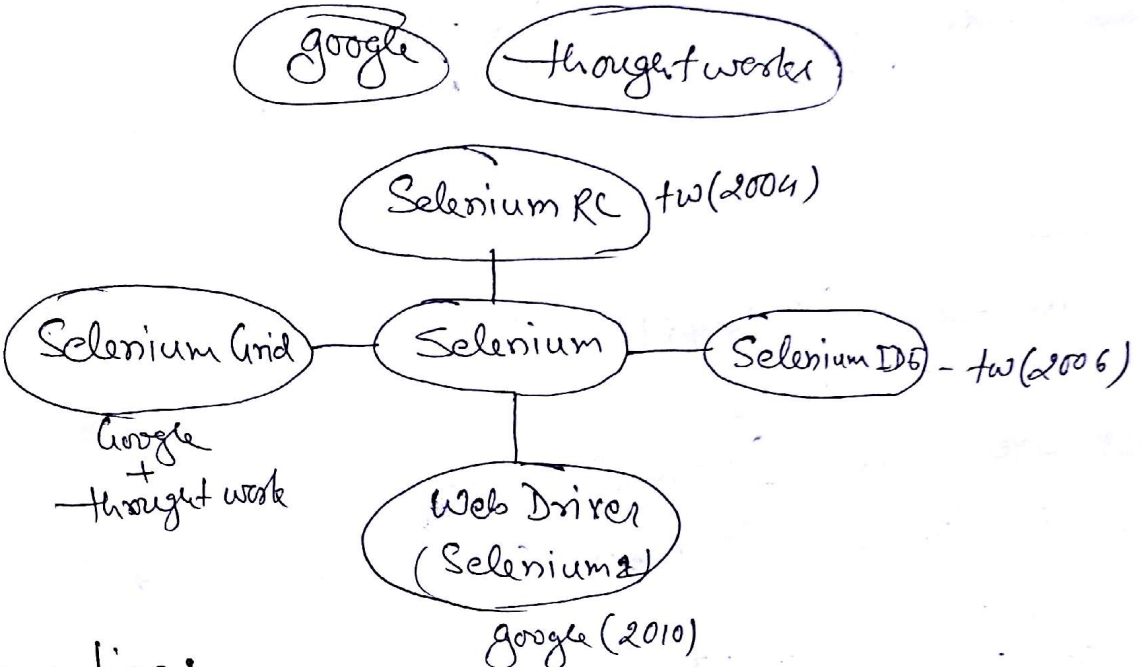
SELENIUM

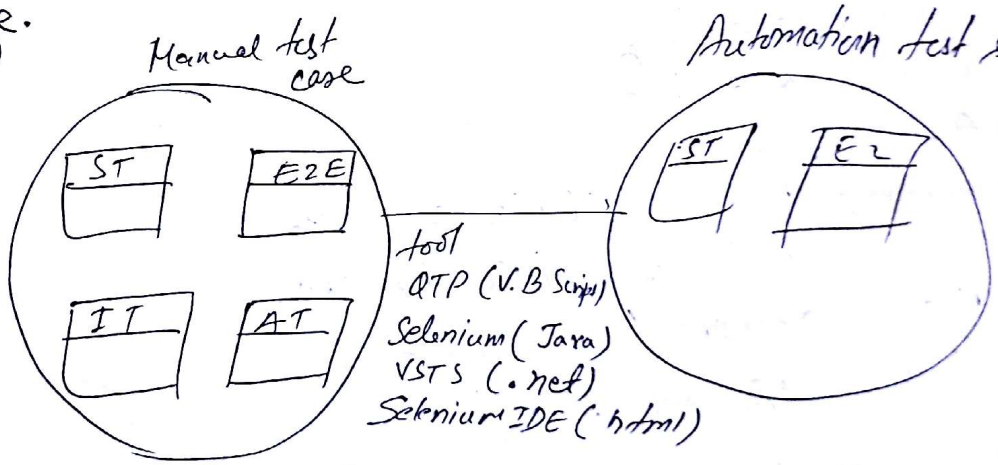
Date 11/04/16
Page

It's an open source community [http://seleniumhq.org/download]



Automation:-

Automation is a process of converting any manual test cases to test scripts using some tools with scripting language.



There are two types of automation tool we have one is functional automation tool other one is non functional automation tool.

Functional
Open Source
- Selenium IDE (html)
- Web Driver (.Java)
- Selenium RC (.Java)
- Appium (Java)
- Soap UI - (Web Services)

Non functional

- Commercial
  - Load Runner
  - RP (Rational performance)
  - VSTS
- Open Source
  - J Meter
  - Neo Load.

Cause content

- Selenium IDE - [2 days]
- html [1 or 2 days]
- Object Identification [4 days]
- Basic knowledge on Selenium RC
- Web Driver
  - Basic methods of webdriver
  - Basic operation on webElement
  - Wait statement
  - Action/Select class.
  - Window handeling
  - Frame handeling
  - Auto suggest.
  - Working with multiple elements
  - Working with multiple browser
  - File attachment / File download.
  - Unit testing tool → testNG, Junit
  - Framework Implementation with Real time project.
    - hybrid driven framework.
  - SVN
  - DB connection.
  - maven (build testing tool)
  - Jenkin (CI-tool)

SELENIUM IDE

→ Selenium IDE is a functional automation testing tool which support only web based application.

→ Selenium IDE is a UI based, record and playback tool.

→ Selenium IDE uses html language (Selenese lang) to generate a test script.

→ Selenium IDE can record all the user action performed on web element.

→ Selenium IDE can only record action performed on web element, keyboard and mouseover movements are not recorded.

→ Selenium IDE supports only firefox browser, because which is implemented as plugin for the firefox browser.

Installation Steps of Selenium IDE

1) In firefox browser go to google.
2) Search for download Selenium.
3) Click on first link and navigate to selenium community.
4) Find Selenium IDE division in community website and click "from addons.mozilla.org" link.
5) Click on Add to Firefox button.
6) Click on install in the popup window.
7) After installing click on restart browser.
8) In order to open Selenium IDE tool go to firefox press alt tools option will be available. Click on tools → Selenium IDE

→ Selenium IDE typically uses 3 steps to automate test script.

① Selenium IDE Command
② Target (locators)
③ Value.

Commands are used to perform some specific action on webelement.

Commands	WebElement
type / sendkeys	Edit Box.
click / click And Wait	Button
Select	Dropdown / Weblist
Click	Link
Click	Checkbox
Click	Radio button

LOCATORS :- / TARGET

Locators is Identify any WebElement in UI based on WebElement html source code.

We can identify an WebElement by using below attribute

1) Id
2) name
3) Xpath
4) CSS - Selector (Cascaded style sheet - color of the object)

Date 13/04/16
Page

Firebug

Firebug is an open source plugin which supports only firefox browser. Firebug is used to view the html source code of the specific element in AUT/UI
↓
Application under test

Istallation steps of firebug

1) Go to google search for download firebug.
2) Click on first link navigate to firebug community
3) Click on add to firefox button.
4) Click on install button and restart the browser.

→ In order to open firebug press F12. or click on bug symbol in the top right of the browser.

* Write a login logout Selenium script for Acti Time Application.

Command	Locator	Value
type	name=username	admin
type	name=pwd	manager
clickAndWait	//input[@typ='submit']	
click	//img[@alt='Logout']	

Xpath syntax :- //htmlTag[@att='value']

Ex ↳ //input[@type='submit']

//img[@alt='Logout']

→ for opening the application we will use below command

command	Locater
open	http://qsp-tyy-24-pc/login.do

Points:- 1) Whenever we open Selenium IDE by default IDE is in recording mode.

2) If the button is partially red - that means its in recording mode.

3) Selenium IDE does not have capability to launch new browser and run the test script.

4) Open command is used to navigate to any web application URL.

5) Whenever any navigation action is performed better practice to use clickAndWait instead of click Method.

6) clickAndWait command is used to perform click operation and wait till entire page to load.

7) Selenium IDE will always run the test case in current active browser or Tab.

Batch / Suite Execution

→ Collection of multiple test script is called batch, execute multiple test script in a single click is called batch execution.

Steps to create test suite

1) Create new test script File → New Test Case

2) Automate test script and save test with .html extn
File → Save Test Case

3) Create a new test suite File → New Test Suite

4) Add multiple test case to suite In IDE right click on Test Case button in left pannal and click on add test case and browse all the test case from local system.

Execute test suite

Use ▷= button to execute test suite

* Save test suite with .html extn.

Firepath

1) Firepath is implemented as plugin for firebug tool.
2) Firepath is used to view the html source code ~~to~~ of an web element and also we can evaluate our custom xpath in browser itself.

Installation Steps

1) Go to google search for download firepath.
2) Click on first link navigate to firepath
3) Click on Add to firefox button.
4) Click on install button and restart the browser.

→ In order to use firepath tool, open firebug and click on firepath button in firebug menu bar.
→ Write own firepath and click Evaluate button to cross verify.

CHECKPOINT

Date:- 14/04/16

Checkpoint is a feature available in selenium IDE which is used to verify the expected result of the test case.

There are two types of checkpoint:-

1) Verify checkpoint
2) Assert checkpoint.

Checkpoints commands are verification command, cannot be recorded by IDE. In order to verify the expected result we should insert checkpoint command manually.

Verify Checkpoint

- Verify title
- assert title.

Manual test case

Verify Invalid Msg

Action	Expected Result
Navigate to actiTime applicatio	Login page should be displayed
Click on Login button with wrong user name and password	Verify Invalid msgs

Case1 :- When verify checkpoint fails.

TC-01

Step 1

Step 2 x → Selenium IDE generate the error message for the current step and continue the remaining test script execution.

Step 3

Step 4

Case2:- When assert checkpoint fails:-

TC-01

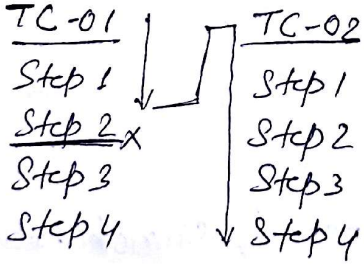
Step 1

Step 2 x → Selenium IDE generate the error message for the current step and stop current test script execution.

Step 3

Step 4

Case3:- When assert checkpoint fails in suite execution.



Selenium IDE generate error msg for the current test and continue execution with next test script.

* Text checkpoint.

Text checkpoint is to use to check any text available in the page

→ Verify Text Present

→ Assert Text Present.

Negative test case (invalid password & invalid msg)

Command	Target	Value
assertTitle	actiTIME - Login	
type	name = username	admin
type	name = pwd	xyz
verifyTextPresent	Username is invalid	
captureEntirePageScreenshot	D:\Users\dsp-tyy\OCMG\test.png	

While writing use all \\ for all

Give the target as the folder location where we have to save the screenshot. Along with the location add the screenshot name.

→ Text checkpoint is used to verify any visible text in UI.

→ Text checkpoint navigate to entire html document and test for the expected text in the UI, it return true if expected value available in UI or else return false.

→ Title checkpoint is used to verify only title of the page.

Advantage of Selenium IDE

1) UI based record/playback tool.
2) Easy to automate and understand the test script.
3) Using Selenium IDE tool we can convert/export selenium IDE test script to webdriver code or RC code.

File → Export test case as → Java/testNG/Webdriver.

Save file with .java extension.

Disadvantage of Selenium IDE

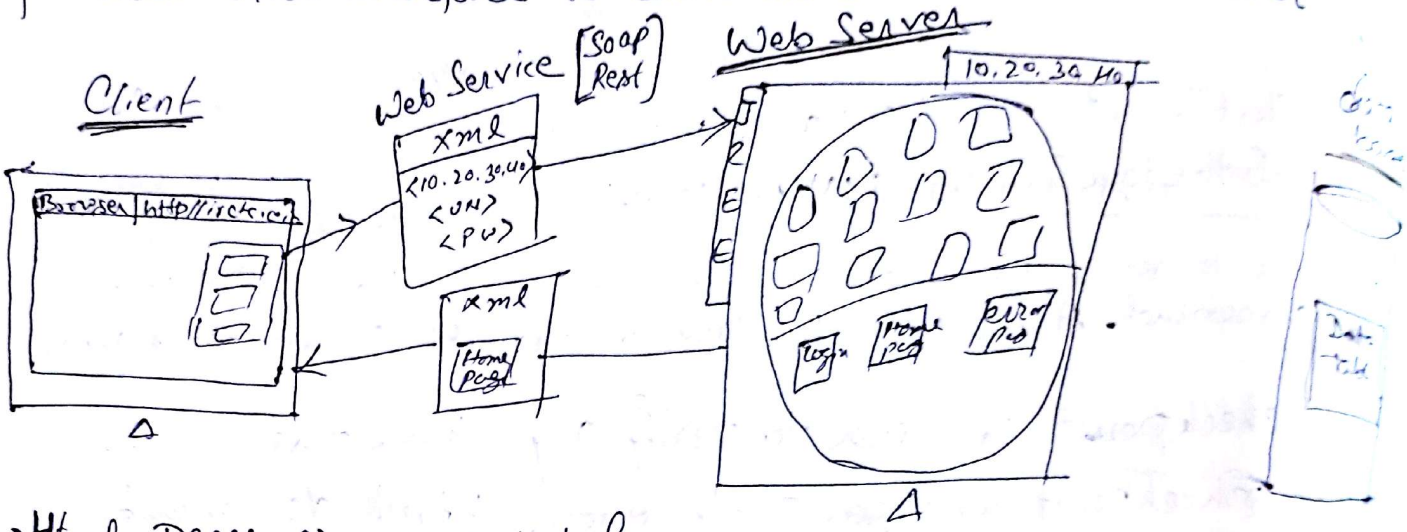
1) Works only with firefox.
2) Can not launch new browser and run the test
3) Can not insert conditional and loop statement.
4) Selenium IDE doesnot support ajax application.
Application which contains lot of dynamic object is called ajax.
Ex:- flipkart, Gmail
5) IDE doesnot support password encryption/decreption. bz All selenium tool

HTML

Date 15/04/16
Page

Hyper text markup language

html language is used to design web pages, html pages provide user interface to communicate to web server



→ Html pages are user interface between client and web server.

