WebDriver, this method returns the address of main webpage on which selenium is currently focusing.

```
String a = driver.getWindowHandle();
```

If we have multiple popups browser opened at the same instanc then inorder to get the address of all the popup's We use another method of WebDriver known as getWindowHandles() methods which returns a Set of string which holds the address of all the Browser popup.

1. getWindowHandles();-

Returns the address of main page and its return type is String.

2. getWindowHandles();-

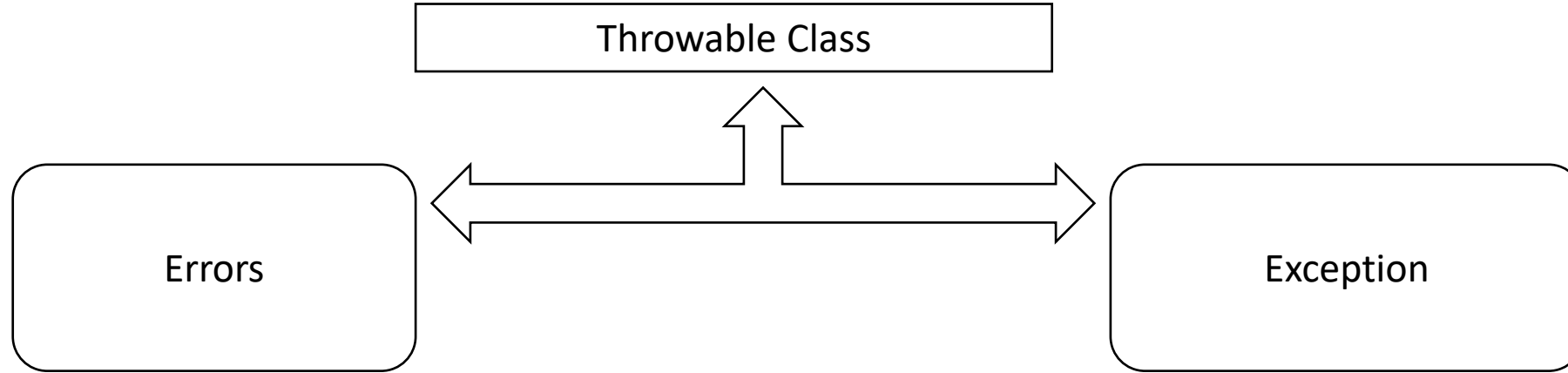
Returns the address of all the window opened by selenium and its return type is set of String ((Set<String>). Now inorder to iterate over Set we have to use iterator method.

```
Set<String> a =driver.getWindowHandles();
```

Now inorder to iterate over set we have use iterator method of Set which returns Iterator class. And then using the ,method of Iterator class we can iterate over a Set.

Selenium

Errors and Exception:-



Errors : - [Error](#) refers to an illegal operation performed by the user which results in the abnormal working of the program. Programming errors often remain undetected until the program is compiled or executed. Some of the errors inhibit the program from getting compiled or executed. Thus errors should be removed before compiling and executing. It is of three types:

- Compile-time
- Run-time
- Logical

Exeption:-

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions. In Java, there are two types of exceptions:

- 1.Checked exceptions.
- 2.Unchecked exceptions

1.Checked Exception:-

These are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using the [throws keyword](#).

2.Unchecked Exception :-

These are the exceptions that are not checked at compile time.



Errors

Recovering from Error is not possible.

All errors in java are unchecked type.

Errors are mostly caused by the environment in which program is running.

Errors can occur at compile time as well as run time. Compile Time:
eg Syntax Error
Run Time: Logical Error.

They are defined in java.lang.Error package.

Examples : java.lang.StackOverflowError,
java.lang.OutOfMemoryError



Exceptions

We can recover from exceptions by either using try-catch block or throwing exceptions back to the Errorsthe The programoccurcaller.


Exceptions include both checked as well as unchecked type.

