Course Name : Automation testing with Selenium Tool

Why you should learn Automation Testing ?

- Get Instant ROI and cost effective
- Time Saving and Obtain Faster Resuls
- Upskill yourself To be with the trend
- Meet Strict Deadlines and get more happy clients.
- Perform tasks impossible with manual testing
- Avoid Repetitive tasks and utilize time productively.

What you will learn by the end of this course ?

- By the end of this course you can automate any Web Appication
- You will be proficient in identifying the elemnts on the webpage and performing various actons on the elements
- You will be able to write test scripts for various testing types with verification
- You will be familiar with the frameworks applied in the industry
- You will be able to write efficient codes applying Code reusability
- You will be able to integrate various Third party tools with Selenium Tool

Description: Automation testing on a software is the best way to increase the effectiveness, efficiency and coverage of your software testing Once automated tests are created they can easily be repeated.

This course helps you gain knowledge theoretically and practically that can be applied on real time Softwares. In this course, we use Seleniu WebDriver as an Automation testing tool.

Automation Testing

- What is Automation testing ?
- When we switch to Automation testing ?
- Why Automation testing (Advantages) ?
- Disadvantages
- Automation Testing Tools

Selenium

- What is Selenium ?
- Why Selenium (Advantages) ?
- What are its versions ?
- What all OS, Browsers and Programming Languages it support ?

- Java-Selenium Architecture
- WebDriver Architecture
- Basic Selenium Program to open and close browser
- Rntime Polymorphism Program in Selenium
- WebDriver abstract methods
- Locatos
- Xpath, its types and cases

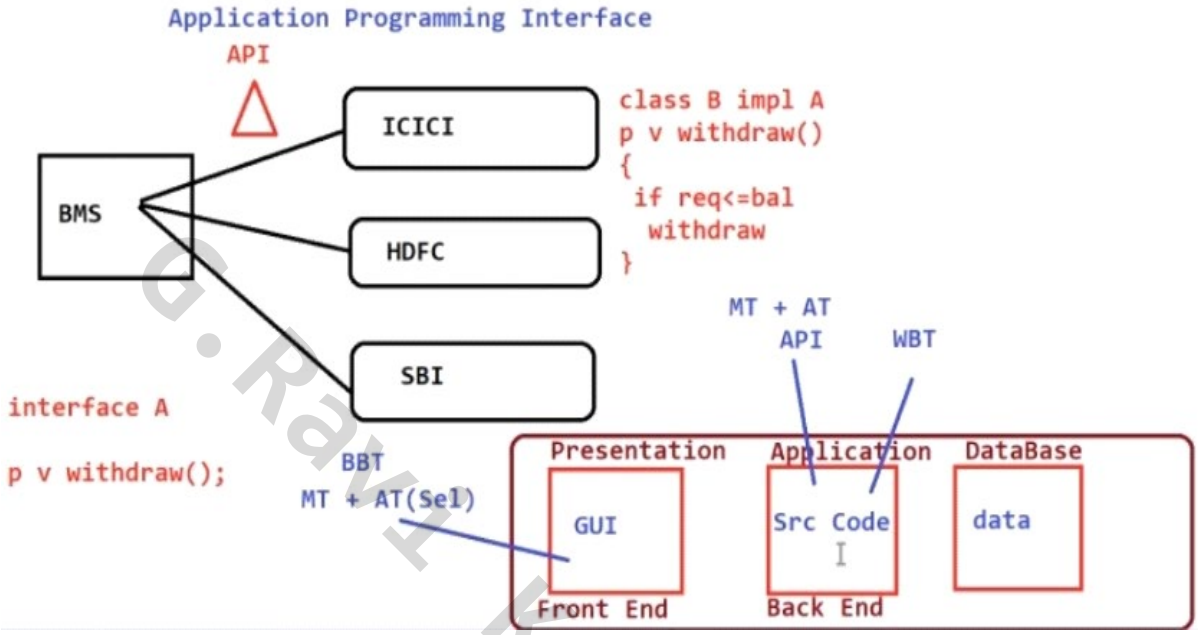
- Handling Multiple Elements
- Handling Synchronisation issue by using implicitly Wait and explicitly Wait
- Handling Dropdown (static and dynamic)
- Handling keyboard and Mouse actions
- Taking Screenshot
- Handling Disabled Element
- Peforming Scroll down Action
- WebElement Interface Methods
- Handling Popups (web-based and Window-based)
- Handling Frames
- Handling New Windows/New Tabs

AUTOMATION FRAMEWORK

=> Stages and Types of Framework
=> Explanation of Hybrid Framework with a combination of
- Data-Driven
- Keyword-Driven
- Method-Driven
    - POM(Page Object model)
- TestNG
    - Fetching TestNG Report
    - Batch Execution
    - TestNG Flags and Annotations
    - Assertion
    - Grouping Execution
    - Data Parameterisation
    - Data driven through DataProvider
    - Parallel Execution
        - Distibuted parallel Execution
        - Cross Browser Parallel Execution
- Modular Frameworks
- Hybrid Framework

=> Hybrid Framework Architecture
=> Introduction to Maven, GitHub and Jenkins

25-09-2020



- BMS - Book My Show application
- BBT - Black Box Testing
- WBT - White Box Testing
- MT - Manual Testing
- AT - Automation Testing
- GUI - Graphical User Interface
- Src Code - Source Code

- HTML - Hyper text Markup Language

- Anything which is present on the webpage is known as WebElements / WebComponents

- WebElements :- Link, textbox, radio button, checkbox, button, dropdown, Images, textarea, popups, frames, text

- These elements are developed by using HTML

- If we want to create any of the WebElements, we need 3 things :
    1. tag
    2. attribute
    3. text

Example to create a link :

```
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<a href='Error! Hyperlink reference not valid. </a>
</body>
</html>
```

- a means anchor

26-09-2020

- Tag should be enclosed within angular brackets
  <html> <a>   <input> <select>

- Attributes are defining characteristics of an element
  e.g: id, name, class, title, value, placeholder, href

- Text should be written either before starting angular brackets or after ending angular brackets
   Hello
  Hi

- Start from <html> tag
  Include <head> and <body> tags

- Inside the <head> tag we use <title> tag to provide title of the webpage

- Inside <body> tag we write all the elements

- To create link we use anchor tag <a> and compulsory attribute called 'href' (href means hyper reference) where we provide address of landing page

- To break the line and write element in next linewe use break rule

- To include simple text we can use <p> (paragraph) (division)

- To give tooltip (info about the element) we use 'title' attribute

Elements	tag	Compulsory attribute
link	<a>	href='address of landing page'
Normal TextBox	<input>	type='text'
password textbox	<input>	type='password'
Checkbox	<input>	type='checkbox'
radio button	<input>	type='radio' name='some value' value should be same for every radio button
button	<input>	type='button' value='some text' what we expect on button
DropDown	<select>	NA
Optons of DropDown	<option>	NA
table	<table>	NA
table head	<thead>	NA
table body	<tbody>	NA
table row	<tr>	NA
table data	<td>	NA
Paragraph	<p>	NA
Image	<img>	src='path of image'

- **different links in seperate lines using break, using title attribute and dropdown**

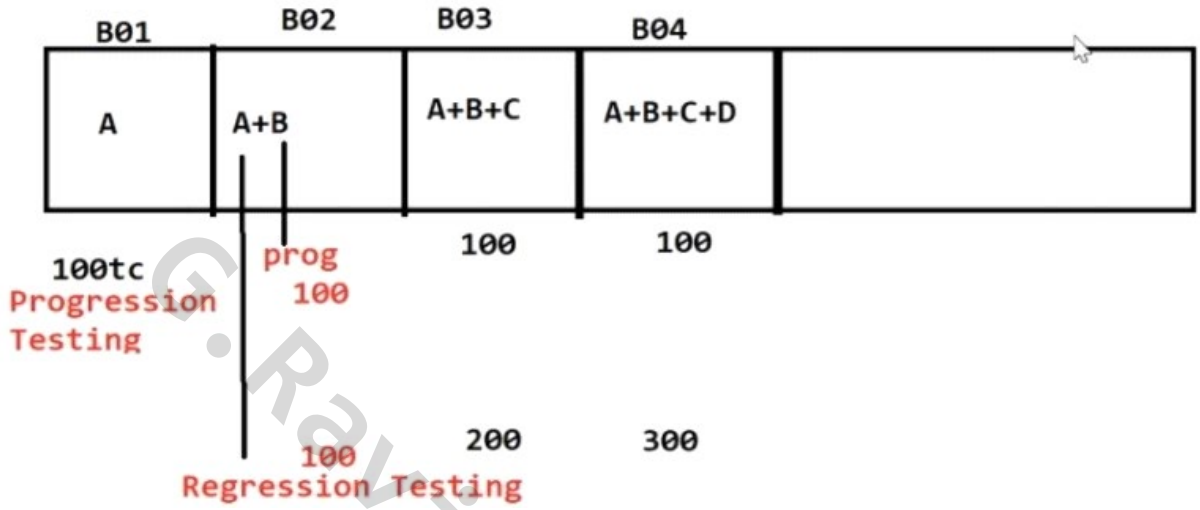
```
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<a href='https://www.google.com' id='i1' name='n1' class='c1'
value='v1'>Google </a> <br>
```

```
<a href='https://www.google.com' id='i2' name='n2' class='c2'
value='v2'>Google </a> <br>
<a href='https://www.facebook.com' title='click here to go to facebook
page'>Facebook </a>
<select name='month'>
<option> </option>
<option>JAN </option>
<option>FEB </option>
<option>MAR </option>
<option>APR </option>
<option>MAY </option>
<option>JUN </option>
<option>JUL </option>
</select>
</body>
</html>
```

- **Creating a Signup page**

```
<html>
<head>
<title>My Registration Page</title>
</head>
<body>
<marquee> Ravi kiran </marquee>
<h1> My WebPage </h1>
<h2> My WebPage </h2>
<h3> My WebPage </h3>
<h4> My WebPage </h4>
<h5> My WebPage </h5><br><br>
<b><span> I dont know what to write <span></b><br><br>
Username<input type='text'></input><br>
Password<input type='password'></input><br>
<input type='radio' name='gender'>MALE</input><br>
<input type='radio'name='gender'>FEMALE</input><br>
<input type='radio'name='gender'>NOT SURE</input><br>
<input type='checkbox'>Manual Testing</input><br>
<input type='checkbox'>SQL</input><br>
<input type='checkbox'>Core JAVA</input><br>
<input type='checkbox'>Automation Testing</input><br>
<input type='button' value='Signup'></input><br>
</body>
</html>
```

28-09-2020



Automation Testing:

- Process of tesing the functionality of an application, by the help of automation tools is known as Automation Testing.

(or)

- Converting manual testcases into automaton scripts, executing them through a framework and getting test results is known as Automation Testing.

Why we switch to Automation Testing ? (or)
Advantages of Automation Testing ?

---

1. We switch to Automaton testing when we have to do regression testing
2. While regression testing, there are lot of repetative tasks, and performing them manually is a boring job.
3. Also in regression when the product also increases, time taken to test also will increase. Hence, we switch to automation to reduce the time taken for testing.
4. To reduce the manpower/number of resources.
5. To obtan the test results faster.
6. To get quick ROI(Returns On Investment)
7. We can expect accurate results.
8. Due to tough competition, software companies need to deliver high quality product within less time. Hence, we switch to Automation Testing.

29-09-2020

- We cannot automate progression testing, because we don't know the flow of the functionality testng.
- We sart automating from the second release.
- Anything which is completely dynamic cannot be automated.
  Ex: Pc-games, OTPs

Drawbacks/Limitations of Automation Testing

- 100% Automation testing is not possible.
- Anything which is completely dynamic cannot be automated.
- Anything which requires manual intervention(interaction) cannot be automated.
  eg, what all cannot be AUTOMATED.....
  1.OTP, CAPTCHA
  2.Making Payments through swiping the Debit/Credit card
  3.Barcode scanner, QR code Scanning
  4.Animation
  5.Testing the quality of audio/video
  6.Game Testing
- We need skilled resources to do automation testing.
- Cost involved in Automation testing is more compared to manual testing, because of tools are licensed and automation skilled resources are paid higher.
- We can't automate an unstable product, we have o wait for the product to become stable.(newly built features will not be automated)

=> free is different and Open source is different
=> **Open source :-** The source code is visible to all of us.
=> But, **Selenium** is both free and Open Source.

- Jason Huggins - 2004 - Comapny:Thoughtworks, place:Chicago,USA
- JavaScript
- JavascriptTestRunner

=> Mercury is a licensed Testing tool , for this a competitive testing tool came into market that is Selenium

- Selenium is a free Open Source, Web-Application Automation tool.

30-09-2020

What is Selenium ?
Selenium is a free, Open-Source, Web-based Application automation tool.

It can be downloaded from https://www.selenium.dev/downloads/

Source code is visible from
https://github.com/SeleniumHQ/selenium
https://github.com/SeleniumHQ/selenium/blob/master/java/client/src/org/openqa/selenium

Selenium Developed by Jason Huggins in 2004 in ThoughtWorks company in Chicago, USA
Earlier name --> 'JavaScriptTestRunner'

To compete with HP-Mercury automation tool, renamed tool as 'Selenium'

Selenium Versions / Selenium Components / Selenium Flavors

- Selenium Core
- Selenium IDE - Record and play back tool
- Selenium RC(Remote Control) / Selenium-1
- Selenium WebDriver/Selenium-2 --- 2007,developed by Google
  current version 3.141.59 -- version 4 in alpha stage
- Selenium Grid
- Appium - automating mobile applications
- Winnium - automating window applications

Note:

Selenese: selenium Commands which are used in Selenium IDE.

