**What are limitations in selenium web driver?**

Following are the limitations of Selenium:

* Selenium supports testing of only web based applications
* Mobile applications cannot be tested using Selenium
* Captcha and Bar code readers cannot be tested using Selenium
* Reports can only be generated using third party tools like TestNG or Junit.
* As Selenium is a free tool, thus there is no ready vendor support though the user can find numerous helping communities.
* User is expected to possess prior programming language knowledge.

**Installing/Configure of Selenium.**

To Download selenium webdriver and install selenium webdriver you need to download selenium jar files. Then configure downloaded selenium jar files in eclipse. Actually there is nothing to install except JDK. Let me describe you step by step process of download, installation and configuration of web driver software and other required components.

**Launch the Eclipse IDE & Create a Workspace**

1. Double click on ‘eclipse.exe‘ to start eclipse. First time when you start eclipse, it will ask you to select your workspace where your work will be stored as shown in below image.
2. Create a ‘working directory’ for all of your projects. Think of it like ‘My Documents’ in the Windows operating system. It’s a folder which contains a lot of your documents, but there’s nothing to prevent you from creating another folder called ‘My Other Documents‘ (for instance) to house other documents.Typically you only need one workspace, and you can think of it as your ‘My Documents‘ for Java code. If you wanted to, you could have more than one, but chances are you won’t have a use for more. I like to choose my own workplace location and will place all my ToolsQA tutorial projects under it.

**Create a new Project**

1. Create new Java Project from File > New > Project .
2. Select Java Project and click Next.
3. Give your Project name ‘OnlineStore‘ as shown in below given figures. Click on Finish button.

**Create a new Package**

1. Right click on Project name ‘OnlineStore‘ and select New > Package.
2. Give your Package name ‘automationFramework‘ and click on Finish button.

**Create a new Class**

1. Right click on Package ‘automationFramework‘ and select New > Class.
2. Give your Class name ‘FirstTestCase‘, check the option ‘public static void main‘ and click on Finish button. This will bring up totally a sweet class creation window.

**Add External Jars to Java build path**

Now you need to add Selenium WebDriver’s Jar files in to Java build path.

1. Right click on Project ‘OnlineStore‘ > Select Properties > Java build path. Then navigate to Libraries tab and click Add External JARs.
2. Add Selenium Java jar, you may add the source file too.
3. Add all the jars from the libs folder as well and Click OK.

That’s all about configuration of WebDriver with eclipse. Now you are ready to write your test script in eclipse and run it in WebDriver.

**What are different ways of locating elements in selenium?**

Locating Elements with Selenium WebDriver, findElement() method returns and WebElement and findElements() returns a list of WebElements.

1. By ID:

in Java: driver.findElement(By.id("element id"))

2. By CLASS:

in Java: driver.findElement(By.className("element class"))

3. By NAME:

in Java: driver.findElement(By.name("element name"))

4. By TAGNAME:

in Java: driver.findElement(By.tagName("element html tag name"))

5. By CSS Selector:

in Java: driver.findElement(By.cssSelector("css selector"))

6. By Link:

in Java: driver.findElement(By.link("link text"))

7. By XPath:

in Java: driver.findElement(By.xpath("xpath expression"))

**Which is fastest way to identify elements in web page?**

Finding elements by ID is usually going to be the fastest option, because at its root, it eventually calls down to document.getElementById(), which is optimized by many browsers.

Finding elements by XPath is useful for finding elements using very complex selectors, and is the most flexible selection strategy, but it has the potential to be very slow, particularly in IE. In IE 6, 7, or 8, finding by XPath can be an order of magnitude slower than doing the same in Firefox. IE provides no native XPath-over-HTML solution, so the project must use a JavaScript XPath implementation, and the JavaScript engine in legacy versions of IE really is that much slower.

If you have a need to find an element using a complex selector, I usually recommend using CSS Selectors, if possible. It's not quite as flexible as XPath, but will cover many of the same cases, without exhibiting the extreme performance penalty on IE that XPath can.

**What is absolute path and relative path in xpath?**

Location path specifies the location of node in XML document. This path can be absolute or relative. If location path starts with root node or with '/' then it is an absolute path. Following are few of the example locating the elements using absolute path.

/class/student − select student nodes within class root node.

<xsl:for-each select = "/class/student">

/class/student/firstname − select firstname of a student node within class root node.

<p><xsl:value-of select = "/class/student/firstname"/></p>

## Example

In this example, we've created a sample XML document students.xml and its stylesheet document students.xsl which uses the XPath expressions.

Following is the sample XML used.

