SELENIUM:

Software Testing:-

Software testing is nothing but verify and validate the software application and find defect to the application. In real time developer the develop the application after that sending to testing department as a tester are need verify application is properly working or not based on customer requirement.

Before release application to the market or customer we need ~~verify~~ Provide application quality and bug free.

⇒ 2 Types :-

(1) manual Testing (Already discuss)

(2) Automation Testing

(2) Automation Testing:-

→ In real time developer developer application to the testing department. If we need automation testing then we testing the application without human efforts. we write some piece of code by using tool automatic link will be open, automatically URL is enter some data enter automatically click on search button.

So i.e Automatic testing Provide quality and bug free product to the customer.

→ also compare actual result based on ~~on~~ expected result.

How can a tool perform testing?

First testers will do some coding and after that and put the code into the automation tool. And the automation tool will perform testing the application.

Selenium: Automation:

- open source
  - ↓
  - Selenium
- Licence version (LV)
  - ↓
  - QTP/UFT/RFT
  - QTP - Quick Test Perfession
  - UFT - Unified Functional Testing
  - RFT - Rational " "

Selenium :-

→ Selenium is one of the most widely used open source web UI automation testing tool.

→ It was originally developed by Jason Huggins in 2004

→ Selenium can only automate web applications

→ Selenium can't automate Desktop app^n & mobile app^n.
only skype, Paint, PDF-Reader.

→ Selenium is not a single tool it is a collection of diff components.

- Selenium IDE
- Selenium WebDriver
- Selenium RC
- Selenium Grid

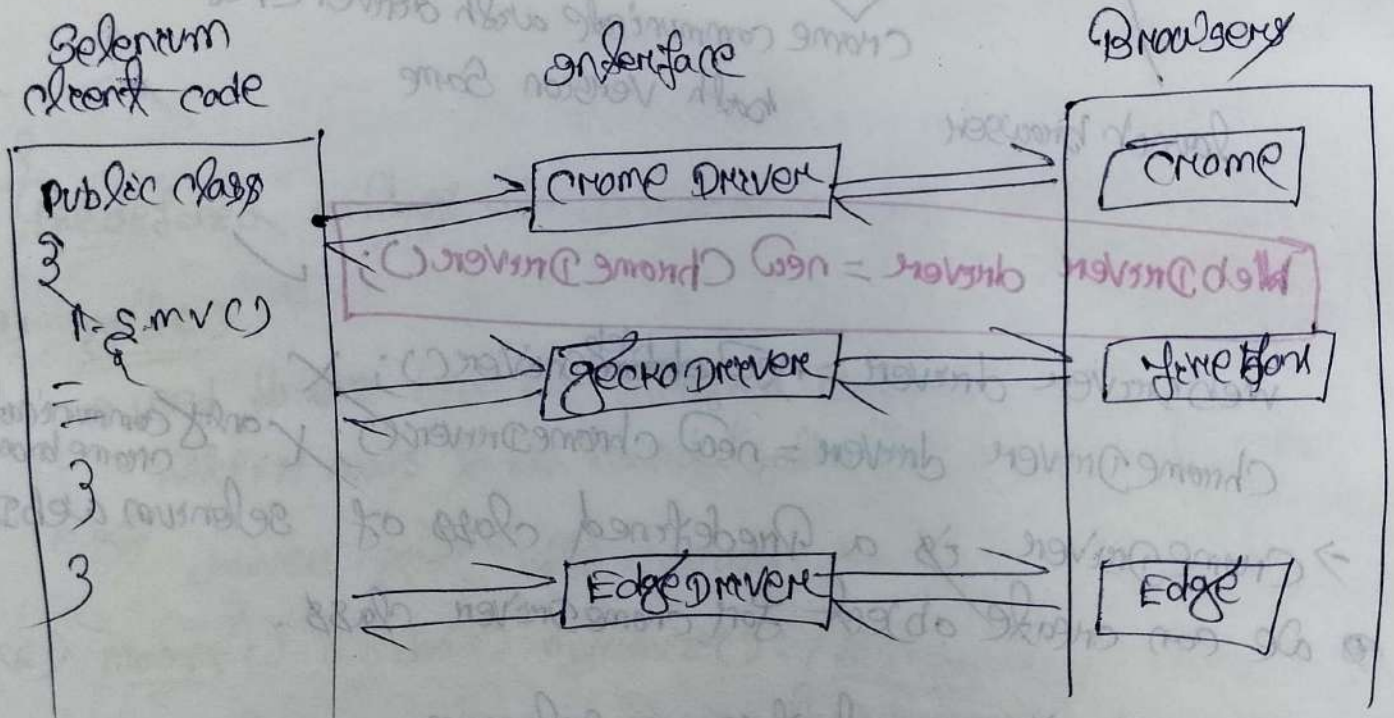
Features of Selenium :-

→ The selenium able to acess the multiple browsers.
ex: crome, firefox, opera etc.

→ The selenium able to acess the multiple operating system.
ex: windows, linux, mac etc.

→ The selenium able to accept many language.
ex: Java, Python, Ruby, Dot net, C#, PHP

→ Selenium Architecture:



Selenium web driver :-

Selenium webDriver is the main component of Selenium.

