SELENIUM

Q1. What is automation testing ? What are its advantages & disadvantages ?
Ans. Doing any task using a tool or system without manual invention is called Automation Testing. You can also call it a replication of human effort.

Advantages	Disadvantages
1. It is faster 2. Saves time 3. Reduces effort 4. Increases the quality	1. Initial investment is high 2. Required skilled manpower

Q2. How can we achieve automation?
Ans. We can achieve automation with the help of some tool and software like QTP, Selenium.

Q3. What is Selenium? What are its advantages and disadvantages?
Ans. It is a free (open source) testing suite, containing tools each with different approach for test automation of web applications.

Advantages	Disadvantages
1. Open source 2. Multiple OS support 3. Multiple language support 4. Multiple browser support 5. Multiple Framework (login page) 6. Web based automated system	1. It does not support window application (ex - excel, paint) 2. We can't automate captcha, OTP and animation (animation-select traffic lights like that)

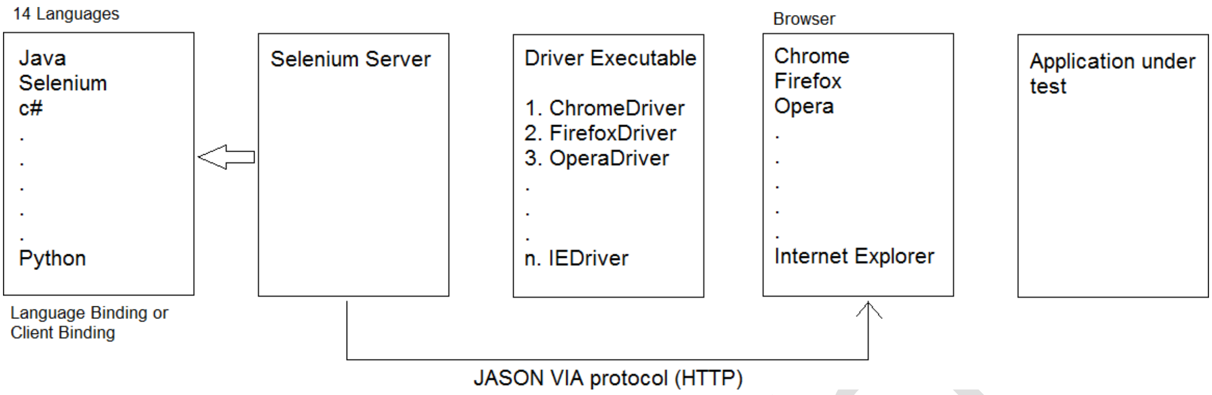
Q4. What is open source?
Ans. It means that we can download for free and we can see source code and modification in that source code.

Q5. Explain System.SetProperty method
Ans. System - It is an inbuilt final class present in java.language package.
SetProperty - It is a static method of system class which will take 2 arguments key and value pair in the form of string.
Key = which browser we want to launch
Value = Path of driver executable

Q6. Explain Selenium architecture.

Ans. 1. Selenium standalone server combined with 14 languages which is known as language binding or client binding.