Program itself is responsible for causing exceptions.

All exceptions occurs at runtime but checked exceptions are known to the compiler while unchecked isnot.

They are defined in java.lang.Exception package

Examples : Checked Exceptions : SQLException, IOException
Unchecked Exceptions : ArrayIndexOutOfBoundsException,
NullPointerException, ArithmeticException.



How to handle a Exception in java :-

We handle the exception in Java by using Try and Catch Block.

Java try block

Java **try** block is used to enclose the code that might throw an exception. It must be used within the method.

If an exception occurs at the particular statement in the try block, the rest of the block code will not execute. So, it is recommended not to keep the code in try block that will not throw an exception.

Syntax:-

```
try
{
    block of code which can throw exception
}
catch(ArithmeticException e) //exception to be handled
{
    Statement/ code to be executed when the exception occurs
}
```

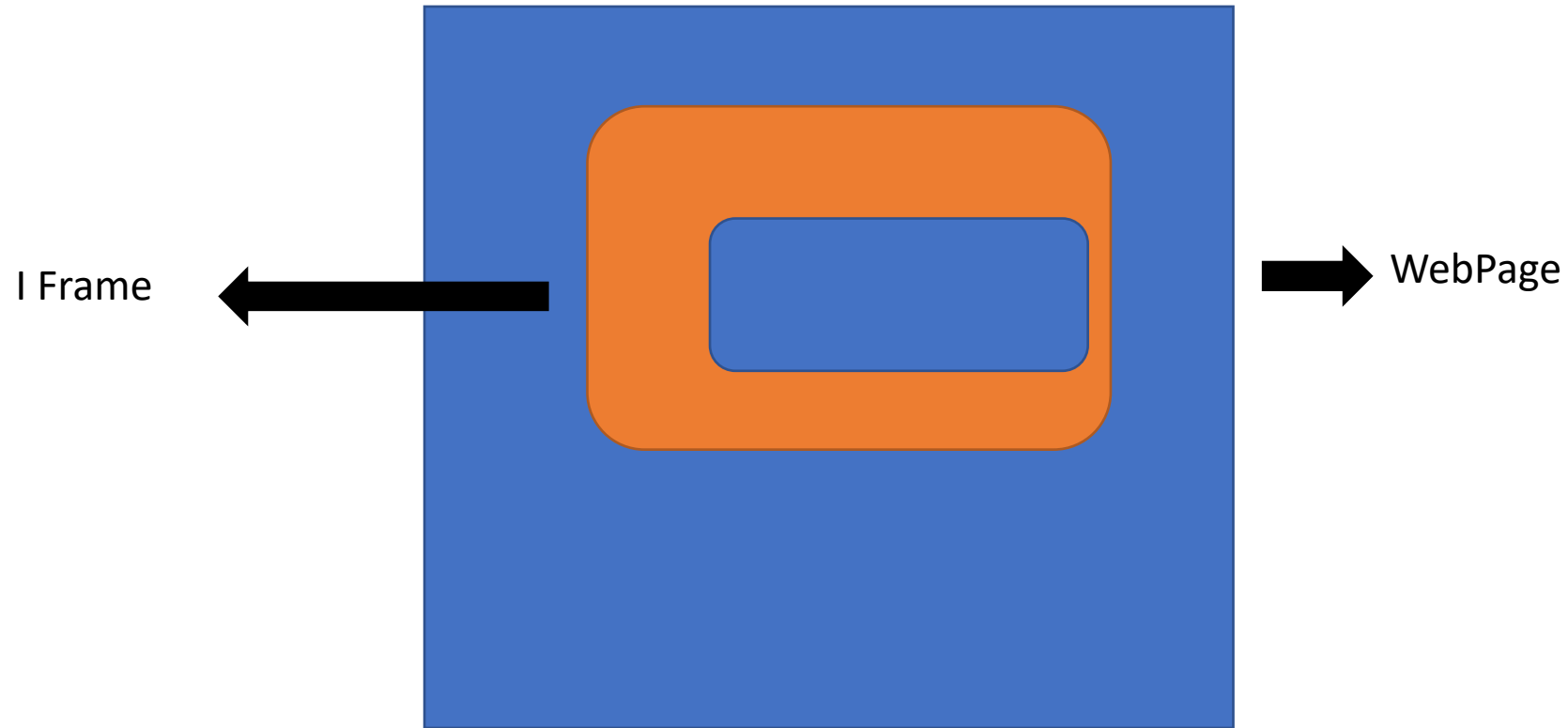

Selenium

Handling of Iframes:-

Displaying a part of webpage as another part of webpage is called as iframe.

Essentially, an iframe is a **HTML document that is embedded inside another document on a website**. Or a webpage embedded in another webpage.

An I frame has always <iframe> as tagname.



Selenium cannot directly focus on iframes, so we cannot directly perform action on the elements present on the iframe. So we have to switch the focus of selenium to the frame.

Inorder to switch the focus of iframe we need to use :-

`driver.switchto().frame(Id or index or webelement)`

The frame method accepts different argument in it to switch to a particular iframe:-

- `Driver.switchTo().frame(id of Iframe as string)`
- `Driver.switchTo().frame (index of iframe)`
- `Driver.switchTo.frame(WebElement of iframe).`

