Selenium

Am currently using jdk 24

JDK and eclipse should be installed, go to selenium website and download the selenium webdriver.

Selenium can be configured with different languages like java, c#, ruby, python, js(node).

Am going with Selenium using java (This webdriver will be in zip file.)

In **Selenium**, a **JAR file** (Java ARchive) is a package file format used to aggregate many Java class files and associated metadata and resources (like libraries, images, etc.) into a single file for distribution.

Done with JDK, Eclipse, Selenium web driver .jar files - now time to configure jar files.

If I wanted to work on with web driver concepts we have add web driver jar files into JRE system library.

To add web drivers…!

1. Right click on the project – click on properties – select java build path.
2. java build path (select libraries) , by deafault JRE libraries will be there ; to work with selenium add selenium web driver to that JRE libraries and get configured. For every new project we have to add selenium jar files newly to JRE to work with selenium.
3. If jar files are in eclipse choose add jars , if jars files are available in local machine choose add external jars. The jar file wch we added into JRE will get configure to our selenium project.
4. Add external jar(Selenium webdriver where we have extracted) – after adding apply – ok.
5. After adding it – new folder is created by default called referenced libraries.

Packages will be in source folder.

Src – new – package

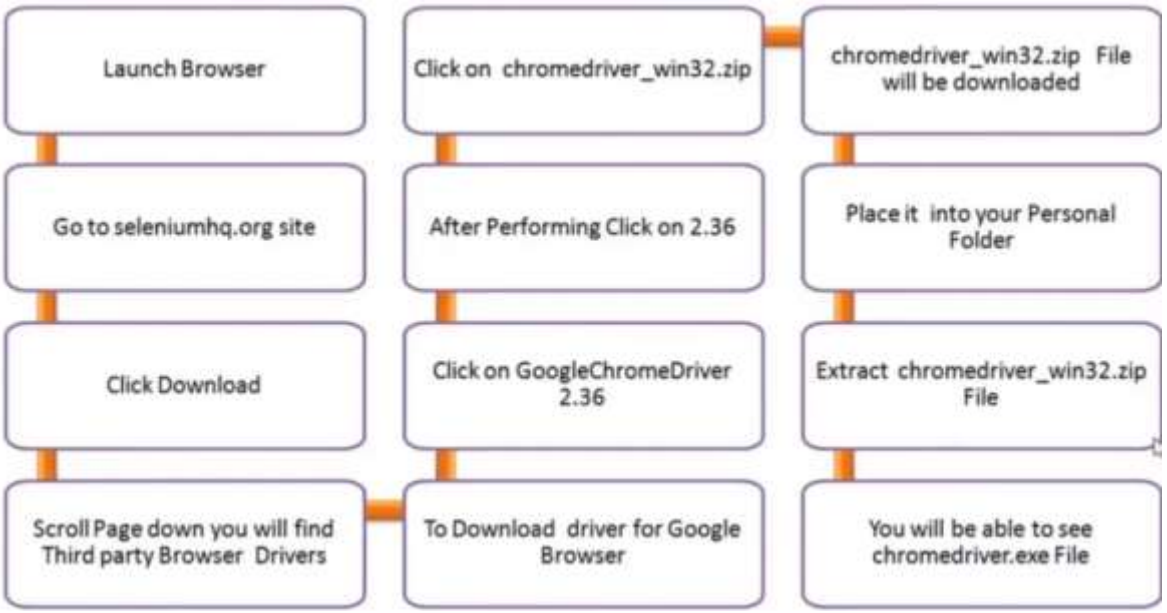
Inside the package create the java classes.

```
package Homepage;
public class Login {

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

Installing Selenium webdriver and 3rd party browser drivers :

- GGL Chrome - Chrome driver.
- Mozilla Firefox - Gecko driver.
- Opera - Opera driver.
- Internet Explorer - IE driver server.



Go to: Third Party Browser Drivers NOT DEVELOPED by seleniumhq

Download the latest version before downloading check the version of the chrome.

Done with Chrome , Gecko , IE Downloading the drivers.

When we launching the driver, we have to select .exe file location while launching it.

WebDriver Class Execution :

Call WebDriver API to perform Test : Launch Browser, Open URL, Interact with elements.

For launching the browser we have to create the object of WebDriver().

In Java (and other statically typed languages like C#):

- WebDriver is an interface provided by Selenium.
- It defines methods like get(), findElement(), quit(), etc.
- Actual browser drivers (like ChromeDriver, FirefoxDriver, etc.) are classes that implement the WebDriver interface.

Launch Browser

- WebDriver driver = new FirefoxDriver()
- A FirefoxDriver class with no parameters means that the default Firefox profile will be launched by our Java program.
- The default Firefox profile is similar to launching Firefox in safe mode (no extensions are loaded).

Open Base URL

WebDriver's get() method is used to launch a new browser session and directs it to the URL that we specify as its parameter.

Drivers

- driver.get("http://www.google.com")
- Must be valid URL with valid host name.

To Launch Chrome Browser:

```
System.setProperty(“webdriver.chrome.driver”, “Path of Chrome driver”);

System.setProperty(“webdriver.chrome.driver”,"C://Users//cheta//Downloads//SeleniumDownloads//Drivers//chromedriver.exe");

WebDriver driver=new ChromeDriver();
```

Same manner for gecko and other drivers.

A **WebDriver** in Selenium is a tool that allows us to **automate interactions with web browsers**, just like a human would — clicking buttons, entering text, navigating between pages, etc.

ChromeDriver

ChromeDriver is a standalone server that lets Selenium control the Google Chrome browser.
It translates WebDriver commands into actions in the Chrome browser using the Chrome DevTools protocol.

GeckoDriver

GeckoDriver is the link between Selenium and Mozilla Firefox.
It enables Selenium to interact with Firefox using the Gecko browser engine via the WebDriver protocol.

IEDriver (IEDriverServer)

IEDriver lets Selenium control Internet Explorer for automation and testing.
It is based on the legacy architecture of IE and is mostly deprecated in favor of modern browsers.

Close(): Used to close the active browser. Syntax: driver.close();

Quit(): This method closes all the browser that opened by WebDriver during execution/close every associated window. Syntax: driver.quit();

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.edge.EdgeDriver;

public class Launch {
    public static void main(String[] args) {
        System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");
        WebDriver driver = new EdgeDriver();
        driver.get("https://www.google.com");
        driver.quit();
    }
}
```

- **`WebDriver`** – Selenium interface to control web browsers like Chrome, Edge, Firefox.
- **`EdgeDriver`** – Selenium class that launches and automates Microsoft Edge.
- **`System.setProperty()`** – Sets the system property to specify the path of the EdgeDriver executable.
- **`new EdgeDriver()`** – Creates and launches a new Edge browser instance.
- **`driver.get()`** – Opens a given URL in the browser.
- **`driver.quit()`** – Closes all browser windows and ends the WebDriver session.

Selenium IDE, it is an integrated development environment for selenium tests. Originally developed as a Firefox add-on**.** Allows us to record, edit, & replay the test in firefox.

Key Features:

- **Record & Playback**: Automatically records user interactions with the browser.
- **No programming needed**: Best for beginners or non-programmers.
- **Browser extension**: Available for **Chrome** and **Firefox**.
- **Export tests**: Can export recorded tests into **Selenium WebDriver code** (Java, Python, etc.).
- **Assertions**: Supports adding assertions to validate elements/text.

Selenium RC(Remote Control): A server is written in java so available in all platforms. Acts as a proxy between the test and the browser. Proxy server is a system or router that provides a gateway between users and the internet.

Selenium RC server was required to run tests. Test scripts communication with this server using http commands. The server is acted acted as proxy, tanslating these commands into actions performed by web browser.

Selenium RC works by:

- Running a server that acts as a middleman.
- Test scripts send commands to this server.
- The server injects JavaScript into the browser to control it.
- Browser performs actions and sends results back through the server.

Selenium RC was replaced because:

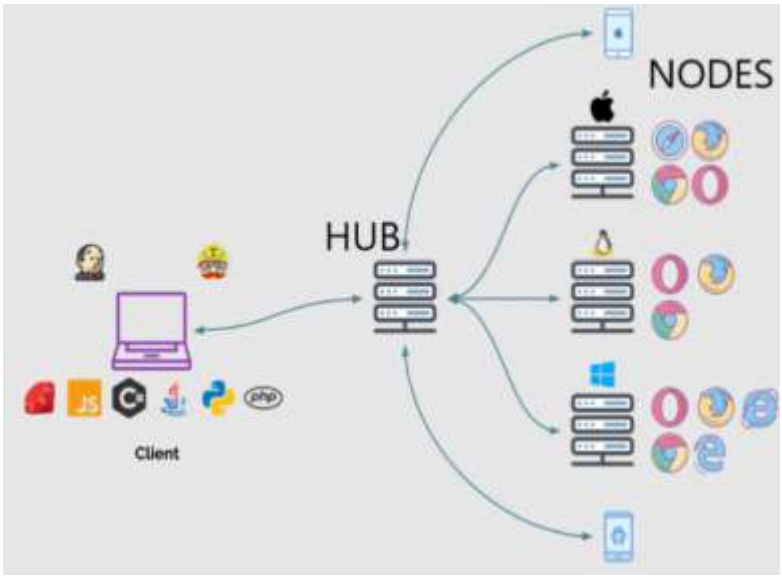
- It’s slow due to the extra server layer between tests and browser.
- Requires manual starting of the RC server.
- Uses JavaScript injection, which has limitations with modern, complex web apps.
- Less reliable and harder to maintain.
- Selenium WebDriver directly controls browsers, making it faster, simpler, and more powerful.