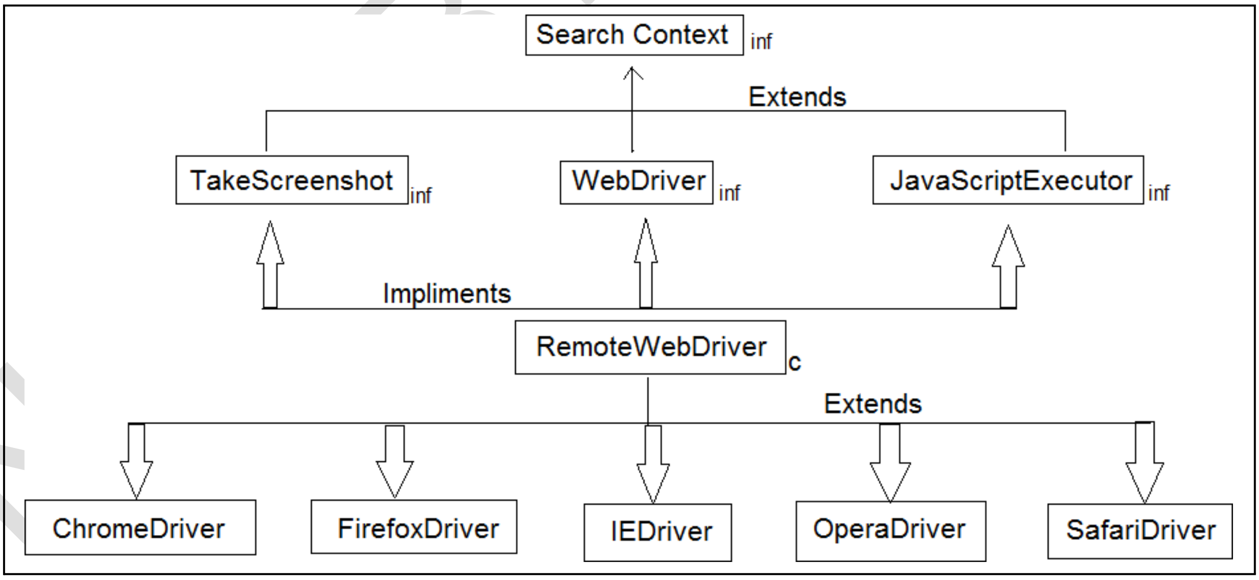
2. With the help of Selenium server we will perform real action on browser with the help of driver execution using JASON (Javascript Object Notation) VIA protocol over HTTP and after that our application under test.



Q7. Explain Java Selenium architecture.

Ans. In Java Selenium architecture supermost interface is search context. After that we have three more interface -

1. Take screenshot
2. WebDriver
3. JavascriptExecutor.



Where, Inf = Interface
C = class

All are interfaces so they will have only abstract methods,with no body no implementation. So we will provide implementation in remote webdriver class and we will use this method in driver class.

(vimp interview qsn)

Q8. Why we are writing WebDriver instead of FirefoxDriver and ChromeDriver ?

Ans. We are performing upcasting over here, in order to avoid unnecessary method. Because if we are creating an object of any driver class, so it will give me the method which we will not use and in feature if we want to change from one browser to any browser we can change it easily because WebDriver is parent of all driver class.

Q9. How to maximize the browser ?

Ans. We can maximize the browser with the help of method chaining. We need to call a non static method call "manage" which is present in WebDriver and it will return options object, with the help of options object we will call "window()". It will return window object we will call "maximize ()" which will return void.

2.optionsObject 4.void()

driver.manage().window().maximize()

1.WebDriver 3. window.

Q10. What is difference between quit() and close() method ?

Ans.

quit()	close()
It will close all the windows or tabs which is associated with the current driver (i.e.chromedriver or any other driver). Basically it will kill the driver instance. Ex - It will close both Parent tab & Child tab.	It will close the current tab in which, driver has focused. It will close the browser if there is only one tab. Ex - It will close only Parent tab which is only in focus.

Q11. How to handle child browser (Windows popup)?

Ans. We can handle child Browser or child browser popup with the help of non static method of WebDriver called getwindowhandle() or getwindowhandles().

Q12. Explain Web driver methods
Ans.

Sr. No.	Method's	Use	Return Type	Argument
1	get()	Used to load URL	Void	String
2	getTitle()	Used to fetch Title of Web Page	String	No
3	Close()	Used to Close Current Tab of Browser	Void	No
4	quit()	Used to close all the web browser windows (parent with child's)	Void	No
5	getCurrentUrl()	Use to Fetch URL of Web Page	String	No
6	Navigate()	Used To Access Browser History	Navigation Object	No
6.1	forward()	Used to go forward in browser history	Void	No
6.2	back()	Used to go back in browser history	Void	No
6.3	refresh()	Used to refresh current page	Void	No
7	Switchto()	It is used to Switch into Frames, Windows, tabs and alerts	Target Locator	No
8	getwindowHandles ()	Used to fetch the Id of both parent and child tab	Set<String >	No
9	getwindowHandle ()	It is used to fetch Id of Parent tab	String	No
10	getPageSource()	It is used to fetch the html code of web page	String	No

11	Manage()	Used mange the window	String	No

LOCATOR'S :

Q13. What are locators ? explain its types

Ans. Locators - They are used to locate the web element or application element, like text field button.