parentframe():

Inorder to switch from iframe to its immediate parent frame we use method `parentFrame()`.

DefaultContent() :-

To switch frame any inner iframe to main page we use method called `defaultContent()`.

<http://demo.automationtesting.in/Frames.html>

<https://demoqa.com/frames>



Selenium

Parametrization:-

How to fetch data from an excel sheet.

We have to use external library called as apache poi.

<https://www.apache.org/dyn/closer.lua/poi/release/bin/poi-bin-5.1.0-20211024.zip>

First Create the object of FileInputStream class which expects the path of file in argument from where we have to fetch the data.

File InputStream opens a connection to the file .

```
FileInputStream file = new FileInputStream(path of file);
```

Then by using the help of ApachePOI's WorkbookFactory we can fetch the data from the excel sheet.

```
WorkbookFactory.create(file).getSheet("Data").getRow(a).getCell(b).getStringCellValue()
```

#Action Class in Selenium is a built-in feature provided by the selenium for handling keyboard and mouse events. It includes various operations such as multiple events clicking by control key, drag and drop events and many more.

In order to use and perform mouse actions we have to first create the object of Actions Class. Which accepts the instance of webdriver in its constructor.

Syntax:-

```
Actions act = new Actions(driver);
```

Now with the help of reference of Actions class we can call the methods of Actions class and perform required actions on the webpage.

Perform()

:- it is used to perform actions and will be used after every method called from Actions class.

Build() : it is used when we call multiple mouse action together, it builds all the actions together.

<https://www.globalsqa.com/demo-site/draganddrop/>
http://demo.guru99.com/test/simple_context_menu.html
<https://demoqa.com/text-box>

