

INTRODUCTION TO AUTOMATION:

AUTOMATION:

The process of converting any manual test cases to automation test scripts using programming language and automation tools.

PROGRAMMING LANGUAGE	SCRIPTING LANGUAGE
Used to develop an application	Used to validate/test the application.
Now a days it is also used to test the application.	Now a days it is also used to develop the application.
It strictly follows syntax Ex: c, C#, C++, Java etc.	Syntax is not mandatory. Ex: Java scripts, vb scripting etc.

ADVANTAGES OF AUTOMATION:

- Saves time.
- Accuracy in result.
- Less resources.
- Code reusability.
- Quality is good.
- Avoid repetitive tasks.

DISADVANTAGES OF AUTOMATION:

- Knowledge of programming language is mandatory.
- We can automate OTP, finger prints, captcha, face recognition, prints related test cases, embedded related QR code, games related, images.

TYPES OF AUTOMATION TOOLS:

1. Functional Automation tool:

To automate functional, integration and end to end test cases.

Ex: Selenium, QTP, cypress, Appium, Postman, rest assured.

2. Non Functional Automation tool:

To automate performance related test cases.

Ex: Jmeter, Neoload, Load Runner, Appload etc.

INTRODUCTION TO SELENIUM:

SELENIUM:

It is open source web application test automation tool.

Open source: available for free

View source code

Customize it.

Web application: Any application which open through browser.

Test automation: To automate test cases.

Tool: Software.

SELENIUM COMPONENTS:

- 1. SELENIUM IDE**
- 2. SELENIUM WEB DRIVER**

SELENIUM IDE: (INTEGRATED DEVELOPER ENVIRONMENT):

If you want to create bug reproductive scripts.

If you want to perform automation aided exploratory testing.

It supports only chrome and Firefox browsers.

SELENIUM WEB DRIVER:

If you want to create robust, regression automation test scripts.

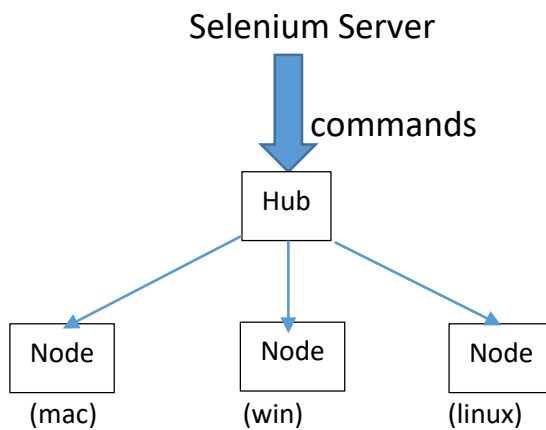
It supports multiple browsers.

Selenium web driver is the successor of selenium RC (remote control) deprecated (outdated).

SELENIUM GRID:

Combination of selenium server + selenium web driver + selenium RC.

It is used to perform system compatibility testing.



To test mobile based application:

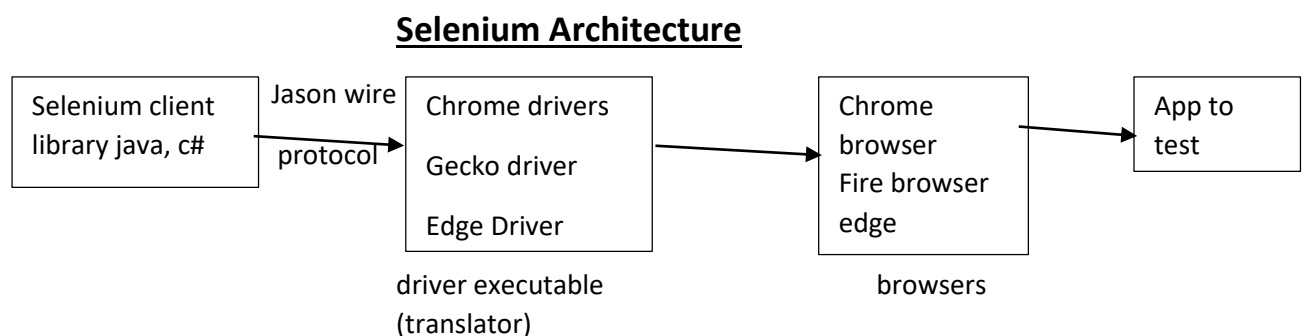
Selendroid : android

Winium: Windows

Appium: IOS and androids.

ADVANTAGES OF SELENIUM:

1. Open Source → Download and installation is easy → cost effective.
2. Open Source → customize it → integrate selenium with any 3rd party tool is easy.
3. It supports multiple programming language.
4. It supports multiple browsers.
5. It supports multiple platforms.
6. Browser compatibility testing is easy.
7. System compatibility testing is easy.



In selenium architecture we have client library, selenium server, driver executables, browsers and application under test. Client library is nothing but selenium supports multiple programming languages. When we write the program and trigger the test it will communicate with selenium server,

selenium server in turn communicate with driver executables, this happens via json via protocol. Selenium server will convert our programmer to Jason format and it will send to driver executables, these drivers executables again convert this Jason format back to HTML. This is because browsers can only understand HTML. This is how the architecture happens and after that all the test execution occurs on the browser which we can see. This is selenium architecture.

Browser drivers➡ each browser contain separate browser driver that is for ChromeDriver for Chrome, FirefoxDriver for firefox,EdgeDriver for edge.

Browsers➡It supports multiple browsers safari,

Launch Empty Browsers:

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

```
FirefoxDriver driver = new FirefoxDriver();
```

```
EdgeDriver driver = new EdgeDriver();
```

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

Explain this?

This statement is used to launch empty chrome browser.

ChromeDriver – It is a class.

Driver- It is a reference variable.

= ➡ assignment operator.

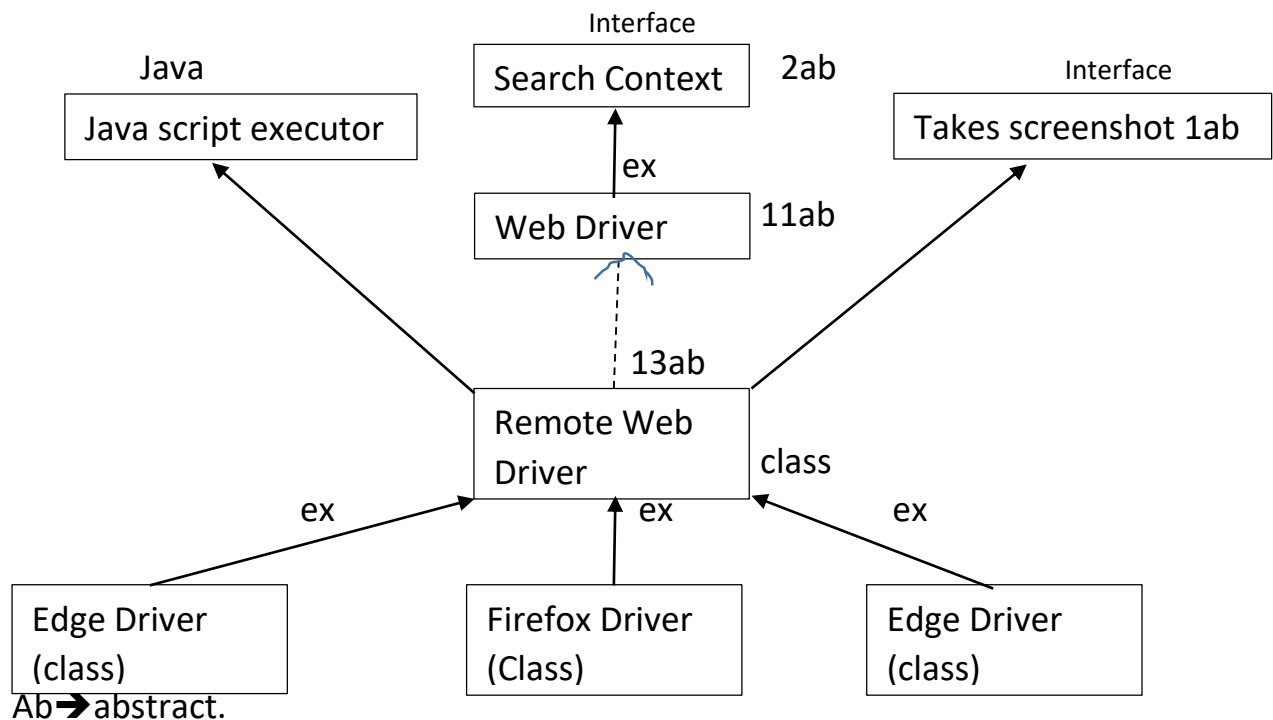
New➡ It is a key word to create a new object.

ChromeDriver()➡It is a constructor of chromeDriver class.

Task:

1. Explain similarly for edge driver and Firefox Driver.
2. Write a script to launch multiple browsers.
3. Write a script to launch user desired browser.