Types of locators -

1. id
2. name
3. className
4. tagName
5. linkText.
6. partialLinkText
7. CSS selector
8. X-path

All locators are present in "By class" and they are "static method" which will take string as an argument.

Ex - By.id(" ")
By.name(" ")

Q14. How to find web element in a web page ?

Ans. To find web element we need to take help of findElement() and findElements() . findElement and findElements are "non static" method which are present in "Search-Context". Since the "Search-Context" is parent of WebDriver so we can call this method with the help of "WebDriver object" both method will accept "By class" as an argument.

For multiple WebElement -

List<WebElement> ElementName = driver.**findElements**(By.LocatorStrategy("LocatorValue"));

For Single WebElement -

WebElement ElementName = driver.**findElement**(By.LocatorStrategy("LocatorValue"));

Q15.What is LinkText & PartialLinkText ?
Ans. linkText () - It is a locator present as static method of by class which will take inner text of Link tag(<a> </a>) in argument as a string.

Ex. -
WebElement forgotPass = driver.findElement(By.linkText("**Forgot your password?**"));
forgotPass.click();

PartialLinkText - It is a locator present as static method of by class which will take inner text of Link tag(<a> </a>) partially in argument as a string

Ex -
WebElement forgotPass = driver.findElement(By.partialLinkText(**"Forgot**"));
forgotPass.click();

Q16. Explain tagName().
Ans. tagName() - It is the locator present as a "static method" of by class, which will take "String" as an argument, when we know the tagName OR when we are working on a single tag that time we can choose tagName as a locator strategy.

Ex -
List<WebElement> Links = driver.findElements(By**.tagName**("a"));

Q17. Explain getText().
Ans. getText() - It is a "non-static" method of web element, which will "not take any argument" & returns inner text of webElement to the console in the form of string.

Ex -
List<WebElement> Links = driver.findElements(By.**getText()**);

Q18. What is the difference between findElement() & findElements() ?
Ans.

findElement()	findElements()
1. Returns the first most web element if there are multiple web elements found with the same locator 2. Throws exception NoSuchElementException if there are no elements matching the locator strategy 3. It will only find one web element	1. Returns list of web elements 2. Returns an empty list if there are no web elements matching the locator strategy 3. It will find a collection of elements whose match the locator strategy

X-PATH -

Q19. What is Xpath ?
Ans. In selenium if elements are not found by the general locators like id, class, name etc. then XPath is used to find an element on web page.

Q20. What are the types of XPath ? Explain each.
Ans. There are 2 types of XPath :

1. Absolute XPath (/)
2. Relative XPath (//)

1. Absolute XPath :

It is characteristic of XPath is that it begins with the single forward slash (/) , which means you can select the element from root tag.(To select child tag)

/html/body/tag_name/tag_name

2. Relative XPath :

For relative XPath the path starts from the middle of the HTML TREE. It starts with the double forward slash (//) , which means it can search web element anywhere in HTML TREE.

You can start from the middle of the html tree & no need to write long XPath

//tag_name[@atribute_name='atribute_value']/tag_name

SYNTAX :

Select Current Node	Select attribute	value of attribute

//tag_name[@atribute_name=’atribute_value’]

Tagname like (input,div,img etc)	attributeName

NOTE - To access the button we have to use below syntax in front of above SYNTAX

.sendKeys(Keys.ENTER); OR .click();

Q. Explain contains function [contains()].

Ans. Contains function is used to find a matching notes in html tree with the help of attribute name & attribute value or the value present in tag like inner text.