Method	Description
clickAndHold()	Clicks (without releasing) at the current mouse location.
contextClick()	Performs a context-click at the current mouse location. (Right Click Mouse Action)
doubleClick()	Performs a double-click at the current mouse location.
dragAndDrop(source, target)	Performs click-and-hold at the location of the source element, moves to the location of the target element, then releases the mouse. Parameters: source- element to emulate button down at. target- element to move to and release the mouse at.
dragAndDropBy(source, x-offset, y-offset)	Performs click-and-hold at the location of the source element, moves by a given offset, then releases the mouse. Parameters: source- element to emulate button down at. xOffset- horizontal move offset. yOffset- vertical move offset.
keyDown(modifier_key)	Performs a modifier key press. Does not release the modifier key – subsequent interactions may assume it's kept pressed. Parameters: modifier_key – any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL)
keyUp(modifier _key)	Performs a key release. Parameters: modifier_key – any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL)
moveByOffset(x-offset, y-offset)	Moves the mouse from its current position (or 0,0) by the given offset. Parameters: x-offset- horizontal offset. A negative value means moving the mouse left. y-offset- vertical offset. A negative value means moving the mouse down.
moveToElement(toElement)	Moves the mouse to the middle of the element. Parameters: toElement- element to move to.
release()	Releases the depressed left mouse button at the current mouse location
sendKeys(onElement, charsequence)	Sends a series of keystrokes onto the element. Parameters: onElement – element that will receive the keystrokes, usually a text field charsequence – any string value representing the sequence of keystrokes to be sent


```
Actions action = new Actions(driver);
WebElement email = driver.findElement(By.xpath("//input[@name='email']"));
email.sendKeys("Subodh");
action.keyDown(Keys.CONTROL);
action.sendKeys("a");
action.keyUp(Keys.CONTROL);
action.build().perform();
Thread.sleep(2000);
action.keyDown(Keys.CONTROL);
action.sendKeys("c");
action.keyUp(Keys.CONTROL);
action.build().perform();

action.sendKeys(Keys.TAB);
action.build().perform();

WebElement pass = driver.findElement(By.xpath("//input[@name='pass']"));

action.keyDown(Keys.CONTROL);
action.sendKeys("v");
action.keyUp(Keys.CONTROL);
action.build().perform();
```

Selenium

#FindElements :-

<http://demo.guru99.com/test/ajax.html>

FindElements in Selenium command takes in By class methods as the parameter and returns a list of web elements. It returns an empty list if there are no elements found using the given locator strategy and locator value. Below is the syntax of find elements command.

Syntax:-

```
List<WebElement> elementName = driver.findElements(By.LocatorStrategy("LocatorValue"));
```

Find Element	Find Elements
Returns the first most web element if there are multiple web elements found with the same locator	Returns a list of web elements
Throws exception NoSuchElementException if there are no elements matching the locator strategy	Returns an empty list if there are no web elements matching the locator strategy
Find element by XPath will only find one web element	It will find a collection of elements whose match the locator strategy.
Not Applicable	Each Web element is indexed with a number starting from 0 just like an array

How to scroll up and down in selenium:-

We use JavascriptExecutor interface of selenium to scroll up and down on a webpage.

We have to first upcast driver to JavascriptExecutor.

```
((JavascriptExecutor)driver)
```

Then with help of executeScript method of JavascriptExecutor we can scroll a page using the page coordinated or by using any WebElement.

```
:- executeScript("arguments[0].scrollIntoView(true)", WebElement)  
:- executeScript("window.scrollTo(100,200)");
```