Limitations of SRC	Advantages of Web Driver
+ Relied on proxy server to comm with browser. + Slower due to intermediate server & single threaded execution. + Limited supports for new browser + Struggled with testing across multiple domains due to security. + Test scripts often needed freq updates.	+ Comm directly with browsers using native browser automation APIs. + Faster exe due to direct comm & supports parallel exe & multi-threading. + Handles dynamic elt & modern web. + Works well with all major browsers. + Test scripts are more stable, regular support updates & community support keep it up to date with new browser features.

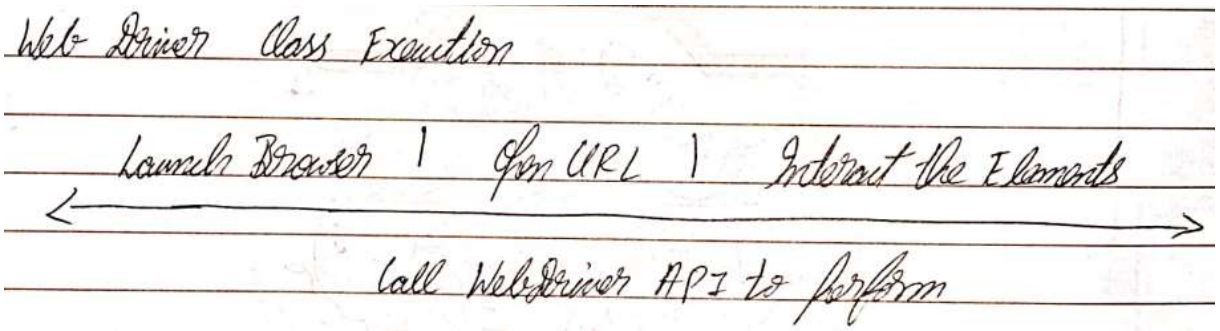
Selenium Grid helps you run many tests at the same time on different computers and browsers. It has one main server (hub) that sends tests to other computers (nodes) to run. This way, you can test your website faster on lots of browsers and devices.

It supports both Selenium RC & Web Driver scripts. Parallel execution is possible by Selenium Grid Environment.

- A hub is the main computer that controls and sends tests to other computers (called nodes) in Selenium Grid.
- A node is a computer connected to the hub that actually runs the tests. There can be many nodes, each running tests on different browsers or systems.



WebDriver is a tool that let us to write programs to **automatically control a web browser,** just like a human would. We can use it to open websites, click buttons, fill out forms, and test how a website works. It talks directly to the browser to perform actions.



Web Element, Anything i.e. present on the web page such as text box, botton etc. Which it represents an HTML element and Selenium allows us to locate & manipulate these elements programmatically. Ex: edit box, link, button (click, submit actions), check box, radio button etc.

Element locators in Selenium are ways to find elements on a web page (like buttons, text boxes, links, etc.) so you can interact with them; like clicking, typing, or reading text. Ex: id, name, classname, tagname, linktext, partiallinktext, css, xpath.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;

public class Facebook {

    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");
        WebDriver driver = new EdgeDriver();
        driver.get("https://www.facebook.com/r.php?entry_point=login");

        // Locate by name attribute
        WebElement component = driver.findElement(By.name("reg_email__"));
        component.sendKeys("chetannrevankar@gmail.com");
        driver.findElement(By.name("firstname")).sendKeys("Chetan");
        driver.findElement(By.className("_1lch")).click();
        Thread.sleep(1000);
        driver.close();
    }
}
```

- `By`: Used to locate elements.
- `WebDriver`: Interface for browser automation.
- `WebElement`: Represents a web element on the page.
- `EdgeDriver`: Class to automate Microsoft Edge browser.
- `throws InterruptedException`: Required because `Thread.sleep()` is used later.

`System.setProperty()`	Tells Selenium **where the browser driver is located**.
`findElement()`	Finds a **specific web element** on a page to interact with it.