- Selenium supports all the Programming languages
  Java, C#, Javascript, Perl, Ruby, Python, R, TCL, Elixir, Haskell

- Selenium supports all the Browsers
  Chrome, Firefox, Opera, Safari

- Selenium supports all the OS like windows, Mac, Linux but except for unix

RC Drawback :
---
1. We need a server without which automation is not possible
2. Same Origin Policy
   www.google.com //Automation possible
   www.google.com/search.html //Automation possible
   www.google.com/goToLink.com //Automation possible
   www.google.com/maps //Automation possible
   www.youtube.com //Automation not possible

Tools

Automation Testing Tools:

Selenium, QTP(Quick Test Professional), TestOptimize, SilkTest, Appium, Winnium, TestComplete, AutoIT

Defect Tracking Tools:

Jira, Bugzilla, Bugzero, Mantis, Bug-genie, rational clear quest

Test Management Tools:

Test Rail, Jira, QTest, Test Manager
QC(Quality Center), ALM(Application Lifecycle Management), TestManager

Performance Testing Tools:

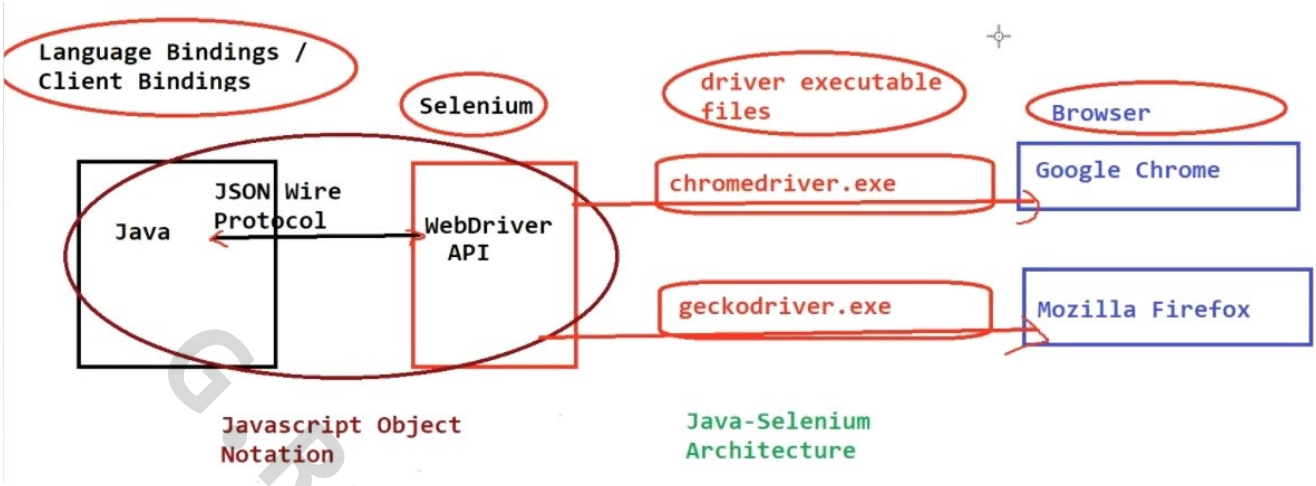
LoadRunner, Jmeter, NeoLoad, Web Load

They are 3 types of applications -
1. Standalone/Desktop application : chrome, firefox, safari
2. Web application : gmail
3. Client server/Mobile application :

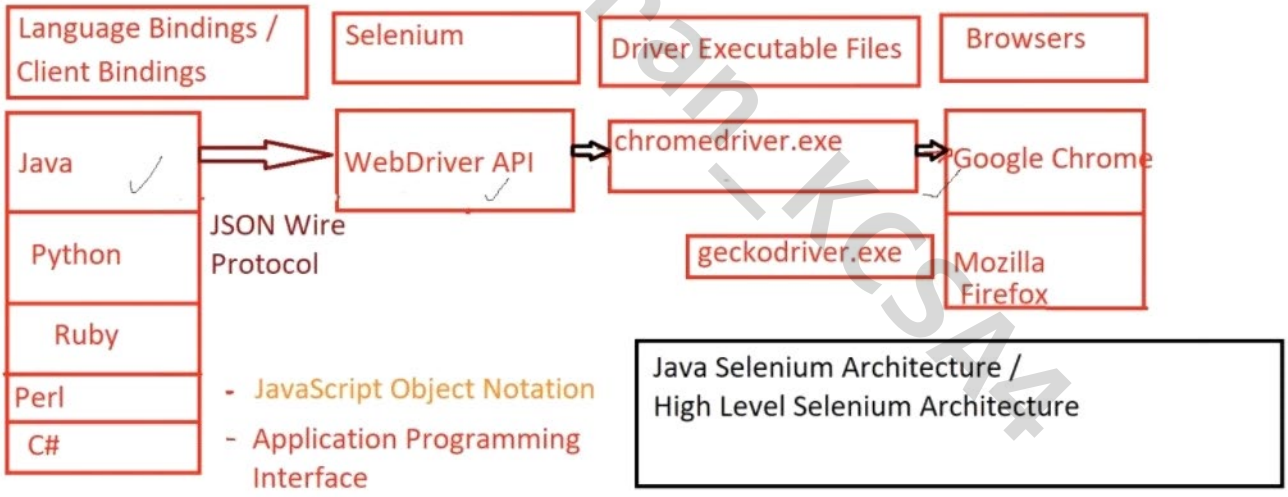
Java-Selenium Architecture

- Selenium WebDriver API communicates with programming language like java by using JASON(Javascript Object Notation) Wire Protocol.
- These 2 combined communicates with the browsers by using respective driver exceutible files.

Java-Selenium Architecture



This diagram talks about how Java-Selenium Communicates with Browser.
Selenium WebDriver API communicates with language Bindings like Java by using JSON Wire Protocol(Javascript Object Notation).
These two combined communicates with the browsers with the help of respective driver executable files like chomedriver.exe, geckodriver.exe etc.



Required Software and files for automating :-

1. JDK
2. IDE(eclipse)
3. Selenium Jar File
4. driver executable file
5. Browser

01-10-2020

Selenium Installation Steps :-
---

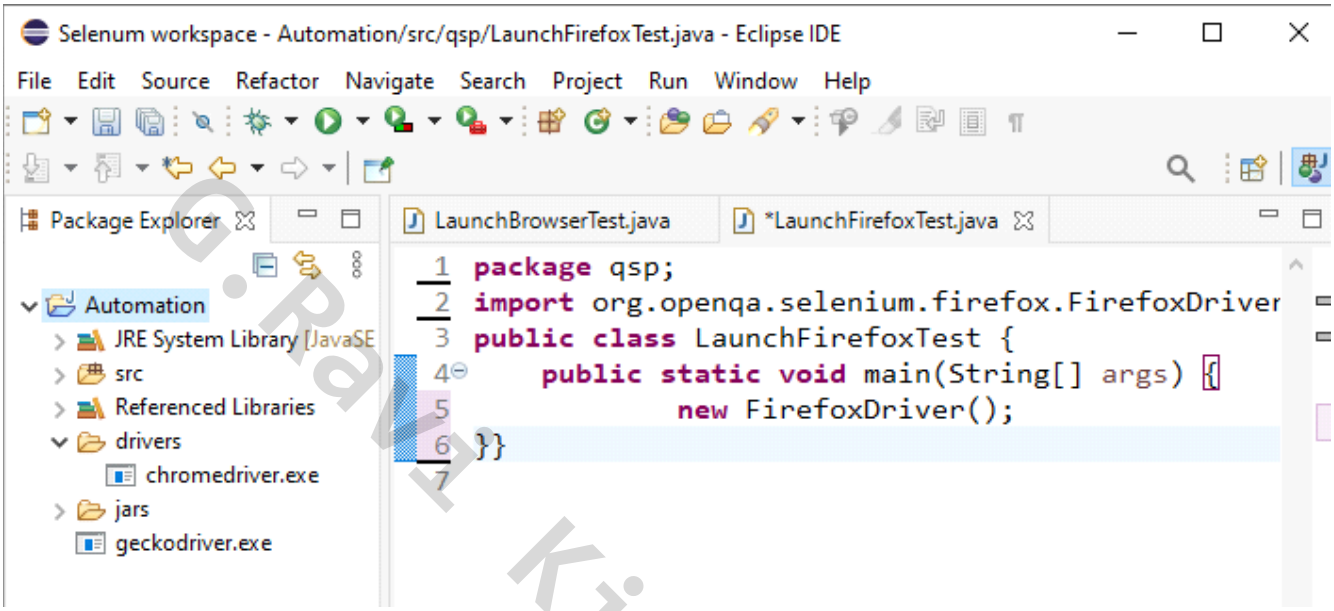
follow the below link
https://drive.google.com/file/d/1OqGqC_xj3UKoZFfaurNkovgvtjyuME-r/view?usp=sharing

- **How to open chrome browser using selenium ?**

```
package qsp;
import org.openqa.selenium.chrome.ChromeDriver;
public class LaunchBrowserTest {
public static void main(String[] args) {
	//For every Browser --> class
	//chrome ---> ChromeDriver
	//Firefox --->FirefoxDriver

	//Set System Property

	//key --> type of browser
	String key="webdriver.chrome.driver";
	//value --> path to the driver executable files
	String value="E:<Error! Hyperlink reference not valid.>";
	System.setProperty(key, value);

	//Open the Chrome browser
	//In selenium --> Just create an object of ChromeDriver class
	new ChromeDriver().close();
}}
```

02-10-2020

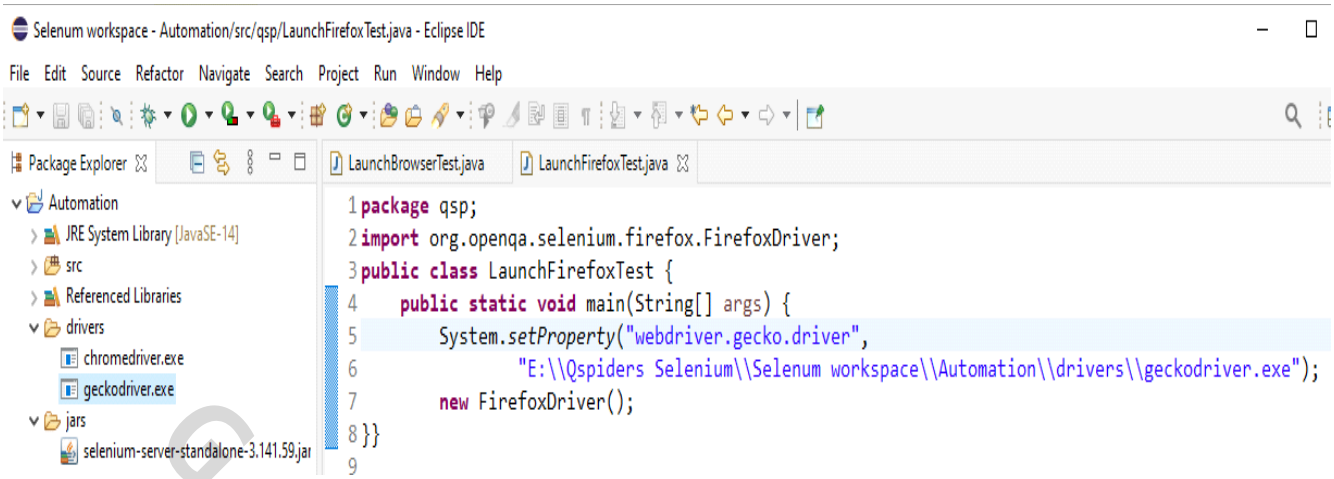
- **How to open Firefox browser without using system property ?**



- by keeping geckodriver.exe directly under automation folder, we can open Firefox browser without using system property.

- **How to open Firefox browser with using system property ?**

```
package qsp;
import org.openqa.selenium.firefox.FirefoxDriver;
public class LaunchFirefoxTest {
public static void main(String[] args) {
      System.setProperty("webdriver.gecko.driver",
                  "E:<Error! Hyperlink reference not valid.>");
      new FirefoxDriver();
}}
```



- **Opening Firefox browser using shortcut for value in System property.**

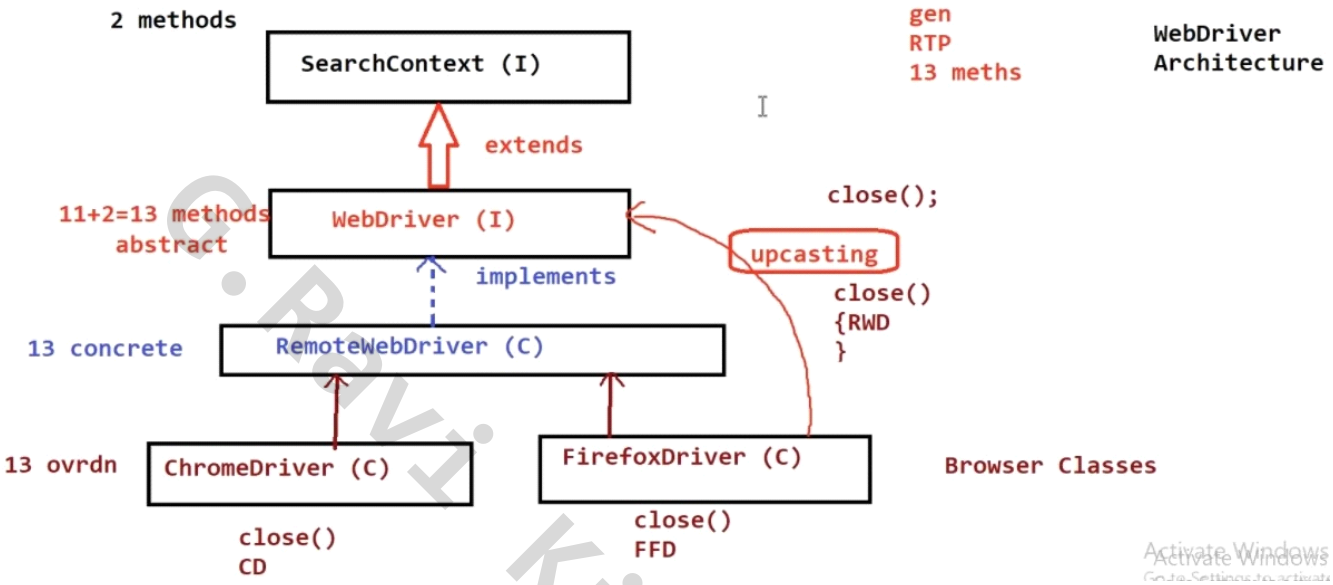
```
package qsp;
import org.openqa.selenium.firefox.FirefoxDriver;
public class LaunchFirefoxTest {
public static void main(String[] args) {

System.setProperty("webdriver.gecko.driver",".\\drivers\\geckodri
ver.exe");
        new FirefoxDriver();
}}
```

- **Opening both chrome and firefox browsers at a time.**

```
package qsp;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class LaunchBrowserTest {
public static void main(String[] args) {

        String key="webdriver.chrome.driver";
        String value="E:<Error! Hyperlink reference not valid.>";
        System.setProperty(key, value);
        new ChromeDriver().close();


System.setProperty("webdriver.gecko.driver",".\\drivers\\geckodri
ver.exe");
        FirefoxDriver firefox = new FirefoxDriver();
        firefox.close();
}}
```

06-10-2020

WebDriver Architecture



- SearchContext is the Supermost interface, WebDriver interface extends it and all the 13 abstract methods of WebDriver are given implementation in RemoteWebDriver Class and all those concrete methods are overridden in respective browser classes like ChromeDriver Class, FirefoxDriver Class etc.