Selenium WebDriver Architecture:



Search context is the super most interface in selenium web Driver architecture. It has 2 abstract methods.

1. FindElement()
- and 2. findElements().

WebDriver is the sub-interface of search context interface. It has 11 abstract methods of its own and inherits 2 abstracts methods from search context interface. All together it has 13 abstract methods.

1. get()
2. getTitle()
3. getCurrentUrl()
4. getPageSource()
5. getWindowHandle()
6. getWindowhandles()
7. manage()
8. navigate()
9. switchTo()
- 10.close()
- 11.Quit()

Remote WebDriver is the implementing class of WebDriver Interface.

Remote WebDriver class also implements to JavaScript Executor interface and takes screenshot interface.

JavaScript Executot has 2 abstract Methods.

1. ExecuteScripts()
2. ExecuteAsyncScript

TakeScreenShot interface has one abstract method.

1. getScreenshotAs()

All the browsers specific classes like

ChromeDriver for Chrome

FirefoxDriver for firefox

EdgeDriver for edge

Extend to RemoteWebDriver class.

Task 2:

Launch Multiple Browsers:

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

```
FirefoxDriver driver = new FirefoxDriver();
```

```
EdgeDriver driver = new EdgeDriver();
```

Task 3:

LaunchUserDesiredBrowser:

```
Scanner sc = new Scanner(System.in); (import scanner)
```

```
System.out.println("enter browser to be launched:");
```

```
String browser = sc.next();
```

```
Switch(browser){
```

```
Case "Chrome" : ChromeDriver driver = new ChromeDriver();break;
```

```
Case "firefox" : FirefoxDriver driver = new FirefoxDriver();break;
```

```
Case "Edge" : EdgeDriver driver = new EdgeDriver();break;
```

```
Default: System.out.println("Enter valid browser Name");break;
```

```
}
```

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

What is upcasting?

The process of converting subclass to super type(class/interface) is called upcasting.

WebDriver – interface.

driver – It is reference variable

= Assignment operator

New – Keyword to create an object

ChromeDriver()- Construct of chromedriver class

; separator.

WebDriver driver = new ChromeDriver();

With respect to java, here we are creating an object for chromeDriver class and up casting it to WebDriver interface in order to access the web driver methods and also to achieve run time poly morphism with this statement.

What is run time poly morphism?

Depending upon the object we create, respective browser will open in run time.

Purpose of this:

W.r.t to selenium launches empty Chrome browser.

It enables us to access all the WebDriver methods.

Window Management:

How to maximize browser?

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

(this is because we might miss the feature in the application.)

How to minimize browser?

```
driver.manage().window().minimize();
```

How to make full screen?

```
driver.manage().window().fullscreen();
```

Few WebDriver Methods:

1. **get()** → It is used to navigate to the application.
usage: `driver.get("url");`
2. **getTitle()** → It is used to return the title of webpage.

Usage: String title = driver.getTitle();

3. **getCurrentUrl()** → It is used to return the current URL

Usage: String url = driver.getCurrentUrl();

4. **getPageSource()** → It is used to return source code of webpage.

Usage → String pagesource = driver.getPageSource();

(pagesource is the HTML of the page right click view page source)

5. **Close()** → To close the current tab

Usage: driver.close(); (java.net.SocketException)

6. **Quit()** → To exit the browser. Usage →

driver.quit();

Task 1: Write a script to launch chrome browser and navigate to amazon.com and fetch title, url, page source and closed the browser.

Task2: Repeat the task1 for demo.actitime.com

Task 3: Repeat task 1 for flipkart.com.

Navigation API's:

Return type is Navigation

It is a class in selenium.

1. **Back** → driver.navigate().back();

2. **After** → driver.navigate().forward();

3. **Refresh** → driver.navigate().refresh();

It is used to refresh the webpage.

4. **To navigate to another application-** to();

Driver.navigate().to(url);

Differences between **get()** and **Navigate()**

Get()	Navigate()
It is used to navigate to the application and waits till the application is completely loaded	It is used to navigate to an application, it also navigate back forward and refresh.
It cannot store cookies	It stores cookies.

LOCATORS:

Why locator?

In order to check the existence particular element on web page we use locator.

What are Locator?

In selenium locators are static methods of 'By' class.

'By' is an abstract class.

Different Types of Locators:

1. Id
2. Name
3. Linktext
4. Partiallinktext
5. Classname
6. Tagname
7. Css selector
8. Xpath

1. Id locator:

If an element node has 'id' as an attribute, then we go for id locator.

Usage: `driver.findElement(By.id("attribute value"));`

Scenario:

Open the browser

Enter facebook.com

Type your name in user name text field.

Close the browser.

Scenario:

Open the browser

Enter demo.actitime.com

Type your name in user name text field.

Close the browser.

2. Name Locator:

If an element node has an attribute as name, then we go for name locator.

Usage: `driver.findElement(By.name("name_attribute_value"));`

Scenario:

Open the browser and enter demo.actitime.com

Enter valid user name and password credentials and

Click on login button.

Validate homepage and close browser.

Scenario:

Open the browser

Enter facebook.com

Enter valid username and password.

Click on login button and validate home page.

3. Linktext Locator:

It is used to identify links only.

How to identify links?

By look and feel and if the tagname is <a>

Usage: `driver.findElement(By.linktext("text_on_link"));`

Ex: <a>skillray

If we have text on the link then we go for link text locator.

Skillrary is the text on the link.

Scenario:

Open the browser and enter facebook.com

Click on create a page

Close the browser.

Task:Scenario:

Open the browser and enter skillrary.com and click on gears and close the browser.

Scenario:

Open the browser

Enter facebook.com

Click on forgotten password and close the browser.

4. Partial linkText:

When the text on the link is lengthy then we go for partial link text.

Ex:<a>forgot your password?

Partial link text is used for links only.

Usage: driver.findElement(By.partialLinktext("partial text on link"));

Scenario:

Open the browser

Enter demo.actitime.com

Click on forgot your password link

Close the browser.

Task Scenario:

Open the browser and enter facebook.com

Click on forgotten password and close the browser.

Tast:Scenario:

Open the browser

Enter demo.actitime.com

Click on forgot your password

Click on return to login page.

7. Css Selector:

Whenever we don't have id or name attributes and whenever we don't have text on the link then we go for Css selector.

Css—cascading style sheet

SYNTAX:

Tagname[attribute name='Attribute value']

Usage: driver.findElement(By.Cssselector("Cssselector syntax expression"));

Steps to write Css selector and xpath expression in html:

1. Right click on the element
2. Click on inspect
3. Press Ctrl f

8. Xpath Locator:

To locate an element when the path of its node is unknown in html tree structure we can go for x path.

There are 2 Types of xpath:

- 1. Absolute xpath.**
- 2. Relative xpath.**

1. Absolute xpath : (not preferred)

The complete path from the root of the html tree to particular element node is called absolute xpath.

Here we use '/' to traverse.

For ex:

```
<html>
  <head>
    <div>
      <input id ='A' />
      <input id ='B' />
    </div>
  <div>
    <input id = 'C' />
    <input id ='D' />
  </div>
</head>
</html>
```

Suppose I want to search C

Html/head/div[2]/input[1]

Suppose I want to search D

Html/head/div[2]/input[2]

Suppose I want to search A

Html/head/div[1]/input[1]

Suppose I want to search B

Html/head/div[1]/input[2]

Drawback:

Absolute path can be lengthy

If we miss a single node the element cannot be identified.

Time consuming to write the entire path.

2. Relative Xpath:

The path from any parent node to particular element is called relative xpath.

Here we use '/' for traversing.

For ex:

```
<html>
```

```

<head>
  <div>
    <input type ='A' />
    <input type ='B' />
  </div>
  <div>
    <input type='C' />
    <input type='D' />
  </div>
</head>
<html>

```

To traverse:

```

//div[1]/input [2] → B
//div[1]/input [1] → A
//div[2] / input [1] → C
//dic[2] / input [2] → D

```

In Relative xpath we have 2 types:

1. **Basic relative xpath**
2. **Advance relative xpath.**

1. **Basic relative xpath:**

- a) X-path by attribute:
- b) X-path by text():
- c) X-path by contains():

- Handles lengthy texts and attributes
- Handles spaces
- Handles partially changing element
- Handles non-breakable spaces.

a) **X-path by attribute:**

Whenever we have attribute in the element node we go for xpath by attributes.

SYNTAX:

//tagname[@attributeName= 'attribute value'].

Scenario:

Open the browser and enter facebook.com
Enter username and password
Click on login button and validate home page and close the browser.

Task Scenario:

Write login script to demo.actitime.com

Drawback:

Attributes are mandatory and it does not support text.

b) Xpath by text():

Whenever we have text on the element we go for xpath by text.

SYNTAX:

`//tagname[text()='text value']`

`//tagname[.='text value']`

Scenario:

Open the browser and enter skillrary.com
Click on gears and close the browser.

Task Scenario:

Open the browser
Enter demoapp.skillrary.com
Click on feedback
Close the browser.