```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.edge.EdgeDriver;

public class FaceBookLinkText {
	public static void main(String[] args) {
		System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");
	 WebDriver driver = new EdgeDriver();
	 driver.get("https://www.facebook.com/r.php?entry_point=login");

	 driver.findElement(By.id("privacy-link")).click();
	 driver.findElement(By.partialLinkText("Cookies")).click();
	 driver.close();    //Closes the parent window
	 driver.quit();     //Closes the child and parent window
	}
}
```

In Selenium WebDriver, `findElement` is used to **locate a single web element on the webpage**. You need to locate elements (like buttons, input fields, links, etc.) before interacting with them (clicking, typing, reading text).

Common Locators:

- `By.id("idValue")`: Finds by element's ID.
- `By.name("nameValue")`: Finds by name attribute.
- `By.className("className")`: Finds by CSS class.
- `By.tagName("tagName")`: Finds by HTML tag.
- `By.linkText("full link text")`: Finds by complete visible link text.
- `By.partialLinkText("partial text")`: Finds by part of the link text.
- `By.cssSelector("cssSelector")`: Finds by CSS selector.
- `By.xpath("xpathExpression")`: Finds by XPath.

Xpath is abt giving the path of the web element. XPath stands for **XML Path Language**.

Right click into attributes – copy – copy xpath.

Types of xpath:

Absolute XPath (Not recommended)

- Specifies the complete path from the root node `/html` to the element. Using attributes.

```
/html/body/div[2]/div/form/input[1]
/html[1]/body[1]
```

- **Fragile**: Breaks if the page structure changes.

Relative XPath (Recommended)

- Starts with `//`, searches from anywhere in the document. Based on position.

```
//input[@id='username']
//body
```

- **Flexible and robust**.

- `//input[@name='firstname']` is a **Relative XPath**.
- Absolute XPaths start from the root `/html/...` and specify the **entire path**.

In XPath, the `*` is a **wildcard** that **matches any node** regardless of its tag name.

WebDrivers Method:

- Get(Return type is void, i/p is str) – Used to open specified url. Syntax: driver.get("url");
- Get Current URL – To get the current url of the browser. Syntax: String Var = driver.getCurrentURL();
- Get Title – To return the title of the browser. Syntax: String Var = driver.getTitle();
- Get Text – Used to fetch inner text of the element. Syntax: element.getText();
- Get Tag Name – To get the tag name of this element.Syntax: element.getTagName();
- Get Css Value – Used to fetch CSS property value of the given element.Syntax: element.getCssValue();
- Get Attribute – To get the value of the given attribute of the element. Syntax: element.getAttribute();
- Submit – Used to perform click operation better than click(). Syntax: element.submit();
- Get Page Source – To get the HTML Page Source. Syntax: String Var = driver.getPageSource();
- Close – Used to close the active browser. Syntax: driver.close();
- Quit - This method closes all the browsers opened by webdriver during execution. Syntax: driver.quit();
- Clear – It clears the value.
- Is Enabled – It checks weather the element is in enabled state or not, return Boolean value.T/F.
- Is Selected – It checks if the element is selected ot not in the current webpage.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;

public class WebElements {
	public static void main(String[] args) {
		System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");
	 WebDriver driver = new EdgeDriver();
	 driver.get("https://www.facebook.com/login");

	 String Title = driver.getTitle();
	 System.out.println("Title is:"+ Title);//Return the Title of the website
	 System.out.println(driver.getCurrentUrl()); //Return the current url link

	 WebElement attribute = driver.findElement(By.id("email"));
	 System.out.println(attribute.getAttribute("id"));
	 System.out.println(attribute.getTagName());

	 driver.close();
	}
}
```

Browser Navigation Methods:

- Navigate to – Loads a new webpage in Current Browser. Syntax: driver.navigate().to("URL");
- Navigate forward – It moves single time forward in the Browser. Syntax: driver.navigate().forward();
- Navigate Refresh – Refresh a current webpage. Syntax: driver.navigate().refresh();
- Navigate back – It moves single time backward in the browser. Syntax: driver.navigate().back();

Tag in HTML	Purpose
a	Link
b	Bold
table	Table
tr	Row
td	Column
select	dropdown
button	button

Tag in HTML	Purpose
span	plaintext
input	Textbox, radiobutton, passwordbox, check box, button
img	Image
i	italic
iframe	frame
div	Ajax Controls, Asynchronous Java Script

In the scenario when we not get the locator available we will go for xpath.

- XPath is a language that describes a way to locate and process items in **Extensible Markup Language (XML)** documents.
- By using an addressing syntax based on a path through the document's logical structure or hierarchy.
- XPath in used in Selenium to uniquely identify an element on a Webpage as element locator.