- As a Selenium standard, we always upcast our browser classes to WebDriver interface to achieve -
  1. Generalization
  2. Runtime polymorphism(at runtime I can decide in which browser my code will run)
  3. To get all those 13 methods required for automation testing.

```
package qsp;                    //Ex
import java.util.*;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class RuntimepolymorphismInSelenium {
      public static void main(String[] args) {
            WebDriver driver;
            System.out.println("Enter Browser Name :");
            Scanner sc=new Scanner(System.in);
            String browserName = sc.next();

            if(browserName.equalsIgnoreCase("chrome"))
      System.setProperty("webdriver.chrome.driver","./drivers/chromedri
ver.exe");
            driver=new ChromeDriver();
            }

            else if(browserName.equalsIgnoreCase("firefox")) {
      System.setProperty("webdriver.gecko.driver","./drivers/geckodrive
r.exe");
            driver=new FirefoxDriver();
            }
}}
```

07-10-2020

WebDriver Abstract Methods(Browser Window Related Methods)

Initiation	get(String url)	void
Page Verification Methods/Data Capture Methods	getCurrentUrl()	String
Page Verification Methods/Data Capture Methods	getPageSource()	String
Page Verification Methods/Data Capture Methods	getTitle()	String
Data Capture Methods	getWindowHandle()	String
Data Capture Methods	getWindowHandles()	Set<String>
Browser Window Handling Methods	manage()	Options
Navigation Method	navigate()	Navigation
Inspection Methods	findElement(By arg)	WebElement
Inspection Methods	findElements(By arg)	List<WebElement>
Control Switching Method	switchTo()	TargetLocator
Termination Methods	close()	void
Termination Methods	quit()	void

1. **get(String url) :** used to enter the URL
2. **getCurrentUrl() :** used to get the URL of the Current WebPage
3. **getPageSource() :** used to get the Source code of the Current WebPage
4. **getTitle() :** used to get the title of the Current WebPage
5. **getWindowHandle() :** used to get the window handle of the current browser window
6. **getWindowHandles() :** used to get the window handle of all the browser windows

7. manage() : used to manage browser window

8. navigate() : used to navigate from ome page to another, previous page, next page, can refresh current WebPage

9. findElement(By arg) : find a particular element on the WebPage

10.findElements(By arg) : find multiple elements on the WebPage

11.switchTo() : used to switch our control from WebPage to popups, frames, windows etc

12.close() : close the current browser window

13.quit() : close all the browser windows opened by selenium

```
package qsp;                                    //Ex-1
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class WebDriverMethods {
      public static void main(String[] args) {
      System.setProperty("webdriver.chrome.driver",
                              "./drivers/chromedriver.exe");
      WebDriver driver=new ChromeDriver();
      //enter the test url
      driver.get("https://www.google.com/");

      String url = driver.getCurrentUrl();
      System.out.println(url);

      String pgSrc = driver.getPageSource();
      System.out.println(pgSrc);

      String title = driver.getTitle();
      System.out.println(title);

      if(title.equals("Google"))
      {
        System.out.println("Google Page is Displayed, Test Step PASSED");
            }
      else
      {
        System.out.println("Google Page is not Displayed, Test Step
                        FAILED");
            }
}}
```

08-10-2020

```
package qsp;                                        //Ex
import org.openqa.selenium.Dimension;
import org.openqa.selenium.Point;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class WebDriverMethods {
      public static void main(String[] args) throws InterruptedException
{
            System.setProperty("webdriver.chrome.driver",
                                    "./drivers/chromedriver.exe");
            WebDriver driver=new ChromeDriver();
            //enter the test url
            driver.get("https://www.google.com/");

            //To maximize the browser window
            driver.manage().window().maximize();

            Thread.sleep(3000);
//          //enter the url
//          driver.navigate().to("https://accounts.google.com");
//
//          Thread.sleep(3000);
//          //navigate to previous page
//          driver.navigate().back();
//
//          Thread.sleep(3000);
//          //navigate to next page
//          driver.navigate().forward();
//
//          Thread.sleep(3000);
//          //Refresh the current page
//          driver.navigate().refresh();

            //method Chaining
            //Return type of current method should be same as parent of
            next


            //To change the size of the browser window
            Dimension d=new Dimension(600,200);
            driver.manage().window().setSize(d);
            Thread.sleep(3000);
```

```
            //To change the position of the browser window
            Point p=new Point(500,600);
            driver.manage().window().setPosition(p);

            //To delete all the cookies stored by the browser window
            driver.manage().deleteAllCookies();
}}
```

**** Differences between get() and navigate()**

S.No	get()	navigate()
1.	Enter URL	Enter URL Navigate to Previous Page Navigate toNext Page Refresh the Current Page
2.	get() will not take advantage of Browser History	navigate() will take advantage of Browser History
3.	get() will wait until complete page is loaded	navigate() will not wait until complete page is loaded

---
09-10-2020

- Before performing action on any element, first we should locate the element on webpage.
- Hence to find its address we have findElement() method and findElements() method inside WebDriver interface.

1. findElement() method of WebDriver interface is used to get the address of first matching element on the webpage.
2. Return type of findElement() method is WebElement interface.
3. If findElement() method is not able to find the element on the webpage then we get NoSuchElementException.
4. findElement() method takes locators as its argument.

What are Locators ?

Definition:

1. Locators are static methods of abstract 'By' class.
2. Locators are used to locate one or more elemens on the webpage.
3. Locators act as argument for findElement() and finElements().

Types of Locators :-

---

1. tagName(String arg)
2. id(String arg)
3. name(String arg)
4. className(String arg)
5. linkText(String arg)
6. partialLinkText(String arg)
7. cssSelector(String arg)
8. xpath(String arg)

```
package qsp;                                    //Ex
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class Locators {
      public static void main(String[] args) throws InterruptedException
{

System.setProperty("webdriver.chrome.driver","./drivers/chromedriver.exe");
```

```
            WebDriver driver=new ChromeDriver();

        driver.get("file:///E:/Qspiders%20Selenium/.html/demo.html");
            Thread.sleep(2000);

//          Locators - Static methods of abstract 'By' Class
//          WebElement ele = driver.findElement(By.tagName("a"));
//          System.out.println(ele);
//          ele.click();

//          driver.findElement(By.id("i2")).click();

//          driver.findElement(By.name("n2")).click();

//          driver.findElement(By.className("c2")).click();

            driver.findElement(By.linkText("Google")).click();

//          <a>Inbox(30)</a>  partially dynamic link
            driver.findElement(By.linkText("Inbox(25)")); //Wrong

//          What is partially dynamic link ?
//          Link in which a part of it is static and part of it is dynamic

//          How to handle partially dynamic link ?
//          By using partialLinkText, partialLinkText means part of the
             complete linktext
//          wherein we use static part and ignore dynamic part

            driver.findElement(By.partialLinkText("Inbox"));

//          Bhanuprakasha - linkText
//
//          partialLinkText
//          Bhanu
//          prakash    //uses of 'partialLinkText' locator
//          akash            1.To handle partially dynamic links
//          asha             2.To avoid writing lengthy linktexts
//          anu
//          hanu
//          anup

//          driver.findElement(By.partialLinkText("akash"));

        }
}
```

10-10-2020

HTML Code to create link

```
<html>
<body>
<a href='https://www.google.com' id='i1' name='n1' class='c1' value='v1'
title='t1'>Googly</a>
</body>
</html>
```

- In the above code, to create a link we use <a> tag and mandatory attribute called 'href' but other attributes like id, name, class, value, title are optional.

- So, in selenium we can use features like tag, attributes like id, name and class and text to identify the element on the webpage.

- So accordng to above example -
  1. **tagName**("a") - we use tagName of the element
  2. **id**("i1") - we use id of the element
  3. **name**("n1") - we use name of the element
  4. **className**("c1") - we use class of the element
  5. **linkText**("Googly") - Here we use text but make sure element is 'link' only
  6. **partialLinkText**("Go") - Here we use a part of the complete linktext
  7. **cssSelector** - Here we can identify any element by giving both tag and attribute
     - Syntax: tagName[AtributeName='AttributeValue']
     - Example: a[href='https://www.google.com']
     - Example: a[id='i1']
     - Example: a[name='n1']
     - Example: a[class='c1']
     - Example: a[value='v1']
     - Example: a[title='t1']

- Shortcuts of Css Selector :-
    - Shortcut for id is #
      div[id='i1']
      div#i1
      #i1
    - Shortcut for class is .
      div[class='c2']
      div.c2
      .c2

- To VERIFY the cssSelector expression and xpath expression, we download an add-on for FIREFOX browser called 'Try Xpath'
- Steps to downlaod and how to use 'Try Xpath'
    a. open Firefox browser --> Go to google and type 'download try xpath for firefox'
    b. Click on 'Add to Firefox' button and click on 'Add' button
    c. Try xpath will be installed and visible at top right corner of firefox browser
    d. Click on TX icon on top right corner
    e. For verifying cssSelector expression, change Way to 'QuerySelectorAll'
    f. Write the cssSelector expression and click on execute button

Limitation of CssSelector :
- To use cssSelector minimum 1 attribute should be present which is unique, if no attribute is present, we can't use cssSelector because cssSelector does not support text.

8. **xpath** -
- xpath is the path travelled in the HTML tree to find an element
- xpath is one of the locator which covers all possible ways to find an element
- There are two types of xpath -
    1. Absolute xpath
    2. Relative xpath

Absolute xpath means complete path we travel from start(html) element to --> it is achieved by '/' (Single Forward Slash) --> immediate child/immediate descendant

HTML Code //Ex

```
<html>
<body>

<div>
A<input type='text'></input>
B<input type='text'></input>
</div>

<div>
V<input type='text'></input>
D<input type='text'></input>
</div>

</body>
</html>
```

HTML Tree Structure

```
--------------------

html [1]
   body [1]
      div [1]
         input A [1]
         input B [2]
       div [1]
         input V [1]
         input D [2]
```

Elements	Absolute xpath
ABVD	html/body/div/input
AB	html/body/div[1]/input
A	html/body/div[1]/input[1]
AV	html/body/div/input[1]
BV	html/body/div[1]/input[2] | html/body/div[2]/input[1]

12-10-2020

Relative xpath :

- Shortest path of element in HTML Tree
- // means any child or any descendant

HTML Tree Structure

```
html [1]
     body [1]
          div [1]
               input A [1]
               input B [2]
          div [1]
                input V [1]
                input D [2]
```

Elements	Relative xpath
ABVD	//input
AB	//div[1]/input
A	//div[1]/input[1]
AV	//input[1]
BV	//div[1]/input[2] | //div[2]/input[1]
AD	//div[1]/input[1] | //div[2]/input[2]

Cases of Relative xpath:

Case 1: xpath by unique attribute

Syntax: //tagName[@AttributeName='AtributeValue']

Ex-1://div[@id='SIvCob'] ---> https://www.google.com/
Ex-2://p[@class='login'] ---> https://www.skillrary.com/user/login
Ex-3://span[@class='icp-nav-link-inner'] ---> https://www.amazon.in/

Case 2: xpath by text() function

Syntax: //tagName[text()='textValue']

Ex-1://a[text()='Images'] ---> https://www.google.com/
Ex-2://label[text()='Do you have group code?']
---> https://www.skillrary.com/user/login
Ex-3://a[text()='Amazon Pay'] ---> https://www.amazon.in/

<div fn='Deepika' ln'Rai'>

//div[@fn='Deepika'] ----> 2
//div[@ln='Rai'] ----> 2
//div[@fn='Deepika' and @ln='Rai'] ----> 1

Case 3: xpath by multiple attributes

Syntax:

- UniqueElement(and): //tagName[@attr1='attrVal' and @attr2='attrVal' and .....]
- ManyElements(or): //tagName[@attr1='attrVal' or @attr2='attrVal' or .....]

Ex-1://input[@class='RNmpXc' or @type='submit']
---> https://www.google.com/
Ex-2://input[@id='groupcode' and @type='text']
---> https://www.skillrary.com/user/login
Ex-3://span[@class='a-size-base' and text()='4GB RAM+ 64GB Storage']
---> https://www.amazon.in/

13-10-2020

```
<div id='ui-id-1'>
<div id='ui-id-2'>
<div id='ui-id-3'>
<div id='ui-id-4'>
<div id='ui-id-5'>
```

Case 4: xpath by contains() function

Syntax: //tagName[contains(@attrName,'AttrValue')]

Write xpath expression to match all the above five

1. Handle Parially Dynamic Elements
   //div[contains(@id,'ui-id')]
2. Avoid Writing lengthy text
   //p[contains(text(),'Uttara Karnataka')]
   ---> http://yuvadhwaja.in/initiatives.html
3. Avoid Spaces present in the values
   //div[contains(text(),'Login']

Case 5: xpath by axis(Relationship)

```
Travel from Parent ---> Child
          parent element/descendant::child element
Travel from Child ---> Parent
          child element/ancestor::parent element
Travel from one element --->next element of same parent
          element/following-sibling::next element
Travel from one element ---> previous element of same parent
          element/preceding-sibling::previous element

<html>
<body>
<table border='1'>
<tbody>

<tr id='t1'>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>

<tr id='t2'>
<td>E</td>
```

```
<td>F</td>
<td>G</td>
<td>H</td>
</tr>

</tbody>
</table>
</body>
</html>
```

HTML Tree Structure :

```
html
      body
            table
                  tbody
                        tr 't1'
                           td A
                           td B
                           td C
                           td D
                        tr 't2'
                           td E
                           td F
                           td G
                           td H
```

```
1. html to body  ---> //html/descendant::body
2. html to tbody ---> //html/descendant::tbody
3. html to B     ---> //html/descendant::td[text()='B']
4. tbody to G    ---> //tbody/descendant::td[text()='G']
5. t1 to A,B,C,D ---> //tr[@id='t1']/descendant::td

6. E to t2    ---> //td[text()='E']/ancestor::tr[@id='t2']
7. t1 to body ---> //tr[@id='t1']/ancestor::body
8. C to html  ---> //td[text()='C']/ancestor::html

9. A to B      ---> //td[text()='A']/following-sibling::td[text()='B']
10. A to B,C,D ---> //td[text()='A']/following-sibling::td
11. F to H     ---> //td[text()='F']/following-sibling::td[text()='H']
12. A to C,D   ---> //td[text()='A']/following-sibling::td[text()='C' or
                                                          text()='D']

13. G to F   ---> //td[text()='G']/preceding-sibling::td[text()='F']
14. D to A   ---> //td[text()='D']/preceding-sibling::td[text()='A']
15. H to F,G ---> //td[text()='H']/preceding-sibling::td[text()='F' or
                                                        text()='G']
```

14-10-2020

Cousin
B to F --->

- //td[text()='B']/ancestor::tr/following-sibling::tr[@id='t2']/descendant::td[text()='F']
- //td[text()='B']/ancestor::tbody/descendant::td[text()='F']

Uncle
G to t1 --->

- //td[text()='G']/ancestor::tr/preceding-sibling::tr
- //td[text()='G']/ancestor::tbody/descendant::tr[@id='t1']