Basic xPath -
//input[@id = ‘email’]
//input[@type = ‘text’]
//span[@class=’menues’]
//*[@class=’name’]

Contains() -
//input[contains(@name,’btn’)]
//span[contains(.,’sale’)]

Q. Explain text function.

Ans. text() - It is used to write xPath using inner text.
Syntax : //TAGNAME[TEXT()=’INNERTEXT’]

Ex. - //span[text()=admin]

Q. Explain OR & AND conditions.

Ans. 1. OR - In or expression, 2 conditions are used, whether 1st condition OR 2nd condition should be true.
SYNTAX -
//tag_name[@attributeName=’attributeValue’ **or** @attributeName=’attributeValue’]

Ex - //input[@type=’submmit **or** @name=’btn’’]

2. AND - In AND expression, 2 conditions are used, both conditions should be true to find the element.If fails to find elements if any one condition false.
SYNTAX -
//tag_name[@attributeName=’attributeValue’ **and** @attributeName=’attributeValue’]

Ex - //input[@type=’submmit **and** @name=’btn’’]

Go to URBANLADDER.com
CLOSE POPUP
Click on stores
Fetch all the locations
Find out all xpath &

TRAVERSING IN HTML TREE USING AXIS -

AXIS :

AXIS	**DESCRIPTION**
1. ancestor	Select all the ancestor(parent,grandParent) of the element
2. descendant	Select all the descendant(child,grandChild) of the element
3. following	Select all the element that follow choosing tag of current element
4. following-sibling	Select all the sibling after the current-element
5. parent	Select the parent of the current element
6. preceding	Select all the element that are before current element
7. preceding-sibling	Select all sibling that are before current element

SYNTAX :

//tagName[@attriubuteName=’attributeValue’]**/axisName::**tagName[@attrubuteName=’attributeValue’]

CLASS IN SELENIUM -

1. Action Class -

In order to do action events, you need to use org.openqa.selenium.interactions Action class. The user facing API for emulating complex user gestures. Use the selenium actions class rather than using the Keyboard or Mouse directly. This API includes actions such as drag & drop, clicking multiple elements.

Syntax :

Actions **a** = new Actions(WebDriver Driver)

a = 1. moveToElement()
2. contextClick()
3. dragAndDrop()
4. dragAndDropBy()
5. doubleClick()

For mouse over action we have to moveToElement(), it will take webEements as an argument.

a.moveToElement(ele).build().perform();

Where **,** moveToElement() = non static

build= the build() method is used to compile all the listed actions into a single step.

Perform= A convenience method for performing the actions without calling build() first.

doubleClick()

It is an overloaded method present in “Actions class” which will help us to perform the doubleClick operation on the WebElement. It will ask for webElement as an argument & it will return “Actions class object “.

Q. How to perform right click action on any web element ?

Ans. We can perform right click action with the help of a “non static” overloaded method called “contextClick()”, which will take webElement as an argument & it will return “Actions class object”.

Google - java - get all text in console & click on 3rd suggestion

Drag And Drop -

To perform dragAndDrop[WebElementSource, target] we are having 2 methods called dragAndDropBy [dragAndDropBy(WebElement, x-axis, y-axis)].

Both are the non-static methods of static class which will help us to drag a WebElement till a particular index in x axis & y axis. And it will drag a WebElement from sourceWebElement to targetWebElement, both will return “actions class object”.

isDisplayed() -

It is an non static method of webElement which is used to verify the web element is present in the web page or not, it will not take any argument & it will return boolean value.

It will return true if the webelement is displayed in the webpage else it will return false.

isEnabled() -

It is an non static method of webElement which will not take any argument, it is used to verify a web element is enabled or not by returning the boolean value.

It will return true if the webElement is enabled else it will return false.

isSelected() -

It is used to check webElement is selected or not, by returning boolean value. It will work only for “input” tag (<input> </input>).

getAttribute() -

It is used to fetch the attribute value to the console in the form of string. By taking the argument attribute name in the form of string.

getCssValue() -

It will fetch the css values to the console in the form of string by taking the arguments css property in the form of string.

getLoc() -

It is used to fetch the location of webElement by returning the “point class object”, using point class object we can get the x-axis & y-axis value or position of webElement.

clear() -
It will clear the value present in input tag like textbox
Return type of clear method is “void”.