Feature	Absolute XPath	Relative XPath
Starts With	/	//
Path Type	From root to element	From anywhere to element
Flexibility	Less flexible	More flexible
Length	Long	Short
Robustness	Low	High
Example	/html/body/div[1]/form/input[2]	//input[@name='username']
Preferred?	No	Yes

Syntax's to create our own xpath:

Syntax1: //tagname[@attribute='Value of Attribute'] Ex: //input[@id=`identifierid`]

Syntax2: //tagname[@attribute='Value of Attribute'] [@attribute='Value of Attribute']

Ex: //input[@id='identifierld'][@name='identifier'] //xpath with multi attr (A^B)

Syntax3: //tagname[@attribute='Value of Attribute' or @attribute='Value of Attribute'] //(A or B) Ex: //input[@id='identifierld' or @name='identifier']

Syntax4: //*[@attribute=['Value of Attribute'] //* specifies anytag. Ex: //*[@id='identifier']

Syntax5: //tagname[text()='Value of Attribute'] Ex: Forgot password / Password not received?

Ex: //*[(text()='Forgot password / Password not received?']

Syntax6: //tagname[starts-with(text(),' Value of Attribute Value Starting')] Ex: Forgot password / Password not received?

Ex: Forgot password / Password not received?

// * [starts-with(text(),'Forgot')] (or) //a[starts-with(text(),'Forg')]

Syntax7: //tagname[contains(@type,'Value of Attribute sub')] Ex: Forgot password / Password not received?

//*[contains(text(),'Forgot')] (or) // * [contains(text(),'orgo')]

Syntax8: //tagname[@attribute='Value of Attribute'][1][Last()][Last()-2]

Syntax9: //tagname[@attribute='Value of Attribute']/tagname/tagname #Relative with Absolute. Ex: //input[@attribute='value']/div/input #Parent to Child.

CSS Selector: Is a combination of an element selector and selector value. It identifies a web element within a webpage. **CSS Selector** is a syntax used to locate elements on a webpage based on their attributes, classes, IDs, and hierarchical structure. It's faster and more readable than XPath for simple selectors but cannot traverse backward in the DOM.

DOM (Document Object Model) is a programming interface that represents the structure of an HTML or XML document as a tree of elements (nodes). It allows developers to access, modify, and manipulate the content, structure, and style of a webpage dynamically.

Methods used in CSS Selector: TagName, Id, ClassName, Attributes, Contains, Starts-with, Ends-with.

Synatx for ID:

css= <HTML tag> <#> <Value of ID attribute>

- **HTML tag** It is the tag which is used to denote the web element which we want to access.
- # - The hash sign is used to symbolize ID attribute. It is mandatory to use hash sign if ID attribute is being used to create CSS Selector.
- **Value of ID attribute** - It is the value ofan ID attribute which is being accessed.
- The value of lD is always preceded by a hash sign.

Synatx for Class:

css= <HTML tag> <.> <Value of Class attribute>

' **.** ' - The sign is used to symbolize Class attribute.It is mandatory to use dot sign if a Class attribute is being used to create CSS Selector. The value of Class is always preceded by a dot sign.

Syntax for Attribute:

css= <HTML tag> <[attribute=Value of attribute]>

Attribute - It is the attribute we want to use to create CSS Selector. It can value, type, name , id,form.placeholer etc. It is recommended to choose an attribute whose value uniquely identifies

the web element.Value of attribute, It is the value of an attribute which is being accessed.

Syntax for Sub-String: StartsWith(^) Match a prefix

It is used to correspond to the string with the help of a matching prefix.

css= <HTML tag> <[attribute^=prefix of the string]>

^ - Symbolic notation to match a string using prefix.

Prefix - It is the string based on which match operation is performed. The likely string is expected to start with the specified string.

Syntax for Sub-String: EndsWith($) Match a suffix

css= <HTML tag> <[attribute$=suffix of the string]>

$ - Symbolic notation to match a string using suffix.

The suffix - It is the string based on which match operation is performed. The likely string is expected to ends with the specified string.

ChroPath is a browser extension (for Chrome and Edge) that helps you:

- **Generate XPath & CSS Selectors** for any web element.
- **Inspect elements easily** (like browser DevTools, but with more automation).
- **Test locators** directly from the browser.
- **Validate XPath/CSS** and see if they select the correct elements.

A **checkbox** is a GUI that allows users to select or deselect the options.