Case 6: xpath by Dependant and independant element
or
xpath by reference of surrounding unique element

ex-1: Real-time example from https://www.selenium.dev/downloads/

Ruby	Download
Java	Download
Python	Download
C#	Download
JS	Download

- //td[text()='Java']/following-sibling::td[@data-label='Links']/descendant::a[text()='Download']
- //td[text()='Java']/ancestor::tr/descendant::a[text()='Download']
- //td[text()='Java']/..//a[text()='Download']

ex-2: Real-time example from
https://www.flipkart.com/mobiles/pr?sid=tyy%2C4io&p%5B%5D=facets.brand%255B%255D%3DRealme&otracker=nmenu_sub_Electronics_0_Realme

- //div[text()='Motorola']/preceding-sibling::div[@class='_1p7h2j']

15-10-2020

Case 7: xpath by group index

```
html [1]
     body [1]
          div [1]
             a [1]
             a [2]
             a [3]
          div [2]
             a [1]
             a [2]
             a [3]
          div [3]
             a [1]
             a [2]
             a [3]

//a --> 9
//a[1] --> 3
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
a[8]
a[9]

(//a)[6]  --> 1
(//div)[1] --> 1

(//a[@href='index.html'])[1] ---> http://yuvadhwaja.in/

(//a[text()='Download'])[2] ---> https://www.selenium.dev/downloads/
```

- Difference between:- //a, //a[1], (//a)[1]
  1. //a - matches all the links on the webpage
  2. //a[1] - match all the first links of their parent
  3. (//a)[1] - matches very first link of the webpage

- //input - matches every input on webpage
- //inpur[1] - matches all the first inputs of their parent
- (//input)[1] - matches very first input of the webpage
- (//input)[last()] - matches very last input
- (//input)[last()-1] - match last but one
- (//input)[position() mod 2 =0] - match all even inputs
- (//input)[position() mod 2 =1] - match all odd inputs

- //* - matches every tag on the whole webpage

- 4 mostly used Locators-
    1. id
    2. name
    3. linktext
    4. xpath

- fastest Locator --> tagName

- slowest Locator --> xpath

- Arranging of fastest to slowest locators :-
  id, name, cssSelector, xpath

- most efficient loactor --> xpath

- Differences between cssSelector vs xpath

cssSelector	xpath
cssSelector does not support text, it requires atleast one attribute	xpath supports text and also attribute
cssSelector is faster than xpath	xpath is slower than cssSelector
cssSelector is uni-Directional(forward)	xpath is Bi-Direcional(forward+backward)

16-10-2020

TestCase to Login to an Application
---

step no	Action Description	input	Expected Result	Actual Result	Status	Comments
1	Open the browser, Enter the test URL	https://demo.actitime.com/login.do	Login Page should be displayed			
2	Enter Valid Username, password and click on login button	UN- PW-	Home Page(Enter Time Track Page)should be displayed			


```
//Write a selenium program to login to an application

package qsp;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class LoginTest {
	public static void main(String[] args) throws InterruptedException
{

	//Set the system property
	System.setProperty("webdriver.chrome.driver",
				"./drivers/chromedriver.exe");

	//Open the Browser
	WebDriver driver = new ChromeDriver();

	//Maximize the browser window
	driver.manage().window().maximize();

	//Enter the test URL
```

```
        driver.get("https://demo.actitime.com/login.do");

        //Get the Login page title
        String loginTitle = driver.getTitle();

        //Verify actual title with expected title
        if(loginTitle.equals("actiTIME - Login"))
        {
         System.out.println("Login Page is Displayed, Test Step PASSED");
        }
        else
        {
        System.out.println("Login Page is not Displayed, Test Step FAILED");
        }

        Thread.sleep(2000);
        //Enter Valid Username in username textbox
        driver.findElement(By.id("username")).sendKeys("admin");

        Thread.sleep(2000);
        //Enter Valid Password in password textbox
        driver.findElement(By.name("pwd")).sendKeys("manager");

        Thread.sleep(2000);
        //Click on Login Button
        driver.findElement(By.xpath("//div[text()='Login ']")).click();

        Thread.sleep(5000);

        //Get Home Page title
        String homeTitle = driver.getTitle();
        //Verify actual title with expected title
        if(homeTitle.equals("actiTIME - Enter Time-Track"))
        {
         System.out.println("Home Page is Displayed, Test Step PASSED");
        }
        else
        {
         System.out.println("Home Page is not Displayed, Test Step FAILED");
        }
}}
```

Handling Multiple Elements
---

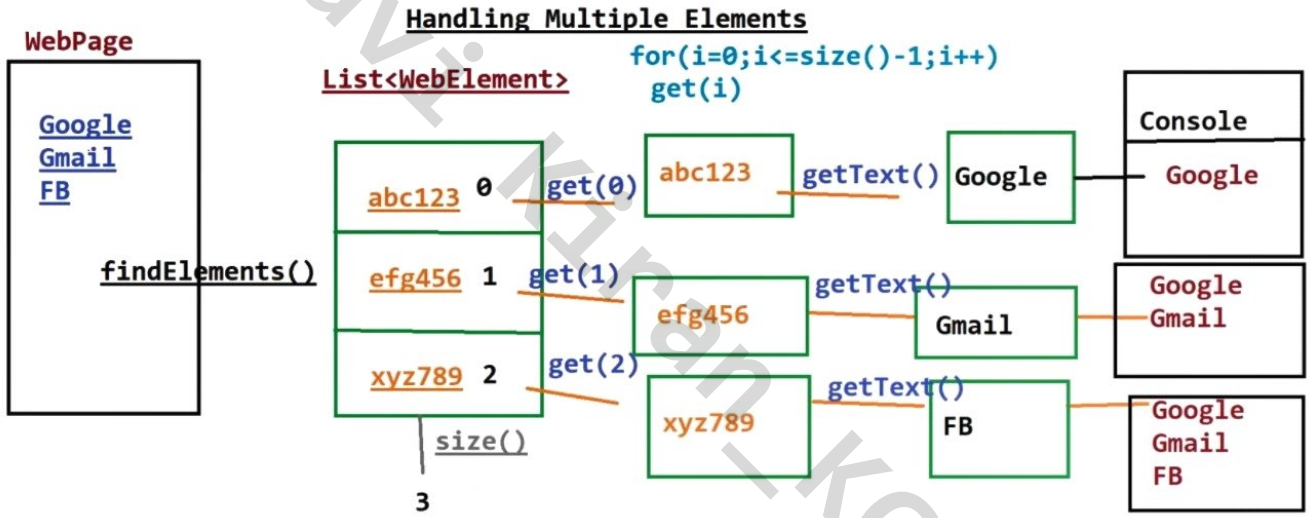
1. We Handle multiple Elements by using findElements() method of WebDriver interface.
2. Return type of findElements() is List<WebElement>
3. If findElements() method is not able to find the elements on the webpage, then we get emptyList.

Note: If findElement() method is not able to find the element on the webpage, then we get "NoSuchElementException".

```
<html>
<body>
<a href='https://demo.actitime.com/login.do'>actitime</a><br>
<a href='https://www.google.com/'>Google</a><br>
<a href='http://www.yuvadhwaja.in/'>Yuvadhwaja</a><br>
</body>
</html>
```

```
package qsp;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class HandlingMultipleElements {
	public static void main(String[] args) {

		System.setProperty("webdriver.chrome.driver",
					"./drivers/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("file:///E:/Qspiders%20Selenium/.html/3links.html
");

	//Getting address of Multiple elements
	List<WebElement> allLinks = driver.findElements(By.tagName("a"));
	//Count Number of elements
		System.out.println("Total Links: "+allLinks.size());
	//
		//Printing text of all elements
	//	for(int i=0;i<allLinks.size();i++)
	//	{
	//		WebElement oneLink = allLinks.get(i);
	//		String text = oneLink.getText();
```

```
//          System.out.println(text);
//    }

      //clicking on the last link
      allLinks.get(allLinks.size()-1).click();

      //Handling Single Element
//    WebElement glink = driver.findElement(By.linkText("Google"));
//    System.out.println("TagName is: "+glink.getTagName());
//    System.out.println("Text is: "+glink.getText());
//    System.out.println("Attribute Value is:
                                  "+glink.getAttribute("href"));

      }
}
```



17-10-2020

WebDriver Interface

S. No	findElement()	findElements()
1.	findElement() is used to get the address of first matching element on the webpage	findElements() is used to get the address of all the matching elements on the webpage
2.	Returntype is WebElement interface	Returntype is List<WebElement>
3.	If findElement() method is not able to find the element on the webpage, we get **NoSuchElementException**	If findElements() method is not able to find the elements on the webpage, we get **emptyList**

Synchronisation:

- Matching the speed of Selenium with the speed of application, to avoid synchronisation issue is called Synchronization

- What is synchronsation issue ?
  Selenium is too fast but application might take time to load, hence this time mismatch is called Synchronisation issue.

- There are two types of waits to match selenium speed -
  1. Static wait
  2. Dynamic wait

1. **Static wait** means waiting time is fixed, here we use Thread.sleep(15000) means compulsorily we have to wait for 15 seconds.

2. **Dynamic wait** means waiting period is not fixed, it is changing according to application speed.
   - There are two types of Dynamic wait -
     1. Implicitlywait
     2. Explicitlywait

Implicitlywait:

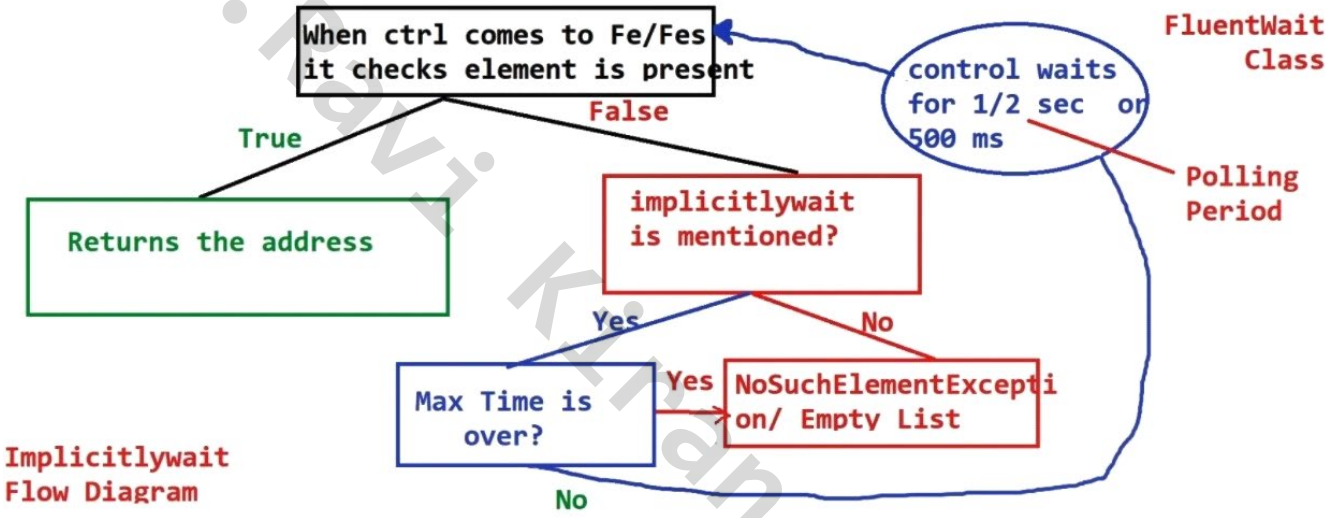
- Here no need to give any condition.
- It will wait for findElement() method and findElements() method

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

- here 20 means maximum time until control should wait before giving NOSuchElementException

Note: implicitlywait statemet can be declared once at the top of the code which can work for all the findElement() method / findElements() methods

ImplicitlyWait Flow Diagram



Explanation of ImplicitlyWait Flow Diagram :

- When the control comes to findElement() / findElements() it will check whether the element is present or not...if the element is present, then findElement() will return the address...

- But if element is not present then it will check whether implicitlyWait statement is mentioned or not...If no then we will get NoSuchElementException / empty List...If it is mentioned, then it will check whether time is over or not ???

- If time is over but element not found, NoSuchElementException...But if time is not over, then control will wait for 1/2 sec or 500 ms and again check whether element is present or not.

- This 500 ms duraton is called polling period and is mentioned in FluentWait Class.

```java
package qsp;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class FlipkartSynchronisation {
	public static void main(String[] args) throws InterruptedException
{
	System.setProperty("webdriver.chrome.driver",
				"./drivers/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://www.flipkart.com/");

	driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

		Thread.sleep(2000);

	driver.findElement(By.xpath("//button[text()='X']")).click();


	driver.findElement(By.xpath("//span[text()='Cart']")).click();

//		Thread.sleep(3000);

//		1/2 sec + 1/2 sec  + 1/2 sec + 1/2 sec

	driver.findElement(By.xpath("//span[text()='Login']")).click();

}}
```

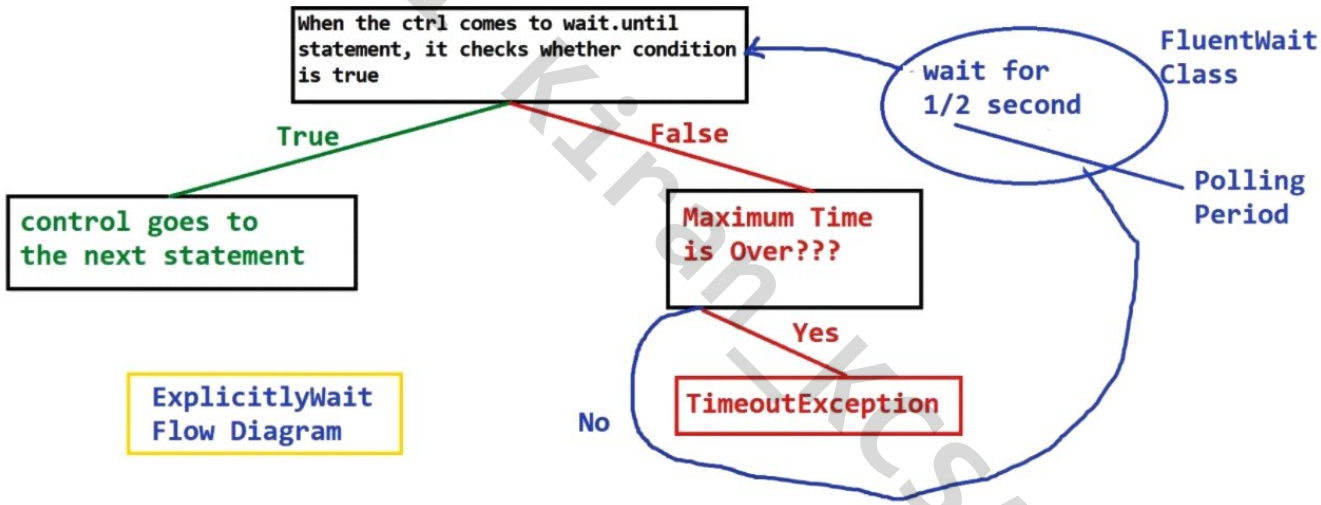
Explicitlywait:

- Here we need to give condition
- It will work for any method
- create an object of 'WebDriverWait' Class

```
WebDriverWait wait = new WebDriverWait(reference variable of WEbDriver, timeoutInSeconds);
```

- ExplicitlyWait will work for any method including findElement() and findElements()
- Here we need to create an object of WebDriverWait Class and we use the static methods of ExpectedConditions Class to give the condition...
- WebDriverWait class and ExpectedConditions Class is imported from 'org.openqa.selenium.support.ui' package

ExplicitlyWait Flow Diagram



Explanation of ExplicitlyWait Flow Diagram :

- When the control comes to wait.until() statement, it will check whether the condition is true or false.
- If condition is true, then control goes to the next statements and start executing them...
- If condition is false, then it will check maximum time is ove ror not
- If yes that means condition never became true within the maximum time, it gives TimeoutException
- If No, it waits for 1/2 sec or 500 ms and then checks again condition became true or not

```
package qsp;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class LoginTest {
	public static void main(String[] args) throws InterruptedException
{

		//Set the system property
		System.setProperty("webdriver.chrome.driver",
				"./drivers/chromedriver.exe");
		//Open the Browser
		WebDriver driver = new ChromeDriver();
		//Maximize the browser window
		driver.manage().window().maximize();
		//Enter the test URL
		driver.get("https://demo.actitime.com/login.do");

		//Get the Login page title
		String loginTitle = driver.getTitle();
	//Verify actual title with expected title
	if(loginTitle.equals("actiTIME - Login"))
	{
	 System.out.println("Login Page is Displayed, Test Step PASSED");
	}
	else
	{
	System.out.println("Login Page is not Displayed, Test Step FAILED");
	}

		Thread.sleep(2000);
		//Enter Valid Username in username textbox
		driver.findElement(By.id("username")).sendKeys("admin");

		Thread.sleep(2000);
		//Enter Valid Password in password textbox
		driver.findElement(By.name("pwd")).sendKeys("manager");

		Thread.sleep(2000);
		//Click on Login Button
		driver.findElement(By.xpath("//div[text()='Login
']")).click();
```

```
//          Thread.sleep(5000);

            WebDriverWait wait= new WebDriverWait(driver, 20);
            wait.until(ExpectedConditions.titleContains("Enter"));
//          1/2 + 1/2 + 1/2 + 1/2

            //Get Home Page title
            String homeTitle = driver.getTitle();
            System.out.println(homeTitle);

      //Verify actual title with expected title
      if(homeTitle.equals("actiTIME - Enter Time-Track"))
      {
      System.out.println("Home Page is Displayed, Test Step PASSED");
      }
      else
      {
      System.out.println("Home Page is not Displayed, Test Step FAILED");
      }
}}
```

19-10-2020 && 20-10-2020

***Differences between ImplicitlyWait and ExplicitlyWait

ImplicitlyWait	ExplicitlyWait
ImplicitlyWait is used to synchronize findElement() and findElements() only	ExplicitlyWait is used to synchronize any method
No need to mention any condition	We should provide condition according to the situation
No need to create Object	We create an object of WebDriverWait class
TimeUnit can be days, hours, minutes, seconds, milliSeconds, microSeconds, nanoSeconds	TimeUnit can only be Seconds
In ImplicityWait, if element not found within maximm time, we get NoSuchElementException/EmptyList	In ExplicitlyWait, if condition does not become true within maximum time, then we get TimeoutException
ImplicitlyWait is a kind of global wait, because we can mention implicitlyWait once at the top of the code which works for every findElement()/findElements() throughout the code	ExpliciltyWait is a kind of local wait, which has to be mentioned before every synchronization issue
Syntax:driver.manage().timeouts().implicitlyWait(long arg, TimeUnit) **Example:**driver.manage().timeouts().implicitlyWait(20,TimeUnit.SECONDS)	WebDriverWait wait=new WebDriverWait(WebDriver reference,TimeoutInSeconds) wait.until(ExpectedConditions.RequiredCondition)

```java
package qsp;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class FlipkartSynchronisation {
	public static void main(String[] args) throws InterruptedException
{

		System.setProperty("webdriver.chrome.driver",
					"./drivers/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://www.flipkart.com/");

//	driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
		Thread.sleep(2000);

	driver.findElement(By.xpath("//button[text()='X']")).click();


	driver.findElement(By.xpath("//span[text()='Cart']")).click();

//		Thread.sleep(3000);

		WebDriverWait wait=new WebDriverWait(driver,20);

	wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath
("//span[text()='Login']")));

//		1/2 sec + 1/2 sec  + 1/2 sec + 1/2 sec

	driver.findElement(By.xpath("//span[text()='Login']")).click();

}}
```

Handling Dropdown(ListBox/WebList/Custom Dropdown)

Two types

1. Static Dropdown
2. Dynamic dropdown

1. Static Dropdown: Options of the dropdown are fixed
    Two types of Static Dropdown
        1. Single-Select Dropdown
        2. Multi-Select Dropdown

Created by using
<select>, <option>

Q. How to handle the static dropdown ?

- Since it is created by using <select>
- We handle it by using Select Class.
- Select Class is imported from 'org.openqa.selenium.support.ui' package
- Select Class Constructor is a pramterized constructor
- which takes WebElement arg, where we need to pass address of the dropdown.

Select Class Methods

- **Selection Methods**
    1. selectByVisibleText(String arg) : void
    2. selectByValue(String arg) : void
    3. selectByIndex(int index) : void

- **DeSelection Methods**
    4. deselectByVisibleText(String arg) : void
    5. deselectByValue(String arg) : void
    6. deselectByIndex(int index) : void
    7. deselectAll()

- **Operational Methods**
    8. isMultiple() : boolean
    9. getOptions() : List<WebElement>
    10. getAllSelectedOptions() : List<WebElement>
    11. getFirstSelectedOption() : WebElement
    12. getWrappedElement() : WebElement

Note:

1. selectByIndex() can handle duplicate options
2. When we try to use deselection methods on a single select dropdown, then we get UnsupportedOperationException (Java)- (Runtime)

isMultiple() :

- Used to verify whether it is a Single Selectdropdown or multi-Select Dropdown.
- ReturnType is boolean.
- Returns true if it is multi-Select dropdown.

getOptions() :

- Used to get the address of all the options present in the dropdown.
- ReturnType is List<WebElement>
- If no option present in dropdown then we get emptyList

getAllSelectedOptions() :

- Used to get the address of all the selected options
- ReturnType is List<WebElement>
- If no option is Selected in dropdown then we get emptyList

getFirstSelectedOption() :

- Used to get the address of first selected option in the dropdown
- ReturnType is WebElement Interface
- If no option is selected, then it returns NoSuchElementException

getWrappedElement() :

- Used to get the address of all the options wrapped into a single address
- ReturnType is WebElement Interface
- If no option present in dropdown then we get NoSuchElementException

Note:

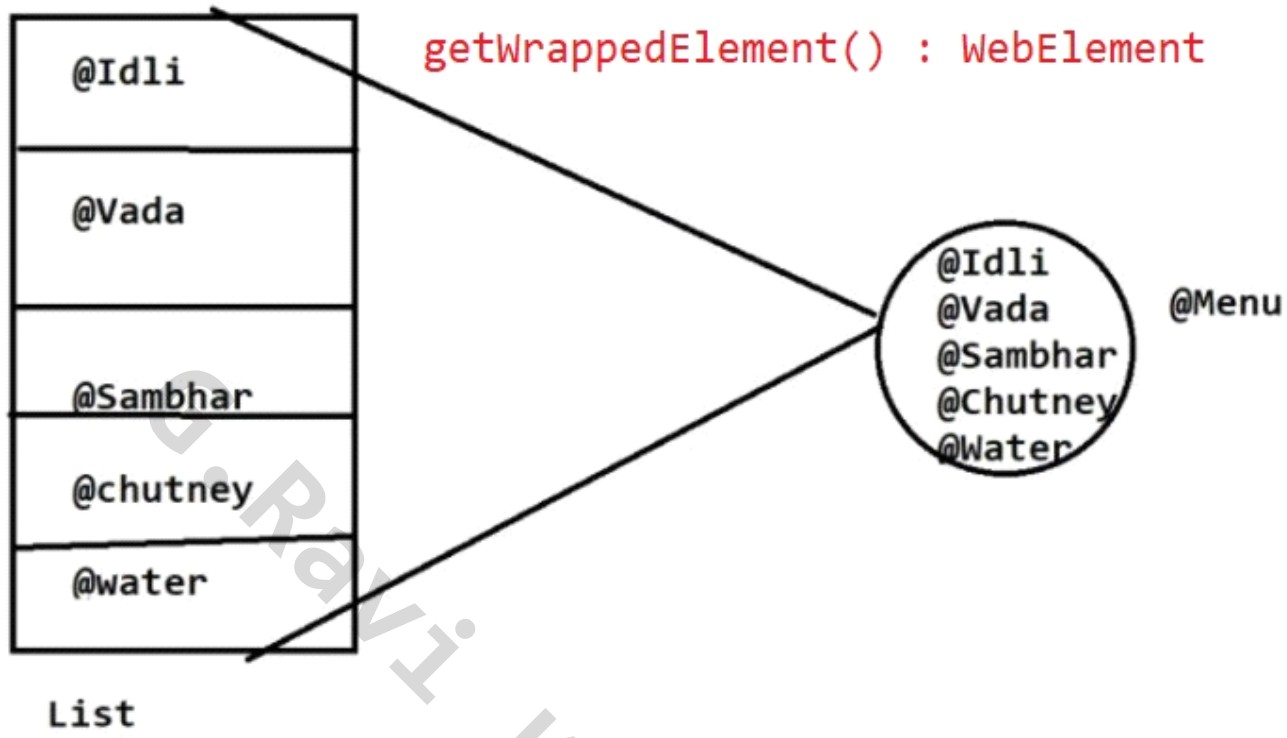
- To create a multi-select dropdown, add the attribute 'multiple' in select tag.

```
<html>
<body>
<select id='novotel' multiple>
<option value='a'>Idly</option>
<option value='b'>Biryani</option>
<option value='c'>Vada</option>
<option value='d'>Dosa</option>
<option value='e'>Pani Puri</option>
<option value='f'>Roti</option>
<option value='g'>Noodles</option>
<option value='h'>Jamoon</option>
<option value='i'>Idly</option>
<option value='j'>Chai</option>
</select>
</body>
</html>
```

```
package qsp;
```

```java
import java.util.List;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class HandlingDropdown {

	public static void main(String[] args) throws InterruptedException
{

		System.setProperty("webdriver.chrome.driver",
					"./drivers/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("file:///E:/Qspiders%
			20Selenium/.html/MultiSelectDropdown.html");

		Thread.sleep(3000);

		WebElement ddAddr = driver.findElement(By.id("novotel"));
		Select sel=new Select(ddAddr);//Address of Dropdown
		List<WebElement> allOptions = sel.getOptions();
		System.out.println("Total Options in DD: "+allOptions.size());

//		for(int i=0;i<allOptions.size();i++)
//		{
//			System.out.println(allOptions.get(i).getText());
//			sel.selectByIndex(i);
//		}
//		sel.deselectAll();

//		for(int j=0;j<allOptions.size();j++)
//		{
//			sel.deselectByIndex(j);
//		}

//		for(int j=allOptions.size()-1;j>=0;j--)
//		{
//			sel.deselectByIndex(j);
//			Thread.sleep(500);
//		}
```

```
            //for(start;end;incr/decr)
//          for(int i=2;i<=5;i++)
//          {
//                sel.selectByIndex(i);
//          }

//          List<WebElement> alSelOp = sel.getAllSelectedOptions();
//          System.out.println(alSelOp.size());
//        WebElement firstSelOp = sel.getFirstSelectedOption();
//        System.out.println(firstSelOp.getText());


            WebElement wrapEle = sel.getWrappedElement();
            System.out.println(wrapEle.getText());

//          for(  WebElement oneOp:alSelOp)
//          {
//                System.out.println(oneOp.getText());
//          }

        //sel.selectByVisibleText("Biryani");


//          if(sel.isMultiple())
//          {
//                System.out.println("It is Multi-Select Dropdown");
//          }
//          else
//          {
//                System.out.println("It is Single-Select Dropdown");
//          }


//          sel.selectByValue("d");
//          sel.selectByVisibleText("Biryani");
//          sel.selectByIndex(0);
//          sel.selectByVisibleText("Vada");
//          Thread.sleep(3000);
//          sel.deselectByVisibleText("Vada");
//          sel.deselectAll();
    }
}
```



Exceptions
---

1. IllegalStateException - Java - Runtime
Reason: System property not set
(or)
It is set incorrectly

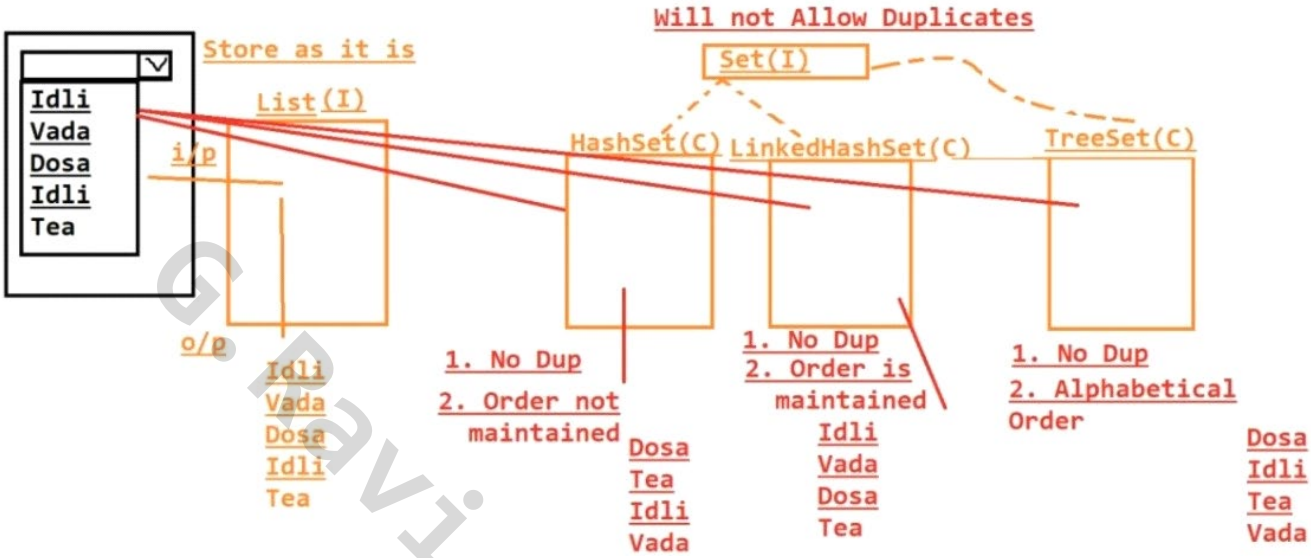
2. NoSuchElementException - Selenium - Runtime
Reason: If findElement() is unable to find the element on the webpage, then we get this exception