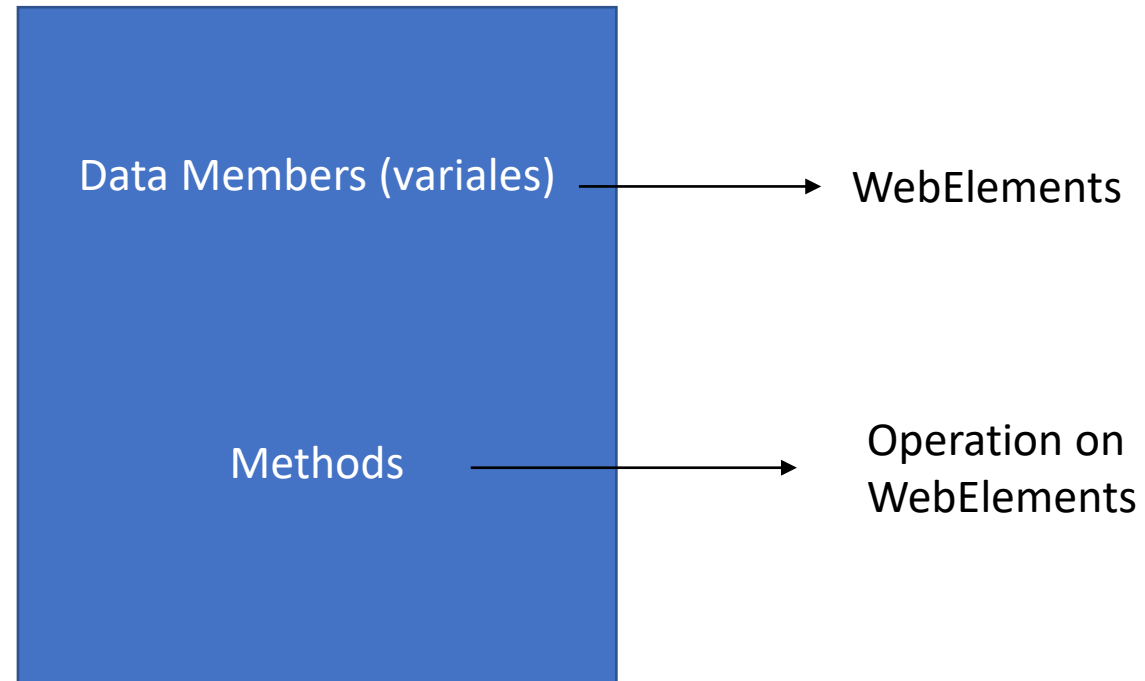
Selenium

Page Object Model (POM) :-

Page Object Model (POM) is a design pattern, popularly used in test automation that creates Object Repository for web UI elements. The advantage of the model is that it reduces code duplication and improves test maintenance.

-Object Repository :-

All the data members and method functions stored in the separate file is called as object repository.



Under this model, for each web page in the application, there should be a corresponding Page Class. This Page class will identify the WebElements of that web page and also contains Page methods which perform operations on those WebElements. Name of these methods should be given as per the task they are performing

Properties of POM classes:-

- **POM** strictly follow the encapsulation and abstraction i.e All the data members declared inside the pom class are restricted to private with private access specifier & data members are declared globally.
- The constructor inside the pom class are declared public.
- All the methods inside the Pom class are declared with public access specifier.
- To access the data members of POM class we need to call it in the methods inside the same class.

```
public class POM {  
    @FindBy(xpath="//input[@id='email']")private WebElement email;  
    @FindBy(xpath="//input[@id='pass']")private WebElement pass;  
    public POM(WebDriver driver) {  
        PageFactory.initElements(driver, this);  
    }  
  
    public void email(String UserName) {  
        email.sendKeys(UserName);  
    }  
  
    public void pass(String password) {  
        pass.sendKeys(password);  
    }  
  
}
```


Selenium

TestNG:-

TestNG is an automation testing framework in which NG stands for “Next Generation”. TestNG is inspired by [JUnit](#) which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make [end-to-end testing](#) easy.

Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

TestNG Annotations:-

@BeforeClass:- The method written under this annotation is called once when the class execution is started.

@AfterClass:- The method written under this annotation will be executed only once at the end of the class execution.

@BeforeMethod:- The method written under this annotation will be executed before every test in the class.

@AfterMethod:- The method written under this annotation will be executed after every test in the class.

@Test :- The method written under this annotation will be executed only once in the class and we can write multiple test annotation in a class.

@Before Test:- It will run before the testing starts i.e before the class annotations.

@Parameter :- if we want to pass some parameter to the test then we use @ Parameters annotation with the test provide the required value.

Advantages of TestNG:-

- TestNG is a [testing](#) framework that is capable of making Selenium tests easier to understand and of generating reports that are easy to understand.
- The Console window in Eclipse generates a text-based result while the TestNG window is more useful because it gives us a graphical output of the test result plus other meaningful details such as:
 - Runtimes of each method.
 - The chronological order by which methods were executed
- TestNG is capable of generating HTML-based reports.