A **dropdown web element** is also known select box or combination box. Which is user interface that allows users to choose one or multiple options from the list.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.edge.EdgeDriver;
public class Checkbox {

    public static void main(String[] args) {
        System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");

        WebDriver driver = new EdgeDriver();

        try {
            driver.get("https://the-internet.herokuapp.com/checkboxes");

            // Locate and click first checkbox if not already selected
            By firstCheckbox = By.xpath("//form[@id='checkboxes']/input[1]");
            if (!driver.findElement(firstCheckbox).isSelected()) {
                driver.findElement(firstCheckbox).click();
            }
```

```
            // Locate and click second checkbox if not already selected
            By secondCheckbox = By.xpath("//form[@id='checkboxes']/input[2]");
            if (!driver.findElement(secondCheckbox).isSelected()) {
                driver.findElement(secondCheckbox).click();
            }
            System.out.println("Checkboxes selected!");

            // Wait for 5 seconds before closing the browser
            Thread.sleep(5000);

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            driver.quit();                          }}}
```

```
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;

public class RadioButton {

       public static void main(String[] args) throws InterruptedException {
             System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");
        WebDriver driver = new EdgeDriver();
        driver.get("https://www.facebook.com/r.php?entry_point=login");

        List<WebElement> gender = driver.findElements(By.name("sex"));

        int cnt = gender.size();
        System.out.println(cnt);
//        ================================================================
        for(int i=0;i<cnt;i++) {
             String txt = gender.get(i).getText();
             System.out.println(txt);
             gender.get(i).click();
             Thread.sleep(1000);
        }
        driver.quit();
       }
}
```

Web alerts refers to a small pop-up message boxes generated by a web page using JS.

- **Notification:** We can skip and proceed forward.
- **Alert/Pop-up:** We can't proceed further unit as we clear the alert/pop-up.

Types of Alerts:

1. Web Alerts.
2. Windows alerts, can't handle by selenium web driver.

Advantage: We can automate web pages only, We can automate web alerts.
Drawback: We can't automate window alert using Selenium WD, we can automate it using extra tools like Java Robot/Auto IT.

Web Alert: Switch To().alert()

Syntax: driver.switchTo().alert().operation();

#operation(): accept(), dismiss(), getText(), sendKeys().

Xpath Axes: In Selenium are used to navigate through elements in an XML or HTML document relative to a given element. They allow us to find elements based on their relationship to other elements in DOM.

Preceding, Ancestor, Parent, Preceding-Sibling.

Element, Child, Descendent, Following-Sibling, Following.

Axis	Definition	Example
preceding	Selects all nodes before the current node (excluding ancestors).	`//div[@id='x']/preceding::div`
ancestor	Selects all ancestors of the current node.	`//span/ancestor::div`
parent	Selects the parent of the current node.	`//span/parent::div`
preceding-sibling	Selects siblings before the current node.	`//div[@id='x']/preceding-sibling::div`
child	Selects all direct children of the current node.	`//div/child::p`
descendant	Selects all descendants of the current node.	`//div/descendant::p`
following-sibling	Selects siblings after the current node.	`//div[@id='x']/following-sibling::div`
following	Selects all nodes after the current node.	`//div[@id='x']/following::div`

Action class in Selenium used to handle advanced user interactions like mouse movements, keyboard events & drag and drop functionalities.

Move mouse pointer to element in page.

Build(), Perform(), KeyPress, KeyRelease, Double Click, Cick & Hold, Drag & Drop.

Actions Class in SWD Jar used to Automate Elements in Special Conditions

Situation 1:

Move Mouse Pointer to Element in Page

```
Web Element e=driver.findElement(By.Locator("Element Name"));

Actions a = new Actions(driver);

a.moveToElement(e).build().perform();
```

```
Web Element e=driver.findElement(By.Locator("Element Name"));

int x=e.getlocation().getX();

int y=e.getlocation().getY();

Actions a = new Actions(driver);

a.moveByoffset(x,y).build().perform();
```

Situation 2:

Double Click on Element in Page

```
Web Element e=driver.findElement(By.Locator("Element Name"));

Actions a=new Actions(driver);

a.moveToElement(e).doubleClick().build().perform();
```

Situation 3:

Right Click On element

```
Web Element e=driver.findElement(By.Locator("Element Name"));

Actions a=new Actions(driver);

a.moveToElement(e).contextclick().build().perform();
```

Situation 4:

Drag and Drop

```
Web Element el=driver.findElement(By.Locator("Element Name"));

Web Element e2=driver.findElement(By.Locator("Element Name"));