Drawback:

Element should have the text on it
Sometimes text can be lengthy
Does not support attributes.

c) Xpath by contains():

Whenever we have lengthy text or lengthy attributes we go for xpath by contains.

Handles lengthy texts and attributes

Handles spaces

Handles partially changing element

Handles non-breakable spaces.

SYNTAX:

```
//tagname[contains(@attributeName, 'attributeValue')]  
//tagname[contains(text(),'text value')]
```

Scenario:

Open the browser and enter facebook.com

Click on forgotten password link and type mobile no

And click on search and close the browser.

Handles partially changing element

Scenario:

Open the browser and enter demo.actitime.com

Enter valid credentials and click on login and click on ? and go to about your actitime inspect the build version element.

- `(build 1405_20)`
`//span[contains(text(),'build')]`
- `<div id ='about popup_useduser Accounts">(91 user accounts used)</div>`
`//div[contains(text(),'user account used')]`
- `actiTime 2020 online`
`//span[contains(text(),'online')]`

Handles non-breakable spaces.

Whenever developer/web designer develops a web page sometimes they give non-breakable spaces in the text of an element using and nbsp (non-breakable spaces).

We cannot identify or locate such elements using text().

We should use Xpath by contains():

```
<td class="switchCell readylabelcell">Ready for approval </td>
```

We cant use xpath by text()

So we go for xpath bycontains()

```
//td[contains(text(),'Ready for approval')]
```

```
<td class="switcherCellnotReadylabelCell">NotReady</td>
```

`//td[contains(text(),'Not Ready')]`

2. Advance Relative X-path:

a)Xpath by group Index:

b)Xpath by traversing:

✓ Independent and dependent xpath:

✓ Xpath by axes:

a) Xpath by group index:

When we have more than one matching elements we go for xpath by group index.

SYNTAX:

(xpath expression)[position Value]

For ex:

`(//a[contains(text(),'Feature')])[1]`

Drawbacks:

There can be n number of matching elements and indexing by searching each and every element is a tedious job.

With frequent GUI changes, the element position might not remain same.

Hence we cannot locate the element uniquely.

b) Xpath by traversing:

Steps to be followed:

1. Identify static element and write the xpath

2. Identify common parent.

3. Write tagname of dynamic element and element index (if required)

There are 2 types in xpath traversing:

1. Independent and dependent xpath

2. Xpath by axes:

1. Independent and dependent xpath:

a)Forward traversing

b)Backward traversing

a) Forwards traversing:

Traversing from parent to immediate child using '/' is called forward traversing.

b) Backward traversing:

Traversing from child to immediate parent using '/../' is called backward traversing.

Scenario:

Open the browser

Click on selenium.dev

Go to download

C# to API docs.

Note:

Read the html identify if there are text go with text() xpath.

Task Scenario:

Open flipcart and type iPhone and traverse from apple iPhone 11 to price.

2. Xpath by axes:

a) Parent axes:

b) Child axes:

c) Ancestor axes:

d) Descendant axes:

a) Parent Axes:

It is used to traverse to immediate parent. It works like '/../'

Usage: **/parent::tagname**

b) Child Axes:

It is used to traverse to immediate child. It works like '/'

Usage: **/child::tagname**

Task Sceario:

Open the browser and click on selenium.dev and click on downloads and traverse from image to API docs
(using parent child traverse)

Task Scenario:

Open demo.actitime.com

Traverse from checkbox to loginbutton

Using parent-child and forward and backward traverse.

Task Scenario:

Open the browser

Click on Ajio.com

Traverse from name to Price.

Using parent and child traverse and forward and backward traverse.

c) . Ancestor axes:

It is used to traverse from child to any parent node.

usage : **/ancestor::tagname**

d) Descendant axes:

It is used to traverse to any child from parent node.

usage : **/descendant::tagname**

scenario:

flipkart browser

apple iphone

from text to price

```
//span[text()='APPLE iPhone 11 (White, 128  
GB)']/ancestor::div[2]/descendant::div[10]
```

scenario:

demo.actitime.com

from checkbox to login

using ancestor and descendant

```
//input[@type='checkbox']/ancestor::tr/descendant::a[@id='loginButt  
on']
```

Task scenario:

flipkar browser

watches

go to 2nd watch

traverse from name to rating's

Sibling Functions :

1. **Preceding- sibling:** It is used to traverse to the nodes above which are present at the same level having same parent.

usage: /preceding-sibling::tagname.

2. **Following-Sibling:** It is used to traverse to the nodes below which are present at the same level having same parent.

Usage: /following-sibling::tagname.

Go to bullymoviereviewz.com

traverse from india lockdown to blurr

```
//b[text()='India Lockdown']/ancestor::tr/following-sibling::tr[6]/descendant::b
```

traverse from black adam to ram setu

```
//b[text()='Black Adam']/ancestor::tr/preceding-sibling::tr/descendant::b[text()='Ram Setu ']
```

tast 1 : In selenium.dev

go to downloads

traverse from c# to java script

```
//p[text()='C#']/ancestor::div[@class='row justify-content-center px-5 pb-5']/descendant::p[text()='JavaScript']
```

using ancestor and dependant.

Task:

traverse from python to ruby .

3. Preceding:

It is used to traverse to the nodes at the same level above having different parents.

Usgae : /preceding::tagname

```
//p[text()='Python']/preceding::p[text()='Ruby']
```

traverse from python to ruby.

elements having at the same level and parents nodes are different.

go to bollymoviereviewz
from thank god to blurr

```
/b[contains(text(),'Thank God')]/preceding::b[text()='Blurr']
```

when you have nbsc element go for xpath by contains.

4. **Following:** It is used to traverse to the nodes at the same level below having different parents.

Usage: **/following::tagname**

```
//b[text()='Kantara']/following::b[text()='Brahmastra (World)']
```

from kantara to Brahmastra.

Operators in Xpath:

We have 2 Operators to pass multiple attributes to locate an element uniquely.

1. **And Operator:**

When both conditions satisfy only then it locates the element.

Usage: `//tagname[@AN1='AV1' and @AN2='AV2']`

or

`//tagname[@AN='AV' and contains()]`

or

`//tagname[contains() and contains()]`

(AN - Attribute name

(AV-Attribute value)

2. **Or operator:**

When atleast one condition is satisfied only then it locates the elements.

Usage: `//tagname[@AN1='AV1' or @AN2='AV2']`

`//tagname[@AN='AV' or contains()]`

`//tagname[contains() and contains()]`

Open olympics.com
scroll down to Featured Athletes
inspect from name to gold medal.

```
//span[text()='Eliud KIPCHOGE']/ancestor::li[@class='b2p-list__item position-relative']/descendant::span[@class='result-medal result-medal--gold']
```

```
//span[.='Carolina MARÍN']/ancestor::li/descendant::span[@class='result-medal result-medal--gold']
```

Open icc-cricket.com
under rankings
inspect player ranking.

DIFFERENCES BETWEEN CSS SELECTOR AND X PATH:

CSS SELECTOR	X PATH
Css selector is faster compared to Xpath.	X path is bit slower.
It is unidirectional.	It is multi-directional.
We cannot use multiple attributes.	We can use multiple attributes using 'And' and 'OR' operator.
It does not support text.	It supports text.

Class Name Locator:

Whenever we have an attribute as class in the element node we go for classNameLocator.

Usage: driver.findElement(By.className("class_attribute_value"));

Drawback:

We might have multiple matching elements using same class name, hence we cannot identify the element uniquely.

TagName Locator:

It is used to identify an element using tagname.

Usage: driver.findElement(By.tagName("tagname"));

We can also fetch the list of elements having same tagname.

Drawback:

We might have multiple matching elements using same tag name, hence we cannot identify the element uniquely.

SYNCHRONIZATION:

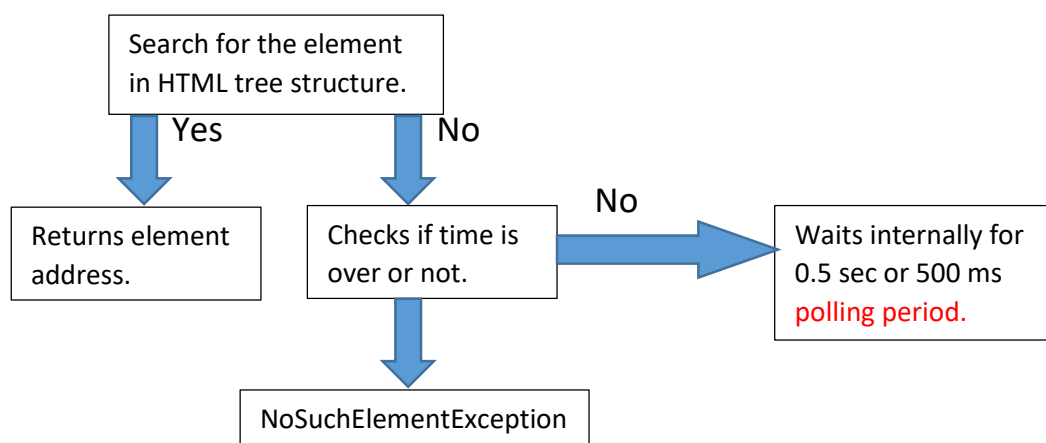
The process of matching selenium speed with application speed is called synchronization.