Q. What is robot class? Explain it.

Ans. Robot Class -

If we want to perform keyboard actions like opening a new tab using keyboard Or opening a new window using a keyboard, we need to take the help of “Robot Class”, which is present in “java.awt package”, We need to create the object of robot class

Robot r = new Robot();

As soon as we will create the object of robot class it will ask to throw “awt exception”. After creating the object of robot class, we need to call non static method “KeyPress & KeyRelease”

r.KeyPress(int KeyCode);
r.KeyRelease(int KeyCode);

Both methods will ask for int KeyCode as an argument.

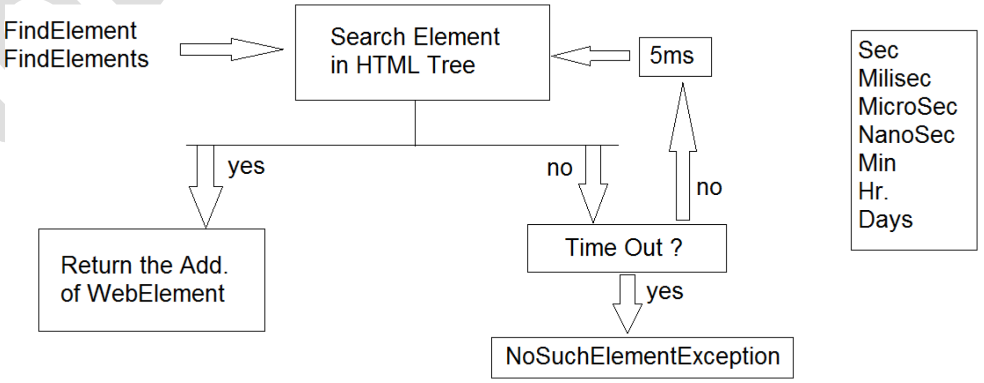
KeyEvent Class -

To pass the KeyCode we need to take the help of “KeyEvent class” which is present in “awt package” & having “static final variable” declared for each keyboard buttons.

KeyEvent.VK_T
KeyEvent.VK_CONTROL

Search java
Double click on java then copy it
Then open new tab

Q. Implicitly Wait



Implicit wait combines with findElement & findElement's, it will check the element OR search the element in "HTML Tree". If it got the element in HTML tree, than it will return the address of web element & execution will continue.If it is not able to find element in HTML tree then it will check for "timeout".

If timeout is over then it will return NoSuchElementException else it will wait for 5 millisec & search for the element in HTML tree again, this process will continue until the timeout is not over OR element got found in HTML tree.

Here timeout will be sec, millisec, nanosec, microsec, min, hr., days.

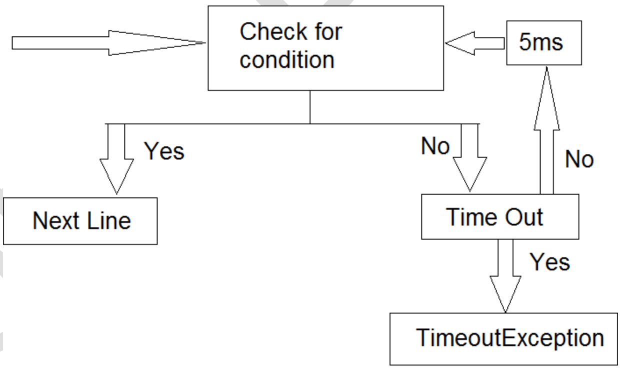
SYNTAX :

driver.manage()**.timesOut().implicitlyWait(long time, TimeUnit);**

Q. Explicit Wait

Ans. Explicit Wait will check for condition if condition is true then it will go to next line, if condition is not true then it will go & check for time out if time out is over then it will give timeout Exception, else after 500ms. It will go & check the condition again.