Actions a=new Actions(driver);

a.dragAndDrop(e1,e2).build().perform();
```

Web Element el=driver.findElement(By.Locator("Element Name"));

Web Element e2=driver.findElement(By.Locator("Element Name"));

int x=e2.getLocation().getX();

int y=e2.getLocation().getY();

Actions a = new Actions(driver);

a.dragAndDrop(e1,x,y).build().perform();

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.interactions.Actions;

public class DragDrop {

	public static void main(String[] args) throws InterruptedException {
		System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");
	  WebDriver driver = new EdgeDriver();
	  driver.get("https://jqueryui.com/droppable/");

	  Actions a = new Actions(driver);
	  driver.findElement(By.tagName("iframe"));
	  driver.switchTo().frame(0);
	  Thread.sleep(2000);

	  WebElement DragSrc = driver.findElement(By.id("draggable"));
	  WebElement DropTgt = driver.findElement(By.id("droppable"));
	  a.dragAndDrop(DragSrc, DropTgt).perform();

	  if (DropTgt.getText().matches("Dropped!")) {
		System.out.println("Successfully Dropped");
	  }else {
		System.out.println("Invalid Element Choosen");
	  }
	  driver.quit();
	}
}
```

Situation 5:

AutoComplete or Cache Element Automation

Web Element e=driver.findElement(By.Locator("Element Name"));

Actions a=new Actions(driver);

a.click(e).sendkeys("Data").build().perform();

a.sendkeys(keys.DOWN).build().perform();

a.sendkeys(keys.ENTER).build().perform();

Reading Properties File:

```
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;
public class ReadProperties {
	public static void main(String[] args) throws IOException {
		Properties p1 = new Properties();
		FileInputStream fis = new
FileInputStream("C:\\Users\\cheta\\Desktop\\SeleniumJava\\Student.properties");
		p1.load(fis);
		System.out.println(p1.getProperty("Name"));
		System.out.println(p1.getProperty("Course"));
		System.out.println(p1.getProperty("Time"));
		System.out.println(p1.getProperty("Sub-Course"));
	}
}
```

Reading Text File:

```
import java.io.FileReader;
import java.io.IOException;
import java.io.BufferedReader;
public class ReadText {
	public static void main(String[] args) throws IOException {
		String Fpath = "C:\\Users\\cheta\\Desktop\\SeleniumJava\\Read.txt";
		BufferedReader bf = new BufferedReader (new FileReader(Fpath));
		String startline;
		while((startline = bf.readLine()) != null) {
			System.out.println(startline);
		}
	}
}
```

Reading Excel File:

```
import java.io.File;
import java.io.IOException;
import jxl. Sheet;
import jxl.Workbook;
import read.biff.BiffException;
public class ReadExcel {
	public static void main(String[] args) throws BiffException,IOException {

		Workbook wb = Workbook.getWorkbook(new
File("C:\\Users\\cheta\\Desktop\\SeleniumJava\\Book.xlsx"));
		Sheet sh = wb.getSheet("Sheet1");
		for(int i=0;i<sh.getRows();i++) {
			System.out.println("Username:
"+sh.getCell(0,i).getContents());
			System.out.println("Course: "+sh.getCell(1,i).getContents());
		}
	}
}
```

Data Driven Testing:

```
import java.io.File;
import java.io.IOException;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.edge.EdgeDriver;
import jxl. Sheet;
import jxl.Workbook;
import read.biff.BiffException;
public class DataDrivenTesting {
	public static void main(String[] args) throws BiffException,IOException {

		System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");
	 WebDriver driver = new EdgeDriver();
	 driver.get("https://www.facebook.com");
		Workbook wb = Workbook.getWorkbook(new
File("C:\\Users\\cheta\\Desktop\\SeleniumJava\\Book.xlsx"));
		Sheet sh = wb.getSheet("Sheet1");

		for(int i=0;i<sh.getRows();i++) {
			WebElement Unm = driver.findElement(By.id("email"));
			WebElement pwd = driver.findElement(By.id("pass"));
			Unm.sendKeys(sh.getCell(0,i).getContents());
			pwd.sendKeys(sh.getCell(1,i).getContents());
			Thread.sleep(2000);
			pwd.clear();
			Unm.clear();
		}
	}
}
```

Java Robot:

Robot is a Class in JDK

- To control keyboard or mouse to interact with OS windows.
- To automate 1-Tier and 2-Tier Applications(Screens) including Pop-up Windows in 3-Tier WebPages.
- Can Automate Native Operation System applications like Notepad, Skype, Calculator, etc.
- It can simulate Keyboard and Mouse Event.
- It helps in upload/download of files when using selenium web driver.
- It can Automate Tire-1 Applications.
- Keyword/mouse event will only works on current instance of Window.

Example: While code is performing any robot class event, and during the code execution user moved to some other screen then keyword/mouse event will occur on that screen.

Installation Process:

- No need for Extra Installations and Jars.
- Robot and Other required classes are inbuilt in JDK Packages.
- Java Robot is Platform Independent.