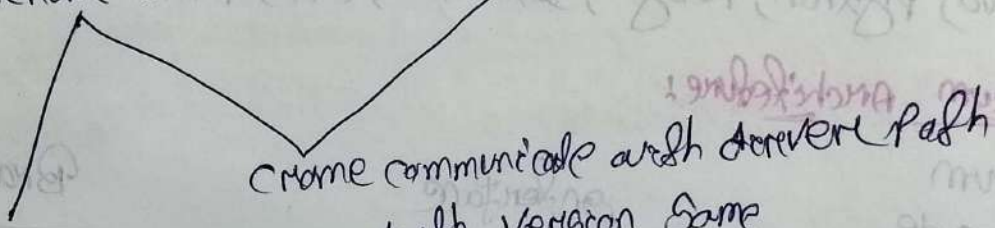
→ Selenium webDriver is an API, which contains a set of predefined library of commands / methods.

How to launch the chrome Browser?

First we need to create a class inside of class we need to create method. If we want to launch the chrome browser we need to set ~~Set.Property~~ System.SetProperty (key, value).

key: webdriver.chrome.driver

value: driverPath




```
WebDriver driver = new ChromeDriver();
```

WebDriver driver = new ~~chrome~~ WebDriver(); ✗

ChromeDriver driver = new ChromeDriver(); ✗ only communicate chrome browser

→ ChromeDriver is a Predefined class of selenium webDriver. we can create object for ChromeDriver class.

→ WebDriver is an interface in selenium.

~~web~~ ~~driver is found~~

ChromeDriver is the child class of WebDriver interface.

→ ChromeDriver() is constructor.

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

Public class Sele
{
    P.S.V.m()
    {
    System.setProperty("webdriver.chrome.driver", "chrome path");
    WebDriver driver = new ChromeDriver();
    }
}
```

Predefined methods of Selenium WebDriver:-

(1) get()
→ used to open the specified URL's web page.
http:// needs to be provided, otherwise the URL not open.
e.g. // driver.get("http://amazon.com/");

(2) manage().window().maximize() :- / fullscreen
→ used to the URL maximize to maximize.

e.g. driver.manage().window().maximize();

(3) close():
→ This is used to close current browser.
e.g. // driver.close();

(3) quit(): all browser will close.

(4) getTitle(); (String)
→ This method used to get the current browser title.
e.g. // S.o.P(driver.getTitle());

(5) getCurrentUrl():- (String)

It is used the current browser URL.

e.g// S.o.p(getCurrentUrl());

(6) getPageSourceCode:- (String)

It is used current browser source code.

e.g// S.o.p(driver.getPageSourceCode());

(7) findElement():-

It is predefined method of WebDriver & it to be used "By" class

Locators	Command:-
Id	→ driver.findElement(By.id(" "));
name	→ driver.findElement(By.name(" "));
className	→ driver.findElement(By.className(" "));
tagName	→ driver.findElement(By.tagName(" "));
linkText	→ driver.findElement(By.linkText(" "));
PartialLinkText	→ driver.findElement(By.PartialLinkText(" ");
cssSelector	→ driver.findElement(By.cssSelector(" "));
xPath	→ driver.findElement(By.xPath(" "));

(8) click():-

It is used to click operation.

e.g// Button, link, check box, radio option.

(9) sendKeys():-

It is used to enter text into the text fields.

e.g// like - text box, text area, Password etc.

(10) clear():-

It is used to clear the text available in the text box or text area.

e.g//

(11) getText() :- (String)

gt is used to retrieve the elements text.

i.e // The text betn starting and ending tags of HTML elements.

Locators :-

Locators help to the selenium in finding the UI elements on web Pages.

Selenium → Locators → UI element

so we can use these locators in our test automation code to locate the UI elements on web pages.

→ The main reason of using locators.
selenium is blind by default.
selenium doesn't have the ability to find/locate UI elements on web Page.

Different types of Locators :-

There are different types of locators which can used for locating the UI elements on web page.

id :→ To find the element by ID of the element.
name :→ To find the element by name of the element.
className → // by className of the element.
linkText → To find the element by text of the link.
cssSelectors → css path also locates elements having no name, class or ID,
xPath :→ x Path required for finding the dynamic elements & dom traverse between various elements of the web page.

Shot on OnePlus
Amita😍

8) How to put data on web application (search box)?

```
Public class Sandy
{
    P.S.V.M(){
        Scanner sc = new Scanner(System.in);
        S.o.p("Enter any url");
        String url = sc.nextLine();
        System.setProperty("webdriver.chrome.driver","path");
        WebDriver driver = new ChromeDriver();
        driver.get(url);
        driver.manage().window().maximize();
        driver.findElement(By.id("seachid")).sendKeys("iphone11");
        driver.findElement(By.id("seachbuttonid")).click();
    }
}
```

9) How to validate the different types of app url without making change.

By use Scanner class

Previous Program

* getAttribute() : (String)

→ It is used to return the value stored in specified value.

eg driver.findElement(By.id(" ")).getAttribute(" ");

getText()
Return the text

getAttribute()
Return attribute name

* isDisplayed() : (boolean)

→ It is used to find out whether the element displayed or not.

eg boolean a = driver.findElement(By.id("but2")).isDisplayed();

also isEnabled() :-

* isSelected() : (boolean) :-

→ It is used to find out whether radio box/checkbox are selected or

eg boolean a = driver.findElement(By.id("box2")).isSelected();

* navigate() :

→ It is used to navigate().
- to()
- back()
- forward()
- refresh()

e.g. driver.navigate().to("need url");
driver.navigate().back();
driver.navigate().forward();
driver.navigate().refresh();

* submit() :

→ It is only use form. e.g.

* How to launch the browser multiple times?