→ Html language is a collection of html tag, each html tag should be enclosed with angular braces. < >.

<html>	<h>
<title>	<select>
<body>	<table>
<input>	<a>

→ Each and every open Html tag should have end tag

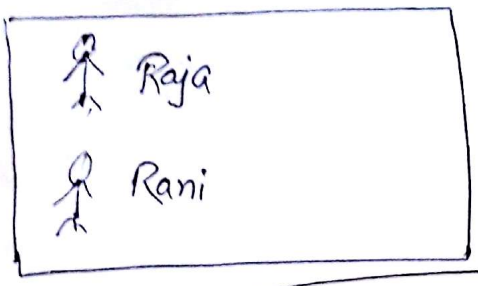
<html> → open tag

</html> → end tag.

or

<input/> → open and closed in same.

→ Html tag may or maynot contains backend attributes, backend attribute should be written within angular braces.



→ backend Attribute

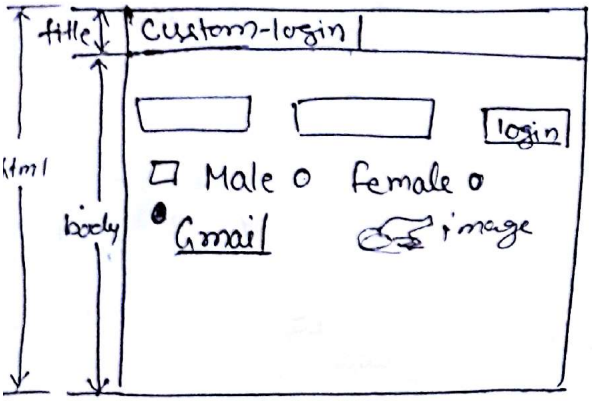
<Person type='men' id='01' name="Raja" />

<person type='women' id='02' name="Rani" />

→ Html tag may or may not contain visible text, visible text always written outside the angular braces.

<person type='men' id='01' name='Raja'> Raja </person>

<person type='women' id='02' name='Rani'> Rani </person>



<html>
<title> custom-login </title>
<body>
<input type='text' id='u1' name='username' />
<input type='password' id='p1' name='pwd' />
<input type='button' id='b1' value='login' />
</body>
</html>

* Use placeholder attribute if we want to show some text inside the input box.

<input type='checkbox' id='check' />
<label> Male </label>
<input type='radio' name='rd' id='rd1' />
<label> FeMale </label>
<input type='radio' name='rd' id='rd2' />

Link → Gmail

* In src we have to give the location if html and image are in different location.

Dropdown

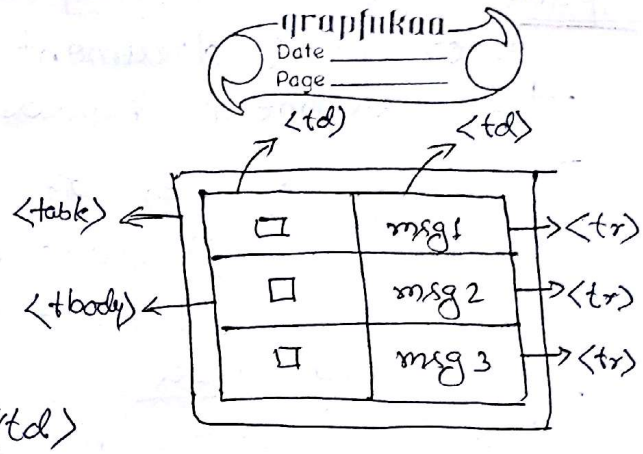
```
<select name = 'sel'>
    <option> select country </option>
    <option> india </option>
    <option> eng </option>
    <option> Aus </option>
    <option> Pak </option>
</select>
</body>
</html>
```

Date:- 18/04/16

Webelement	htmltag	mandatory att	value	additional	value
EditBox	input	type	text/ password	id name value	custom value
Button	input	type	button/ Submit	"	"
Checkbox	input	type	CheckBox	"	"
RadioButton	input	type	Radio	name att should be same for group	"
Webtable	<table> <tbody> <tr> <td>	N/A	N/A	Id name value	"
Dropbox	select option	N/A	N/A	"	"
Link	a	href	URL	"	"
Image	img	src	location of the image	"	"

To design a Table

```
<html>
<title> Custom page </title>
<body>
<table name = 'tab1' border = 'b'>
<tbody>
<tr>
<td> <input type = 'checkbox'/> </td>
<td> <a href = "https://msg1.com'> msg1 </a> </td>
</tr>
<tr>
<td> <input type = 'checkbody"/> </td>
<td> <a href = 'https://msg2.com'> msg2 </a> </td>
</tr>
</tbody>
</html>
```



Header → <h1> </h1>

Pragraph → <P> </P>

division (dividepage) →

Color →

Object Identification

All selenium tool will identify the object based on four locators, locators are written looking at the webelement Html source code.

four locators :- id, name, xpath, css selector.

id :→ Used to identify the webelement using Id backend att.

name :→ Used to identify the webelement using name attribute.

Xpath :→ Used to identify the webelement using any attribute.

Css :→ Used to identify the webelement class (or) Id attribute.

Xpath:- Xpath is a way of navigate to enter html document and identify the webelement based on webelement attributes.

There are 2 types of Xpath

1) Absoulute Xpath
2) Relative Xpath.

While writing xpath

// → Go to enter html source code
/ → Go to child node [html tag]
[] → Go to parent node.
Provide backend for the webelement.
@ → attribute symbol.
* → match any html tag.

Note: → In real time application most of the elements will never find Id & name attribute. In such case we have to go for xpath to identify webelement.

Absolute Xpath

Case 1

```
<html>
<title> Custom page </title>
<body>
  <h1> welcome to Amazon </h1>
<input type = 'text' name = 'username'/>
<input type = 'text' name = 'pwd'/>
<input type = 'button' name = 'pwd' value = 'login'/>
</body>
</html>
```

Ex

```
//html/body/input[1]
//html/body/input[@name = 'username']
//html/body/input[2]
```

* Whenever Xpath is written from route //html tag to child html tag followed by / is called absolute Xpath.

→ whenever we inspect webelement using firepath tool its always generate absolute xpath which can't be used in realtime selenium script. because absolute xpath will failed to identify the object. whenever object location or requirment changes.

Relative Xpath:-

Syntax:- `//html tag [@att = 'value']`

Ex:- //input [@name='username']

//input [@name='pwd and @type='text']

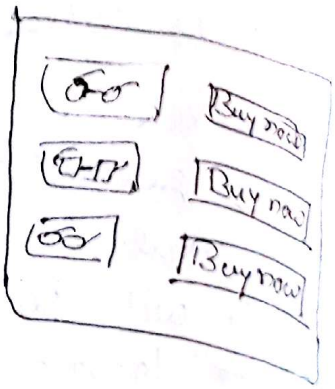
Whenever Xpath is written directly to ~~seper~~ specific webelement is called relative xpath.

→ relative Xpath never failed to identify the object even though object location changes.

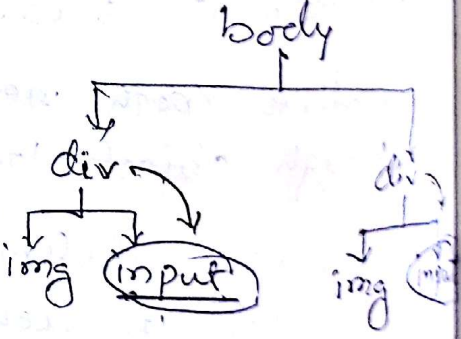
→ whenever webelement not able to identify using one attribute in such case we can provide multiple attribute using 'and' ~~Symbol~~ keyword.

Case2 :- How to identify similar objects present in UI

Here Buy now is similar for all.




```
<html>
 <title> custom-login </title>
 <body>
   <h1> welcome to amazon </h1>
 <div class = 'fs'>
    <img src = 'image.png' />
    <input type = 'button' value = "buy now' />
 </div>
    <div class = 'rb'>
      <img src = 'image1.png' />
      <input type = 'button' value = 'buy now' />
    </div>
 </body>
</html>
```



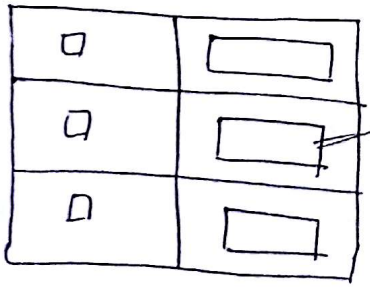
To identify fastbach(fs) buy now button

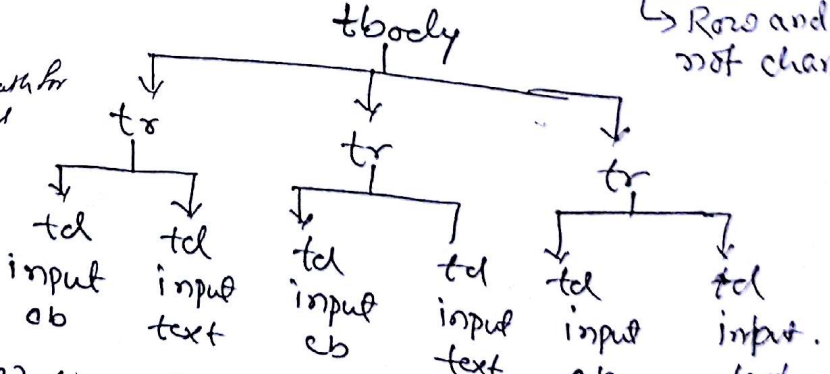
//div [@ class = 'fs'] / input [@ value = 'buy now']

To identify rayban(rb) buy now button

//div [@ class = 'rb'] / input [@ value = 'buy now']

Case3 Working with static webtable





Static webtable
↳ Row and column not changing

Xpath //tr[2]/td[2]/input

row index

column index.

Point for Case 2

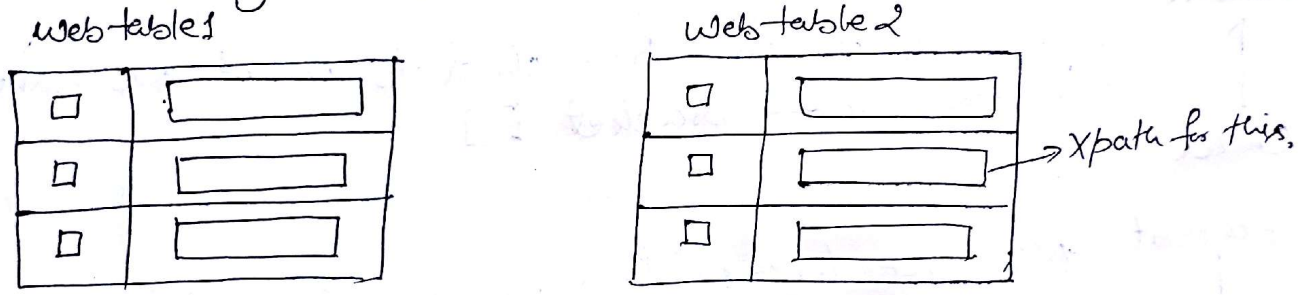
Whenever web element not able to identify using one or more attributes in such case we can take help of parent and grandparent to identify the object uniqly.