Different Wait Statements:

1. **Thread. Sleep(time_in_milliseconds)** → It is simple java wait statement. It blindly waits for specified amount of time and continues executing next statement. It throws interrupted Exception.
2. **Implicitly Wait Statement** → It is selenium wait statements which synchronizes only find element () and find elements () methods.
Timeouts....>Return type.

SYNTAX:

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(time_in_seconds));
```



Implicit Wait Work Flow:

- Whenever implicitlyWait statement is given along with find element () or find elements () methods, it searches for the element in HTML tree structure.

- If element is not found, then it checks if given time is over or not.
 - ➔ If given time is not over, then internally it waits for 0.5 seconds or 500 milliseconds. This waiting time is called as polling period. Then it goes back and searches for the elements in html tree structure again.
 - ➔ If time is over then it throws NoSuchElementException.
 - ➔ If element is found it returns element address.
 - ➔ This process repeats until element is found or specified time is over.

3. Explicitly Wait:

It is selenium wait statement which synchronizes all the WebDriver methods including find element () and find elements () methods.

SYNTAX:

```
WebDriverWait wait = new WebDriverWait(driver, Duration.Ofseconds(timeinsec));
```

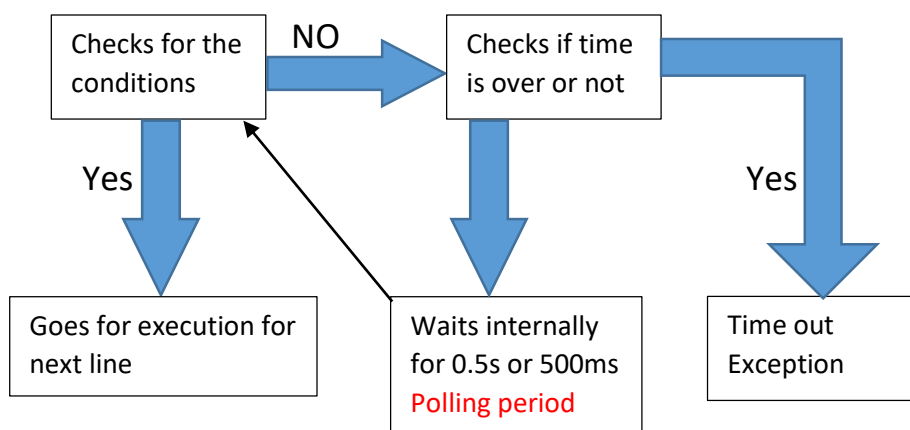
```
wait.until(ExpectedConditions.visibilityOf(element));
```

Or

```
wait.until(ExpectedConditions.elementToBeClickable(element));
```

Or

```
Wait.until(ExpectedConditions.titlecontains("title"));
```



Explicitly Wait work flow:

- ➔ Whenever explicitly Wait statement is given along with any WebDriver methods, it will first check if the condition mentioned is satisfied or not.
- ➔ If the condition is satisfied it will go to the next statement in the scripts.

➔ If the condition is not satisfied, then it checks if the time is over or not.

➔ If time is over, it throws Timeout Exception.

➔ If time is not over, it will wait internally for 5 sec or 500ms. This waiting period is called as polling period. Then re-checks if the condition is satisfied or not.

➔ This process repeats until the condition is satisfied or time is over.

ImplicitlyWaits	ExplicitlyWait
It synchronizes only findElement() and findElements () methods.	It synchronizes all the WebDriver methods including findElement() and findElements () methods.
No need to specify the condition	Condition should be specified.
Once implicitlyWaits is given it works Implicitly, need not to give again.	ExplicitlyWait should be given every time for method to synchronize it
It throws NoSuchElementException If element not found.	It throws TimeoutException if condition is not satisfied.

4. FluentWait:

It is selenium waits statement which is used to customize the polling period.

SYNTAX:

```
FluentWait wait = new FluentWait(driver)
    .withTimeout(Duration.ofseconds(time))
    .pollingEvery(Duration.ofseconds(polling_time))
    .ignoring(Exception e);
```

WebElement:

It is an interface.

Element which appears on the web page are called WebElements.

WebElement methods:

Actions	Getters	Verification
Sendkeys()	getText()	isEnabled()
Click()	getLocation()	isDisplayed()
Clear()	getSize()	isSelected()
Submit()	getAttribute()	

Actions:

1. sendKeys() : It is used for typing.
2. Click() : It is used for clicking.
3. Clear() : It is used to clear data in text field.

Scenario:

Open the browser

Enter google.com

Type some text or data in search text field.

And clear the data

Close the browser.

Task Scenario:

Open the browser

Enter amazon.com

Type some text or data in search text field.

And clear the data

Close the browser.

4. **Submit()**: It is similar to click method but it is used for forms and it works only if the element node has attribute type = "submit"

Scenario:

Getters:

1. **getText ()**:

It is used to fetch the text on the element.

Returntype is String

Scenario:

Open the browser

Enter demo.actitime.com

Fetch the page header text.

2. getLocation ():

It is used to fetch the location of the element on web page.

Return type is Point

Point is a class in selenium which provided the X and Y coordinates of the element.

For x coordinates → getX() for y coordinates → getY()

Scenario:

Open the browser

Enter amazon.com

Location for game accessories.

Priority Locators:

Id-----→name-----→linktext.....>xpath

(only if it is link)

Suppose id has alpha numeric don't go for id

For ex: id= u 0 5 f

Same for name as well if it has alpha and numeric we can't go for name.

3. getSize():

It is used to get size or dimensions of the element on the web page.

Return type is getSize() is Dimension.

Dimension is a class in selenium. It has 2 methods to get height and width of the element.

Height → getHeight()

Width → getWidth()

Scenario:

a) Open the browser

Enter amazon.com

Fetch dimension of any element on the web page

Close the browser.

b) Open the browser

Enter amazon.com

Fetch dimension of logo page

Close the browser.

c) Open the browser

Enter facebook.com

Fetch dimension of facebook

Close the browser.

Note: If you want to check, go to computed in html page only and you can check font size colour dimensions etc.

4. `getAttribute(String attributeName):`

It is used to fetch attribute value of the element when attribute name is passed as key to this method.

Return type is String

Scenario:

Open the browser and enter google.com

Fetch the attribute of search text field

Close the browser.

Open the browser and enter Instagram.com

Fetch the attribute of any web element on webpage

Close the browser.

Verification methods:

1. `isDisplayed():`

It is used to check if the element is displayed on the web page or not.

return Type = Boolean

Scenario:

Open the browser

Enter vtiger.com

Check if the vtiger logo is displayed or not and close the browser.

Open the browser and Enter demoactitime.com and check if the logo is displayed or not and close the browser.

2. `isEnabled():`

It is used to check if the element on the web page is enabled.

Return type is Boolean

Scenario:

Open the browser and enter the amazon.com and

Check if the search button is enabled or not and close the browser.

Scenario:

Open the browser and enter Instagram.com

Check if login button is enabled or not

Enter valid credentials and click if login button is enabled or not

If enabled, click on the button else print disabled
Close the browser.

3. isSelected():

It is used to check if the element is selected or not. Generally used for checkboxes and radio buttons.

Return type is Boolean

Scenario:

Open the browser and enter demo.actitime.com

Select the checkbox 'keep me logged in'

Check if the checkbox is selected or not

Close the browser.

Scenario:

Open the browser and enter facebook.com

Click on 'create new account' then select gender

Check if gender radio button is selected or not and close the browser.

CHAPTER 2

HANDLING WEB ELEMENTS

1. Auto Suggestions:

List of suggestions that appear automatically on the web page when we search for resource.

We can handle auto suggestion using **findElements()** method.

Scenario:

Open the browser

And enter google.com

Type 'selenium' in the text field

Fetch all the auto suggestion and print in console and close the browser.

Scenario:

Open the browser and enter amazon.com

Type 'laptops' in search text field

Fetch all the auto suggestions and print in console and close the browser.

Scenario:

Open the browser and enter google.com

Type 'yourname' in search field

Fetch the 5th element in auto suggestions then close the browser.

findElements() :

It is used to fetch the list of matching web elements on the web page

Return type of findElements() is List<WebElement>.

Difference bwtween findElement() and findElements()

findElement()	findElements()
It is used to fetch first matching element.	It is used to fetch the list of all matching elements.
The return type is WebElement	The return type is List<WebElement>
If element is not found it throws NoSuchElementException	If the elements are not found it returns empty array list.

2. Mouse Actions:

It can be handled using Actions class.

Different mouse actions:

- Mouse Hovering
- Right Click
- Double Click
- Drag and Drop.

➤ Mouse Hovering

Step 1: Create an instance for Actions class and pass WebDriver reference to the constructor.