WebDriverWait ww = new WebDriverWait(driver,15);
ww.until(**ExpectedCondition**(ele)); //It is an interface
ww.until(**ExpectedConditions**(ele)); //It is an Class



Until is non static method which is present in WebDriverWait class which will take expected condition class as an argument.

Q. Difference between Implicitly & Explicitly.

Implicitly	Explicitly
1. It will check for findElement OR findElements.	1. It will check for conditions.
2. Address of webElement.	2. It will go to the next line.
3. Sec, millisec, microsec,hr,days etc.	3. Sec
4. It will throw NoSuchElementException	4. It will throw timeoutException

Assignment - If Un = admin in textbox then only send the password as manager (Using explicit wait)

- Open Autoit
software scit
What is control id ?
It is combination of class & instance (copy from Autoit software)
Which file to upload
Right click on file =properties=security=path(question mark should not be there)

Right click on file = click on compile script

Assignment - (Selenium: ActiTime4Aug)

1. Lunch browser
2. Load actitime
3. Pass UN & PW
4. Click on login page
5. Click on "?" button
6. Click on "about your actiTime"
7. Fetch "ActiTime2020online" to console

// write 4 aug ppt in this DOC.

Take Screenshot

5Aug
Q.How to take Screenshot?
Ans.(Only 1 abstract method is present in take screenshot functional interface.)

To take screenshot we have to cast the driver to takeScreenShot & get the use of getScreenshotAs() it will take OutputType.File, OutputType is an interface & file is static member, return type of this method is file class object.

Then we have to make object of File class & we have to pass the path of location where we want to save our screenshot.

And we have make use of the files class which is present google.common.io.file packages & we have to use static method called copy which will take two argument from file & to file to copy img from that link & paste into file.

To get the ScreenShot refresh the folder

```
TakesScreenshot ts = (TakesScreenshot) driver;
File S = ts.getScreenshotAs(OutputType.FILE);
File SS = new File("./S" + name + ".png");
Files.copy(S,SS);
```

JSExecutor :

To perform the scroll operation we have to go for JSExecutor.
Scroll operation like - ScrollUP, ScrollDown, ScrollLeft, ScrollRight.
To make use of JSExecutor we have to cast the webdriver object to JSExecutor.

WebDriver driver = new ChromeDriver();
JavascriptExecutor js = (JavascriptExecutor) driver;

In JSExecutor we have one method called executeScript() inside this we have to write the script.

ScrollDown -

In this we have to write the index upto where we want to scroll in y-axis & we have to make x-axis as 0.
Ex - js.executeScript("window.scrollBy(0,**1000**)");

ScrollUP -

If you want to go up we have to put negative axis in place of y-axis
Ex - js.executeScript("window.scrollBy(0,**-500**)");

ScrollRight -

In this we have to write the index upto where we want to scroll in x-axis & we have to make y-axis as 0
Ex - js.executeScript("window.scrollBy(**1000**,0)");

ScrollLeft -
If we want to go up we have to put -ve axis in place of x-axis
Ex - js.executeScript("window.scrollBy(**-500**,0)");

Scroll Till a WebElement -
To scroll till a WebElement we have to store the WebElement in variable & we have to pass the variable in the method
Ex - js.executeScript("arguments[0].scrollIntoView()" , ele);

Scroll Full Down -

Ex - js.executeScript("window.scrollTo(**0,document.body.scrollHeight**)");

Scroll Full UP -

Ex - js.executeScript("window.scrollTo(**document.body.scrollWidth,0**)");

Read Data from Excel File :

1. Create 1 folder in selenium & save excel file in that folder.
2. Then create an path of that excel file in class in which you want to import that excel file

1. To read data from excel file we have to make us of poi-jar's. We have to add those jar to build path. After that we have to make the object of fileInputStream & pass the path of excel file.
2. After that we have to call a static method create() which is present inside WorkBookFactory class & it will take fileInputStream object as an argument & it will return WorkBook interface object.
3. After that we have to call a non static method getSheet which will take "String Sheet name" as a argument & it will return sheet interface object.
4. In sheet interface we have one method called getRow which will take int as an argument from which row we want to read the data, it will return RowInterface object.
5. Using that RowInterface object we will call "getCell()" which will take int argument & return cellInterface object.