Keywords in TestNG:-

1. Priority:- When we have multiple test in a class and if we have to set the priority of the test then we have to “priority” keyword with the @Test annotation, it tells the priority of test and sequence of execution is followed with priority.
Syntax :-

@Test(priority = 1)

- Priority can be by default zero(0) i.e @Test
- Priority can be +ive integer i.e @Test(priority = 1)
- Priority can be -ive integer i.e @Test(priority = -1)
- Priority can be duplicate in this the test execution will be based on alphabetic sequence.
- Priority cannot be decimal or variable.

For Ex:- @Test(priority = 1)
 public void Test()
 {
 System.**out.println**("Test3");
 }

2. Invocation Count :- When we have to execute a single test multiple time then we can use the keyword “invocation count “ with the @Test annotation, I will execute the test number of times we want.

Syntax :- @Test(invocationCount = 2)

For ex:-

```
@Test(invocationCount =2)
    public void Test2()
    {
        System.out.println("Test2");
    }
```

3. Enabled = false :-

If we want to skip some test from execution then in order to skip it we use “enabled= false” keyword with @Test annotation in TestNG.

for ex:-

```
@Test(enabled = false)
    public void Test2()
    {
        System.out.println("Test2");
    }
```

4. timeOut :- (time in milliseconds)

If a test contains multiple test methods and if we want the test to be complete and in a time duration and if it takes more time the test should automatically fail then we use “timeout” keyword with @Test annotation. Time is in milliseconds

for ex:-

```
@Test(timeOut = 1000)
    public void Test() throws InterruptedException
    {
        Thread.sleep(2000);
        System.out.println("Test3");
    }
```

5. DependsOnmethod = {method name} :-

If a test execution depends on another test then we need to use “dependsOnMethod”

Keyword with @Test annotation.

for ex:-

```
@Test(dependsOnMethods = {"Test2"})
public void Test() throws InterruptedException
{
    System.out.println("Test3");
}
```

TestNG.xml:-

TestNG.xml file is a configuration file that helps in organizing our tests. It allows testers to create and handle multiple test classes, define test suites and tests. It makes a tester's job easier by controlling the execution of tests by putting all the test cases together and run it under one XML file. This is a beautiful concept, without which, it is difficult to work in TestNG.

Threads:-

- It provides parallel execution of test methods.
- It allows the dependency of one test method on another test method.
- It helps in prioritizing our test methods.
- It allows grouping of test methods into test groups.
- It supports the parameterization of test cases using @Parameters annotation.
- It helps in Data-driven testing using @DataProvider annotation.
- It has different types of assertions that help in validating the expected results with the actual results.
- It has different types of HTML reports, Extent reports, etc. for a better and clear understanding of our test summary.

Concepts Used In TestNG.xml

#1) A Suite is represented by one XML file. It can contain one or more tests and is defined by the <suite> tag.

#2) A Test is represented by <test> and can contain one or more TestNG classes.

#3) A Class is a Java class that contains TestNG annotations. Here it is represented by the <class> tag and can contain one or more test methods.

#4) methods are the methods that are present in a class, here it is represented under <methods>.

#5) If you want to include specific test methods of class you have to mention then with <include> tag.

#6) If you want to exclude a particular test method from a class then you have to mention the method with <exclude> tag.

#7) If you want to pass parameters to a method from testNG xml then we have to pass then with <parameter> tag.

TestNG.xml:-

```
<test name="Basic Validation">
  <parameter name="a" value="10"/>
  <parameter name ="b" value ="20"/>
  <parameter name = "file" value ="velocity"/>
  <classes>
    <class name="demo.TestExecution"/>
    <methods>
      <include name = "loginToZerodha()"/>
      <exclude name = "method name"/>
    </methods>
  </classes>
</test> <!-- Test -->

</suite> <!-- Sanity -->
```

Selenium

Waits in Selenium :-

When a page is loaded by the browser the elements which we want to interact with may load at different time intervals. Not only it makes this difficult to identify the element but also if the element is not located it will throw an “**ElementNotVisibleException**” exception. Using Selenium Waits, we can resolve this problem.