Point for Case 3

In order to identify the webelement available inside a static table better practise to use row and column index of the webtable.

```
//tr[row index]/td[column index]
```

Case 4 Working with multiple webtable




```
//table[@name='tab2']/tbody/tr[2]/td[2]/input
```

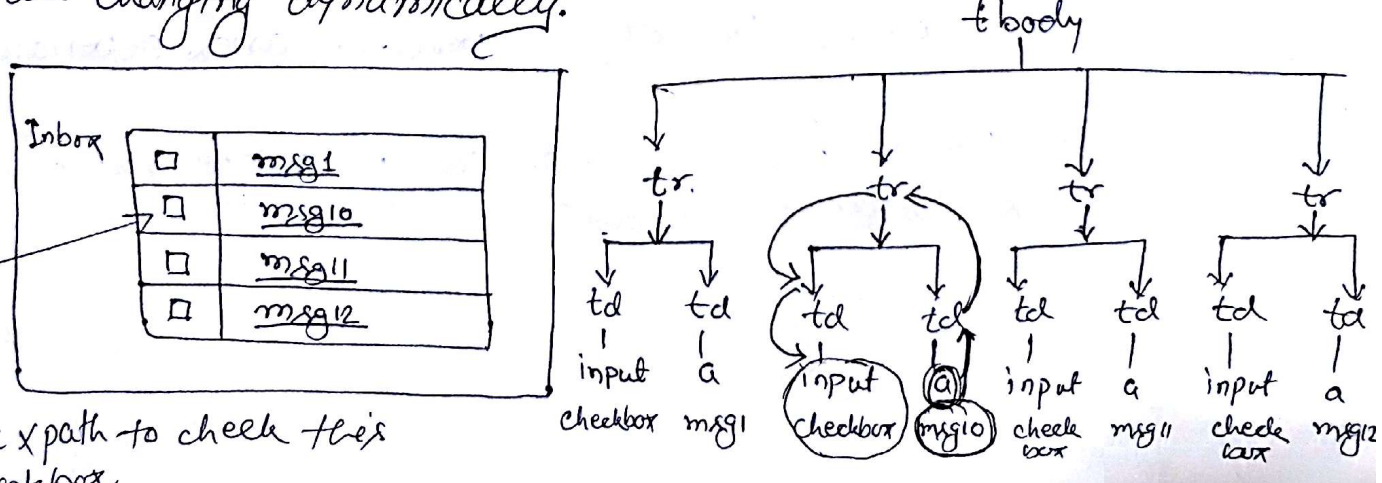
Date:- 20/04/16

Case 5 Working with dynamic webtable

→ The content of the table is changing dynamically.

Ex: of dynamic table

1) Gmail Inbox table is dynamic because the messages are changing dynamically.



We can easily write xpath for the link as there is unique attribute (href) but we cannot write xpath to select the checkbox as it's changing dynamically. If someone delete the previous msg then the row number will change for second message.

Here xpath for msg10 link is

//a[@href='http//msg10.com'] → this will identify the msg10 link.

Now xpath for checkbox

//tr[td[a[@href='http//msg10.com']]]/td/input.

in above xpath for child td we can give index also.

Parent
↑
Child

When we traverse from child to parent we have to use square bracket [ ]

Parent
↓
Child

When we traverse from parent to child we have to use forward slash /

Point: Steps to work with dynamic webtable

1) Find an independent (unique) element and dynamic element in a webtable.
2) Write down the tree structure for both independent and dynamic element in the webtable.
3) find a common parent for both independent and dependent element.
4) Construct the xpath (by taking reference of independent element try to identify dynamic element).

Xpath Function

1) text()
2) normalize-space()
3) Contains()
4) Following-Siblings()
5) Decendent()

1) text()

Syntax:- `text() = 'Expected value'`

how to use in xpath.

`//htmlTag [text() = 'Expected value']`

Ex

```
<html>
  <title> custom-login </title>
  <body>
    <h1> welcome to amazon </h1>
    <h1 class='hd'> amazon product list </h1>
    <input type='text' name='search' />
    <input type='checkbox' name='search' />
    <a href='http://gmail.com' id='_link'> Gmail </a>
  </body>
</html>
```

→ `<h1> welcome to 10.1 million users </h1>`

Xpath:- `//h1 [text() = 'welcome to amazon']`

`//a [text() = 'Gmail']` or `//a [@href = 'http://gmail.com']`

Write xpath for each element of above html

1) Header 1

`//h1 [text() = 'welcome to amazon']`

2) Header 2

`//h1 [text() = 'amazon product list']` → Wrong

text() function cannot ignore space before and after string.

~~class = hdl~~
// ~~h1~~ [@ class = hdl] → Correct.

3) Search textbox
// input [@ type = 'text' and @ name = 'search'] → Correct
// input [@ type = 'text'] → wrong.
Mandatory attribute alone should not be used to identify editbox or radio button or checkbox.

4) Search checkbox.
// input [@ type = 'checkbox' and @ name = 'search'] → Correct
// input [@type = 'checkbox'] → Wrong.

5) Link
//a [text() = 'Gmail'] → Correct
//a [@ href = "http://gmail.com'] → Correct
//a [@id = 'link'] → wrong (As space is given in id)

6) Header 3.
//h1 [text() = 'million user'] → Wrong
text function cannot identify ~~x~~ Partial (part) of the string.
using
//h1 [text() = 'welcome to 10.1 million user'] :- Wrong
text() function cannot identify the dynamic ~~elem~~ object here 10.1 is dynamic text.

Points for text() function

Date:- 21/04/16

1) text() function is used to identify any web element using visible text.
2) text() function navigate to entire html document and check for the expected value in UI, if expected value completely matches with UI text it return true or else false
3) text() function is complete pattern matching function, it cannot match part of the string
4) text() function cannot ignore the spaces before and after the string.

5) text() function cannot identify dynamic text.

2) normalize-space()

Syntax- normalize-space(text()/@attribute)

//HtmlTag [normalize-space(text()/@attribute)='Expected value']

Ex Xpath for Header:

//h1[normalize-space(text())='amazon product list']

Ex Xpath for Gmail login and Header

① //h2[@class='hidden-sm al']

② //h2[normalize-space(text())='Sign in to continue to Gmail']

Ex Xpath for Link in previous amazon example

//a[normalize-space(@id)='link']

Ex Xpath for Show button in actiTime application

//input[normalize-space(@value)='Show']

Points:-

1) normalize-space() can ignore space before and after the string or visible text. not inbetween the string.

2) normalize-space() can also ignore the spaces of any backend attribute of the web element.

3) Contains()

Syntax:- contains(text()/@attribute, 'Expected value')

//HtmlTag[contains(text()/@attribute, 'Expected value')]

Ex Xpath for Header3 in amazon

//h1[contains(text(), 'llion users')]

> part of text million users.

To use start of the string and end of the string

//h1[contains(text(), 'welcome') and contains(text(), 'user')]

//input[contains(@value, 'RS')] ⟶

₹499 RS

Price can change ...

Points:-

1) Contains function is not a complete string matching function it can identify the object by using part of the string or part of the word or complete string.

2) Contains() function navigate to entire html document and check for the expected value, if expected value partially matches with UI it returns true or else false.

3) Contains() function automatically ignore the spaces before and after the string.

4) Using contains() function we can identify any dynamic object.

5) Contains() function play major role to identify the object in ajax application.

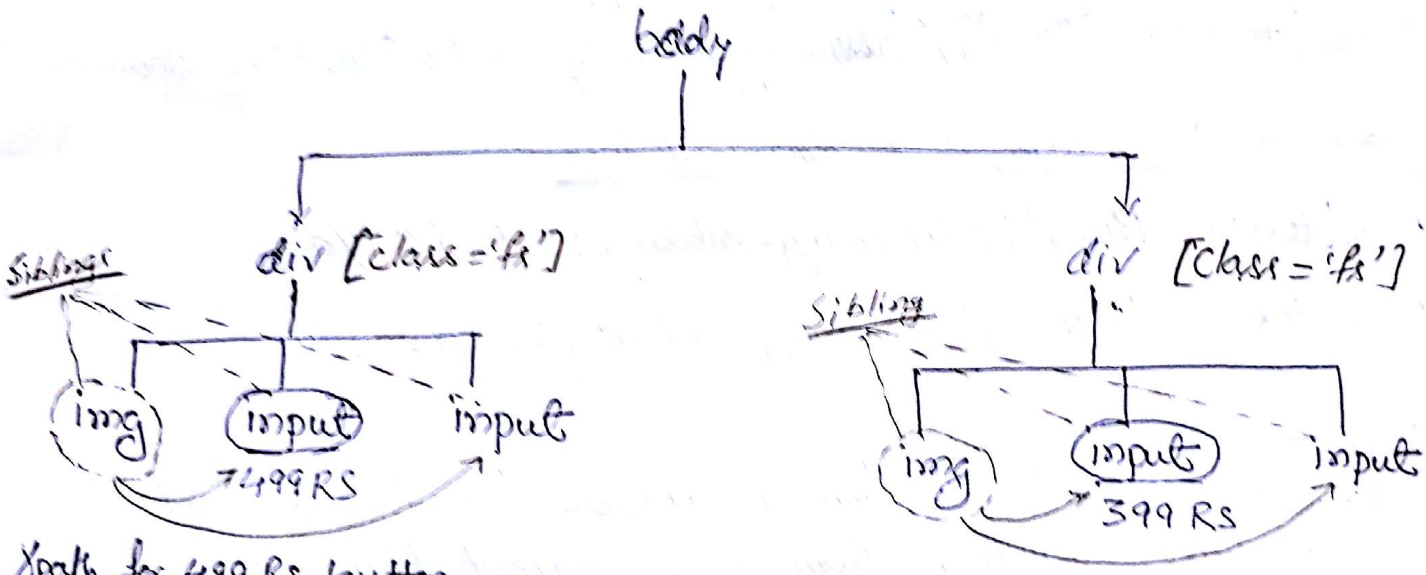
4) following-sibling()

Syntax //following-sibling::

:: → Scope resolution operator

//htmlTag/following-sibling::

```
<html>
  <title> custom_login </title>
  <body>
    <h1> Welcome to amazon </h1>
  <div class='fx'>
    <img src='image.png'/>
    <input type='button' value='499 Rs'/>
    <input type='button' value='cancel'/>
  </div>
   <div class='fx'>
    <img src='image1.png'/>
    <input type='button' value='399 Rs'/>
    <input type='button' value='cancel'/>
   </div>
```



Xpath for 499 Rs button

```
//img[@src='image.png']/following-sibling::input[contains(
                                    @value,'Rs')]
```

Xpath for cancel button beside 499 Rs

```
//img[@src='image.png']/following-sibling::input[contains(@value,
                                    'cancel')]
```

Xpath for 399 Rs button

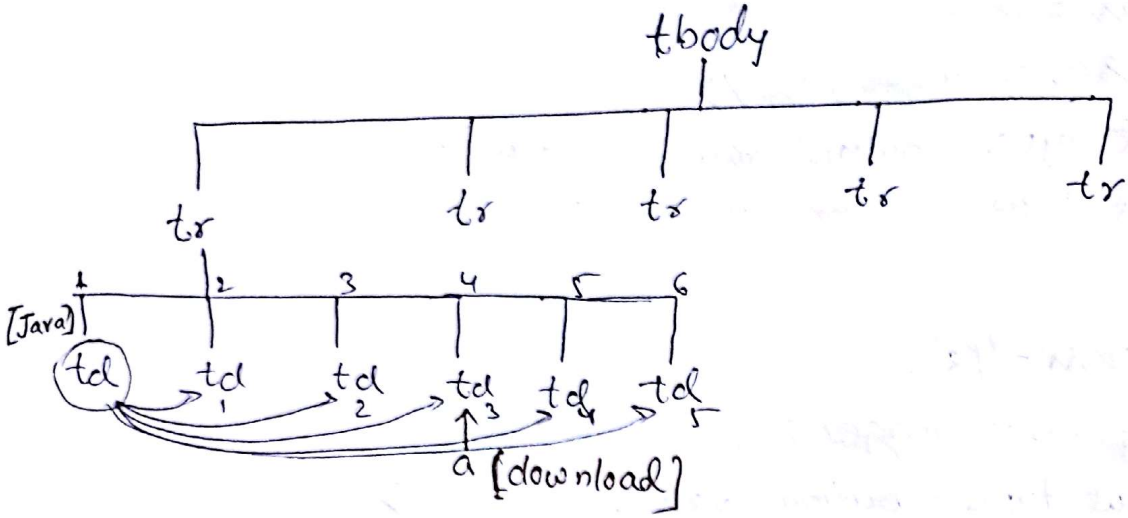
```
//img[@src='image1.png']/following-sibling::input[contains(@value,
                                    'Rs')]
```

Xpath for cancel beside 399 RS

graphikaa
Date
Page

//img[@src='image1.png']/following-sibling::input[@value='cancel']

Ex:- Xpath for download link for java in seleniumhq.org/download



//a[contains(@href,'selenium-java')]

or

//td[text()='Java']/following-sibling::td[3]/a

or

//td[text()='Java']/following-sibling::td[text()='download']

Xpath for Ruby and C# download link

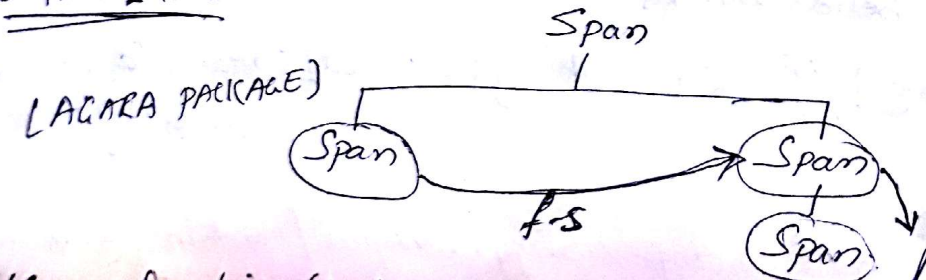
//td[text()='Ruby']/following-sibling::td[3]/a

//td[text()='C#']/following-sibling::td[3]/a

Points:-

1) following-siblink function is used to identify next immediate siblings from the current html tag.
2) whenever webelements changes its attributes completely, in such cases we take help of siblink to identify the dynamic objects.