*To Select Particular cell Value -

```
FileInputStream fis = new FileInputStream( “loc + name + Extension” );
workBook  wb = workBookFactory.Create( fis );
Sheet sh = wb.getSheet( “SheetName” );  // whatever you want to do
Cell c1 =sh.getRow( int row ).getCell( int cell );
```

*To Select All Cell Values at once -

```
FileInputStream fis = new FileInputStream( “loc + name + Extension” );
workBook  wb = workBookFactory.Create( fis );
Sheet sh = wb.getSheet( “SheetName” );  // whatever you want to do
Cell c1 =sh.getRow( int row ).getCell( int cell );
int rCount= sh.getPhysicalNumberOfRows();
            int cCount= sh.getRow(0).getLastCellNum();
            System.out.println(rCount + " " + cCount);
            for(int  row= 0; row<rCount; row++) {
                  for(int col=0; col<cCount;col++) {
                        Cell c1 = sh.getRow(row).getCell(col);
                        System.out.println(c1);
                  }
            }
```

POM (Page Object Model)

(Here we are taking help of encapsulation)

Click here to checkout programs

1. **StaleElementReferenceException** to avoid this exception we go for POM class.
2. StaleElementReferenceException means the reference of element is old.
3. After finding the element & before performing any action on that element if page is refreshed we get StaleElementReferenceException.
4. POM class is used to test the webPages. In POM class we have **@FindBy** annotation to find WebElement.

   ```
   @FindBy( locator-name=”locator-value” )
   Private webElement ele;

   Ex- @FindBy(xpath=””)
       Private webElement ele;+
   ```

   We can go for any locators inside findBy annotation.
5. In order to initialize //write from others notebook

   ```
   pageFactorry.initElements( webDriver, POMObject );
   pageFactorry.initElements( webDriver, this );
   ```

6. In POM we have to define methods of every webElement which are performing on that webElement.

Why we use POM ?

1. To avoid StaleElementReferenceException
2. To create separate class for each page

TestNG Annotations :

@BeforeSuite - The annotated method will be run before all tests in this suite have run

@AfterSuite - The annotated method will be run after all tests in this suite have run

@BeforeTest - The annotated method will be run before any test method belonging to the classes inside the tag is run.

@AfterTest - The annotated method will be run after any test method belonging to the classes inside the tag is run.

@BeforeClass - The annotated method will be run before the first test method in the current class is invoked.

@AfterClass - The annotated method will be run after the all the test methods in the current class have been run.

@BeforeMethod - The annotated method will be run before each test method.

@AfterMethod - The annotated method will be run after each test method.

@Test - The annotated method is a part of a test case

Data Provider :

```
package TestNG;

import org.testng.Reporter;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

public class DemoDataProvider17Aug {

        @DataProvider
        public static String[][] demo(){
                String st[][] = new String [3][2];
                st[0][0]="User A";
                st[0][1]="A";

                st[1][0]="User B";
                st[1][1]="B";

                st[2][0]="User C";
                st[2][1]="C";

                return st;

        }

        @Test(dataProvider = "demo")
        public void tc(String a, String b) {

                Reporter.log(a+" "+b, true);
        }}
```

# Selenium
# TestNG
# Maven

*** Exceptions in Selenium ***

IllegalStateException : The path to the driver executable is not correct
(Check : System.SetProperty)

WebDriverException : If we directly close the browser while automation script is
Running.

NoSuchElementException : No such element: Unable to locate element
(Check : Xpath).

ElementNotInteractableException : Element not interactable.

SessionNotCreatedException : If our chrome version is updated to latest version
(Check : Browser Version)

FileNotFoundException :

StaleElementReferenceException : If id or reference of element is got old.