```
class A
{
public void launch()
{
Scanner a = new Scanner(System.in);
S.o.P("Enter any url");
String url = a.nextLine();
for(int i=1; i<=5; i++)
{
System.setProperty("webdriver.chrome.driver", "Path");
WebDriver driver = new ChromeDriver();
driver.get("url");
driver.manage().window().maximize();
driver.close();
}
}
P.S.V m()
{
A a = new A();
a.launch();
}
}
```

* How to launch the firefox browser? (gecko)

```
class A {
    P.S.V.m()
    {
        System.setProperty("webdriver.gecko.driver", "Path");
        WebDriver driver = new FirefoxDriver();
        driver.get("https://amazon.in");
        driver.manage().window().maximize();
    }
}
```

* How to launch the edge browser?

```
class A
{
    P.S.V.m()
    {
        System.setProperty("webdriver.edge.driver", "Path");
        WebDriver driver = new EdgeDriver();
        driver.get("https://amazon.in");
        driver.manage().window().maximize();
    }
}
```

How to login Salesforce application?

```
class login
{
    public void Log()
    {
        System.setProperty("webdriver.chrome.driver","Path");
        WebDriver driver = new ChromeDriver();
        driver.get("https://salesforce/login/");
        driver.manage().window().maximize();
        driver.findElement(By.id("username")).sendKeys("Sarea85");
        driver.findElement(By.id("password")).sendKeys("Sarea@123");
        driver.findElement(By.id("login")).click();
        driver.close();
    }
P.S.V.M()
    {
        login n = new login();
        n.log();
    }
}
```

* How do do the login Negative Testing ?

```
class A
{
  WebDriver driver;
  Public void login()
  {
    System.setProperty("webdriver.chrome.driver", "Path");
    WebDriver driver = new ChromeDriver();
    driver.get("https://salesforce/login");
    driver.manage().window().maximize();
  }
Public void multi(String username, String Password)
  {
    driver.findElement(By.id("username")).sendKeys(Username);
    driver.findElement(By.id("Password")).sendKeys(Password);
    driver.findElement(By.id("login")).click();
  }
P.S.V.M()
  {
    A a = new A();
    a.login();
    a.multi("sarvas5", "sarvas@123");
    a.multi("aadesh", "sarvas123");
    a.multi("a...", "....");
  }
```

How to take Screenshot.

```
Public class Screens
{
    WebDriver driver;
Public void launch()
{
    System.setProperty("webdriver.chrome.driver", "Path");
    WebDriver driver = new ChromeDriver();
    driver.get("url");
    driver.manage().window().maximize();
driver.findElement(By.id("username")).sendKeys("name");
driver.findElement(By.id("Pass...")).sendKeys("Password");
driver.findElement(By.id("login")).click();
}
Public void screenshot() throws Exception
{
    File src = ((TakesScreenshot)driver).getScreenshotAs
                                    (OutputType.FILE);
    File des = File("Path");
    FileHandler.copy(src, des);
}
P.S.Vm()
{
Screens s = new Screens();
    s.launch();
    s.screenshot();
```

How to launch an app url in multiple browser.

```
class Browser
{
    WebDriver driver;
    public void launch()
    {
        Scanner s = new Scanner(System.in);
        S.O.P("Enter any Browser");
        String url = s.nextLine();
        if(url.equalsIgnoreCase("chrome"))
        {
            System.setProperty("webdriver.chrome.driver","Path");
            driver = new ChromeDriver();
        }
        else if(url.equalsIgnoreCase("edge"))
        {
            System.setProperty("webdriver.edge.driver","Path");
            driver = new ChromeDriver();
        }
        else
            S.O.P("Not found any browser");
        }
        driver.get("https://amazon.in/");
        driver.manage().window().maximize()
    }
```

```
P.S.V.M()
{
  Browser b = new Browser();
  b.launch();
}
```

How to do the Parallel Execution?

```
Public class SaMV@
{
Public void launch()
{
  String [] Broo = {"chrome", "edge"};
for(Browser : Brow)
{
  if(Browser.equals ignoreCase("chrome"))
  {
    System.setProperty("webdriver.chrome.driver", Path);
    WebDriver driver = new ChromeDriver();
  }
else if(Browser.equal("edge"))
  {
    "                "
    else { S.O.P("Not Found"); driver.get("url")
                               driver.maximize()
P.S.V.M()
{
  SaMV re = new SaMV()
  re.launch();
}
```

Uses of Navigate() :-

```
Public class Navi
{

    Public void launch()
    {
      System.setProperty("webdriver.chrome.driver", "Path");
      WebDriver driver = new ChromeDriver();
      driver.get("https://google.com/");
      driver.manage().window().maximize();
      Thread.sleep();
      driver.navigate().to("https://amazon.in");
      driver.navigate().back();
      driver.navigate().forward();
      driver.navigate().refresh();
      driver.close();
    }

    public static void main(String[] args)
    {
      Navi n = new Navi();
      n.launch();
    }
}
```

Text - Black

Handling Multiple windows in Selenium :-

Selenium only locate one window. Sometime our browser different window will open new window. So if we handle main window and all child window, Selenium provide diff methods.

Syntax: → So every window have diff ID.

→ get.windowHandle() :- This method will handle current window.

→ get.windowHandles() :- This method will handle the all window.

→ Set: This method get the window handles in the form of a String

Set<String> set = driver.get.windowHandles();

→ Switch to : This method helps to switch between the windows.

→ action :- This method helps to perform certain actions on the window

e.g