```
Actions a = new Actions(driver);  
a.moveToElement(element).perform();
```

Note: Action is imported from selenium.interaction.

Step 2: Call respective methods to perform mouse actions using Actions class reference.

Mouse Hover:	<code>a.moveToElement(element).perform();</code>
Right Click:	<code>a.contextClick(element).perform();</code>
Double Click:	<code>a.doubleClick(element).perform();</code>
Drag and Drop:	<code>a.dragAndDrop(src,target).perform();</code>

Perform(): is the default method to be given after each mouse action method.

Scenario:

Open the browser and enter demoapp.skillrary.com

Mouse hover to the course. Click on cucumber tab and close the browser.

Scenario:

Open the browser enter myntra.com

Mouse over on kids and click on kids

And close the browser.

Note: When the element is not inspectable by right click and vanishes within seconds even before you perform inspections. Follow the steps as given below:

Step 1: Go to sources in developer tools.

Step 2: Mouse hover to the element.

Step 3: Press f8+ctrl+\

This will pause the script and puts the screen in debugger mode. Take the arrow mark to the element and inspect it.

➤ **Right Click:**

Step 1: Create an instance for Actions class and pass WebDriver reference to the constructor.

```
Actions a = new Actions(driver);  
a.contextClick(element).perform();
```

Note: Action is imported from selenium.interaction.

Scenario:

Open the browser enter amazon.com. Right click on search text field. And close the browser.

Scenario:

Open the browser enter myntra.com Right click on beauty tab and close the browser.

➤ **DoubleClick:**

Step 1: Create an instance for Actions class and pass WebDriver reference to the constructor.

```
Actions a = new Actions(driver);  
a.doubleClick(element).perform();
```

Note: Action is imported from selenium.interaction.

Scenario:

Open the browser enter demoapp.skillrary.com
Mousehover to course and click on 'selenium training' and
Double click on plus(+) button and
Close the browser.

➤ **Drag and Drop:**

Step 1: Create an instance for Actions class and pass WebDriver reference to the constructor.

```
Actions a = new Actions(driver);  
a.dragAndDrop(src,target).perform();
```

Note: Action is imported from selenium.interaction.

Scenario:

Open the browser

enter <http://www.dhtmlgoodies.com/scripts/drag-drop-custom/demo-drag-drop-1.html>

Drag and drop cat to first box and close the browser.

3. Drop Downs:

We can handle drop downs using 'Select' class.

Select class is present in **org.openqa.selenium.support.ui** package.

We have 3 methods to perform select from drop down

- a. **selectByIndex(int index)**
- b. **selectByVisibleText(String text)**
- c. **selectByValue(String value)**

Steps to handle drop downs:

Step 1: Create an instance of select class and pass element reference as argument to the constructor.

Select s= new Select(element);

Step 2: Using the select class reference call one of the above methods to select an element from dropdown list.

Note: Dropdown can be identified using tagname<select>

Scenario:

Open the browser and enter amazon.com

Select an item from all dropdown and close the browser.

To check the first selected option

getFirstSelectedOption()

This method returns the **WebElement** which is selected first.

To get all the options from dropdown lists- **getOptions()**

This method returns **list<WebElements>**

We have 2 types of dropDowns:

- **single select dropdown** → We can select only one option at a time. We cannot deselect single select dropdown it throws **UnsupportedOperationException**.

- **Multi select dropdown**→ We can select multiple options at a time. We can deselect from multi select dropdown.

To check if the dropDown is single select or multi select we use the method-
isMultiple()

It returns true if the dropDown is multi select

It returns false if the dropdown is single select.

Scenario:

Open the browser enter demoapp.skillrary.com

Check if the dropdown is single or multi select and

Close the browser.

In order to deselect the option from the dropdown we have methods:

1. `deselectByIndex(int index)`
2. `deselectByVisibleText(string text)`
3. `deselectByValue(string value)`
4. `deselectAll()`

Scenario:

Open the browser enter demoapp.skillrary.com

Select first 3 options

Get all selected options and

Deselect the options and close the browser.

To get all selected options→`getAllSelectedOptions()`

This method returns `List<WebElement>`

4. ScreenShot:

To get the screenshot of the web page we use `TakesScreenShot` interface and we should add 'apache common io' libraries to the project.

Pre requisite: download apache common io library.

Open the browser and Enter maven repository and Click on the first link

Type apache common io and click on search.

Click on the first link and Click on 2.11.0 version because it has 4000+ usages.

Click on the jar. (it will be downloaded automatically)

Copy the downloaded file and paste in the project(eclipse)and Right click on the paste jar in eclipse and build path and add build path.

Steps to achieve screenshot of Webpage:

1. Typecast WebDriver reference to TakesScreenshot interface.
TakesScreenShot ts = (TakesScreenshot) driver;
2. With TakesScreenshot reference call the method getScreenshotAs()
File src = ts.getScreenshotAs(OutputType.FILE);
3. Create a new file in project
File dest = new File("./screenshot/screenshot.png");
4. Copy the src file(temporary file) to dest file(permanent file)
FileUtils.copyFile(src,dest);
The above statement throws IOException.
5. After execution refresh the project and you can see screenshot folder created.

Note: In order to copy the Screenshot file from temporary memory(RAM) to permanent memory (Local disk) we have to use **apache commons io libraries**. **FileUtils** is the class provided by **apache common io libraries**, using this class we call a method copyFile() to copy the file from RAM to permanent memory. Both are standalone application which is why we are using **apache.common io libraries**.

Task Scenario:

Open the browser and enter demo.actitime.com

Enter invalid credentials and click on login and

Takes the screenshot and close the browser.

Using WebElement method also we can take the screenshot try it.
(element.getScreenshotAs() method.)

5. ScrollBars:

We can handle scroll bars using **JavascriptExecutor interface**.

Steps to perform scroll action:

1. Typecast WebDriver reference to javascriptExecutor interface.
JavascriptExecutor js = (javascriptExecutor)driver;
2. Call the method executeScript() using JavascriptExecutor reference.
Js.executeScript("java script code for scroll action");

Case 1: Hard coding the coordinates

```
Js.executeScript("window.scrollTo(0,5000)");
```

Scenario:

Open the browser and Enter myntra.com

Scroll the page

Close the browser.

Task Scenario:

Repeat above scenario for amazon and ebay.com

Case 2: Using the location of the element and scrolling till that point.

1. Find the location of the element.
2. Concatenate the x and the y coordinates in the above script.

```
js.executeScript("window.scrollTo("+x+", +y+)");
```

Scenario:

Open the browser and Enter myntra.com

Scroll the page till sarees element

Close the browser.

Task Scenario:

Repeat above scenarios for amazon.com

Case 3: Using element reference

```
js.executeScript("arguments[0].scrollIntoView(true)",elementReference);
```

6. Frames :

The web page inside another web page is called frame.

To handle frames, we should switch the control to the frame.

To switch to the frames:

```
driver.switchTo().frame(index);
```

```
driver.switchTo().frame(id_or_name);
```

```
driver.switchTo().frame(text);
```

To switch back to from the frame:

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

Scenario:

Open the browser and enter snapdeal.com

Mouse Hover on sign in and click login

Then enter mobile number and click on close button.
Inspect the search field type toys and click on search button and close the browser.

How to identify the frames:

Right click new frames source

The tagname should be iframes.

7. Popups:

Popups are the windows which appear on the screen.

Different types of popups:

➤ **Alert popup:**

Not inspectable and not movable also.

To handle this popup first we have to switch the control to the popup window.

To click on ok -----➔ `driver.switchTo().alert().accept();`

To click on cancel-----➔ `driver.switchTo().alert().dismiss();`

To get text on the popup➔ `driver.switchTo().alert().getText();`

To pass data to the popup➔ `driver.switchTo().alert().sendKeys("data");`

Scenario:

Open the browser click on demoapp.skillrary.com

Mouse hover to course and click on selenium training

And click on add to cart button and alert popup will display.

Switch to alert popup and get the text and accept and close the browser.

Scenario:

https://the-internet.herokuapp.com/javascript_alerts Jan 11, 10:39 AM

➤ **HiddenDivision/Calendar popup:**

Inspectable but not movable

We can handle this popup using `findElement()`.

Scenario:

Open the browser, enter makemytrip.com and select a departure date

Click on search and close the browser.

➤ **Notification popup:**

Not inspectable and not movable.

To disable this popup we have browser specific classes like ChromeOptions for Chrome and FirefoxOptions for firefox.

Steps to handle:

Step 1: Create a reference to ChromeOptions class even before launching the browser.

`ChromeOptions option = new ChromeOptions();`

Step 2: Call the method addArguments() using ChromeOptions reference.

`Option.addArguments("- - disable-notifications");`

Step 3: While launching browser call parameterized constructor with Chromeoptions class reference as argument.

`WebDriver driver = new ChromeDriver(option);`

Scenario:

Open the browser

Enter ajio.com

Handle notification popup

Close the browser.

Task Scenario:

Open the browser and Enter spicejet.com

Handle notification popup and close the browser.

➤ **Child browser popup:**

We handle this popup using 2 methods:

1. getWindowHandle() → This method returns parent window address
Returntype is String

2. getWindowHandles() → This method returns both parent and child browser addresses.

Returntype is Set<String>

Scenario:

Open the browser

Enter skillrary.com

Click on 'GEARS' and click on 'SKILLRARY ESSAY'

Type your name and click on "yes its my name" and close the browser.

Task Scenario:

Open the browser and enter skillrary.com

Click on 'GEARS' button click on 'SKILLRARY DEMO APP'

Mouse over to Course and click on selenium training

And close the child browser.

Type selenium in search bar. And close the browser.

getWindowHandle()	getWindowHandles()
It is used to return parent window ID	It returns both parent and child window ids
The return type is string	The return type Set<String>

Note: The window id's are in Hexa Decimal format.

8. FileUpload:

This can be handled in 3 ways

1. Using sendKeys()
2. AutoIT tool
3. Robot class

1. Using sendKeys():

We use sendKeys() to upload file only if we have Type='File' attribute.

We should pass the path of the file to the sendKeys()method.

➤ AutoIT:

It is third party tool which automates StandAlone applications.

Steps to download AutoIT:

1. Open the browser and type autoIT download.
2. Click on the first link.
3. Scroll the page bit down and click on 'Download AutoIT'.
4. After downloading, double click on the file and install.

Scenario:

Open the browser

Enter naukri.com

Click on Register button

Click on upload resume.

Close the browser

→ Selenium eclipse IDE

→ Auto IT scite script Editor.

Steps to launch Scite script Editor:

1. Type scite script Editor in search.
2. Open the app.

Standalone application scenario steps:

- a. Switch to the opened file popup window.
- b. Switch the control to Filename – text bar.
- c. Type file path
- d. Click on open button.

Step 1: WinWaitActive("title")

This method is used to wait until the file uploads popup window is active and switches the control to it.

Step 2: Sleep(2000)

In autoIT sleep() is the only wait statement.

Step 3: ControlFocus("title","text","controlID")

In order to inspect an element in stand alone application

- a. Search for 'autoIT v3 window info' in computer.
- b. Drag and drop the 'Finder Tool' to the element to be inspected.

Click on control in AutoIT v3 window info:

- c. ControlFocus method is used to switch the control over to the elements.

ControlID:

It is the combination of class and instance.

Step 4: Send("path_of_the_file_to_be_uploaded")

Send method is used to type data into the element.

Step 5: Sleep(2000);

Step 6: ControlClick("title","text","controlID")

Control click method is used to click on the element.

➔After writing the program, save it in a folder in desktop with au3. Extension.

➔To compile the program in scite script editor.

➔Click on tools and click on compile.

➔Click on 'Compile Script' button

➔We can see .exe file auto saved to the folder on desktop.

Now integrate the above code with selenium program.

```
Runtime.getRuntime().exec("path of .exe file")
```

➤ **File Upload Using Robot Class:**

Robot class is used to automate for all keyboard related actions.

It is available in java.awt package.

awt- abstract Window ToolKit

Step 1: Create an instance for Robot class

```
Robot robot = new Robot();
```

Step 2: Create an instance for StringSelection class and pass the path of the file to be copied to the FileUpload popup.

```
StringSelection path = new StringSelection("file_path");
```

StringSelection class is available in java.awt.data transfer package.

It is used for all data transfer purposes.

Step 3: Set the content to the clipboard using Toolkit class.

```
Toolkit.getDefaultToolkit().getSystemClipboard().setContents(path,null);
```

Step 4: Press ctrl+v to copy file path from the clipboard.

```
robot.keyPress(KeyEvent.VK_CONTROL);
```

```
robot.keyPress(KeyEvent.VK_V);
```

Step 5: Release the pressed keys

```
robot.keyRelease(KeyEvent.VK_CONTROL);
```

```
robot.keyRelease(KeyEvent.VK_V);
```

Step 6: Press and release enter button

```
robot.keyPress(KeyEvent.VK_Enter);
```

```
robot.keyRelease(KeyEvent.VK_Enter);
```

9. Windows:

Open a new tab

```
driver.switchTo().newWindow(WindowType.TAB);
```

Open new window:

```
driver.switchTo().newWinow(WindowType.Window);
```


Task Scenario: (write script) (interview questions)

Open the browser

Enter ndtv.com

Click on crosswords and click on reveal button

And select word and close the browser.

Task Scenario: (write script) (interview question from Samsung)

Open the browser

Enter flipkart.com

Search for any item of your choice that is search something on search text filed

And click on search button and add to cart.

Verify whether if it is added or not.

Close the browser.

CHAPTER 3

1. Data Driven Testing:

The process of driving the data from external resources like properties file or excel file and utilizing it in the test scripts and performing test execution is called **Data Driven testing**.

Types of data:

1. Common data:

The data which is common to all the test scripts is called common data.

For ex: url, login credentials, configuration settings

2. Test data:

The data which is specific to the test scripts is called test data.

For ex: in amazon → search field

Properties file:

The data is stored in the form of key-value pairs in properties file.

Ex: browser = Chrome

Username=admin

All the values stored will be in String format.

time =10 (10 is also considered to be string)

Pre-requisite:

Create a properties file in the project:

Steps:

Right click on the project → New → folder

Right click on the folder → New → file

Name the file with '.properties' extension

Click on finish.

Steps to read data from properties file:

1. Convert physical file into java readable object.

```
FileInputStream fis = new FileInputStream("file path")
```

The above statement throws FileNotFoundException.

2. Create an instance for properties class

```
Properties property = new Properties();
```

Properties class is present in java.util package

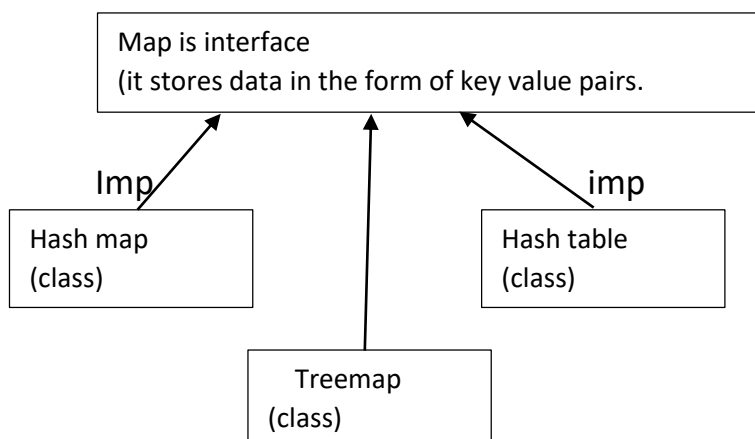
3. Load all the key value pairs to Properties object.

```
Property.load(fis);
```

The above statement throws an exception-IOException.

4. Read data from Properties file

```
String data = Property.getProperty("key");
```



It stores data in the form of key-value pairs.

It does not allow duplicate keys

It allows duplicate values.

It has get() method to fetch the data and it has put() method to write data.

Java Concept used internally in Properties file:

Internally Properties files utilizes Map interface concept to load data to properties object.

When `property.load(fis)` method is called, it internally creates a Hash Table and stores all the data from `fis` into HashTable in the form of key-value pairs.

Hash Table is the implementing class of Map interface.

Drawbacks of properties file:

1. We can fetch only single data at a time.
2. It does not allow duplicate keys.
3. It is not organized.
4. When we have lot of data it is tedious to fetch particular key from the file.

Excel:

In excel the data is stored in the form of tables. It is organized. We store test data in excel.

We use apache poi libraries to read data from excel.

Whenever you want to open Excel in eclipse:

Go to test data excel in eclipse (which is created under test data folder in eclipse) Right click open with system editor. Then only whatever you do on excel it will get saved.

Abstract class

WorkbookFactory
Create()

Interfaces in apache poi

Workbook
getSheet()

Sheet
getRow()

Row
getCell()

Cell
getStringCellValue()
getNumericCellValue()] data



Object Repository:

It is the collection of locators, elements and their respective business libraries.

POM (Page Object Model):

It is java designed pattern preferred by Google to develop an object repository.

Why POM?

1. Handles StaleElementReferenceException.
2. Maintenance of web elements is easier.
3. Modification of web elements is easy.
4. Code can be optimized.
5. Reusability of elements and business libraries.
6. Faster test script development.
7. Increase code readability.

POM has 3 steps:

1. Declaration

We declare the element as private.

```
@FindBy(LocatorName = "LocatorValue")
```

```
private WebElement/List<WebElement>elementRef;
```

@FindBy is used in POM to find element/elements as well as declare the element/elements.

It returns WebElement/List<WebElement>based on the element declared.

Difference between @FindBy and find element?

2. Initialization

Here we call parameterized constructor.