### students.xml

<?xml version = "1.0"?>

<?xml-stylesheet type = "text/xsl" href = "students.xsl"?>

<class>

<student rollno = "393">

<firstname>Dinkar</firstname>

<lastname>Kad</lastname>

<nickname>Dinkar</nickname>

<marks>85</marks>

**Different types of waits or synchronization in selenium webdriver?**

Generally in Test Automation, we have two components  
1. Application Under Test  
2. Test Automation Tool.

Both these components will have their own speed. We should write our scripts in such a way that both the components should move with same and desired speed, so that we will not encounter "Element Not Found" errors which will consume time again in debugging.

Synchronization can be classified into two categories:

1. Unconditional   
2. Conditional Synchronization

Unconditional :  
In this we just specify timeout value only. We will make the tool to wait until certain amount of time and then proceed further.

*Examples: Wait() and [Thread.Sleep();](https://docs.oracle.com/javase/tutorial/essential/concurrency/sleep.html" \t "_blank)*

The main disadvantage for the above statements are, there is a chance of unnecessary waiting time even though the application is ready.

The advantages are like in a situation where we interact for third party systems like interfaces, it is not possible to write a condition or check for a condition. Here in this situations, we have to make the application to wait for certain amount of time by specifying the timeout value.

Conditional Synchronization:

We specify a condition along with timeout value, so that tool waits to check for the condition and then come out if nothing happens.

It is very important to set the timeout value in conditional synchronization, because the tool should proceed further instead of making the tool to wait for a particular condition to satisfy.

1. Implicit Wait.

An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available.

The default setting is 0. Once when we define the implicit wait, it will set for the life of the WebDriver object instance.

It is a mechanism which will be written once and applied for entire session automatically. It should be applied immediately once we initiate the Webdriver.

Implicit wait will not work all the commands/statements in the application. It will work only for "FindElement" and "FindElements" statements.

If we set implicit wait, find element will not throw an exception if the element is not found in first instance, instead it will poll for the element until the timeout and then proceeds further. We should always remember to add the below syntax immediately below the Webdriver statement.

Syntax:

driver.manage.TimeOuts.implicitwait(6,Timeunit.SECONDS);

Example using implicit timeout

WebDriver driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("[http://www.google.com"](http://www.google.com/));

Explicit Wait:

We need to define a wait statement for certain condition to be satisfied until the specified timeout period. If the Webdriver finds the element within the timeout period the code will get executed.

Explicit wait is mostly used when we need to Wait for a specific content/attribute change after performing any action, like when application gives AJAX call to system and get dynamic data and render on UI.

Example: Like there are drop-downs Country and State, based on the country value selected, the values in the state drop-down will change, which will take few seconds of time to get the data based on user selection.

    WebDriverWait wait = new WebDriverWait(driver, 10);

    wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("statedropdown")));

The above statement waits up to 10 seconds before throwing Exception (TimeoutException - Timed out after 10 seconds waiting for visibility of element) or if it finds the element, it will return in 0 - 10 seconds.

There are different waits that can be used based on the needs which we frequently come across when automating web applications.

-- how to handle multiple windows in selenium webdriver

public class MultiWindowHandle {

WebDriver driver;

public void setup() throws Exception {

driver=new FirefoxDriver();

String URL="https://www.abc.co.in/";

driver.get(URL);

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

}

public void test() throws Exception {

// Opening Calender

driver.findElement(By.xpath("//img[@alt='Calender']")).click();

// Storing parent window reference into a String Variable

String Parent\_Window = driver.getWindowHandle();

  // Switching from parent window to child window

for (String Child\_Window : driver.getWindowHandles())

{

driver.switchTo().window(Child\_Window);

// Performing actions on child window

driver.findElement(By.id("calendar\_month\_txt")).click();

List  Months=driver.findElements(By.xpath("//div[@id='monthDropDown']//div"));

int Months\_Size=Months.size();

System.out.println("Month size is:"+Months\_Size);

Months.get(1).click();

driver.findElement(By.xpath("//\*[@id='calendarDiv']/div/table/tbody/tr/td[contains(text(),'16')]")).click();

}

//Switching back to Parent Window

driver.switchTo().window(Parent\_Window);

//Performing some actions on Parent Window

driver.findElement(By.className("btn\_style")).click();

}

  public void close() {

  driver.quit();

  }

}

-- how to lanuch webpage using chrome driver

import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
public class Chrome {  
WebDriver driver;  
public void launchChrome()  
{  
System.setProperty("webdriver.chrome.driver", "E:\\DD MISHRA\\workspace\\chromedriver\_win\_26.0.1383.0\\chromedriver.exe");  
driver = new ChromeDriver();  
}

public void testChrome()  
{  
driver.get("http://www.google.co.in");  
driver.findElement(By.id("gbqfq")).sendKeys("Selenium");  
}  
public void kill()  
{  
driver.close();  
driver.quit();  
}  
}