```
Public class Sam
{
  Public static void main(String[] args)
  {
    System.setProperty("webdriver.chrome.driver", "Path");
    WebDriver driver = new ChromeDriver();
    driver.get("amazon.blogspot.com");
    driver.findElement(By.linkText("open a PopUp")).click();
    Set<String> window = driver.getWindowHandles();
    Iterator<String> ie = window.iterator();
    String main = ie.next();
    String child = ie.next();
```

driver.switchTo().window(child);

String s = driver.findElement(By.id("Bread")).getText();

s.o.p(s);

driver.close(); // child window close

driver.switchTo().window(main);

driver.close(); // main window close.

}

How to Handle the DropDown ?

```
Public class Sony
{
    WebDriver driver;
    public void launch()
    {
        System.setProperty("webdriver.chrome.driver", "Path");
        driver = new ChromeDriver();
    driver.get("Salesforce//login/");
    driver.manage().window().maximize();
    driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
}
public void Drop()
{
    Select se = new Select(driver.findElement(By.name("Company")));
    se.selectByVisibleText("21-200 Employees");
    se.deselectByVisibleText("  ");
}
```

```
public static void main(String[] args)
{
    Sanu n = new Sanu();
    n.launch();
    n.Drop();
}
```

→ If we want to handle the dropdown on webpage first we need to findout ie dropdown or not.
→ first the dropdown we need to inspect then check the dropdown related tag name, class, select or not.
If tag name select i.e dropdown or not select tag so that is not dropdown.
→ If we handle dropdown selenium provide diff types of methods first we need locators from the dom after the write commands.

```
Select obj = new Select(driver.findElement(By.name("  ")));
```

→ If we want select the created element from down we have diff types of methods.

```
// obj.selectByVisibleText(" ");
// obj.deselectByVisibleText(" ");

// obj.selectByIndex( );
// obj.deselectByIndex( );
// obj.selectByValue( );
// obj.deselectByValue( );
// obj.isMultiple( );
// obj.deselectAll( );
```

How to handle the alerts ?

If we want handle any alert we have diff types of methods are there.

→ driver.switchTo().alert().accept();

one display alert we accept means click on button.

→ driver.switchTo().alert().dismiss();

dismiss means cancel button.

→ driver.switchTo().alert().sendKeys("");

on this alert box we are unable to insert. because the developer develop the alert by using javascript executor.. if we want send data manual data is visible but by using automation tool is not visible but we can send data.

→ driver.switchTo().alert().getText();

if we want to get any texts in alert box we need to use this method.

e.g.