3. TimeoutException - Selenium - Runtime
Reason: In ExplicitlyWait, If condition is not becoming true within the maximum time, then we get this exception

4. InterruptedException - Java - Compiletime
Reason: Thrown when a thread is in sleeping state

5. UnsupportedOperationException - Java -Runtime
Reason: If we try to deselect any option from single-select Dropdown

---
21-10-2020




```
package qsp;

import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;
import java.util.TreeSet;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class RemoveDuplicateUsingSet {

	public static void main(String[] args) {
		System.setProperty("webdriver.chrome.driver",
					"./drivers/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("file:///E:/Qspiders%
		20Selenium/.html/MultiSelectDropdown.html");
```

```
            //print all the options as it is....
//          WebElement ddAddr = driver.findElement(By.id("novotel"));
//          Select s =new Select(ddAddr);
//          List<WebElement> allOpts = s.getOptions();
//
//          for(WebElement oneOpt:allOpts)
//          System.out.println(oneOpt.getText());


            //print all the options without duplicate
//          WebElement ddAddr = driver.findElement(By.id("novotel"));
//          Select s = new Select(ddAddr);
//          List<WebElement> allOpts = s.getOptions();
//
//          Set<String> st = new HashSet<>();
//
//          for(WebElement oneOpt:allOpts)
//          {
//                String text = oneOpt.getText();
//                st.add(text);
//          }
//
//          for(String oneText:st)
//          {
//                System.out.println(oneText);
//          }


            //Print options without Duplicate + Order should be maintained
//          WebElement ddAddr = driver.findElement(By.id("novotel"));
//          Select s =new Select(ddAddr);
//          List<WebElement> allOpts = s.getOptions();
//
//          Set<String> st = new LinkedHashSet<>();
//
//          for(WebElement oneOpt:allOpts)
//          {
//                String text = oneOpt.getText();
//                st.add(text);
//          }
//
//          for(String oneText:st)
//          {
//                System.out.println(oneText);
//          }
```

```
            //Print options without Duplicate + Sorted Order
            WebElement ddAddr = driver.findElement(By.id("novotel"));
            Select s =new Select(ddAddr);
            List<WebElement> allOpts = s.getOptions();

            Set<String> st = new TreeSet<>();

            for(WebElement oneOpt:allOpts)
            {
                  String text = oneOpt.getText();
                  st.add(text);
            }

            for(String oneText:st)
            {
                  System.out.println(oneText);
            }

      }
}
```

Output:

as it is	HashSet	LinkedHashSet	TreeSet
Idly Biryani Vada Dosa Pani Puri Roti Noodles Jamoon Idly Chai	Vada Roti Pani Puri Jamoon Biryani Dosa Idly Chai Noodles	Idly Biryani Vada Dosa Pani Puri Roti Noodles Jamoon Chai	Biryani Chai Dosa Idly Jamoon Noodles Pani Puri Roti Vada

---
22-10-2020

Dynamic Dropdown / Custom Dropdown

In this kind of dopdown the Options are changing.....
To create it we use tags like input, div, ul, li, a ......

Q. How to handle dynamic dropdown ???

- Since the options are inspectable, we use findElement(), findElements(), sendkeys(), click()
- Since it is not created by select tag, hence we can't use Select class and its methods. If we use, we get UnexpectedTagNameException.

```
package qsp;

import java.util.List;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.support.ui.Select;

public class HandlingDynamicDropdown {

	public static void main(String[] args) throws InterruptedException
{

		System.setProperty("webdriver.chrome.driver",
"./drivers/chromedriver.exe");
		ChromeOptions co = new ChromeOptions();
		co.addArguments("--disable-notifications");
		WebDriver driver = new ChromeDriver(co);
		driver.manage().window().maximize();
		driver.get("https://www.cleartrip.com/");
		driver.manage().timeouts().implicitlyWait(20,
TimeUnit.SECONDS);
	//	Thread.sleep(3000);
		WebElement ddAddr = driver.findElement(By.id("FromTag"));
		ddAddr.sendKeys("del");
```

```
            Thread.sleep(6000);

//          ddAddr.sendKeys(Keys.DOWN);
//          Thread.sleep(3000);
//          ddAddr.sendKeys(Keys.DOWN);
//          Thread.sleep(3000);
//          ddAddr.sendKeys(Keys.ENTER);


            List<WebElement> allOptions =
driver.findElements(By.xpath("//a[contains(@id,'ui-id')]"));
            System.out.println(allOptions.get(0).getText());
            Thread.sleep(3000);
            allOptions.get(allOptions.size()-1).click();

            //System.out.println(allOptions.size());

//          System.out.println(allOptions.get(0).getText());

//          for(WebElement op:allOptions)
//          {
//                System.out.println(op.getText());
//          }

//          for(int i = 0;i<allOptions.size();i++)
//          {
//                WebElement oneOpt = allOptions.get(i);
//                String text = oneOpt.getText();
//                System.out.println(text);
//          }

      }
}
```

Assignment-1:

---

1. Go to Flipkart.com and type laptops in search textbox
2. Count and print the options of the dropdown and select 2nd option
3. In Search Results page, Apply filter
   - Max Price - 50000
   - Processor - Core i3
   - Brand - Dell
   - OS - Windows 10
4. Print the RAM and Price Details of the First laptop

```
package qsp;

import java.util.List;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Assignment1 {

	public static void main(String[] args) throws InterruptedException
{
		System.setProperty("webdriver.chrome.driver",
"./drivers/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://www.flipkart.com/");

		driver.manage().timeouts().implicitlyWait(20,
TimeUnit.SECONDS);

	driver.findElement(By.xpath("//button[text()='✕']")).click();

		WebElement ddAddr1 =
driver.findElement(By.xpath("//input[@class='LM6RPg']"));
		ddAddr1.sendKeys("laptops");

	   List<WebElement> allOptions =
driver.findElements(By.xpath("//li[@class='_1va75j']"));
		System.out.println(allOptions.size());

		for(WebElement oneOpt:allOptions)
		{
			System.out.println(oneOpt.getText());
		}

		Thread.sleep(3000);
		allOptions.get(1).click();

		Thread.sleep(3000);

	driver.findElement(By.xpath("//div[@class='_1YoBfV']/descendant::
option[@class='OMc8Rd' and @value='50000']")).click();
```

```
            Thread.sleep(3000);
            driver.findElement(By.xpath("//div[text()='Core
i3']/preceding-sibling::div[@class='_1p7h2j']")).click();

            Thread.sleep(3000);

      driver.findElement(By.xpath("//div[text()='Dell']/preceding-sibli
ng::div[@class='_1p7h2j']")).click();

            Thread.sleep(3000);
            driver.findElement(By.xpath("//div[text()='Operating
System']")).click();

            driver.findElement(By.xpath("//div[text()='Windows
10']/preceding-sibling::div[@class='_1p7h2j']")).click();


            List<WebElement> allPrizes =
driver.findElements(By.xpath("//div[@class='_1vC4OE _2rQ-NK']"));
            //System.out.println(allPrizes.size());
            System.out.println(allPrizes.get(0).getText());



            List<WebElement> allRams =
driver.findElements(By.xpath("//li[contains(text(),'RAM')]"));
            //System.out.println(allRams.size());
            System.out.println(allRams.get(0).getText());
      }
}
```

Assignment-2:

---

1. Go to google.com and type 'Qspiders'
2. Count and print all the options
3. If 'Qspiders Review' option is present, then click on it

```
package qsp;

import java.util.List;

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

```
public class Assignment2 {

	public static void main(String[] args) throws InterruptedException
{
		System.setProperty("webdriver.chrome.driver",
"./drivers/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://www.google.com/");

		WebElement Addr =
driver.findElement(By.xpath("//input[@title='Search']"));
		Addr.sendKeys("Qspiders");

		Thread.sleep(3000);
		List<WebElement> allOptions =
driver.findElements(By.xpath("//ul[@class='erkvQe']/descendant::li[con
tains(@class,'sbct')]"));
		System.out.println(allOptions.size());


		Thread.sleep(3000);

		for(WebElement everyOp:allOptions)
		{
			System.out.println(everyOp.getText());
		}


		for(WebElement everyOp:allOptions)
		{
			if(everyOp.getText().equals("qspiders reviews"))
			{
				Thread.sleep(3000);
				WebElement ReqOpt =
driver.findElement(By.xpath("//b[contains(text(),'reviews')]"));
				ReqOpt.click();
			}
		}
	}
}
```

26-10-2020

Handling Keyboard and Mouse Actions

Action class :

- Used to handle keyboard and mouse actions
- Imported from org.openqa.selenium.interactions package
- Actions class constructor takes webDriver References or Browser class reference as its argument.
- We handle mouse and KeyBoard Actions by using non-static methods of Actions class.
- Also Note for every method of Actions Class, we need to compulsorly use .perform() at the end.

- Robot Class is used to handle keyboard and muse Actions
- Its imported from java.awt.package
- robot class constructor throws a compile time exception called AWTException
- Robot class uses keyPress method in which KeyEvent class is present
- In KeyEvent Class all the keyboard keys are stored as virtual keys and since they are static final variables , we can access directly through classname.VariableName (keyEvent.VK_T)
- Also remember to release a key after it is presses, so that it will be released virtually

1. Handling mouse hover action - moveToElement(addr).perform()
2. a) Perform Right Click - contextClick(addr).perform()
   b) Handle Right Clicked Options - Robot Class - keyPress/KeyRelease
3. Perform Double Click - doubleClick(addr).perform()
4. Perform Drag and Drop action - dragAndDrop(fromAddr, toAddr).perform()

```
package qsp;

import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;
```

```
public class HandlingMouseAndKeyboardAction {

	public static void main(String[] args) throws InterruptedException,
AWTException {

		//Mouse Hover Actions

//		System.setProperty("webdriver.chrome.driver",
"./drivers/chromedriver.exe");
//		WebDriver driver = new ChromeDriver();
//		driver.manage().window().maximize();
//		driver.get("https://www.flipkart.com/");
//
//		Thread.sleep(5000);
//
	driver.findElement(By.xpath("//button[text()='X']")).click();
//
//		WebElement elec =
driver.findElement(By.xpath("//span[text()='Electronics']"));
//		Thread.sleep(5000);
//		Actions a = new Actions(driver);
//		a.moveToElement(elec).perform();
//
//		Thread.sleep(3000);
//
//		driver.findElement(By.xpath("//a[text()='Power
Banks']")).click();


		//Performing Double click Actions

//		System.setProperty("webdriver.chrome.driver",
"./drivers/chromedriver.exe");
//		WebDriver driver = new ChromeDriver();
//		driver.manage().window().maximize();
//		driver.get("https://demo.actitime.com/login.do");
//
//		Thread.sleep(3000);
//
//		WebElement input = driver.findElement(By.id("username"));
//		input.sendKeys("admin");
//
//		Actions a=new Actions(driver);
//		a.doubleClick(input).perform();
```

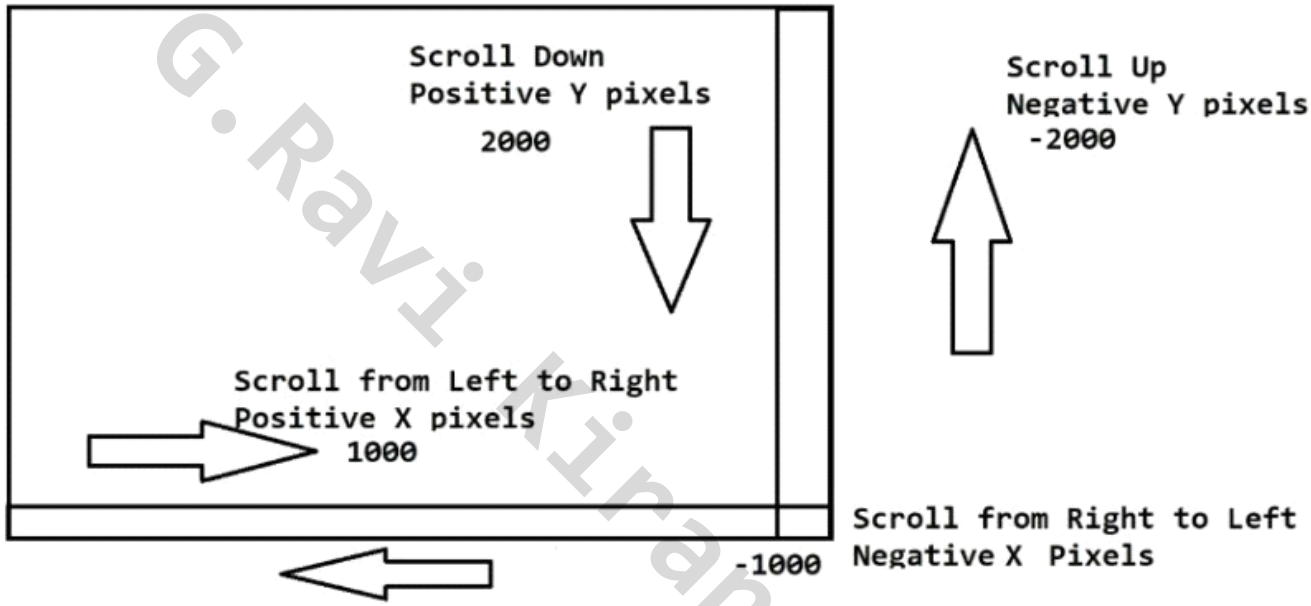
```
            //Performing Drag and Drop Actions

//          System.setProperty("webdriver.chrome.driver",
"./drivers/chromedriver.exe");
//          WebDriver driver = new ChromeDriver();
//          driver.manage().window().maximize();
//
      driver.get("http://www.dhtmlgoodies.com/scripts/drag-drop-custom/
demo-drag-drop-3.html");
//
//          Thread.sleep(5000);
//
//          WebElement src = driver.findElement(By.id("box7"));
//          WebElement dest = driver.findElement(By.id("box107"));
//
//          Actions ac = new Actions(driver);
//          ac.dragAndDrop(src, dest).perform();


            //Perform Right Click
            //Handle right clicked options

            System.setProperty("webdriver.chrome.driver",
"./drivers/chromedriver.exe");
            WebDriver driver = new ChromeDriver();
            driver.manage().window().maximize();

      driver.get("file:///C:/Users/Ravi%20Kiran/Downloads/Programs/.htm
l/demo.html");