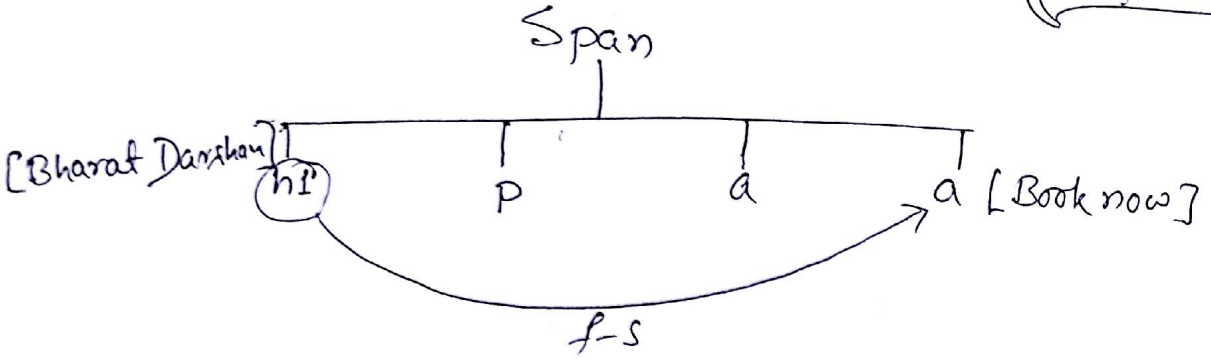
Ex from IRCTC



//span[contains(text(),'AGARA PACKAGES')]/following-sibling::span

* From IRCTC

Xpath for Book now button for Bharat Darshan.



~~//h1[text()='BHARAT DHARSHAN']/~~

//h1[text()='Bharat Darshan']/following-sibling::a[text()='Book Now']

Interview questions on X-path and IDE

Date 25/04/16
Page

1) What is the difference bet^n relative and absolute X-path.

2) How to identify webelement when element present inside the webtable.

→ If static - td, tr
If dynamic - take unique column as ref then dynamic element

3) How to identify dynamic webelement in UI.

→ if dynamic changing partially - use contains
If dynamic changing completely - use following-sibling element.

4) What is Locator and how many locators are there in Selenium ⇒ and which locator is preferred.

→ id
name
Xpath
CSS Selector

5) Difference between Verify and Assert checkpoint?

6) Difference between Selenium IDE and WebDriver?

SELENIUM RC

→ Selenium RC is not a UI based and Record and Play-back tool.

→ Selenium RC is just a collection of Java-Script methods.

→ Selenium RC supports multiple browser. (Ex FF, IE, Chrome, Safari, Opera...)

→ Selenium RC supports multiple OS/platform - Windows, Linux, Mac-OS.

→ Selenium RC supports multiple language (Java, .net, Perl, python, Rubby, PHP)

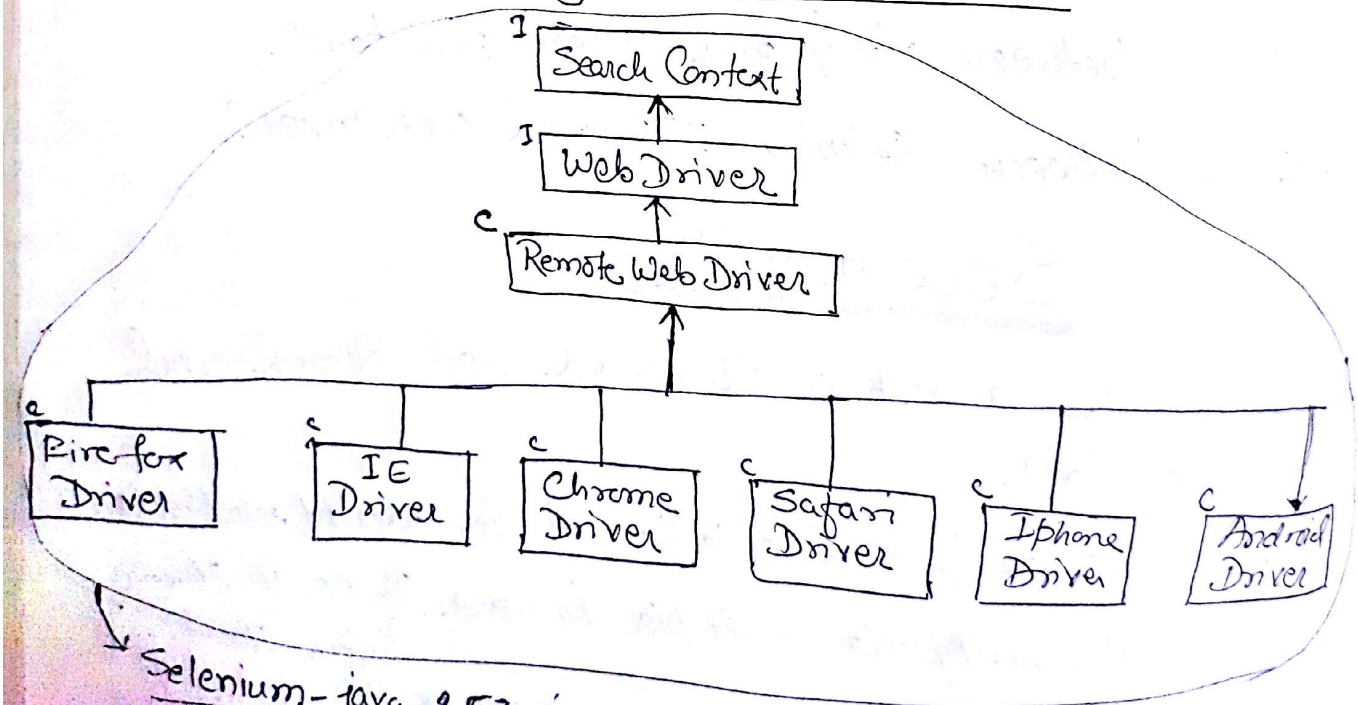
Disadvantage of Selenium RC

→ Selenium RC was failed to support secure network, because secure browser will not allow other java script to work on secure browser.

WebDriver

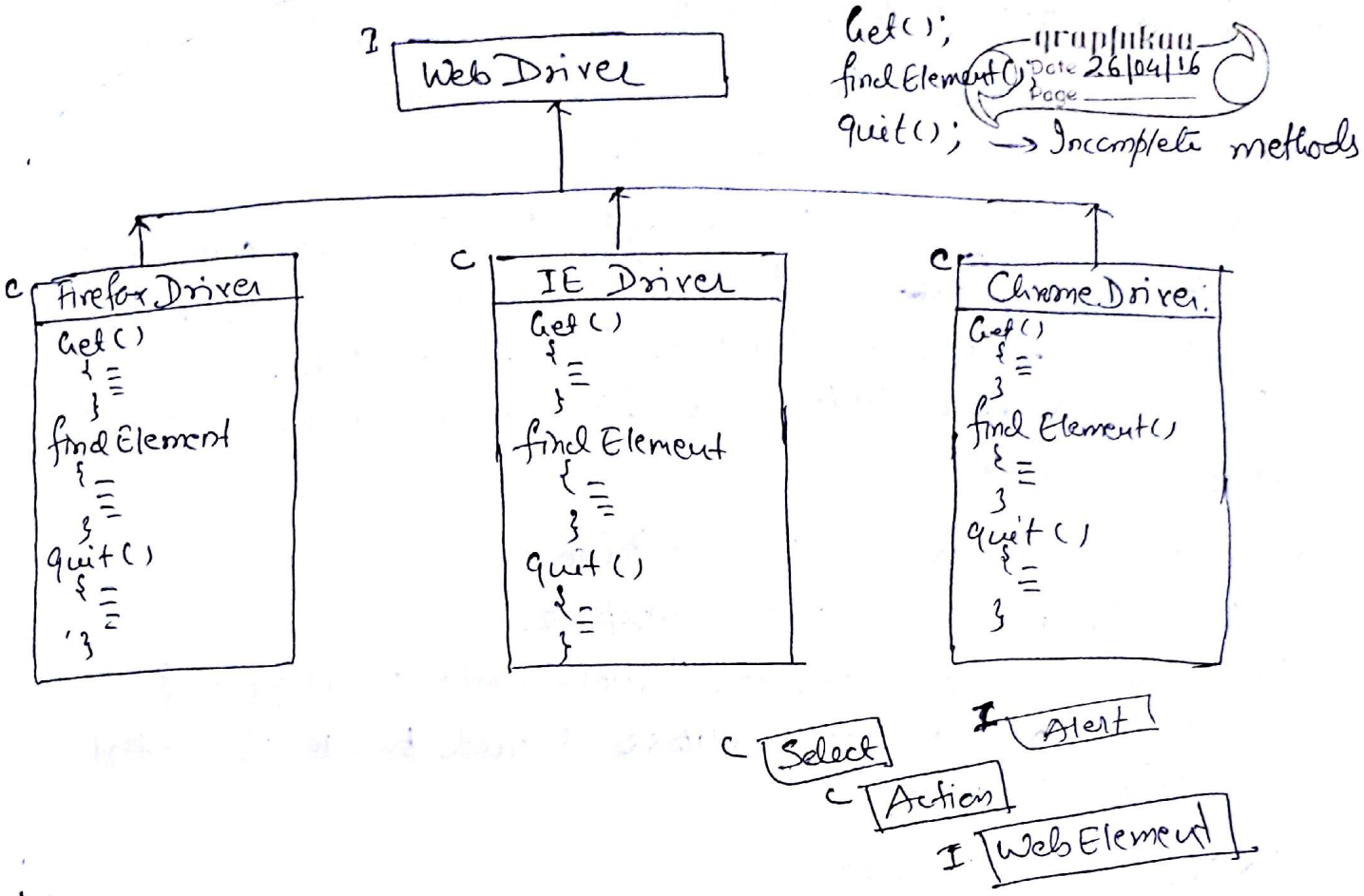
1) WebDriver is not a UI based & Record/playback tool.
2) WebDriver is a just collection of core-Java API's.
3) Support multiple browser — firefox, IE, Chrome, Safari, Opera.
4) Support multiple OS/platform — Windows, Linux, mac-OS, IOS, Android.
5) Support multiple language - Java, .Net, Perl, Rubby, Python, Php.
6) WebDriver supports secure network also.

Class diagram of WebDriver



Selenium-java-2.53.jar
- .class
+ jar
+ jar
+ jar

1 Webdriver is a collection of Interface and class.



Note:- Technicaly web driver is an interface All browser classes implements webdriver incomplete method.

→ All browser classes are subclass of webdriver Interface.

→ WebDriver is a collection of classes and Interface/collection of .jars.

→ WebDriver is not a UI based tool, its just a .zip folder, In order to write a webdriver testScript we will take help of Eclipse platform.

→ All api's of WebDriver are purely object oriented.

application peripheral Interface

→ WebDriver supports all native browser, so that webdriver can directly communicate to browser.

Installation Steps of WebDriver

1) Download WebDriver jars
   - ↳ Go to google search for selenium.
   - ↳ Click on first link to navigate to Selenium community
   - ↳ In selenium community find WebDriver language binding division. Click on download link beside java.
   - ↳ Download .zip folder and place in C Drive.
2) Extract zip folder.
3) Include webdriver jars to Eclipse
   - ↳ Open Eclipse with new workspace.
   - ↳ Create new Java project. (Note:- while creating a java project make sure Eclipse should point to latest version of JDK.
   - ↳ Create new Package :- Src → New → Package
   - ↳ Create new Java file. Package → New → Java file.
   - ↳ Select source folder → Right click → Build path → Configure build path → Click on Library button
   - ↳ Click on Add External jar button and include all the jars from C Drive / Local system.
   - ↳ Now write Webdriver code.