**What is desired capabilities in selenium webdriver?**

importorg.openqa.selenium.WebDriver;

importorg.openqa.selenium.ie.InternetExplorerDriver;

importorg.openqa.selenium.remote.DesiredCapabilities;

public class IEtestforDesiredCapabilities {

public static void main(String[] args) {

DesiredCapabilities capabilities = DesiredCapabilities.internetExplorer();

capabilities.setCapability(CapabilityType.BROWSER\_NAME, "IE");

capabilities.setCapability(InternetExplorerDriver.

INTRODUCE\_FLAKINESS\_BY\_IGNORING\_SECURITY\_DOMAINS,true);

System.setProperty("webdriver.ie.driver", "C:\\IEDriverServer.exe");

WebDriver driver = newInternetExplorerDriver(capabilities);

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

driver.get("http://gmail.com");

driver.quit();

}

}

**How to set language while opening website?**

Using Firefox Browser :

FirefoxProfile profile = new FirefoxProfile();  
//setting the locale french : ‘fr’  
profile.setPreference(“intl.accept\_languages”,”fr”);  
driver = new FirefoxDriver(profile);  
driver.get(“[http://google.co.in&#8221](#8221););

Using Chrome Browser :

System.setProperty(“webdriver.chrome.driver”,”D:/DollarArchive/chromedriver.exe”);  
ChromeOptions options = new ChromeOptions();  
options.addArguments(“–lang= sl”);  
ChromeDriver driver = new ChromeDriver(options);  
driver.get(“[http://google.co.in&#8221](#8221););

Unfortunately it wont work for IE browser, We need to change it manually

**Write code to use textbox, button click events**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Actions;

public class Facebook {

public static void main(String args[]){

WebDriver driver = new FirefoxDriver();

driver.get("http://www.facebook.com");

WebElement email= driver.findElement(By.id("email"));

Actions builder = new Actions(driver);

Actions seriesOfActions = builder.moveToElement(email).click().sendKeys(email, "gati.naveen@gmail.com");

seriesOfActions.perform();

WebElement pass = driver.findElement(By.id("pass"));

WebElement login =driver.findElement(By.id("u\_0\_b"));

Actions seriesOfAction = builder.moveToElement(pass).click().sendKeys(pass, "naveench").click(login);

seriesOfAction.perform();

driver.

}

}

**How to select items from dropdown/select box?**

package automationFramework;

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.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.Select;

public class DropDownCommands {

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

WebDriver driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("http://toolsqa.wpengine.com/automation-practice-form");

Select oSelect = new Select(driver.findElement(By.id("continents")));

oSelect.selectByVisibleText("Europe");

Thread.sleep(2000);

oSelect.selectByIndex(2);

Thread.sleep(2000);

List<WebElement> oSize = oSelect.getOptions();

int iListSize = oSize.size();

for(int i =0; i < iListSize ; i++){

String sValue = oSelect.getOptions().get(i).getText();

System.out.println(sValue);

if(sValue.equals("Africa")){

oSelect.selectByIndex(i);

break;

}

}

driver.quit();

}

}

-- how to know if checkbox is checked or not in webpage

WebDriver driver = new InternetExplorerDriver();  
WebDriverBackedSelenium rc = new WebDriverBackedSelenium(driver, "");  
driver.get("[http://www.thejekels.com/dojo/demo/checkboxtree/](http://www.google.com/url?q=http%3A%2F%2Fwww.thejekels.com%2Fdojo%2Fdemo%2Fcheckboxtree%2F&sa=D&sntz=1&usg=AFQjCNFg3O4m8F9ZpfaRGboJ48iI1XQCQw)");  
if (driver.getWindowHandles().size()==0){  
throw new RuntimeException("Hello!");  
}  
String xpath = "//INPUT[@type='checkbox' and @id='cbt\_CheckBox\_1']";  
new WebDriverWait(driver, 5000).until(ExpectedConditions.presenceOfElementLocated(By.xpath(xpath)));  
WebElement checkbox = driver.findElement(By.xpath(xpath));   
System.out.println("============Before click");  
System.out.println("isSelected via webdriver: "+checkbox.isSelected());  
System.out.println("isSelected via rc: "+rc.isChecked(xpath));  
System.out.println("isSelected via value attribute: "+checkbox.getAttribute("value"));  
checkbox.click();   
System.out.println("============After click");  
System.out.println("isSelected via webdriver: "+checkbox.isSelected());  
System.out.println("isSelected via rc: "+rc.isChecked(xpath));  
System.out.println("isSelected via value attribute: "+checkbox.getAttribute("value"));