```
Public class Baru
{
    P.S VM()
    {
        System.setProperty("webdriver.chrome.driver", "Path");
        WebDriver driver = new ChromeDriver();
        driver.get("https://omayo.blogspot.com/");
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
```

driver.findElement(By.cd("alert1")).click();
driver.switchTO().alert().accept();

driver.findElement(By.cd("alert1")).click();
driver.switchTO().alert().dismiss();

driver.findElement(By.cd("alert1")).click();
driver.switchTO().alert().sendKeys("sona");
driver.switchTO().alert().accept();

driver.findElement(By.cd("alert1").click();
driver.switchTO().alert().getText();

XPath Expressions:-

out of all the locators, xPath Expressions are the powerful locators and can be able to locate any type of UI element.

→ 2 Types of xPath Expressions:-

(1) Absolute xPath:
(2) Relative xPath:

xPath = //tagname[@Attribute='value']

(1) Absolute xPath:-

→ of containing the complete path from the root element to the desire element.

→ The main disadvantages of absolute path so there is any changes made on the path of element then that xPath gets failed.

→ The absolute xPath starts with single forward slash (/)

e.g. /html/body/div[2]/div[1]/div[4]/...

Relative xpath:-

Relative xpath tries to locate the element directly instead of locating from root.

→ It can search the elements anywhere on the webpage, means no need to create a long xpath and we can search from the middle of Html dom structure.

→ The Relative xpath starts with double slash (//).

//tagname[@Attribute='value']	tag==p

eg: //input[@id='username']

: orginally use.

Id → //input[@id='username']

name → //input[@name='username']

//input[@id='username' and @name='username']

or

//*[@id='password']

text → //a[text()='mobiles']

type → (//input[@type='text'])[2]

and → //input[@id='email' and @name='email']

or → //input[@id='username' or @name='username']

following → //label[text()='last name'] / following::input

Preceding → //table/tbody/tr[2]/td[2][text()='mohan anders'] / preceding::td[1]

[@name='name']

//P[@id='Para1'] → Locates the Paragraph having the id attribute as Para1

//P[@class='main'] → locates the // having class attribute as main.

//P[@id='Para1'][@class='main'] → locates all the Plags having id as 'Para1' and class as 'main'.

//input[@name='username'] → locates the flags having name attribute.

//input[@value='orange'] → locates the flags having name & value attribute.

//input[@checked] → locate the input flags having checked attribute.

//img → locate the image flag.

//select[@class='combobox'] | //a[@value='link2'] → locate the dropdown field having class & hyperlink value.

//a[@id='link1' or @value='link2'] → locate the hyperlink having id & value.

//a[@href='https://www.selenium143.com'] → locate the hyperlink on web page.

//a[@href='https://www.selenium143'][1]
[2]
//.

Functions

Text() :-
find the flags having the exact text in black colour.

e.g// //input[text()='Para1']

Containing() :-
It is used when the value of any attribute changes dynamically.
Then we use Partial Text.
eg: id='para123'
//input[contains(@id,'para')]

and also use

//p[contains(text(), 'Some is')]

Some as starts-with(): diff env/id Some)123

contains(): env/id=123 Some 567

Path Axes:-

are used to locate an element which doesn't have id/name, class etc. with help of path axes we can locate such element.

following:-

It select the everything in the document after the closing of the current node.

after head tag find body tags.

//head/following::body

<head>
child
</head>
following

Preceding

select the nodes that appear before the current node in the document.

before body find head.

//body/preceding::head

following-sibling:

select the nodes after current node.

after body find div

//body/div[1]/following-sibling::div

Preceding-sibling:

Select the siblings before the current node.

before body find div

//body/div[4]/preceding-sibling::div

Parent
Select the Parent ~~node~~ current node.

child:
Select the children of current node
//html/child::head

ancestor:
Select the ancestor (Parent, grandParent) of the current node.
//title/ancestor::html

descendant:
Select the descendant (children, grandchildren, etc) of the current node
//html/descendant::title

Css Selector :-

→ css selectors is use to locating UI elements, it needs to be used as a friendly over xPath expressions.

Why css selectors needs to be preferred over xPath Expressions

→ when compared to xPath expressions, css selectors locate ~~to be~~ UI elements faster.
→ Selenium may not be able to locate few UI elements using xPath expressions, while executing the automation scripts.

Css selectors two types:-
1. Absolute Css Selectors
2. Relative Css Selectors

(1) Absolute Css Selectors:
Absolute Css Selectors syntax to locates the elements from root that means complete Path.

→ html > body > p[id = 'para1'] // Normal

html > body > p#para1 // css selectore of id

→ html > body > p[class = 'sub'] // Normal

html > body > p.sub // CSS selectors.

Xpath	CSS Selectore
/	(>) # (id)
//	(space) . (class)
@	

Relative CSS Selectors :-

→ Relative css selectors locates the elements directly instead of locating from root.

eg: p[id = 'para1'] are p#para1

p[class = 'sub'] are p.sub

→ input[value = 'blue'] or [value = 'blue'] = locate input tag having value = 'blue'.

→
- * → contains
- $ → ends with
- ^ → starts with
- # → id
- . → class
- + → following sibling
- , → or operator

→ html > body > p[id = 'para1'] // Normal

html > body > p#para1 // Css selectore of id

→ html > body > p[class = 'sub'] // Normal

html > body > p.sub // Css selectors.

Xpath	Css selectore
/	(>) # (id)
//	(place) . (class)
@	

Relative Css selectors :-

→ Relative css selectors locates the elements directly instead of locating from root.

eg: p[id = 'para1'] are p#para1

p[class = 'sub'] are p.sub

→ input[value = 'blue'] or [value = 'blue'] = locate input tag having value = 'blue'.

→
- * → contains
- $ → ends with
- ^ → starts with
- # → id
- . → class
- + → following sibling
- , → or operator

→ html > body > p[id = 'para1'] // Normal

html > body > p#para1 // Css selectore of id

→ html > body > p[class = 'sub'] // Normal

html > body > p.sub // Css selectors.

Xpath

/

//

@

Css Selectore

(>) # (id)

(space) . (class)

Relative Css Selectors :-

→ Relative css selectors locates the elements directly instead of locating from root.

eg: p[id = 'para1'] or p#para1

p[class = 'sub'] or p.sub

→ input[value = 'blue'] or [value = 'blue'] = locate input tag having value = 'blue'.

→

* → contains

$ → ends with

^ → starts with

→ id

. → class

+ → following sibling

, → or operator

Handling Frames :-

→ Frame is a web page which embedded on another web page, and is used to display multiple pages inside a single web page.

→ developers can also embedded a document to be scrolled inside a frame

→ first we need to inspect and check on HTML page <I.frame> is tag used i.e frame on page.

first we need to switch frame.

```
WebElement iframe2 = driver.findElement(By.id("iframe2"));
driver.switchTo().frame(iframe2);
```

Then perform frame ~~any~~ page

again switch to main page :-

```
driver.switchTo().defaultContent();
```

Then perform main page.

Finding the number of frames available on the page.

```
S.o.P(driver.findElements(By.id("iframe")).size());
```

Handling Light box :-

If we want to handle light box not need to switch perform

e.g,

```
driver.findElement(By.id("lightbox")).click();
// no need to switch light box.
driver.findElement(By.cssSelector(".close.curser")).click();
```

Action class in Selenium:-

→ Action is a Predefined class of Selenium WebDriver.
→ Actions class contain various Predefined methods which can simulate mouse and keyboard events.

(1) mouse Action (moveToElement(), perform())

If we want to handle any mouse actions elements we need to use actions the action will provide moveToElement() method.

Syntax:-

```
Actions ob) = new Actions(driver);
WebElement src = driver.findElement(By.xPath("//a[text()"
ob).moveToElement(a).perform();
WebElement b = driver.findElement(By.xPath("//a[text()']"));
ob).moveToElement(b).click().perform();
```

(2) dragAndDrop() : (mouse action)

First we need valuable draganddrop inside of frame or not. If the draganddrop inside of frame we need switch frame.