Sample Code

```
Public class SampleSeleniumTest {
Public static void main (String [] args) {
   // Launch empty firefoxDriver browser
WebDriver driver = new FirefoxDriver ();
   // navigate to any web app URL
driver.get ("https://gmail.com");
driver.quit ();
  }
}
```

* Writing WebDriver Test Script typically uses 5 steps

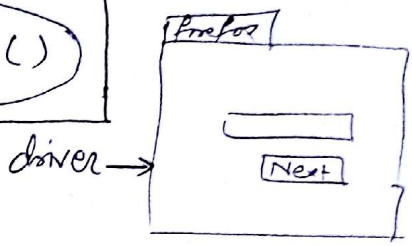
Step1 :- Launch Empty browser.

Note:- whenever we create instance or object to any browser class, automatically empty browser is getting launch and give the control of the browser to driver reference variable.

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

or

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



Step2 :- Navigate to any web based application using get() method.

Note:- URL should be start with http/https, because WebDriver uses http protocol for internal browser communication.

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

Step3 :- Find Email editbox in UI.
```
WebElement wbUs = driver.findElement(By.id("Email"));
```

Step4 :- Perform an action on webElement
```
wbUs.sendKeys("qspider.selenium");
```

Step5 :- Close browser
```
driver.quit();
```

Locators Available in WebDriver

graphikon Date 27/06/16 Page

*Locators are used to identify any webElement in UI based on html source code.

- By
  - Static Method
    - id()
    - name()
    - Xpath()
    - CSS-Selector()
    - Link Text()
    - Partial Link Text()
    - Tag Name()
    - Class Name()

→ These 8 locators are static method available in By class.

* Driver.findElement(Locater)

* In WebDriver findElement method is used to identify an object, there are 8 ways we can identify the object in UI.
* findElement Method takes locators as an argument, it always navigate to entire html source code check for the expected webElement (expected locater) in UI.
* If object is available in UI, findElement method always return webElement or else throws NoSuchElement exception and stop the entire execution.
* HTML → `<input id=usr name='username' class='cs' type='email'/>`

```
WebElement wb = driver.findElement(locater)
```

- By.id("usr")
- By.name("username")
- By.className("cs")
- By.TagName("input")
- By.xpath("//input[@id='usr']")

* Write a program for Gmail application login-logout.

```
class Sample {
Public static void main
```

→

```
Public class GmailLoginLogout()
Public static void main(String[] args) throws InterruptedException
WebDriver driver = new FirefoxDriver();
driver.get("https://gmail.com");
driver.findElement(By.id("Email")).sendkeys("tricon.arun");
driver.findElement(By.id("next")).click();
driver.findElement(By.id("Passwd")).sendkeys("sanjaykar");
driver.findElement(By.id("signIn")).click();
Thread.sleep(15000);
driver.findElement(By.xpath("//a[contains(@title,'@gmail.com')]")).
                                                    click();
driver.findElement(By.xpath("//a[text()='Sign out']")).click();
driver.quit();
```

Assignment:- Write a program for facebook login logout.

Date:- 28/04/16

Notes:- 1) Whenever Webdriver launches a browser automatically all the ad-ons getting disabled.

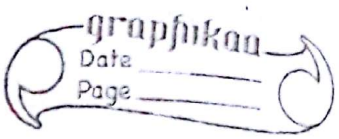
2) WebDriver launches new browser for every execution, so webDriver cannot work with existing browser.

3) While writing a webDriver code better practice to use ctrl + space for automatic package import.

4) WebDriver has backward compaitability (no-fwward compatability)

5) Wherever we download latest version of WebDriver jar better practice to work with latest version of Browser otherwise we may get compatible issue.

WebDriver's basic Methods



WebDriver

- get()
- navigate().to()
- navigate().forward()
- navigate().back()
- navigate().refresh()
- findElement()
- findElements()
- getTitle()
- getCurrentUrl()
- getWindowHandle()
- manage().deleteCookies() → deleteAllCookies
- manage().window().maximize()
- quit()
- close()

```
* → public class SampleTest {
    Public static void main(String[] args) throws InterruptedException {
      WebDriver driver = new FirefoxDriver();
      driver.navigate().to("https://gmail.com");
// driver.findElement(By.xpath("//a[contains(text(),'Need help?')]")).
                                                                  click();
      driver.findElement(By.LinkText("Need help?")).click();
      Thread.sleep(5000);
// Click on back button on the browser
      driver.navigate().back();
// Click on forward button on the browser
      driver.navigate().forward();
// Click on refresh button on the browser
      driver.navigate().refresh();
    }
}
```

// maximize the current browser which is launched by webDriver
driver.manage().window().maximize();

// delete all cookies from the browser
driver.manage().deleteAllCookies();

* Verify actiTime homePage

Step1) Login to actiTime application.

Expected Result:- Application should navigate to open Tasks page

→ Public class VerifyHomePage {
Public static void main(String[] args)
// Login to APP
WebDriver driver = new FirefoxDriver();
driver.get("http://qsp-tyy-24-pc/login.do");
driver.findElement(By.name("username")).sendKeys("admin");
driver.findElement(By.name("pwd")).sendKeys("manager");
driver.findElement(By.xpath("//input[@type='submit']")).click();
Thread.sleep(4000);

// test data
String expectedVal = "actiTIME - Open Tasks";

// Verify home page
String actVal = driver.getTitle();
if(expectedVal.equals(actVal)){
System.out.println("home page is verified == PASS");
} else {
System.out.println("home Page is not verified == FAIL");
}
driver.close();
}

Points:

1) get() method is used to navigate any html webpage. or any web application.

2) navigate().to() method also used to navigate to any web application but always hold the browser history.

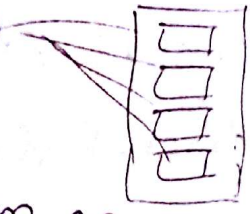
3) Using navigate() method we can also perform back, forward and refresh operation on the browser.

4) findElement() is use to find any single webElement in UI, which always return single webElement.

Ex:- WebElement wb = driver.findElement(locater);

5) findElements() is use to identify multiple webElements in UI which always return collection of webElements.

Ex:- List<webElement> list = driver.findElements(locater)



6) getWindowHandles() is use to perform action on multiple browser or multiple tabs.

7) getTitle() is use to capture the title from the current page.

8) getCurrentUrl() is use to capture the URL from the current page

9) quit() method is use to close all the windows which is opened by webDriver.

10) close() method is use to close only one current active window.

Note:- All the methods available in webDriver is used to perform action/operation on the webbrowser.

Code Optimization.

Ex-1

→ Get same output with different or less number of java statement is called code optimization.

int i=10;
int j=20;
int z = i+j;
Sopln(z);
	int i=10;
	int j=20;
	Sopln(i+j);
		int j=20;
		Sopln(j+10);
			Sopln(10+20);

O/P = 30	O/P=30	O/P=30	O/P=30

Ex-2

```
class F {
  W find() {
    System.out.println("execute find");
    W w1=new W();
    return w1;
  }
}
class W {
  void send() {
    System.out.println("perform type");
  }
  void click() {
    System.out.println("perform click");
  }
}
Public class Test {
  Public static void main(String[] args) {
```

I
```
F f1 = new F()
  f1.find();
W w1 = new W()
  w1.send();
  w1.click();
```

II
```
F f1 = new F()
W w1 = f1.find()
```

III
```
new F().find().send()
  or
new F().find().click()
```

WebElement basic methods

- sendKeys()
- click()
- submit()
- getText()
- getAttribute
- clear()
- getCssValue()
- getTagName()
- getSize()
- getLocation()

- isEnabled()
- isDisplayed()
- isSelected()

WebElement is an inbuit api available in .jar which is used to perform specific action on the webelement or object.

Note:- webElement method should be used after finding webElement in UI.

Ex:- In order to perform action on editbox, we will go for findElement method to identify the object in UI, then findElement method return webElement reference object if object is available and using webElement reference we can perform sendkey operation on editbox.

* Verify Gmail invalid message.

Date:- 02/05/16

Step1) Navigate to Gmail Applicatia

Step2) Enter invalid username and click on Next button

Expected Result:- Sorry, Google doesn't recognize that email" message should display.

→ public class VerifyGmailInvalidMsg {

public static void main(String[] args)

//test Data

String username = "abcdefgh";

String expMsg = "Sorry, Google doesn't recognize that email.";

//Step 1: navigate to Gmail app

WebDriver driver = new FirefoxDriver();

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

//Step 2: enter invalid username and click on next button.

driver.findElement(By.id("Email")).sendKeys(username);

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

Thread.sleep(2000);

// Verify invalid msg

WebElement wb = driver.findElement(By.id("errormsg-0-Email"));

String actVal = wb.getText();

System.out.println(actVal);

if(expMsg.equals(actVal)){

System.out.println("Invalid msg is verified == PASS");

}else{

System.out.println("Invalid msg is verified == FAIL");

}

driver.close();

G-2 Manual test case:- Verify Gmail login Page objects.

Step1:- Navigate to Gmail application.

Step2:- Verify Gmail logo and editbox default value.

Expected result:- 1) Google logo should be display in the login page

a) Enter your email Placeholder should be present in the editbox.

```
Public class VerifyGmailLoginPageObject {
  Public static void main(String[] args) {
   //test Data
    String expVal = "Enter your email";
   //Step1 : navigate to Gmail app.
   WebDriver driver = new FirefoxDriver();
   driver.get ("https://gmail.com");
   //Step2 : verify gmail Logo Image.
   WebElement wb = driver.findElement (By.xpath ("//div [@
                                    class='logo logo-w']"));
   boolean imgstatus = wb.isDisplayed();
    if (imgstatus == true) {
      System.out.println("Google logo image is displayed==PASS");
   }else {
     System.out.println("Google logo image is not displayed==FAIL");
    }
  //Verify username Editbox default value
    String actVal = driver.findElement (By.id ("Email")).
                          getAttribute ("placeholder");
    If(expVal.equals(actVal)) {
      System.out.println("default value is verified==PASS");
    }else {
      System.out.println("default value is not verified==FAIL");
    }
   driver.close();
```

Ex3 :- Verify Stay signed in checkbox ☐ in login page.

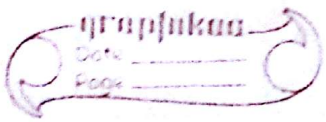
Step1 :- Navigate to Gmail application

Step2 :- Enter valid username and click on next button.

Expected Result :- 1) Stay signed in checkbox should be enabled in UI

2) Stay signed in checkbox should be selected by default.

→ Public class VerifyStayCheckbox {
Public static void main (String[] args) {



// Step 1 : navigate to Gmail app.
WebDriver driver = new FireforDriver ();
driver.get ("https://gmail.com");

// Verify checkbox
driver.findElement (By.id("Email")).sendKeys ("arunkar1506");
driver.findElement (By.id("next")).click ();
Thread.sleep (2000);
WebElement wb = driver.findElement (By.id("PersistentCookie"));

boolean s1 = wb.isEnabled ();
boolean s2 = ~~wb.isDisp~~ wb.isSelected ();
if (s1 == true && s2 = true) {
System.out.println ("Test case is PASS");
} else {
System.out.println ("Test case is FAIL");
}
driver.close ();

Date:- 03/05/16

Points:- 1) WebElement API is used to perform operations on the webElement.

2) Before using WebElement method we have to find an element in HTML source code using findElement or findElements method.

3) WebElement method are used to perform click, sendkeys and checking the visibility. operations on the object.

4) getText() method is used to capture visible Text from the webElement, it always return string value if web-Element has visible text or else return Empty String.

5) getAttribute() :- This method is used to capture any backend attribute value of the webElement, it always return string value.

Ex:- Gmail

getTagName() — a; getAttribute() — href='http://gmail.com' id='lnk'; getText() — Gmail

6) getTagName() :- This method is used to capture HTML Tag from source code, it returns string value.

7) isDisplayed() :- This method is used to check whether webElement is available in UI, it returns boolean (TRUE) value if object available in UI or else return FALSE.

8) isEnabled() :- This method is used to check wheather webElement is active or disabled in UI, it always return boolean value.

9) isSelected() :- This method is used to check wheather radio button or checkbox is already selected or not, it returns TRUE value if webElement is already selected

10) clear() :- This method is used to clear the text from the editbox.

11) sendkeys() :- This method is used to enter value in editbox.