            Thread.sleep(3000);
            WebElement google = driver.findElement(By.id("i2"));

            Actions ac = new Actions(driver);
            ac.contextClick(google).perform();

            Robot r = new Robot();
            r.keyPress(KeyEvent.VK_T);
            r.keyRelease(KeyEvent.VK_T);

      }
}
```

27-10-2020

Scrolling Down a WebPage

Q. How do we scroll down ?

A. We use javascript



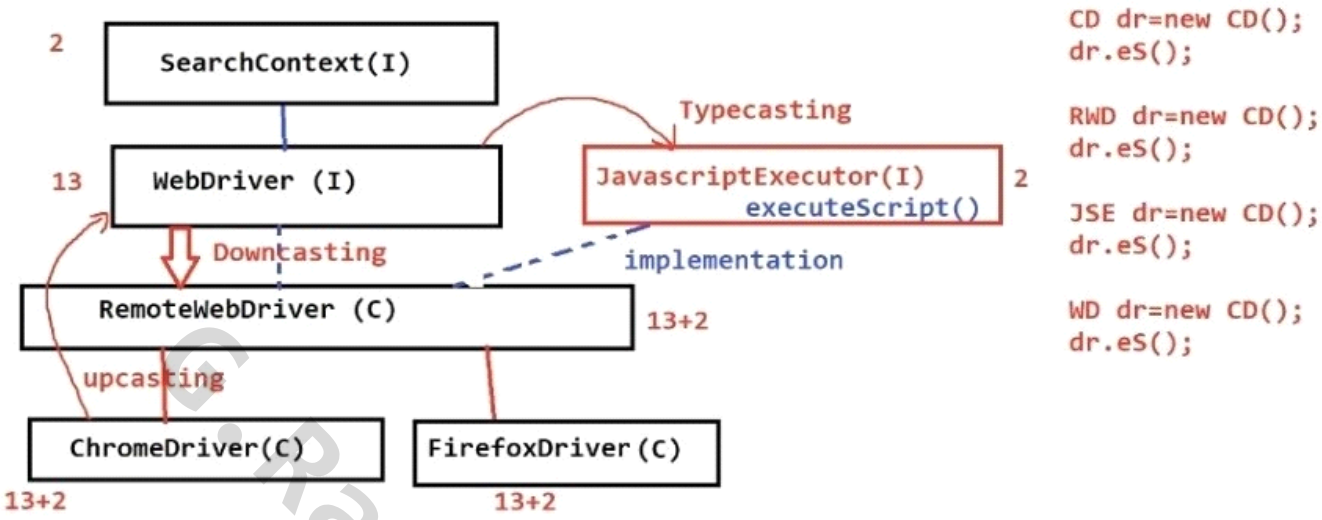
Q. How to scroll down manually???

1. Go to the required webpage and Press ctrl shift I or F12/ Fn F12, Developer Tools will be displayed
2. Click on the console tab and write the javascript as follows
   Syntax: window.scrollBy(x axis, y axis)
   Example: window.scrollBy(100, 800)

Q. How to scroll down through Automation(Selenium webDriver)???

A. In Selenium, we use executeScript() of JavascriptExecutor interface to execute the required javascript automatically...

JavascriptExecutor Diagram



- Since we are upcasting our driver to Webdriver interface, executeScript() method is not accessible.
- Hence to access it, we either Typecast from WebDriver interface to JavascriptExecutor interface or Downcast from WebDriver inerface to RemoteWebDriver class.

Q. How to scroll down to a particular element ???

A.

- If we want to scroll down to a particular element, then we get the location of the element(location means X axis and Y axis) by using getLocation() method of WebElemnt interface.
- getLocation() returns Point(x axis, y axis) of the element.
- window.scrollBy+location of Element

Example:

```
WebElement ele =driver.findElement(By.xpath("//h2[text()='Selenium Level Sponsors']"));
Point loc = ele.getLocation();
JavascriptExecutor jse = (JavascriptExecutor)driver;
jse.executeScript("window.scrollBy"+loc);
```

```java
package qsp;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Point;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;

public class ScrollingDown {

	public static void main(String[] args) throws InterruptedException
{
		System.setProperty("webdriver.chrome.driver",
"./drivers/chromedriver.exe");
		//upcasting from ChromeDriver class to WebDriver Interface
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://www.selenium.dev/downloads/");

		Thread.sleep(3000);

		WebElement ele =
driver.findElement(By.xpath("//h2[text()='Selenium Level Sponsors']"));
		Point loc = ele.getLocation();
		System.out.println(loc);

		//Through typeCasting
		JavascriptExecutor jse = (JavascriptExecutor)driver;
		jse.executeScript("window.scrollBy"+loc);

		//Through Downcasting
//		RemoteWebDriver rwd = (RemoteWebDriver)driver;
//		rwd.executeScript("window.scrollBy"+loc);
	}
}
```

28-10-2020

Handling Disabled Elements

How do we handle Disabled elements ?

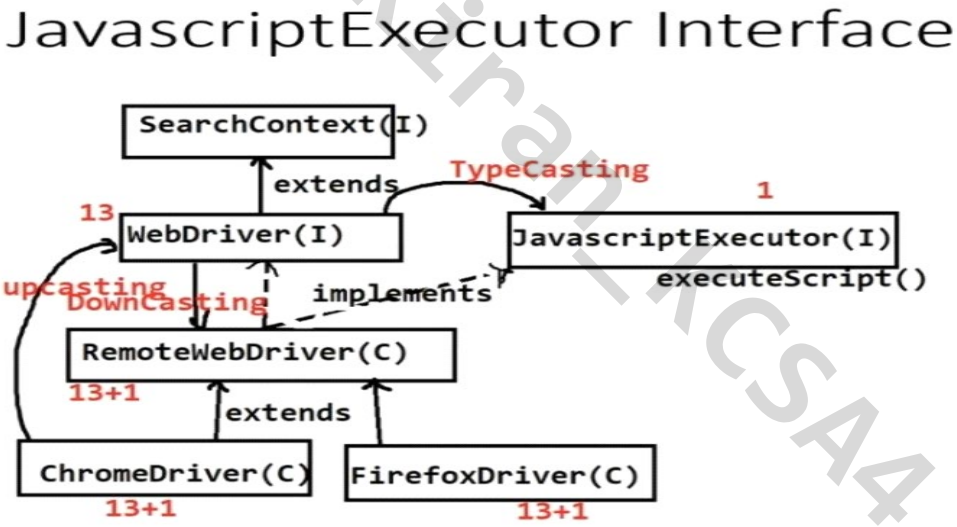
- By using Javascript

First we need to handle it manually and then we pass the Javascript into executeScript(script) method of JavascriptExecutor Interface

Handling it Manually :-

1. Go to the required webpage and press F12 key(in some laptops Fn + F12), Developers tool will be displayed
2. Go to the console tab and start writing the below Javascript and press Enter

Syntax: document.getElementByID('id value').value='text'
Example: document.getElementByID('i2').value='manager'



- As a selenium standard, we always upcast our browser classes to WebDriver Interface.
- Hence executeScript() is not accessible through WebDriver reference variable(driver).
- So to access executeScript(),
  1) we either typecast from
  WebDriver interface to JavascriptExecutor interface

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

```
      j.executeScript("document.getElementById('i2').value='manager'");
                        (or)
      2) Downcast from
      WebDriver Interface to RemoteWebDriver Class
      WebDriver driver = new ChromeDriver();
      RemoteWebDriver rwd = (RemoteWebDriver)driver;
      rwd.executeScript("document.getElementById('i2').value='manager'"
);

package qsp;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class HandlingDisabledElements {

      public static void main(String[] args) throws InterruptedException
{

      System.setProperty("webdriver.chrome.driver","./drivers/chromedri
ver.exe");
            WebDriver driver = new ChromeDriver();
            driver.manage().window().maximize();

      driver.get("file:///C:/Users/Ravi%20Kiran/Downloads/Programs/.htm
l/DisabledElement.html");
            Thread.sleep(2000);

            WebElement untb = driver.findElement(By.id("i1"));
            if(untb.isEnabled()) {
                  System.out.println("Untb is Enabled");
                  untb.sendKeys("admin");
            }
            else {
                  System.out.println("Untb is not Enabled");
                  JavascriptExecutor j = (JavascriptExecutor)driver;

      j.executeScript("document.getElementById('i1').value='admin'");
            }
            Thread.sleep(3000);
            WebElement pwtb = driver.findElement(By.id("i2"));
            if(pwtb.isEnabled()) {
                  System.out.println("Pwtb is Enabled");
```

```
                pwtb.sendKeys("manager");
            }
            else {
                System.out.println("Pwtb is not Enabled");
                JavascriptExecutor j = (JavascriptExecutor)driver;
      j.executeScript("document.getElementById('i2').value='manager'");
            }
}}
```

WebElemental Methods Element Level

S.No	Method Name	ReturnType
1	click()	void
2	clear()	void
3	sendKeys(charSequence)	void
4	getTagname()	String
5	getAttribute(String attrName)	String
6	getText()	String
7	getLocation()	Point
8	getRect()	Rectangle
9	getCssValue(String cssName)	String
10	getSize()	Dimesnion
11	getScreenshotAs(Output Type)	File
12	isDisplayed()	boolean
13	isEnabled()	boolean
14	isSelected()	boolean
15	findElement(By arg)	WebElement
16	findElements(By arg)	List<WebElement>
17	submit()	void

Method Name	Method Use
click()	Used to click on any element
clear()	Used to clear the content of the element
sendKeys(charSequence)	Used to send charSequence into the element
getTagname()	Used to get the tagName of the element
getAttribute(String attrName)	USed to get a particular attribute value of the element by giving the attribute name
getText()	Used to get the text of the element
getLocation()	Used to get Location of the element (means X pixel, Y pixel)
getRect()	Used to get X Axis, Y Axis, height, width
getCssValue(String cssName)	Used to get a particular CSS attribute value of the element by giving the css attribute name
getSize()	Used o get the size of element(means length and breadth)
getScreenshotAs(Output Type)	Used to get the screenshot of a particular element
isDisplayed()	Used to verify if an element is present on the webpage
isEnabled()	Used to verify if an element is enabled or diabled(enabled means ready to interact)
isSelected()	Used to verify if an element is selected or not
findElement(By arg)	Used to find a particular element within the current element
findElements(By arg)	Used to find all matching elements within the current element
submit()	Used like click() method but only can be used on buttons with type='submit'

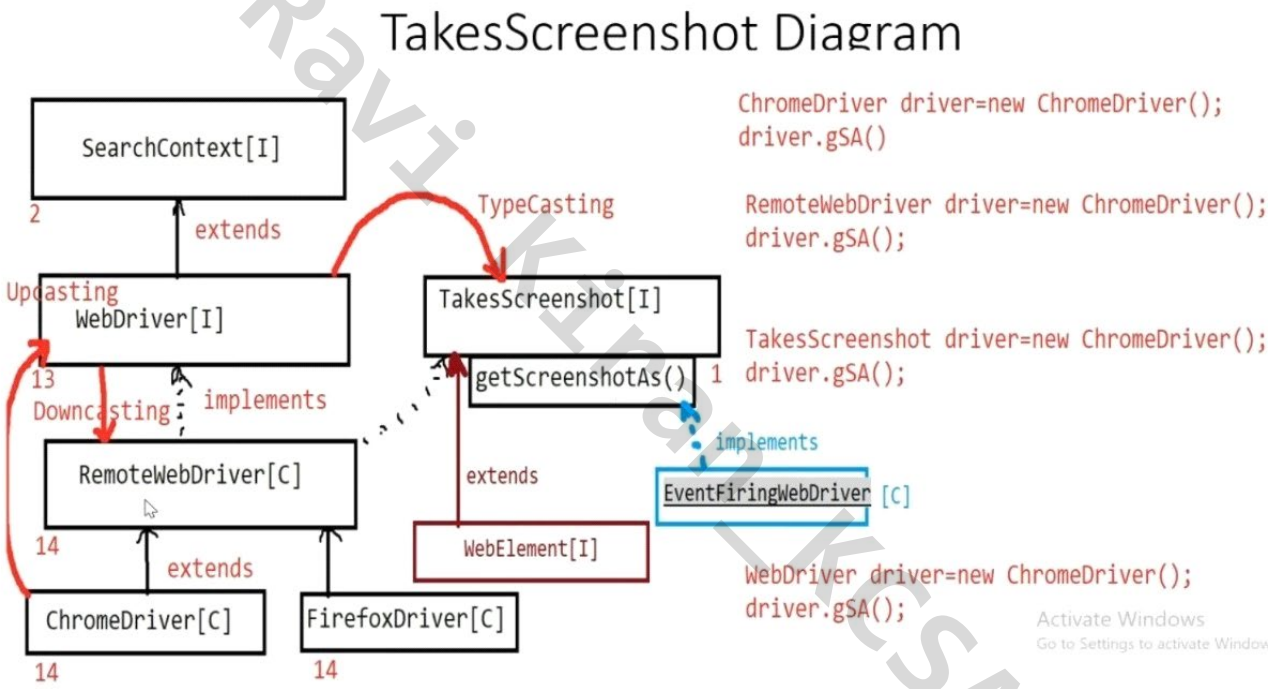
```
package qsp;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class WebElementMethods {
      public static void main(String[] args) throws InterruptedException
{
      System.setProperty("webdriver.chrome.driver",
                                        "./drivers/chromedriver.exe");
            WebDriver driver = new ChromeDriver();
            driver.manage().window().maximize();
            driver.get("https://demo.actitime.com/login.do");
            Thread.sleep(3000);
WebElement chkbox = driver.findElement(By.id("keepLoggedInCheckBox"));
chkbox.click();
            Thread.sleep(3000);
            if(chkbox.isDisplayed())
            {
                  System.out.println("Checkbox is Displayed");
                  if(chkbox.isEnabled())
                  {
                        System.out.println("Checkbox is Enabled");
                        if(chkbox.isSelected())
                        {
                              System.out.println("It is already selected,
                                    dont  touch");
                        }
                        else
                        {
                              System.out.println("It is not selected, click
                              now");
                              chkbox.click();
                        }
                  }
                  else
                  {
                        System.out.println("Checkbox is not Enabled");
                  }
            }
            else
            {
                  System.out.println("Checkbox is not Displayed");
            }
}}
```

29-10-2020

```
package qsp;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class WebElementMethods {
	public static void main(String[] args) throws InterruptedException
{

	System.setProperty("webdriver.chrome.driver",
				"./drivers/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://demo.actitime.com/login.do");

		Thread.sleep(2000);

		WebElement untb = driver.findElement(By.id("username"));
		untb.sendKeys("admin");
		Thread.sleep(2000);
		untb.clear();
		System.out.println("Tag: "+untb.getTagName());
		System.out.println("Attribute:
"+untb.getAttribute("class"));
	WebElement header = driver.findElement(By.id("headerContainer"));
		System.out.println("Text: "+header.getText());