Types of wait:-

1. Implicit Wait
2. Explicit wait
3. Fluent Wait

1.Implicit Wait:-

The **Implicit Wait in Selenium** is used to tell the web driver to wait for a certain amount of time before it throws a “No Such Element Exception”. The default setting is 0. Once we set the time, the web driver will wait for the element for that time before throwing an exception.

Implicit wait is applicable every driver instance i.e when we use implicit wait it wait for every element to be visible.

Syntax:- `driver.manage().timeouts().implicitlyWait(20,TimeUnit.SECONDS);`

Implicit wait will accept 2 parameters, the first parameter will accept the time as an integer value and the second parameter will accept the time measurement in terms of SECONDS, MINUTES, MILLISECOND, MICROSECONDS, NANOSECONDS, DAYS, HOURS, etc.

2.Explicit wait:-

The **Explicit Wait in Selenium** is used to tell the Web Driver to wait for certain conditions (Expected Conditions) or maximum time exceeded before throwing “ElementNotVisibleException” exception. It is an intelligent kind of wait, but it can be applied only for specified elements.

In explicit wait we use time parameter along with a particular condition to happen.

Steps to use:-

- First create reference wait for “**WebDriverWait**” class and instantiating using “**WebDriver**” reference and time.

syntax:-

```
WebDriverWait wait = new WebDriverWait(WebDriverReference,TimeOut);
```

```
WebElement w = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("xpathexpression")));
```

500 milli sec

Different condition to be used with explicit wait:-

- 1.alertIsPresent() *
- 2.elementSelectionModeToBe()
- 3.elementToBeClickable()*
- 4.elementToBeSelected()*
- 5.frameToBeAvaliableAndSwitchToIt()
- 6.invisibiltyOfTheElementLocated()
- 7.invisibiltyOfElementWithText()
- 8.presenceOfAllElementsLocatedBy()
- 9.presenceOfElementLocated()
- 10.textToBePresentInElement()
- 11.textToBePresentInElementLocated()
- 12.textToBePresentInElementValue()
- 13.titleIs()
- 14.titleContains()*
- 15.visibilityOf()*
- 16.visibilityOfAllElements()
- 17.visibilityOfAllElementsLocatedBy()
- 18.visibilityOfElementLocated()*