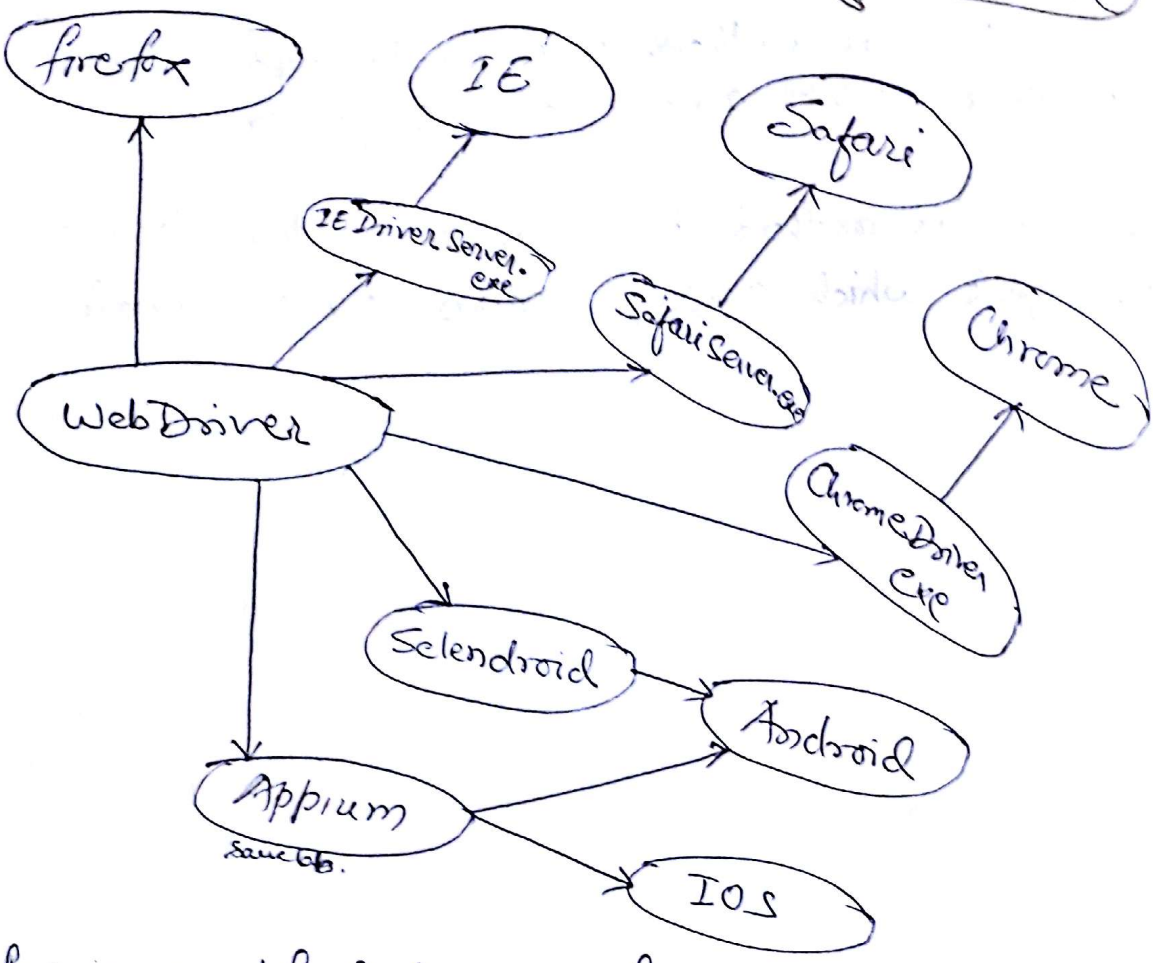
12) Submit() :- This method is used to submit HTML page to server

13) Click() :- This method is used to perform click operation on any webElement like button, radio button, checkbox, clickable image etc.

14) getCssValue(): - This method is used to capture color of the object, which always return string value.

15) getLocation(): - This method is used to capture location of the object which always return x, y co-ordinate

Working with Multiple Browser



→ firefox is a default browser for webDriver so no need to configure the settings to work with firefox Browser

→ In order to work with other than firefox browser, we have to take a help of driver server and configure the settings in Browser.

Working with InternetExplorer browser

Step1) Navigate to selenium community and download IE Driver Server.

Step2) In Selenium community website find internetExplorer Driver server division and click on windows IE link based on the available windows configuration.

Step3) Download .zip file from the community and unzip the folder then we will get IEDriver server.exe.

Step4) Before launching InternetExplorer browser we have to run the server in webDriver code.

System.setProperty("webdriver.ie.driver",
"C:\\Users\\oar1611\\Desktop\\selenium\\IEDriverServer.exe");

WebDriver driver = new InternetExplorerDriver();

Steps) Before running the code in IE Browser we have to configure changes in browser side.

a) Browser zoom level should be 100%.

b) Security level of the browser should be Medium-high for all the zone.

c) enable protected mode checkbox should be checked for all the zone.

Tools → Internet option → security tab and do all changes.

Date:- 04/04/16

ACTIONS

Working with mouse movement & Keyboard operations

Actions class

Methods:
- moveToElement()
- sendKeys()
- contextClick()
- dragAndDrop()
- perform()
- doubleClick(), doubleClick(wb)
- clickAndHold()
- Release()

Scenario:- Find Android option in IRCTC

Steps):- Navigate to irctc.com.

Steps) Find mobileApps dropdown menubar, Perform mousehover operation on mobileApps.

Steps) Click on Android link available inside the mobile Apps dropdown menu.

Expected Result

Application should navigate to mobile Apps Page, and it should contains download App button

→ Public class FindAndroidAppInIRCTC {
Public static void main (String [ ] args) {
WebDriver driver = new FirefoxDriver();
driver.get("https://irctc.co.in");
// find Mobile Apps dropdown menu in UI
String x = "//div[@id='bluemenu']/ul/li/a[contains(text(), 'Mobile Apps')]";
WebElement wb = driver.findElement(By.xpath(x));
// Create object to Actions class
Actions act = new Actions(driver);
// Using actions class and performs mouse hover operation.
act.moveToElement(wb);
// perform actual Actions class action
act.perform();
// click on Android link
driver.findElement(By.xpath("//a[text()='Android']")).click();
}
}

* Working with Keyboard operation.

Step1) Navigate to actiTime application

Step2) Enter username and password and pass enter ↳ Keyboard operation.

Expected Result :- Should be able to login to the application without clicking login button.

→ public class KeyBoard {
public static void main (String [] args) {
WebDriver driver = new FirefoxDriver();
driver.get("https://oar16-pc/logic.do");
// Enter user name and password.
driver.findElement(By.name("username")).sendKeys("admin");
driver.findElement(By.name("pwd")).sendKeys("manager");
// Click on Enter button in keyboard.
Actions act = new Actions(driver);
act.sendKeys(Keys.ENTER).perform();
}
}

Date:-05/04/16

Functional Keys
- Alt
- Control
- Tab
- Space
- delete
- Escape

sendkeys(Keys.control)

Character Keys
- a
- b
- c
- ...
- $
- @
- !

SendKeys("c")

* Write a program to perform 2 concurrent keyboard operation. Ex- ctl c

→
Public class MultipleKeyBoard {
Public static void main (String [] args) {
WebDriver d = new FirefoxDriver();
d.get("http://oar16-pc/logic.do");
// Find Username Edit box in UI.
WebElement uNamewb = d.findElement(By.name("username"));
uNamewb.sendkeys("admin");

// Select Username data by double clicking the editbox.

```
Actions act = new Actions(d);
act.doubleClick(uNameWb).perform();
```

// Copy the data.

```
act.sendKeys(Keys.chord(Keys.CONTROL,"c")).perform();
```

// to go to password editbox.

```
act.sendKeys(Keys.TAB).perform();
```

// paste the data in password box.

```
act.sendKeys(Keys.chord(Keys.CONTROL,"v")).perform();
}
}
```

→ Sequencial Keyboard operation.

* Chord is not required

```
act.sendKeys(Keys.TAB, Keys.TAB, Keys.ENTER).perform();
```

* Write a program to perform Right click operation.

```
Public class RightClick {
Public static void main(String[] args) {
WebDriver d = new FirefoxDriver();
d.get("tittps://gmail.com");
WebElement wb = d.findElement(By.linkText("Need help?"));
Actions act = new Actions(d);
act.contextClick(wb).perform();
act.sendKeys("w").perform();
```

→ opening in new window

Points :-

1) Actions class is an independent class available in Interactions package.
2) Actions class is used to perform user interactions operation. like Keyboard, mousehover, Right click, double click etc.

3) In order to perform concurrent Keyboard operation we should take help of chord method available in Keys class.

4) Keys is another independent class, all the functional Keyboard (static keywords) available in Keys class.

5) Move to Element :- This method is used to take mouse cursor middle of the webElement.

6) SendKeys :- This method is also available in Actions class use to perform functional or character Keyboard operation

7) doubleClick() :- This method is use to perform double click operation where mouse cursor is currently present.

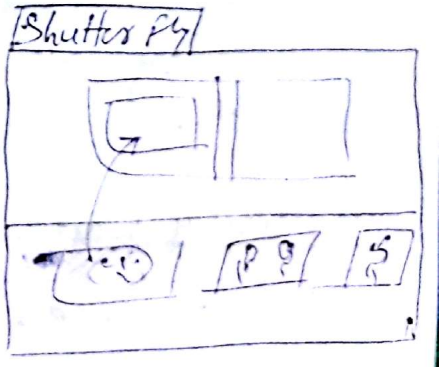
8) doubleClick(wb) :- This method is use to perform double click operation in the middle of the specified element.

9) Context Click ()
   Context Click (wb) } - Use to perform Right click operation.

10) Perform () :→ This method is use to perform actual Actions class action, perform method also release driver control from actions class.

11) Drag And Drop

```
Srcwb = d.fE(By.xpath("//imp[@----]));
Dstwb = d.fE(By.xpath("//div[----]));

Actions act = new Actions
act.dragAndDrop(Srcwb, Dstwb).perform();
```



WebDriver Wait Statement

There are four types of wait statement

1. Normal wait.
2. Implicitly wait
3. Explicitly wait
4. Page load wait.

Normal wait

```
Syntax - thread.sleep(10000)
```

→ Normal wait is a hard coded wait should not be used in real time selenium script, because normal wait always going to wait specify amount of time.

→ Whenever application loaded within 2 sec, normal wait unneccesary wait for another 8 sec so that execution speed will be decreased.

Synchronization :-

The process of matching application speed with automation tool speed is called synchronization.

Synchronization issue :-

Whenever Automation tool try to perform some Action on webElement, but webElement not yet loaded in UI, in such cases webDriver stop execution by throwing No SuchElement Exception. This is synchronization issue.

Implicitly Wait :-

```
Syntax :- driver.manage().timeouts().implicitlyWait(duration, type of duration)
```

→ Implicitly wait wait till page to load before performing Action on webElement.

→ Implicitly wait always monitor or polling entire HTML document before perform Action on webElement, if webElement is

getting loaded within 2 sec Implicitly wait release the control to next line instead of waiting entire 20 sec.

→ Whenever we introduce implicitly wait before get method then that wait statement is applicable for each and every action done by WebDriver.

Disadvantage of Implicitly Wait

→ Implicitly Wait cannot wait for dynamic object, so that cannot be used in Ajax application.

Explicitly Wait

Syntax:-

WebDriverWait
→ untill (Expected condition)
→ presence of ElementLocated (Locator)

```
WebDriverWait wait = new WebDriverWait (driver, 20);
wait.until(ExpectedConditions.presenceOfElementLocated(By.
        xpath("//a[contains(@title, 'gmail.com')]")));
```

→ Explicitly wait. wait till element to be present in UI.

→ Technically Explicitly wait check for the expected condition for every 500 millisecond, if expected element appears in UI, explicitly wait release the control to next line instead of waiting entire 20 sec.

Working with Dropdown

graphikon Date 07/05/16 Page

Select

- selectByVisibleText()
- selectByValue()
- selectByIndex()
- deselectAll()
- deselectByIndex()
- deselectByVisibleText()
- getoptions()
- getAllselectedOptions()
- ismultiple()
- getFirstSelectedOption()

* Write script for selecting a month in Facbook login page.

```
Public class DropDown
Public static void main (String[] args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("https://facebook.com");
// Step1: identify dropdown in UI
    WebElement wb = driver.findElement(By.id("month"));
// Step2: create an object to SELECT class
    Select sel = new select (wb);
// Step3: select value from dropdown.
    sel.selectByVisibleText ("Mar");
  }
}
```

* End to End test_ verify Task.

Step1): Login to actiTIME application.
Step2) Click on task image and navigate to task page
Step3) Click on Project & Customer link and navigate to Customer page

Step4) Click on Add New Customer button and create a customer

Step5) Click on Add new Project button and create a project for above customer

Step6) Navigate to task page

Step7) Create 5 task for above project. and verify the succeful message " 5 new tasks were added to the customer "ICICI", project "testing".

→ Public class EndToEndTest_VerifyTask {

Public static void main (String[] args) {

// Test Data.

String userName = "admin";

Date:- 9/06/15

Sel selectByVisibleText ("Aug");
Sel. selectByValue ("8");
Sel. selectByIndex (8);

* Write a program to work with multiselect dropdown.

→ public class Multiselect

Public static void main (String[] args) {

WebDriver driver = new FirefoxDriver();

driver.get ("file:///C:/Users/oar16/Desktop/page.html");

// find the multi select dropdown.