Robot r = new Robot(); #Robot is Instance Class, import Robot java.awt

Method of Robot Class

keyPress( ): This method will press CONTROL Key from of Keyboard.
Example: r.keyPress(KeyEvent.VK_CONTROL):

mousePress(): This method will press the right click of your mouse.
Example: r.mousePress(lnputEvent.BUTTON3_DOWN_MASK):

mouseMove(): This will move mouse pointer to the specified X and Y coordinates.
Example: r.mouseMove(point.getX(), point.getY()):

keyRelease(): This method with release down arrow key of Keyboard.
Example: r.keyRelease(KeyEvent.VK_DOWN):

mouseRelease(): This method will release the right click of your mouse.
Example: r.mouseRelease(lnputEvent.BUTTON3_DOWNMASK):

// To Enter Data as Input Data in Clipboard
StringSelection s1=new StringSelection("10");
// String Selection is an Instance Class

//Send Data to Screen from Clipboard
Toolkit.getDefaultTooIkit().getSystemCIipboard().setContents(s1,null);
// Toolkit is Static Class

//Operations

r.keyPress(KeyEvent.VK_CONTROL);

r.keyPress(KeyEvent.VK_V);

r.keyRelease(KeyEvent.VK_V);

r.keyRelease(KeyEvent.VK_CONTROL);

File Downloading:

```
import java.awt.Robot;
import java.awt.event.KeyEvent;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.Keys;

public class RobotExample {

    public static void main(String[] args) throws Exception {
        System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");
        WebDriver driver = new EdgeDriver();
        driver.get("https://www.google.com");

        driver.findElement(By.name("q")).sendKeys("java download");
        driver.findElement(By.name("q")).sendKeys(Keys.ENTER);

        System.out.println("Please solve the CAPTCHA manually.");
```

```
        Thread.sleep(20000);  // wait 20 seconds for manual CAPTCHA

        Thread.sleep(2000);

        driver.findElement(By.xpath("//h3[contains(text(),'Download Free Java
Software')]")).click();
        Thread.sleep(2000);

        driver.findElement(By.xpath("//span[contains(text(),'Free Java
Download')]")).click();
        Thread.sleep(2000);

        driver.findElement(By.xpath("//span[contains(text(),'Agree and Start Free
Download')]")).click();
        Thread.sleep(2000);

        // Create Robot object (from java.awt)
        Robot r = new Robot();
        r.keyPress(KeyEvent.VK_TAB);
        r.keyRelease(KeyEvent.VK_TAB);
        r.keyPress(KeyEvent.VK_TAB);
        r.keyRelease(KeyEvent.VK_TAB);
        r.keyPress(KeyEvent.VK_ENTER);
        r.keyRelease(KeyEvent.VK_ENTER);
    }
}
```

File Uploading:

```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.datatransfer.StringSelection;
import java.awt.event.KeyEvent;
import org.openqa.selenium.Keys;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.edge.EdgeDriver;

public class FileUpload {
       public static void main(String[] args) throws AWTException {
              System.setProperty("webdriver.edge.driver",
"C:\\Users\\cheta\\Downloads\\SeleniumDownloads\\Drivers\\msedgedriver.exe");
        WebDriver driver = new EdgeDriver();
        driver.get("https://www.naukri.com");

        Robot r = new Robot();
        driver.findElement(By.id("input_resumeParser")).click();
        r.setAutoDelay(1000);
        StringSelection ss = new
StringSelection("C:\\Users\\cheta\\Desktop\\SeleniumJava\\Read.txt");

        Toolkit.getDefaultToolkit().getSystemClipboard().setContents(ss, null);

        r.keyPress(KeyEvent.VK_CONTROL);
        r.keyPress(KeyEvent.VK_V);

        r.keyRelease(KeyEvent.VK_CONTROL);
        r.keyRelease(KeyEvent.VK_V);

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

Project: Testing an eCommerce site with Selenium

Industry: E commerce

Problem Statement: How to successfully run an eCommerce site for various product search queries.

Topics: In this Selenium project we will develop Page Object Model Framework and use it for buying a product on Flipkart. We will test the site for a search query like iPhone with the specific phone memory size requirement. The product will be added to the cart, ensuring shipping is done to the entered address and proceeding to the checkout page.

Highlights:

Eclipse with Maven & TestNG plugin, Deploying POM for handling dependency,Fetching and storing result data in a file.

https://www.flipkart.com/

Steps:-

1. Find search WebElement: Enter iPhone product.
2. Select the Memory.
3. Select the product
4. Add to cart
5. Select place order
6. Login
7. Address