```
public ClassName(WebDriver driver)
{
    PageFactory.initElements(driver,this);
}
```

3. Utilization:

Here we give methods for the declared elements i.e., business libraries.

StaleElementReferenceException:

It is selenium exception which occurs when we try to fetch an element using same old address.

Whenever the web page is relocated its element reference changes and when we try to fetch the element with same old reference then it throws StaleElementReferenceException.

In POM the above exception is handled by the constructor in the second step. When the web page gets relocated, the current addresses are reinitialized to the driver which avoids the occurrence of the above exception.

Java Concept used in POM: (v imp)**

Encapsulation technic is used in POM.

Encapsulation is the process of binding states and behaviors of an object in a class. Data hiding is achieved in encapsulation since we declare states of the object using private keyword.

In POM we declare the elements as private achieving data hiding and access these elements using the respective business libraries in test script.

TestNG: Test Next Generation

- ❖ It is unit testing framework tool used by both developers and automation engineers.
- ❖ Developers use it for White Box Testing.
- ❖ Automation Engineers use it for batch, group, parallel executions of test scripts.
- ❖ testNG is developed as a plugin for IDE(Integrated Development Environment).

We have other framework tools:

1. java= JUnit
2. .net= NUnit
3. Javascript= Jasmine, Mocha
4. Python= Pydev

testNG is developed with combination of features of JUnit and NUnit and has have additional features. It supports both java and .net that is why we prefer testNG.

Advantages:

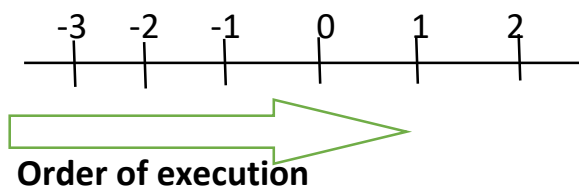
1. It is open source tool- downloading and installation is easy.
 2. Set priority to the test scripts.
 3. Run same test script multiple times using invocation Count.
 4. Disable test scripts.
 5. We can do batch execution.
 6. Parallel execution can be done
 - Distributed Parallel
 - Cross browser parallel/browser compatibility testing
 7. Group execution.
 8. We can Re-run the failed test scripts.
 9. Generates html reports automatically.
 10. Assertions-Validation of test scripts
 11. Create dependency between test scripts-dependsOnMethods
 12. Annotations are used for controlled flow of test execution.
-
- **@Test acts like main method in java.**
 - **Execution in testNG start from @Test.**
 - **Ideally we can have 15 @Test methods in a class.**

How do we prioritize the test scripts:?

To prioritize test scripts, we have a parameter priority = int

Usage: @Test (priority = int)

- Default priority is 0
- Priority follows number line order



- If the priority of the @Test methods is same then it will execute in the order of ASCII values of the method names.

Invocation Count:

In order to run same test method multiple times with same data we use parameter invocationCount = int.

Default invocationCount = 1

Usage: @Test (invocationCount = int)

If invocationCount is 0 or negative value, that @Test method will not be considered for execution.

Disabling the test scripts:

To disable the test script, we use parameter enabled = false

By default, enabled = true

Usage : @Test (enabled = false)

Steps to convert test script class to testNG.xml file:

1. Select the class file and right click on it.
2. Go to testNG and click on 'Convert to TestNG'
3. Rename the xml file with .xml extension and click on 'Finish'
Xml file will be created in the project.
4. Double click on the file to open it.
Xml is the advanced level of html language
Comments = <!--comment-->

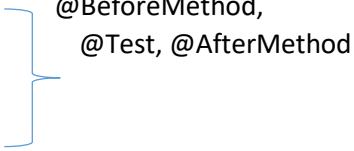
Annotations in testNG:

1. @BeforeSuite : It executes before the <suite> tag in xml.
2. @BeforeTest : It executes before <test> tag in xml.
3. @BeforeClass : It executes before <class> tag in xml.
4. @BeforeMethod : It executes before @Test method in a class.
5. @AfterMethod: It executes after @Test method in a class.
6. @AfterClass: It executes after </class>tag in xml.
7. @AfterTest : IT executes after </test> tag in xml.
8. @AfterSuite : It executes after </suite> tag in xml.

```

<!--BeforeSuite-->
<suite name="Suite">
<!--BeforeTest-->
  <test thread-count="5" name="Test">
    <classes>
<!--BeforeClass-->
    <class name="testNg.EnabledFalsePractice"/>
    <!--AfterClass-->
  </classes>
</test> <!-- Test -->
<!--AfterTest-->
</suite> <!-- Suite -->
<!--AfterSuite-->

```



Batch Execution:

Step 1: Select all the class files which are to be run in batch.

Step 2: Right click on the selected files and convert to testNG.xml file

Step 3: Run the .xml file.

Reports:

TestNG has special feature to generate html reports automatically when .xml file is run.

Step 1: Run the xml file

Step 2: Click on test output folder in the project.

Step 3: Right click on emailable-report.html.

Step 4: Open with web browser.

Step 5: html will have opened.

Re-Run failed test scripts:

When the test scripts fail. Testing creates separate .xml file for these failed test scripts.

Step 1: Refresh the project

Step 2: Expand test output folder.

Step 3: Open test-failed.xml file

Step 4: Fix the defects and re-run test-failed.xml file.

Group Execution:

In order to execute specific group of test scripts testNG has the feature "Groups".

Step 1: We should specify the group for each @Test method as parameter.

Step 2: Convert all the class files to .xml file.

Step 3: Mention the group to be run after suite tag.