Select sel = new Select (driver. findElement (By. name ("sel")

//Check weather select box is multiselect or single select

if (sel. isMultiple () == true) {

// if dropdow in multiselect then select all the options

for (int i=0; i<6; i++) {

sel. selectByIndex (i);

}

```
System.out.println("dropdown is multiselect");
} else {
  sel.selectByVisibleText("Java");
System.out.println("dropdown is singleselect");
  }
  sel.deselect();
 }
}
```

graphikon
Date 09/06/15
Page

* Write a program to select expected value from dynamic select dropdown.

→
```
public class DynamicSelectDD
 public static void main(String[] args){
  WebDriver driver = new FirefoxDriver();
  driver.get("file:///C:/Users/oar16/Desktop/page.html");
// find the dynamic multi select dropdown.
Select sel = new Select(driver.findElement(By.name("sel")));
// Capture all the options from the dropdown.
 String expectedVal = "etl testing";
 int tmp=0;
 List<WebElement> lst = sel.getOptions();
// display the content of collection list.
  for(int i=0; i<lst.size(); i++){
      String actVal = lst.get(i).getText();
      if(actVal.equals(expectedVal)){
       sel.selectByVisibleText(actVal);
        tmp=1;
        break;
      }
 }
if (tmp=1){
    Sopln("expected value is available");
  } else {
    Sopln("expected value is not available");
 }
}
```

Points for Select Class

→ Select class is an independent class available in webdriver jar it is used to work with single or multi or dynamic select dropdown.

→ In order to work with dropdown webElement, we have to take help of select class method instead of webElement.

→ There are 3 ways we can select the value from dropdown

1) SelectByVisibleText :- Select the value based on Text available in UI.

2) SelectByIndex :- Select the value based on position of the option (position always start from 0).

3) SelectByValue :- Select the value based on value of the value attribute from option HTML tag.

→ There are 3 ways we can get the value from the dropdown

1) GetOption() :- Get all the option available in dropdown. which always return list (Collection) of webElements, In order to get the text from the webElement we go for getText

2) GetAllselectedoption() :- Get only already selected option from the webElement which returns collection list.

3) GetFirstSelectedOption() :- Get only first selected option from the dropdown.

→ deSelectAll() → Unselect all the value from multiselect dropdown

→ isMultiple() → Is use to check wheather dropdown is single select or multiselect, it returns boolean TRUE value if dropdown is multiselect.

***

Note:- Whenever WebElement is not implemented using Select HTML tag in such cases we can't take help of select class method.

Ex:- Google search editbox is looks like a dropdown but it is implemented using input HTML tag. so we can not use select class method.

Note:- In order to work with dynamic select dropdown we have to capture all the value using getOption method and check for the expected value in dropdown, select the value if data is available.

* Auto Suggest Editbox.

→ Auto complete Editbox are customized editbox which is implemented using HTML tag with auto complete backend attribute.

→ Auto complete Editbox are always looks like dropdown but we can not take help of select class.

→ Whenever we tried to write something on autocomplete Editbox, which always enable automatic suggested options.

```
* Public class GoogleSearch {
    Public static void main (String[] args) {
        WebDriver d = new FirefoxDriver();
        d.get("https://google.com");
        d.findElement(By.id("lst-ib")).sendKeys("selenium tutorial",
                                                Keys.ENTER);
    }
}
```

* Another way

```
Public class GoogleSearch {
    Public static void main (String[] args) {
        WebDriver d = new FirefoxDriver();
```

```
d.manage().timeouts().implicitlyWait(20, TimeUnit.SECOND);
d.get("https://google.com");
d.findElement(By.id("lst-ib")).sendKeys("wipro c");
d.findElement(By.xpath("//b[text()='careers']")).click();
}
}
```

→ There are two ways we can work with Autocomplete Edit box.

1) Write a complete string in Editbox using sendkeys and pass ENTER Keyboard operation.

2) write a part of the string in Editbox using sendkeys and go for one more click operation to click/select expected value.

* Working with Multiple WebElements

* Write a program to capture all the links from Yahoo.com

```
Public class WorkingwithMultipleWebElement {
 Public static void main(String[] args) {
  WebDriver d = new FirefoxDriver();
  d.get("https://in.yahoo.com/?p=us");
 //find all the Links available in UI
  List<WebElement> lst = d.findElements(By.xpath("//a"));
 //get the Link count
  System.out.println(lst.size());
 //display all the Link name
  for(int i=0; i<lst.size(); i++){
    System.out.println(lst.get(i).getText());
  }
}
```

* Delete All message

Precondition:- Message should be available in inbox table.

Test Data:- Username
Password

Step1:- Login to Gmail
Step2:- Select all message checkbox from inbox.
Step3:- Click on Delete button.

Expected Result:- All the message should be deleted from inbox table.

```
* Public class GmailMessageDelete
  Public static void main (String[] args) {
  WebDriver driver = new FirefoxDriver();
  driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
  driver.get("http://gmail.com");
  // Login to Gmail.
     ==
     ==
  // Wait till inbox table to load.
  WebDriverWait wait = new WebDriverWait(driver, 20);
   wait.until(ExpectedConditions.presenceOfElementLocated
              (By.xpath("//table[@class='F cf zt']")));
  // find all the checkbox available in inbox table
   String x = "//table[@class='F cf zt']/tbody/tr/td[2]/div";
  List<WebElement> lst = driver.findElements(By.xpath(x));
  // Select all checkbox.
   for(int i=0; i<lst.size(); i++)
       {
         lst.get(i).click();
       }
  // Click on delete button.
```

Assignment 1:-

Scenario:- Capture all subject in inbox.

Step1:- Login to Gmail application
Step2:- Capture all the subject of the message
Step3:- Display the subject in console.

Assignment 2:-

Scenario:- Capture all link.
Step1:- Navigate to google.com
Step2:- Search for wipro careers.
Step3:- Capture all the URL available for wipro careers
Step4:- Display URL in console.

```
List = d.findElements(By.tagName("a")
for (int i=0; i<lst.size(); i++)
  Sopln(lst.get(i).getAttribute("href");
```

Assignment 3

Scenario:- Capture all suggested list
Step1:- Navigate to google.com
Step2:- Search for wipro career
Step3:- Capture all suggested option and display it in console.

Interview Question for WebDriver Basic API/Methods

1) What is webdriver.
   ↳ Its an interface, its collection of jars
   Webdriver is an interface, all browser class implement webdriver
2) How to maximize the browser
3) How to delete cookies
   → driver.manage().deleteCookies();

4) Did we use Overriding concept in Webdriver.

→ Yes we used overriding concept.
All the methods available in webdrive (ex - get(), findElement()) are implemented in Browser classes.

5) Did we use Upcasting and Runtime Polymorphism concept in webdriver.

→ WebDriver d = new FirefoxDriver() upcasting.
(sub — Super)

```
d.get("http://gmail.com")
d.F.E(
```

Same driver object behave differently for different different class instance is called Run time polymorphism here same code will work for different browser.

6) Why webdriver doesn't support stand alone application.

→ All selenium tool identify the object based on HTML and standalone application doesn't have HTML, Hence webdriver doesn't support.

7) How many language webDriver supports:

→ Java Python
.net Ruby
Perl C#

8) How many operating system webDriver supports:-

Question on WebElement basic methods

1) How to capture visible text from the UI

→ getText()

2) How to check whether the webElement is available in UI

→ isDisplayed() / isEnabled()

3) How to clear the value from the Editbox
→ driver.findElement(find the webElement).clear()

4) How to capture the value available in Editbox
→ getAttribute("value")
Sopln(wb.getAttribute("value"));

5) How to check wheather Radio button or checkbox is already selected or not.
→ isSelected().

6) How to capture color of the object.
→ getCssValue()

7) What is synchronization and synchronization wait statement available in webDriver.
→ Implicitly
Explicitly

8) How to handle dynamic object.
→ Explicitly wait

9) How to work with Ajax application.
→ Explicitly wait.

10) Difference between Implicitly and Explicitly wait.

11) How to work with Dropdown menubar or how to perform mouse hover.
→ We have to use Action class.
moveToElement() → This is available in Action class.

12) How to perform Right click, double click and drag and drop

13) How to capture tooltip of the image.
→ moveToElement() :- to enable tooltip.
→ getText() :- capture visible text

14) How to work with dropdown.

→ Find a dropdown using findElement method based on locator, and using select class method select the values from the dropdown.

graphikon Date 13/05/16 Page

There are three way we can select

- → Select By Visible Text
- → Select By index
- → Select By Value

15) How to select all the values from multiselect dropdown

→

```
Select
lst = → getOptions()
for (int i=0; i < lst.size(); i++)
    {
    sel.selectByIndex(i)
    }
```

Use getoptions to get the list of all the value and using for loop select all value.

16) How to work with dynamic select listbox.

→ Ans is given in note

17) Write a program to capture all the value from the drofdown and display in console.

→

```
lst = getOptions()
for (int i=0; i < lst.size(); i++)
  {
  Sopln (lst.get(i).getText());
  }
```

18) How to work with IE and Chrome browser.

→ Before launching IE or Chrome browser we have to download IEdriver and ChromeDriver server and run the server, in order to run the server we have to use System.setproperty method which takes 2 argument argument 1 is Keyword, argument 2 is path of the driver server.

19) How to identify All the links available in UI.

→ findElements (By.xpath("//a")

Use findElements method to identify all the links which always returns collection of link webElement

20) Difference between findElement and findElements.

→ find Element	findElements
1) Use to identify single Element	1) Use to identify multiple similar webElement
2) findElement alway return single webElement	2) findElements always returns list (collection) of webElement.
3) find Element throws NoSuch Element Exception if no matching found	3) findElements returns Empty list if No matching found.
4) findElement return first available webElement if multiple matching found	4) findElements always returns multiple webElements.
	Note:- In order to capture specific column data from dynamic webtable we go for findElements.

21) Write a program to capture second column data from dynamic webtable.

```
String x = //table[@---]/tbody/tr/td[2]/a
list = d.findElements(By.xpath(x));
```

Account	Id	Amount
	-	
	-	
	-	
	-	

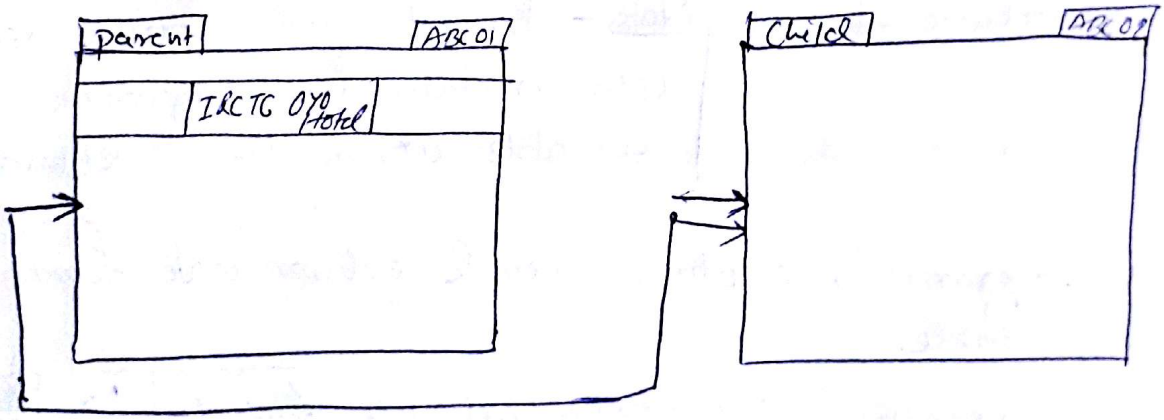
WINDOW HANDLING

→ There are 3 types of windows in web application
1) HTML window or new browser or new tab.
2) Confirmation popup or Alert window
3) Invisible popup.

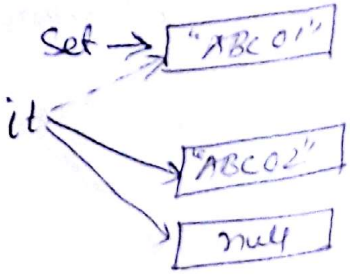
Scenario:- Find OYO Hotel in IRCTC website.
Precondition:- Hotel should be available in db.
Test data:- Location, checkIndate, checkOut date

Step 1) Navigate to irctc.co in
Step 2) click on irctc OYO Hotels which opens new tab.
Step 3) Enter all hotel details and click on search button.
Expected Result:- Hotel should be available.