-- tell me code to pass values from parent window to child window

Parent Window:

Our parent window contains a simple TextBox and a LinkButton. The TextBox will display the data which is sent from the child window and the LinkButton will simply opens the child window also known as the popup window. Below is the simple code that is used to attach the onclick attribute to the LinkButton.

protected void Page\_Load(object sender, EventArgs e)

{

string openWindow = @"window.open('Child.aspx')";

this.LinkButton1.Attributes.Add("onclick", openWindow);

}

Now let's see the child window. In the code below we attach the onclick event to the LinkButton which is on the Child page.

Child Window:

protected void Page\_Load(object sender, EventArgs e)

{

this.LinkButton1.Attributes.Add("onclick", "PassValues()");

}

Child Page also contains a TextBox. Whatever you type in the TextBox will be transferred to the parent page. Now all we need is JavaScript so that we can pass the information to the parent page.

function PassValues()

{

var txtValue = document.getElementById("TextBox1").value;

// Now pass the value to the Parent form

window.opener.form1.txtUserName.value = txtValue;

window.close();

}

This is it! Now when you type something in the TextBox of the child window and press the LinkButton that value is passed to the parent window and displayed in the TextBox of the parent window. After that the child window is automatically closed.

--Write code for right click in selenium

package com.pack.rightclick;

import org.openqa.selenium.Alert;

import org.openqa.selenium.By;

import org.openqa.selenium.NoSuchElementException;

import org.openqa.selenium.StaleElementReferenceException;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.Assert; import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class RightClickExample { WebDriver driver;

public void Setup() { driver = new FirefoxDriver();

driver.manage().window().maximize(); }

public void rightClickTest()

{

driver.navigate().to(URL);

By locator = By.cssSelector(".context-menu-one.box");

WebDriverWait wait = new WebDriverWait(driver, 5);

WebElement element=driver.findElement(locator);

rightClick(element);

WebElement elementEdit =driver.findElement(By.cssSelector(".context-menu-item.icon.icon-edit>span")); elementEdit.click();

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

String textEdit = alert.getText();

Assert.assertEquals(textEdit, "clicked: edit", "Failed to click on Edit link");

}

public void rightClick(WebElement element)

{

{

Actions action = new Actions(driver).contextClick(element);

action.build().perform();

System.out.println("Sucessfully Right clicked on the element");

} catch (StaleElementReferenceException e) {

System.out.println("Element is not attached to the page document " + e.getStackTrace());

} catch (NoSuchElementException e) { System.out.println("Element " + element + " was not found in DOM " + e.getStackTrace());

} catch (Exception e)

}

}

-- Write code for drag/drop in selenium

import java.util.concurrent.TimeUnit;  
   
import org.openqa.selenium.By;  
   
import org.openqa.selenium.WebDriver;  
   
import org.openqa.selenium.WebElement;  
   
import org.openqa.selenium.firefox.FirefoxDriver;  
   
import org.openqa.selenium.interactions.Action;  
   
import org.openqa.selenium.interactions.Actions;  
   
public class DragAndDrop {  
   
public static void main(String[] args) throws InterruptedException {  
   
  WebDriver driver = new FirefoxDriver();  
   
  String URL = "[http://sandbox.checklist.com/account/](http://www.google.com/url?q=http%3A%2F%2Fsandbox.checklist.com%2Faccount%2F&sa=D&sntz=1&usg=AFQjCNGjYNEE_7XDf1xAru0lYS4R6BPJRQ)";  
   
  driver.get(URL);  
   
  driver.findElement(By.name("j\_username")).sendKeys("Username");  
   
  driver.findElement(By.name("j\_password")).sendKeys("Password);  
   
  driver.findElement(By.name("login")).click();  
   
  driver.manage().window().maximize();  
   
  driver.manage().timeouts().implicitlyWait(10000, TimeUnit.MILLISECONDS);  
   
  WebElement From = driver.findElement(By.xpath(".//\*[@id='userChecklists']/li[1]/a/span"));  
   
  WebElement To = driver.findElement(By.xpath(".//\*[@id='userChecklists']/li[4]/a/span"));  
   
  Actions builder = new Actions(driver);  
   
      Action dragAndDrop = builder.clickAndHold(From)  
   
     .moveToElement(To)  
   
    .release(To)  
   
   .build();  
   
  dragAndDrop.perform();  
   