```
<groups>
  <run>
    <include>
    <exclude>
  </run>
</groups>
```

Parallel Execution:

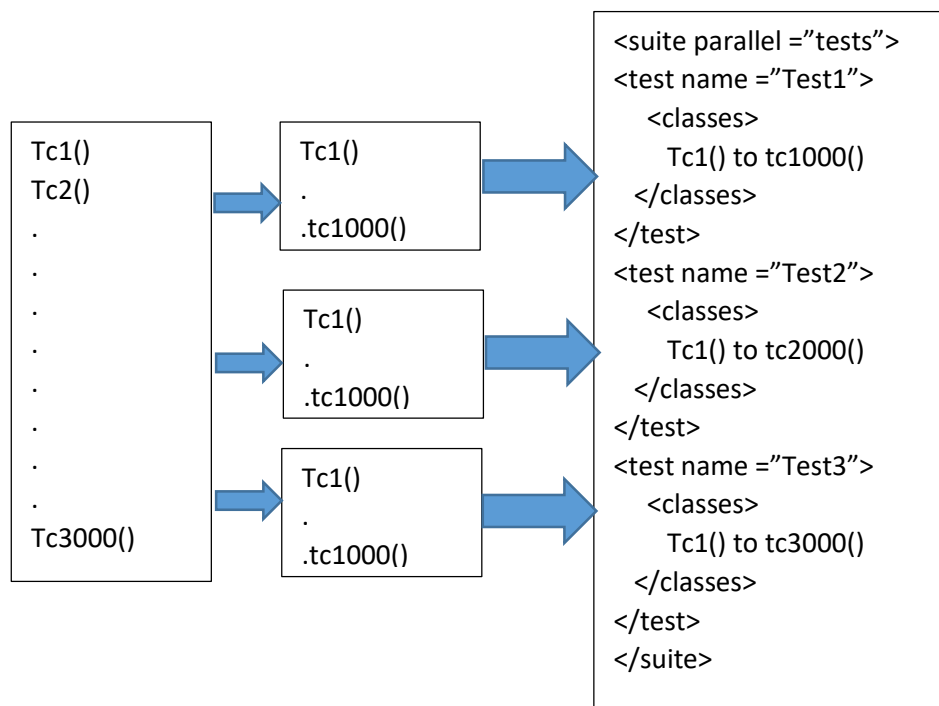
Execution the test scripts in parallel.

1. Distributed Parallel Execution.

Distributing the test scripts to different test runners and executing all the test runners in parallel is called distributed parallel execution.

Ex: 3000 test scripts

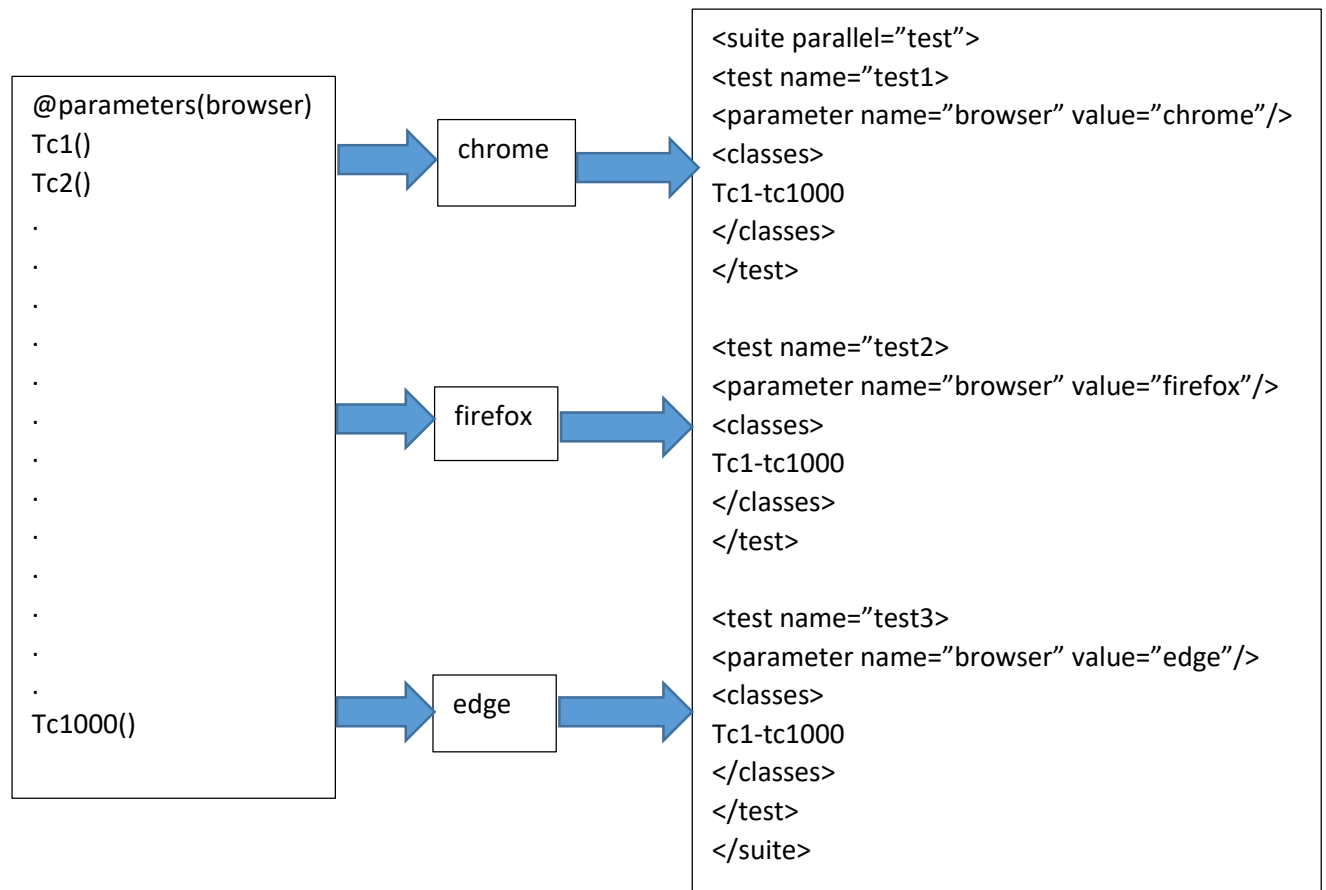
1000 takes 8 hrs



Note: Thread-count should be equal to number of test runners. Default thread-count is 5 and there is not maximum limit for thread-count.

Cross browser parallel execution/ compatibility testing:

Executing same set of test scripts on multiple browsers is called cross browser parallel execution.



Assertions:

It is testing feature which is used to validation or verification in the test scripts.

Why Assertions:

Normal if-else statement does not have capability to fail the test script since depending on the condition mentioned it will either execute if block or else block but never fails the test script. That's why we use Assertions to do accurate validations in test script.

We have 2 types in Assertions:

1. Hard Assert/Assert
2. Soft Assert

1. Hard Assert/Assert:

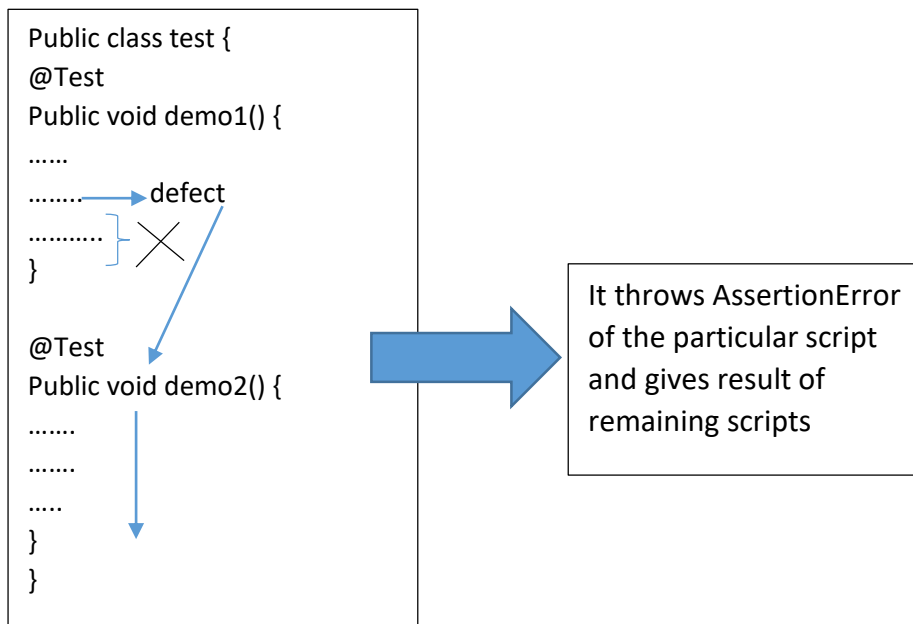
Assert is the testNG class which has static methods to perform validations in test script.

1. **assertEquals(actualValue, expectedValue)**
2. **assertNotEquals(actual, expected)**
3. **assertTrue(condition)**
4. **assertFalse(condition)**
5. **fail()**- to fail the test script.

Usage : Assert.assertTrue(condition);

(any method you can use Assert. (any above methods))

How Assertion works:



When Assert is given it starts with normal execution of test script. When error occurs in one test block it throws Assertion Error and skips the execution of remaining statements in the current block and transfers the control to next block. Finally, it gives the result of execution of all the test block.

2. Soft Assert:

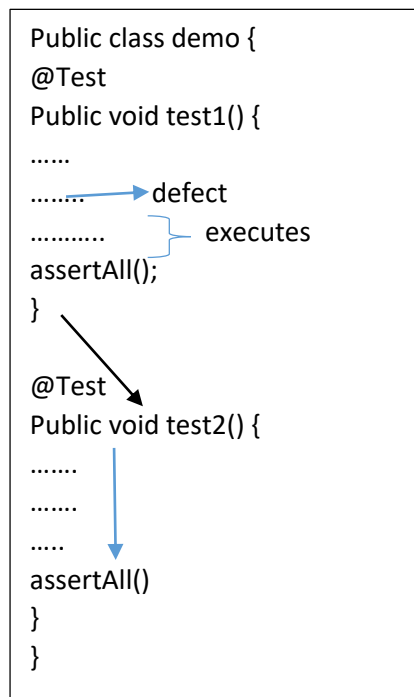
It is a class in testNG which has all non-static methods to perform validations in test script.

1. **assertEquals(actual, expected)**
2. **assertNotEquals(actual, expected)**
3. **assertTrue(condition)**

4. `assertFalse(condition)`
5. `assertAll()` - mandatory method in soft assert and it should be given at the end of the block.

**Usage: `softAssert s = new softAssert();`
`s.assertTrue(condition)`**

How Soft Assertion works:



When softAssertion is given, it starts with normal execution test script. When error occurs in particular block it will still execute the remaining statements in the block and then transfer the control to the next block.

At the end it will throw `AssertionError` of the particular defect and also gives the result of execution of other test script.

Note: `AssertAll()` should be given mandatorily at the end of each block while using `softAssert`.

Difference between HardAssert and softAssert:

HardAssert	softAssert
It contains all the static methods.	It contains all the non static methods.
When AssertionError occurs it skips the execution of the remaining statements in the block	When AssertionError occurs it still executes the remaining statements of the current block.
assertAll() method does not exist	assertAll()method is mandatory and should be given at the end of the block.

Chapter -4

Testcase 1:

Open the browser
Enter skillrary.com
Click on 'GEARS' tab
Click on SKILLRARY DEMO APP
MouseHover to course tab
Select 'Selenium training'
Double click on '+' button
Click on Add to cart
Handle alert popup
Close the browser.

TestCase 2:

Open the browser
Enter skillrary.com
Click on 'GEARS' tab
Click on SKILLRARY DEMO APP
MouseHover to course tab
Select 'Testing' from category dropdown
Drag and Drop 'Junit' course to 'MyCart'
Scroll the page till facebook icon and
Click facebook icon
Close the browser.

TestCase 3:

Open the browser
Enter skillrary.com
Type 'core java for selenium' in search tab
Click on search button
Click on 'Core java for selenium' course
Click on play button
Click on pause button
Click on 'Add to wishlist'
Close the browser.

TestCase 4:

Open the browser
Enter skillrary.com
Click on GEARS tab
Click on SKILLRARY DEMO APP
Scroll till the end of the page
Click on contact us
Enter all the details
Click on 'send us mail'
Close the browser.