```
driver.switchTo().frame();
Actions act = new Actions(driver);
WebElement a = driver.findElement(By.id("draggable"));
WebElement b = driver.findElement(By.id("droppable"));
act.dragAndDrop(a,b).perform();
```

(3) Right click (contextClick()):-

```
Actions ob) = new Actions(driver);
WebElement a = driver.findElement(By.xPath("//a[text()='cut']"));
ob).contextClick(a).Perform();
```

(4) Double Click():-

```
Actions ob) = new Actions(driver);
WebElement a = driver.findElement(By.name("q")).sendKeys("java");
ob).doubleClick(a).Perform();
```

(5) KeyDown() and KeyUP():- (action class) (sendKeys)

```
Actions ob) = new Actions(driver);
WebElement a = driver.findElement(By.linkText("compendiuedev"));
ob).keyDown(Keys.Control).click(a).keyUp(Keys.Control).Perform();
```

Keys class

ESCape:- If we want to handle any window like PUPUP we need to apply ESC key from keyboard.

```
Action act = new Actions(driver);
act.sendKeys(Keys.ESCAPE).Perform();
```

Enter:- once we want to enter any data we need to click on enter

```
act.sendKeys(search, "seleniumwithjava").Perform();
act.sendKeys(Keys.ENTER).Perform();
```

* Handle Auto suggestion in Search box Selenium:-

```
Actions act = new Actions(driver);
WebElement a = driver.findElement(By.id("tsf-search"));
act.sendKeys(a, "seleniumac").perform();
for(int i=0; i<=4; i++)
{
    act.sendKeys(Keys.DOWN).perform();
}
act.sendKeys(Keys.ENTER).perform();
```

Waits in Selenium:-

→ Wait commands in Selenium direct test script to pause for certain time before throwing "ElementNotVisibleException".

→ 3 Types waits

(a) implicit wait
(b) Explicit wait
(c) Fluent wait

Implicit Wait

→ implicitly wait tells the webdriver to wait for some time before throwing a "No such element Exception"

→ implicit wait is global, it is applicable to all the webelements in the scripts.

Explicit wait

→ explicit wait tell the webdriver to halt the execution until a particular condition is met or the maximum time has passed

→ explicit wait time is applicable only to those webelements which are identified by the user.

Fluent wait:

Fluent wait tells the webdriver many wait time after searching for elements before throwing exception.

→ Same as explicit wait

JavaScript Executor :-

How to Send data / login / Enter text by using JavaScriptExecutor.

```
JavascriptExecutor var = (JavascriptExecutor)driver;
WebElement a = driver.findElement(By.id("javaobsearch"));
var.executeScript("arguments[0].value='Selenium with Java';", a);
Thread.sleep(2,000);
WebElement b = driver.findElement(By.id("nav-search"));
var.executeScript("arguments[0].click();", b);
```

MAVEN :-

→ maven is nothing but Powerful management tool that is based on Pom (Project object model).
→ It is used for Projects build, dependency and documentation.

Archetype:-

maven-archetype-quickstart

Pom.xml

It is Project object model.
The Pom.xml file contains information of Project & configuration information for the maven to build the Project.
Such as dependencies, build directory, source directory.
and also add jar file.

what is Repository?

A repository is a directory or Place where all the Jars and Pom.xml file are stored.
There are 3 types:-

1. local Repository
2. central Repository
3. Remote Repository

1. **local Repository:-**
maven repository is created by maven in your local system when you run any maven command.

2. **Central repository:-**
maven community creates maven central repository on web.

3. Remote

How to get the dependency :-

every dependency is available maven repository.
we need to get the repository code related from maven repository.

WebDriverManager :-

Provide the dependency for webdrivermanager from mvnrepository.com in pom.xml file.
→ if we not provide system.setProperty("path", "path")
then we need provide webdrivermanager.

TESTNG

→ Testng is a testing framework inspired from Junit & Nunit but it introducing some new functionalities that make it more powerful and easier to use.
→ The testng able to support parallel execution and grouping execution.
→ on the testng we no need use main method.
we need to use annotations

Testng Annotation :-

→ Testng annotation is a piece of code which inserted inside a program or business logic used to control the flow of execution of test methods.
of the testing code we need to provide @ method one @Test then it execute only one. if we execute testing code every method before we use @Test mandatory.

List of Testng Annotations :-

@Test :-

The purpose of this annotation is to represent the methods inside Java class as Tests.

→ This annotations replace main method().

→ If we provide every method as @Test then we are able to get the output alphabetical orders.

→ We can use priority attribute of @Test.

e.g// @Test(priority=1)

→ If we want skip exception

then throw new SkipException();

e.g:→ SkipException is the predefined class of Testng.

@BeforeSuite :-

If we provide @BeforeSuite annotation method will run before the execution of all the test methods.

@AfterSuite :-

If we provide @AfterSuite annotation method will run after all the execution of test methods.

e.g// 4 methods is there if we provide @AfterSuite then it will execute after all the test methods.

same :- BeforeTest / one time before
AfterTest. / one time after

@ Beforemethod :-

If we Provide @Beforemethod annotation method will be execute before every each test method will run.

ex:- 4 test method is there we Provide one method @Beforemethod then it will be execute every test method before.

@ Aftermethod :-

If we Provide @Aftermethod annotation method will be execute after each test method will run.

ex:-

If we Provide @Beforemethod & @Aftermethod annotation will be execute out of 5 method every method before method and every method after.

@ Beforeclass :- (it will be execute before the class execution)

If we Provide the @Beforeclass execute of Testng class it will be before the first method of current class

@ Afterclass :- (- " "

If we Provide the @Afterclass execute of testng class it will be execute after all test method of current class

Assert in Testng :-