}  
   
}

-- write code to find out if all links are working or not

import [org.openqa.selenium.By](http://org.openqa.selenium.by/" \t "_blank);

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

public class FindBrokenLinks {

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

  WebDriver driver = new FirefoxDriver();

  driver.get("[https://www.xyz.com/](https://www.google.com/url?q=https%3A%2F%2Fwww.xyz.com%2F&sa=D&sntz=1&usg=AFQjCNG02baXRkxW4WuokCXBFj8cBmZugw)");

  Thread.sleep(5000L);

  int size = driver.findElements(By.tagName("a")).size();

  System.out.println(size);

  List<String> Linkarray = new ArrayList<String>();

      List<WebElement> Linklist = driver.findElements(By.tagName("a"));

      for (WebElement link : Linklist) {

       String links = link.getText();

       Linkarray.add(links );

      }

      for (String linkToTest : Linkarray){

       driver.findElement(By.linkText(linkToTest)).click();

       Thread.sleep(15000L);

    if(driver.getTitle().contains("Problem")) {

     System.out.println("Fail");

    }

    else

    {

     System.out.println("pass");

    }

    driver.navigate().back();

    Thread.sleep(5000L);

   }

  driver.close();

}

}

-- write code on how to use javascriptexecutor?

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.Test;

public class JavaSE\_Test {

{

WebDriver driver= new FirefoxDriver();

JavascriptExecutor js = (JavascriptExecutor)driver;

Site. driver.get("[http://demo.guru99.com/V4/](http://www.google.com/url?q=http%3A%2F%2Fdemo.guru99.com%2FV4%2F&sa=D&sntz=1&usg=AFQjCNEJcoeaavmMJfDgPKK5fjwCWGsmuw)");

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

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

start\_time = System.currentTimeMillis();

executeAsyncScript() method to wait for 5 seconds js.executeAsyncScript("window.setTimeout(arguments[arguments.length - 1], 5000);"); System.out.println("Passed time: " + (System.currentTimeMillis() - start\_time));

}

}

up one javascript

diff b/w assert and verify

package stmTutorial;  
  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.firefox.FirefoxDriver;  
import org.testng.Assert;  
import org.testng.annotations.Test;  
  
public class AssertionExample {  
  
    public void assertion(){  
        //Instantiation of driver object. To launch Firefox browser  
        WebDriver driver = new FirefoxDriver();  
  
        driver.get("[http://www.softwaretestingmaterial.com](http://www.google.com/url?q=http%3A%2F%2Fwww.softwaretestingmaterial.com&sa=D&sntz=1&usg=AFQjCNEOeqPQ8ldYCekTftAHOPBpNkwyeg)");  
              String Title = "Software Testing Material";  
        String GetTitle = driver.getTitle();  
        System.out.println("Assertion starts here...");  
        Assert.assertEquals(Title, GetTitle);  
        System.out.println("A blog for Software Testers");  
                driver.quit();          
    }  
}

--dif b/w driver and close

public class BrowserClose()

{

public static String baseUrl;

public void setup(){

driver = new FirefoxDriver();

baseUrl="[http://www.wikishown.com](http://www.google.com/url?q=http%3A%2F%2Fwww.wikishown.com&sa=D&sntz=1&usg=AFQjCNGrxVdqqZU37URV6sEsIbyibjQKjA)";

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

}

public void wikiShown(){

driver.manage().timeouts().implecitlyWait(30,TimeUnit.SSECONDS);

}

public void tearDown()

{

driver.quit();

}

}

--ajax cal using selinum

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.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class Ajaxdemo { private String URL = "http://google.com /selenium/ajax.html"; WebDriver driver;

WebDriverWait wait;

public void setUp()

{

System.setProperty("webdriver.chrome.driver",".\\chromedriver.exe");

driver = new ChromeDriver(); driver.manage().window().maximize(); driver.navigate().to(URL);

}

public void test\_AjaxExample() {

By container = By.cssSelector(".container");

wait = new WebDriverWait(driver, 5); wait.until(ExpectedConditions.presenceOfElementLocated(container));

WebElement noTextElement = driver.findElement(By.className("radiobutton"));

String textBefore = noTextElement.getText().trim();

driver.findElement(By.id("yes")).click();

driver.findElement(By.id("buttoncheck")).click();

TextElement = driver.findElement(By.className("radiobutton")); wait.until(ExpectedConditions.visibilityOf(TextElement));

String textAfter = TextElement.getText().trim();

Assert.assertNotEquals(textBefore, textAfter);

System.out.println("Ajax Call Performed");

String expectedText = "Radio button is checked and it's value is Yes";

Assert.assertEquals(textAfter, expectedText); driver.close();

}

}