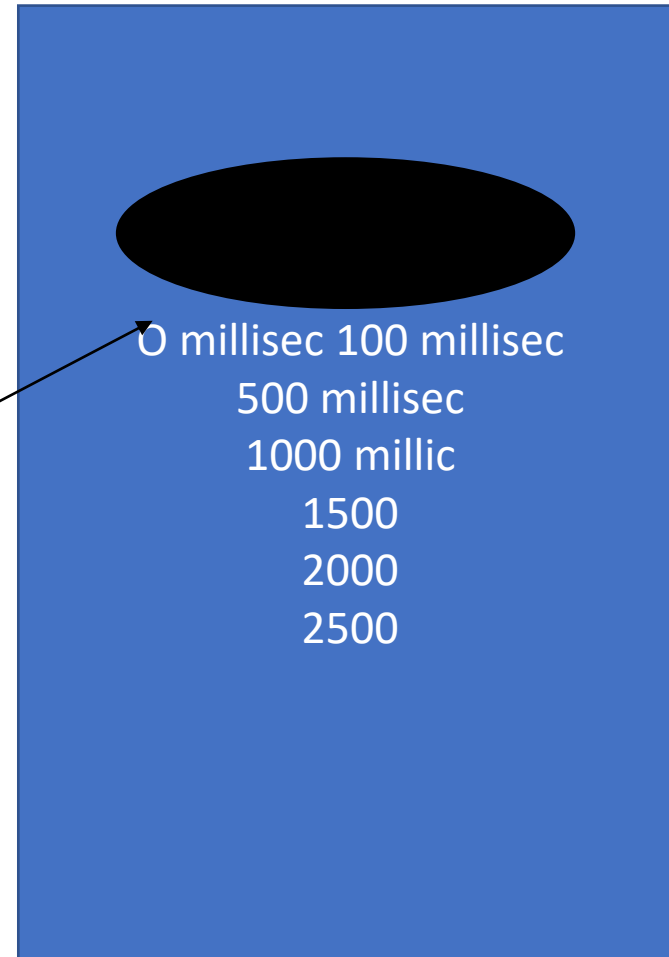
3. Fluent wait:-

The **Fluent Wait in Selenium** is used to define maximum time for the web driver to wait for a condition, as well as the frequency with which we want to check the condition before throwing an “ElementNotVisibleException” exception. It checks for the web element at regular intervals until the object is found or timeout happens.

Syntax:-

```
FluentWait<WebDriver> wait = new FluentWait<WebDriver>( driver);  
wait.withTimeout(Duration.ofMillis(30000)); //Total wait time  
wait.pollingEvery(Duration.ofMillis(100));//interval in which it to and try  
fetch element  
wait.ignoring(Exception.class); // to ignore exception  
wait.until(ExpectedConditions.visibilityOf(WebElement));
```

ExplicitWait 5000
milliisec
by 500 millisec



Selenium

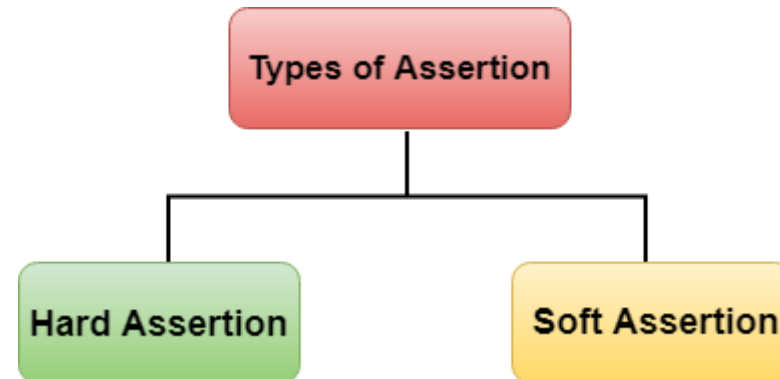
#Assertions :-

Assertion determines the state of the application whether it is the same what we are expecting or not. If the assertion fails, then the test case is failed and stops the execution.

Assertions are basically the validation point which we apply in a test.

There are two types of Assertion:

- Hard Assertion
- Soft Assertion



1. Hard Assert :-

In Hard assert when assertions get fail immediate exception will be thrown for current @Test.

The next line in the current @Test will not execute and it fails the test and stops the execution. The test then executes the next test.

We use static method Assert class to apply Hard assert in Test.

- AssertEquals
- AssertNotEquals
- AssertTrue
- AssertFalse
- AssertNull
- AssertNotNull

2. Soft Assert:-

In soft assert when assertion get fail then it record the failure event and continue the execution, it will not fail the test.

Soft Assert is class which are used to add assertion and the method inside SoftAssert class are nonstatic.

Syntax:-

```
SoftAssert assertion = new SoftAssert();  
assertion.assertFalse(pom.loginIsEnabled());
```

Methods of SoftAssert:-

1. assertEquals();
2. assertNotEquals();
3. assertNotNull();
4. assertTrue();
5. assertSame();
6. assertAll();

Note:- when we use soft assertion in a test immediate failure is not displayed when the condition fails, so inorder to get failure we have to call asserAll() method which will the assertion error that occurred at the instance.