		System.out.println("Size: "+untb.getSize());

		System.out.println("Width: "+untb.getRect().width);
		System.out.println("Height: "+untb.getRect().height);

		System.out.println("Location: "+untb.getLocation());

		System.out.println("X axis: "+untb.getRect().x);
		System.out.println("Y axis: "+untb.getRect().y);

	System.out.println("Font-Size: "+untb.getCssValue("font-size"));
	}
}
```

Output:
Tag: input
Attribute: textField
Text: Please identify yourself
Size: (214, 32)
Width: 214
Height: 32
Location: (644, 294)
X axis: 644
Y axis: 294
Font-Size: 14px

TakesScreenshot



- We use getScreenshotAs() method of TakeScreenshot Interface to take a screnshot.
- getScreenshotAs() is given implementation in RemoteWebDriver Class and it is extended in browser classes.
- But as a Selenium standard we always upcast our browser object to WebDriver Interface, [WebDriver driver=new ChromeDriver();]
- As a result getScreenshotAs() is hidden and inaccessible.

Hence, to access it,

- we either typecast from WebDriver Interface to TakesScreenshot Interface

```
TakesScreenshot ts = (TakesScreenshot)driver;
```

(or)

- We Downcast from WebDriver Interface to RemoteWebDriver Class

```
RemoteWebDriver rwd = (RemoteWebDriver)driver;
```

(or)

- Since, EventFiringWebDriver Class is also Implementing it, we can Create object and access it

```
EventFiringWebDriver e = new EventFiringWebDriver(driver);
```

Explaining TakesScreenshot program in interview

---

1. Upcasting Browser class to WebDriver Interface
2. Typecasting from WebDriver to TakesScreenshot interface
3. Getting the screenshot by calling getScreenshotAs() which takes argument OutputType which is file type
4. We create an object of File Class and in constructor we pass three things
   a. Path where we need to store the screenshot
   b. FileName with which we want to store
   c. Extension (.png / .jpg)
5. Using Third party class called 'Files' which is imported from com.google.common.io package, we copy the obtained screenshot and pass it in the specified path.
6. Files.copy() throws a compile time exception called IOException

```
package qsp;

import java.io.File;
import java.io.IOException;

import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.support.events.EventFiringWebDriver;

import com.google.common.io.Files;
```

```
public class TakingScreenshots {

	public static void main(String[] args) throws IOException,
InterruptedException {
		System.setProperty("webdriver.chrome.driver",
				"./drivers/chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();
		driver.get("https://www.google.com/");

		//1. Through Typecasting
//		TakesScreenshot ts = (TakesScreenshot)driver;
//		File ss = ts.getScreenshotAs(OutputType.FILE);
//		//path+name+extension
//		File f = new File("C:<\\Users\\Ravi Kiran\\Downloads>
				\\\\screenshots\\Google.png");
//		Files.copy(ss, f);


		//2. Through Downcasting
//		RemoteWebDriver rwd = (RemoteWebDriver)driver;
//		File src = rwd.getScreenshotAs(OutputType.FILE);
//		File dest = new File("C:<\\Users\\Ravi Kiran\\Downloads>
				\\\\screenshots\\Image.png");
//		Files.copy(src, dest);

		//3. Through EventFiringWebDriver Class
//		EventFiringWebDriver ef = new EventFiringWebDriver(driver);
//		File src = ef.getScreenshotAs(OutputType.FILE);
//		File dest = new File("C:<\\Users\\Ravi Kiran\\Downloads>

	\\\\screenshots\\GImage.jpg");
//		Files.copy(src, dest);

		//Taking Screenshot of a particular Element
		Thread.sleep(3000);
		WebElement gSearchBox = driver.findElement(By.id("hplogo"));

		File src = gSearchBox.getScreenshotAs(OutputType.FILE);
		File dest = new File("C:<\\Users\\Ravi Kiran\\Downloads>

	\\\\screenshots\\GLogo.jpg");
		Files.copy(src, dest);
	}
}
```

30-10-2020

POPUPS

Popups usually appear on the webpage to give some warning or get some data from users or get confirmation from users.....

These are of two categories -
1. Web-Based
2. Window-Based

WEB-BASED
1. Javascript popup
2. Hidden-Division popup
3. Browser Notification popup

1. Javascript popup - Created by using Javascript Language, Hence the name
1.a Alert - Has only one button - OK button
1.b Confirmaton - Has two buttons - OK CANCEL buttons

Alert can be created by javascript -> alert("message")
Confirmation can be created by javascript -> confirm("message")

Characteristics of Javascript Popup
1. Is it colorful ? NO
2. Is it movable ? NO
3. Is it Inspectable? NO
4. Until we handle it, we can't do futher action on the webpage

How to Handle Javascript Popup ??
- First we need to switch our control from WebPage to alert Popup
- Alert al=driver.switchTo().alert();
- alert() is coming from TargetLocator Class
- alert() returntype is Alert Interface

In Alert interface, we have the following abstract methods :-
1. a1.accept() - Used to click on OK button
2. a1.dismiss - Used to click on CANCEL buton
3. a1.getText() - Used to get the message on the popup
4. a1.sendKeys() - Used to type input on the popup

Note: If there is no alert popup on the webpage, but still we are trying to switch to it and handle it, then we get NoAlertPresentException

- **for Alert Popup**

```
<html>
<body>
<h2>JavaScript Alert</h2>
Username<input type='text'><br>
Password<input type='password'><br>
<button onclick='myFunction()'>Login</button>
<script>
function myFunction() {
  alert('I am an alert box!');
}
</script>
</body>
</html>
```

- **for Confirmation Popup**

```
<html>
<body>
<p>Click the button to display your Love.....</p>
<button onclick="myFunction()">I Love You</button>
<p id="demo">You Pressed OK...Now you have to Love..haha..!</p>
<script>
function myFunction() {
   var txt;
   var r = confirm("Will You Marry me ???");
   if(r == true) {
        txt = "You Pressed Ok, so you should marry...." ;
    } else {
       txt = "You Pressed Cancel, Now you are safe...." ;
    }
    document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```

```
package qsp;

import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class HandlingPopups {
```

```
	public static void main(String[] args) throws InterruptedException
{

		//Handling Alert Popup
//		System.setProperty("webdriver.chrome.driver",
//				"./drivers/chromedriver.exe");
//		WebDriver driver = new ChromeDriver();
//		driver.manage().window().maximize();
//
	driver.get("file:///C:/Users/Ravi%20Kiran/Downloads/Programs/.htAlert%20Popupml/.html");
//		Thread.sleep(3000);
//
	driver.findElement(By.xpath("//button[text()='Login']")).click();
//		Thread.sleep(3000);
//		Alert al = driver.switchTo().alert();
//		System.out.println(al.getText());
//		al.accept();
//		//al.dismiss();

		//Handling Confirmation Popup
		System.setProperty("webdriver.chrome.driver",
				"./drivers/chromedriver.exe");

		WebDriver driver = new ChromeDriver();
		driver.manage().window().maximize();

	driver.get("file:///C:/Users/Ravi%20Kiran/Downloads/Programs/.html/Confirmation%20Popup.html");
		Thread.sleep(3000);
driver.findElement(By.xpath("//button[text()='I Love You']")).click();

		Alert al = driver.switchTo().alert();
		Thread.sleep(3000);
		System.out.println(al.getText());
//		//OK button
//		al.accept();

		//CANCEL button
		al.dismiss();
	}
}
```

31-10-2020
---

2. Hidden Division Popup

- Characteristics:
  1. Is it Colorful? Yes
  2. Is it Movable? No
  3. Is it Inspectable? Yes

How to handle it ?

- Since it is inspectable, we use fndElement(), findElements(), click(), sendKeys() etc to handle it.

3. Browser Notification Popup

- To avoid getting browser notification popup, we need to change few browser settings before opening the browser.

- In Selenium WebDriver, for every browser we have a browser setings class
  1. For Chrome browser --> ChromeOptions Class
  2. For Firefox browser --> FirefoxOptions Class etc

- First we need to create an object of this class and we use a method called addArguments("settings")

```
FirefoxOptions fo = new FirefoxOptions();
fo.addArguments("--disable-notifications");
fo.addArguments("start-maximized");
```

- Then while opening the browser, we need to pass FirefoxOptions Object reference as parameter for FirefoxDriver class

```
//Open the browser with the changes
WebDriver driver = new FirefoxDriver(fo);
```

```
package qsp;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class HandlingPopups {

	public static void main(String[] args) throws InterruptedException
{
```

```
            //Handling Hidden Division Popup
            //Handling Browser Notifications Popup
            System.setProperty("webdriver.chrome.driver",
                                    "./drivers/chromedriver.exe");
            ChromeOptions co = new ChromeOptions();
            co.addArguments("--disable-notifications");
            co.addArguments("start-maximized");
            //Open Chrome browser with the settings
            WebDriver driver = new ChromeDriver(co);

            //driver.manage().window().maximize();
            driver.get("https://www.cleartrip.com/");

            Thread.sleep(3000);
            driver.findElement(By.id("DepartDate")).click();
            Thread.sleep(3000);
driver.findElement(By.xpath("//td[@datamonth='10']//a[text()='4']")).c
lick();
      }
}
```

WINDOW-BASED POPUP

- Selenium can't automate window based applications
- Hence, we take help from third party tools like Robot Class or AutoIT
- Characteristics:
    1. It is Colorful
    2. It is movable
    3. It is not Inspectable
- Examples:
    4. FileUpload
    5. FileDownload
    6. Print

- File Upload, How to handle ?
    a. Trick: use sendKeys(complete path of the file ehich has to be uploaded)
    b. org.openqa.selenium.InvalidArgumentException
    c. Shift+RightClick --> Copy as path
    d. Proper Way: AutoIT

- File Download, How to handle ?
    a. Trick: use Robot Class
    b. Proper Way: AutoIT

Note: No need to handle download popup in chrome browser, because it will not show popup but it auto-downloads any file.

```
package qsp;

import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class HandlingWindowPopups {

	public static void main(String[] args) throws InterruptedException,
AWTException {


System.setProperty("webdriver.gecko.driver","./drivers/geckodriver.exe
");
		WebDriver driver = new FirefoxDriver();
		driver.manage().window().maximize();
		driver.get("https://www.selenium.dev/downloads/");

		Thread.sleep(3000);

	driver.findElement(By.xpath("//td[text()='Java']/..//a[text()='Do
wnload']")).click();
		Thread.sleep(3000);
		Robot r = new Robot();
		r.keyPress(KeyEvent.VK_DOWN);
		Thread.sleep(3000);
		r.keyPress(KeyEvent.VK_ENTER);

		r.keyRelease(KeyEvent.VK_DOWN);
		r.keyRelease(KeyEvent.VK_ENTER);
	}
}
```

02-11-2020

AutoIT

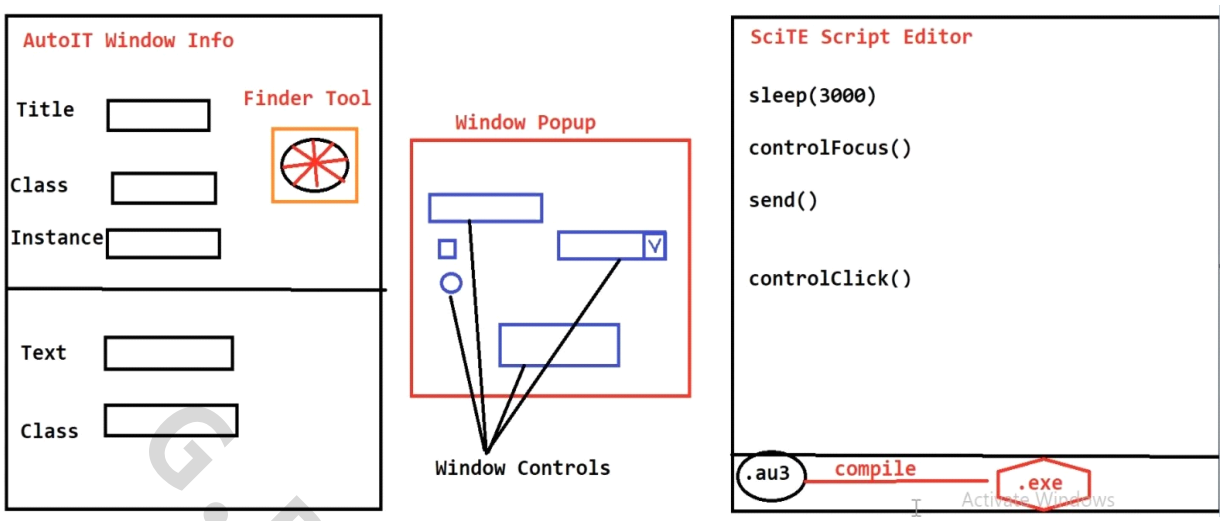
- AutoIT is a free third party tool, used to automate window based applications.
- It can be downloaded from https://www.autoitscript.com/site/autoit/downloads/
  Download AutoIT Zip file
- In AutoIT we have two tools :-
  1. AutoIT Window Info
  2. SciTe Script Editor
- In AutoIT window info, we have a finder tool, which we can drag and drop on the required control. We will get all the attributes details related of that control in Window Info tool.
- With the attributes details, we start writing code in SciTe Script editor in Scripting language.
- Once completed, we save as .au3 format...
- Then we compile(In Editor, Go to Tools -> Compile)
- The au3 autoIT file to get an executable file(.exe)
- Double click on the exe file and click on the window popup to execute manually.

Note:

Control ID = combination of Class and Instance
Class='Button'
ID='10'
Control Id = 'Button10'

How to integrate autoIT script in our selenium script ???

- All the automation of window popup will be done through autoIT and we get an executable file. But we just need to execute that. Hence we use 'exec("full path of exe file")' of Runtime Class to execute(run) it.




```
package qsp;
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import java.io.IOException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class HandlingWindowPopups {
      public static void main(String[] args) throws InterruptedException,
AWTException, IOException {
            System.setProperty("webdriver.gecko.driver",
                                    "./drivers/geckodriver.exe");
            WebDriver driver = new FirefoxDriver();
            driver.manage().window().maximize();
            driver.get("https://www.selenium.dev/downloads/");

            Thread.sleep(3000);
            //Getting Window Popup(Print popup)
            Robot r = new Robot();
            r.keyPress(KeyEvent.VK_CONTROL);
            r.keyPress(KeyEvent.VK_P);
            r.keyRelease(KeyEvent.VK_CONTROL);
            r.keyRelease(KeyEvent.VK_P);

            Thread.sleep(3000);
            //Executing the autoIT Script
            //Integrating AutoIT Script with our selenium script
            Runtime.getRuntime().exec("E:\\Qspiders
Selenium\\autoIT\\Print.exe");
}}
```