```
Assert.assertEquals(6,6); // TRUE
Assert.assertNotEquals("Hye","Byy"); // TRUE
Assert.assertTrue(6>3); // TRUE
Assert.assertFalse(3>5); // TRUE
Assert.fail();
```

Groups in Testng :-

Testng groups allow you to perform groupings of different test methods.

Grouping of test methods is required when you want to access the test methods of different classes.

→ Different is here

(1) Group level
(2) include level
(3) Exclude level
(4) Define level

ex// @Test (groups = {"sanity", "usability"})

How to Execute group level ?

If we want to execute group level we need to create one xml file in that xml we need to provide below commands.

```
<groups>
  <run>
    <include name = "releasing" />
    <include name = "sanity" />
  <run>
<groups>
```

include & exclude :-

→ In this include what ever we provide that type of testng will execute.

→ In this exclude what ever we provide that type of testng will not execute.

How to write the code with real time by using TestNG?

If we want to write we need create Package inside Package we need diff^n class.

→ main method
→ common method
→ Screens.

Parameters in TestNG:- (cross browsing Testing)

If we want to execute any app^n without making change the code in diff browsers. we need to use cross browsing testing.

@Parameters ({"url"})
@Test
Public void Test1 (String url)
{
s.o.p ("firstone");
web driver;
:
:
}
@Parameters
@Test
Public void Test2 (String url2)
{
//
}

.xml

<Parameter name = "url" value = "https://www.amazon"/>

Data Provider :-

```
Public class Datac
{
@Test(dataProvider = "Logindata")
Public void DataProvider1 (String user, String Password)
  {
  WebDriverManager.chromedriver().setup();
  WebDriver driver = new ChromeDriver();
  driver.get("https://login.salesforce.com/");
  driver.manage().window().maximize();
  driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
driver.findElement(By.id("username")).sendKeys(user);
driver.findElement(By.id("password")).sendKeys(password);
driver.findElement(By.id("Login")).click();
}
```

(2) DataProvider

```
Public object[][] logindata()
{
  object[][] data = new object[2][2];
    data[0][0] = "sonu kumar";
    data[0][1] = "sonu123";
    data[1][0] = "sonukumar";
    data[1][1] = "kumar123";
    return data;
}
```

Data Driven Testing :-

after

2010 before every year

XSSFWorkbook → [Excel book] ← HSSFWorkbook

XSSFSheet → [Sheet] ← HSSFSheet

XSSFRow → [Row] ← HSSFRow

XSSFCell → [cell] ← HSSFCell

→ data driven testing is a software testing method in which test data is stored in table or excel format.

→ data driven testing allows testing to input a single test script that can execute tests for all test data from a table and expect the output in the same table.

→ It is also called Parameterized testing.

How to read the data from excel ?

```
Public class data
{
  Public static void main (String[] args)
{
FileInputStream file = new FileInputStream("location");
XSSFWorkbook wb = new XSSFWorkbook(file);
XSSFSheet Sheet = wb.getSheet("Sheet1");
int no = sheet.getLastRowNum();
S.o.P("the no. of column"+no);
```

```
int cell = sheet.getRow(0).getLastCellNum();
S.o.P("the no. of cell is: "+cell);
for(int i=0; i<row; i++)
{
  XSSFRow curr = sheet.getRow(i);
  for(int j=0; j<cell; j++)
  {
    String data = curr.getCell(j).toString();
    S.o.P("  " +data);
  }
  S.o.P();
}
```

How to write the data from excel (update):-

```
class write
{
Public void update(String data)
{
FileInputStream file = new FileInputStream("location");
XSSFWorkbook wb = new XSSFWorkbook(file);
XSSFSheet sheet = wb.getSheet("sheet1");
sheet.getRow(0).getCell(1).setCellValue(data);
FileOutputStream fot = new FileOutputStream("location");
wb.write(fot);
wb.close();
```

```
Public class update
{
    @Test
    Public void fileupdate()
    {
        Scanner sc = new Scanner(System.in);
        S.o.p("enter data");
        String name = sc.next();
        Oracle ob = new Oracle();
        ob.update(name);
    }
}
```

cucumber.java
cucumber.xml
Selenium with Java
cucumber core
gherkin

Cucumber:-

→ Cucumber is a behavior driven development tool.
→ It is used to develop test cases for the behavior of software functionality.
→ Provides an easily understandable testing script for system and automation testing.
→ It will provide diff^t terminologies. Bdd, feature file, Scenario, step definition, Tag, etc of cucumber testing.

BDD (Behavioral Driven Development):-

→ BDD is a software development approach that was developed from Test Driven Development (TDD).
→ BDD provides are to write the scenario in plain language.

How does Cucumber Testing works?

Cucumber test cases are written with the code development of software.

These test cases are called step definition language.

→ First, cucumber tools reads the step definition or a Gherkin or Plain language text inside the feature file.

→ Now, it searches for the exact match of each step in the step definition file. When it finds the match, then executes the test case and provides the result as passed.

Advantages of Cucumber Tool:

→ The main focus of the cucumber testing is on the end-user experience.

→ The test case writing is very easy and understandable.

→ It provides end-to-end testing framework.

Feature File:

In the feature file we need to provide feature, scenarios and feature description to be tested.

→ We need to provide high level description with Gherkin language.

→ Gherkin is plain english, every one able to understand

→ Some file:

- → Background
- → Scenario
- → Scenario outline

Scenario:-

The Scenario we are able to execute only one time.

Scenario outline:-

It helps to execute a scenario multiple times, it will be execute set of data.

Step_Definition:

Here we need to map exact feature file information inside of the step definition file.

Test Runner:-

If we want to execute the cucumber code we are always execute test runner file.

→ In test runner file we need provide cucumber.class

@CucumberOptions inside of the cucumber options we need to provide different types of methods (options).

@RunWith(Cucumber.class)

↓

@CucumberOption(features = "feature file location",
glue = {"step_definition"})

→ features:- we need to provide the feature file location

glue = we need to provide step definition package name.

dryRun:- They will be validate the user able to map feature file on step definition pass or not.

monochrome:- It is able to print output inside of console readable format.

Strict:

If we want to execute some steps only

e.g out of 5 only 3 execute.

Then we need to Provide strict = True;

Then only we are able to execute 3 step. we are unable to get the error remaining 2 steps.

How to execute types of Testing in cucumber:

we need to create diff^n types of scenario under feature.

e.g// Feature:

Scenario:

@Regression

given:

Then:

⇒ If we want to execute any type of testing we need to create below command inside of the testrunner class under cucumber option.

tags = {"@regression"}

⇒ If we want to execute diff^n types of testing we need to create below command inside of cucumber option.

tags = {"@smoke, @Regression"}

How to do the examples in cucumber:-

If we want to provide any examples under feature file we need to create scenario outline.

Feature:
Scenario outline:
Given: Launch the application
when: user enters <username> and <password>
Then: click on login

Examples:

username	password
Sonu@123	sonu@123

→ we need to provide inside of step definition file.

@when ("user enters (.*) and (.*)")

Explain of Gherkin language:-

Given: we need to create step. set of content.

when:→ This step use to perform the action and information and the system.

Then:- This step used to perform representing outcome.

And:- This step used to extend the previous step.

API Testing:

→ API testing is type of ~~testing~~ software testing where application Programming interfaces (APIs) are tested to determine if they meet expectations for functionality, reliability, performance and security of applications.

→ multiple API system can performed API testing. an API testing our primary focus is on business logic layer of the software architecture.

→ The API is the two types.

i. Soap (Simple object access Protocal)
ii. Rest (Representational State Transfer)

Soap:-

→ They is completely old version.
→ They is not a open source completely Paid version.
→ Now a days most of people are prefer rest.

Difference between Web Services and API:-

Postman:

Postman is a collaboration platform for API development. It is a recognized API client to create, design, testing and document APIs.

We can send HTTP requests to mock and receive their response using Postman.

Benefits of using

Postman is a powerful tool for API testing that offers many benefits.

→ Postman provides a user friendly interface for creating and sending requests to APIs.

→ We can quickly test various scenarios by sending different types of requests (GET, POST, PUT, DELETE)

SQL → Structured query language

SQL is a non-procedural programming language developed by IBM in the 1970s and then taken by ANSI.

SQL is used by almost all relational databases to write queries, access, edit, and retrieve data.

SQL

SQL stands for structured query language

It is a used to interact with database, i.e

to create a database, to create table in the database, to retrieve data or update a table in database etc. We can create a database, can create table, delete a table etc.

maven

maven is a nothing but Powerful management tool that is based on pom (Project object model).

→ It is use to Project build, dependency and documentation.

maven-archetype-quickstart

(1) What is Selenium IDE?

→ Selenium IDE is implemented as firefox extension which provides record & playback function on testcases

(2) What is Selenium Commands?

Selenium - 3 Types:
1- Action → It is used for performing the operations and interaction with the target elements
2. Assertions, 3. Accessors

3. What do you mean by the assertion in Selenium?

→ The assertion is used a verification point. It is used to store values in a variable.
→ It verifies the application conforms to what is expected
→ The types of assertion are "assert", "verify" and "waitfor".

4. Diff' bet' assert and verify commands?

→ Assert command check if the given condition is true or false. If the condition is true, the testing is execute next phase. and if the condition is false, execute are stop and nothing will be executed.

→ Verify command check if the given condition is true or false of condition is false it not stop the execution, all test phases should be executed.

(6) Selenium Grid :

→ It allows Parallel execution of tests on different browser and operating systems by distributing commands to other machines simultaneously.

(7) Selenium RC (Remote control) :-

→ It is a server that enables users to generate test scripts in preferred Programming language.

(8) Selenium web driver :

→ It is a Programming interface that helps create and run test cases by directly communicating with web browser