→



Set s = d.getWindowHandles()

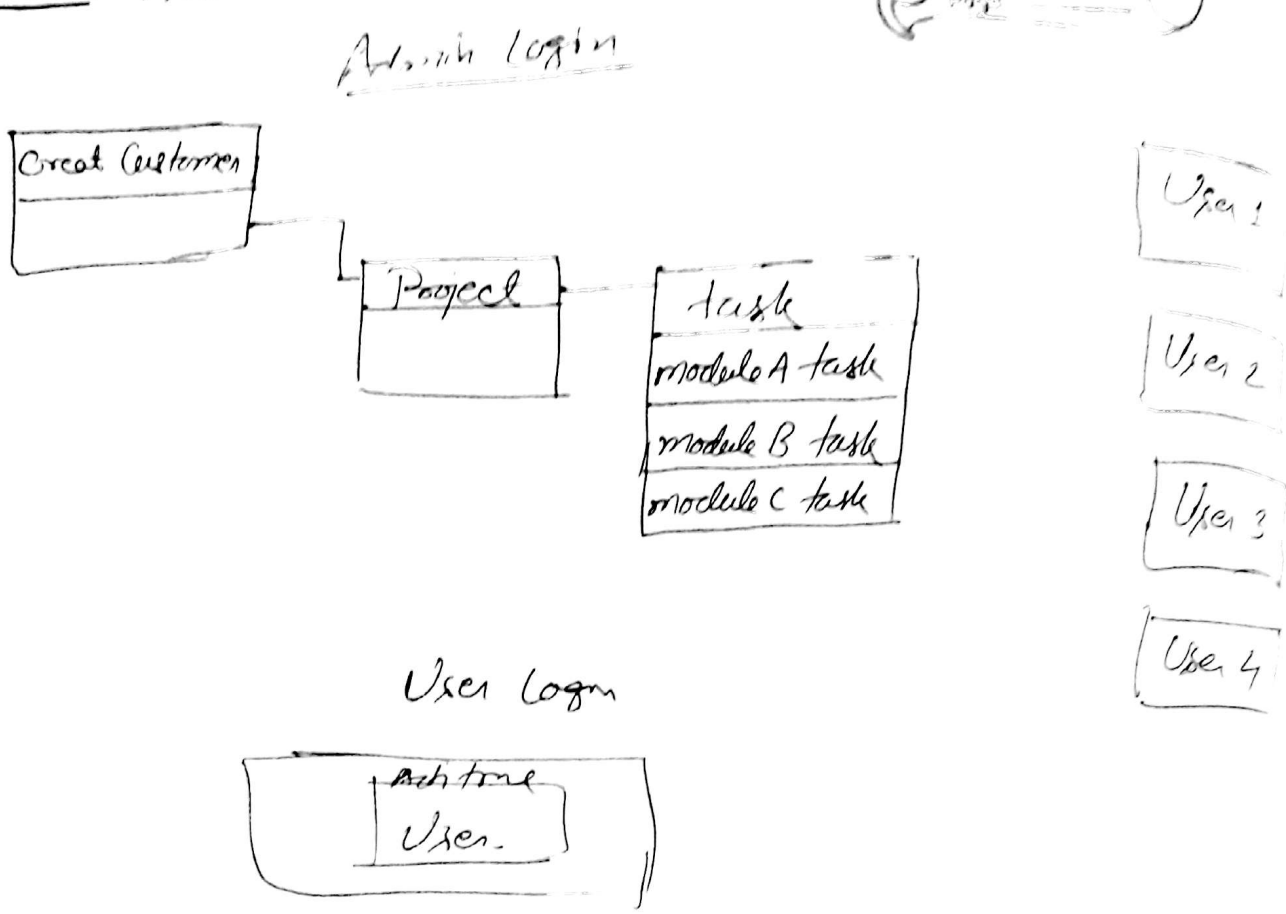


Iterator it = set.iterator()
String ParentId = it.next();
String childId = it.next();

→ class WindowHandle

```
Public class static void main(String[] args) {
 WebDriver driver = new FirefoxDriver();
d.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
d.get("https://www.irctc.co.in");
// Click on Hotel
```

Scenario:- Add user to Task.



Points for window handeling

Date:- 16/05/16

→ Whenever webdriver launches new browser, webdriver cannot have control on new window automatically.

→ In order to work with new window we should get window Id using getWindowHandles method, which always returns set of String.

→ Using Iterator get the windowId from the set collection and pass driver control to new window befor performing an action. `driver.switchTo.window(windowId)`

→ Application itself allocate dynamic windowId automaticaly for ever new window.

→ Dynamic Id will change for every execution

→ There are two ways we can get the windowId.

① driver.getWindowHandles()

② driver.getWindowHandle()

driver.getWindowHandles() → captured all the windowId opened

by web driver.

getWindowHandle() → capture only windowId where driver control is currently active.

→ There are two ways we can close the window

① driver.quit() → close all the window opened by webdriver

② driver.close() → close only one window where driver control is currently active.

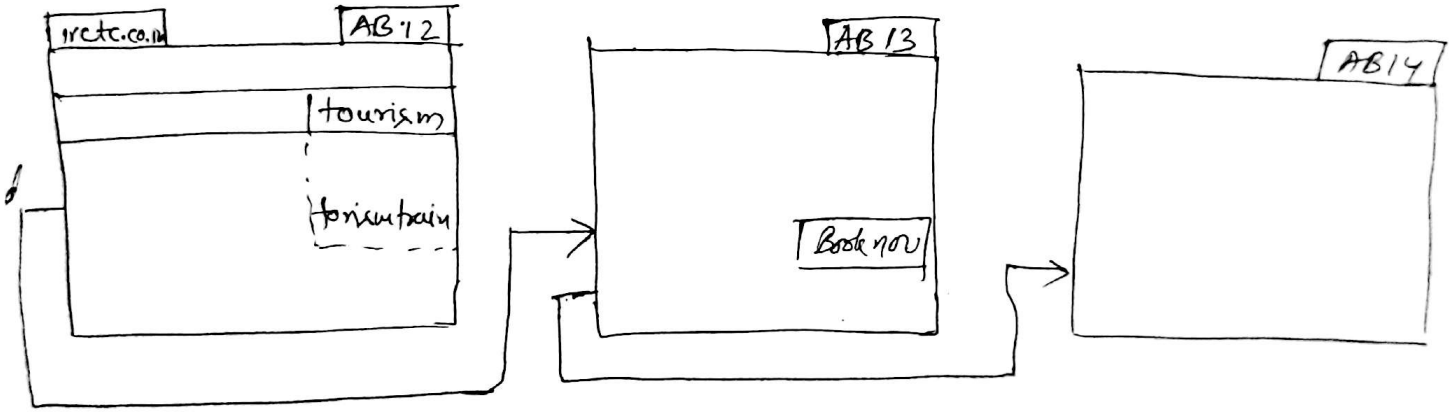
Assignment

Step1) Navigate to irctc.co.in

Step2) Click on Tourist Train available in IRCTC Tourism dropdown menu.

Step3) Click on Book Now button for Maharaja Exp.

Step4) Take mouse cursor middle of the language link and capture all the language available and display it in console.



set1 = d.getWindowHandles()

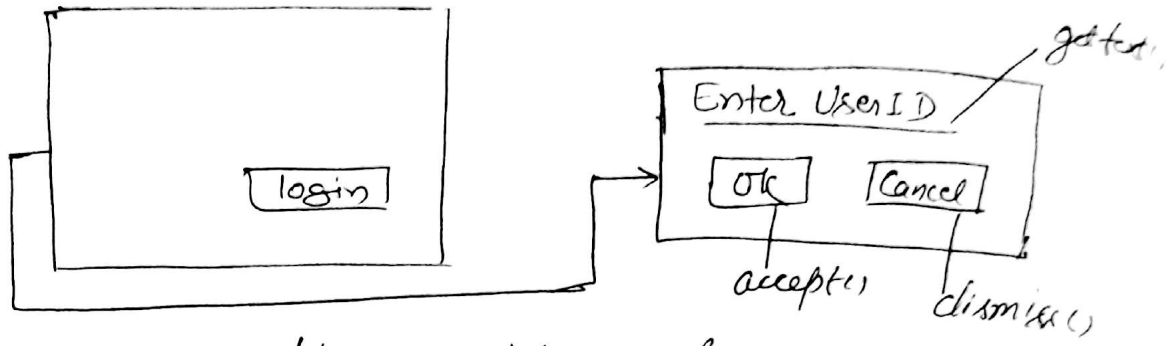
It1 → AB12, AB13

set2 = d.getWindowHandles()

It2 → AB12, AB13, AB14

Alert PopUp / Confirmation Window

Scenario:- Click on Login button in irctc.co.in without entering UserName and Password.



driver.switchTo().alert()

Alert
- getText()
- accept()
- dismiss()

```
* Public class AlertHandle {
  Public static void main (String [] args) {
  WebDriver driver = new FirefoxDriver();
  driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
  driver.get("https://irctc.co.in");
 // Perform action to open alert popup.
  driver.findElement(By.id("loginbutton")).click();
 // pass driver control to Alertbox.
  Alert alt = driver.switchTo().alert();
 // Capture the message from the alertbox.
  String altMsg = alt.getText();
  System.out.println(altMsg);
 if (altMsg.equals("Enter User ID")) {
    System.out.println("alert msg is verified == PASS");
 } else {
    System.out.println("alert msg is not verified == FAIL");
 }
```

```
// click on ok button
alt.accept();
// perform action on parent window
driver.findElement(By.id("usernameId")).sendKeys("admin");
}
}
```

Points :→

→ whenever action performed to open an alert, webdriver cannot have a control automatically.

→ In order to work with Alert we should pass driver control to Alert and using alert API methods perform action on alert.

→ Alerts are always implemented using Java Script so we cannot inspect element using firebug.

→ whenever Action performed with alertbox, driver control automatically come back to main window.

Invisible Popup

1) Expected hidden popup

→ Expected hidden popup are always implemented using div HTML tag but it will be hidden initially.

→ In order to work with hidden popup perform action to enable hidden popup and directly write xpath and perform an action.

*
```
Public class Calendar {
  Public static void main (String [] args) {
  WebDriver driver = new FirefoxDriver();
  driver.get ("https://www.oyorooms.com");
// Click on editbox it opens calendar hidden popup.
  driver.findElement(By.id("js-checkin")).click();
```

```
// Select date
String date = "//td[@data-month='11']/a[text()='30']";
while (true) {
  try {
    driver.findElement(By.xpath(date)).click();
    break;
  } catch (Throwable t) {
    // Click on next link
    driver.findElement(By.xpath("//span[text()='Next']")).click();
  }
}
```

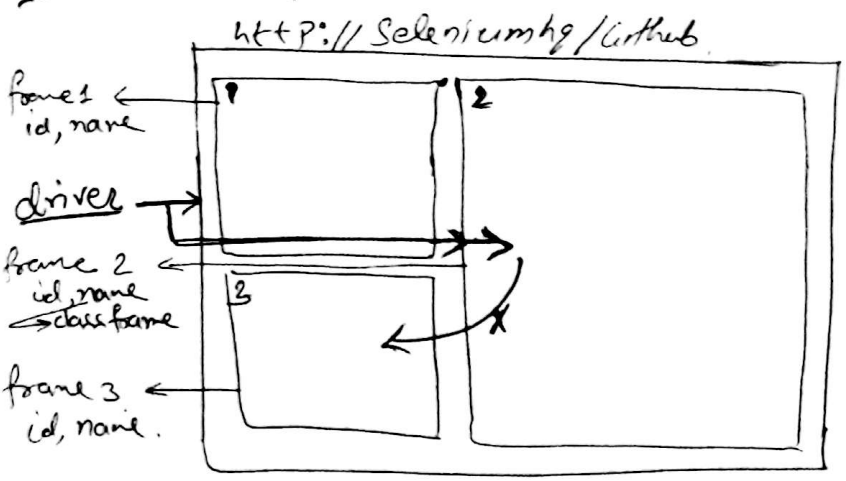
* Advertisement Popup

```
Public class HandleAdvertisePopup {
Public static void main(String[] args) {
  WebDriver driver = new FirefoxDriver();
  driver.get("https://www.oyorooms.com");
  try {
    driver.findElement(By.xpath("//i[contains(@class,'close-icon')]")).
                                                    click();
  } catch(NoSuchElementException t) {
    System.out.println("handle excepetion");
  }
// Continue webdriver code.
}
```

* Capturing moving text on irctc website.

```
WebDriver driver = new FirefoxDriver();
  driver.get("https://irctc.co.in");
String text = driver.findElement(By.xpath("//marquee[@style=
                                'color:#000;']/span")).getText();
  System.out.println(text);
}
```

FRAME HANDLING



driver.switchTo().frame(String) — id/name

driver.switchTo().frame(int) — index.

```
* Public class FrameHandle {
  Public static void main (String[] args) {
  WebDriver driver = new FirefoxDriver();
   driver.get ("https://seleniumhq.github.io/selenium/docs/api/
               java/index.html");
   // pass driver control to frame 2
     driver.switchTo().frame ("classFrame");
   // Click on Link, available in frame 2
     driver.findElement(By.LinkText ("com.thoughtworks.selenium")).
                                                    click();
   // pass control back to parent window
     driver.switchTo().defaultContent();
   // pass driver control to frame 3
     driver.switchTo().frame ("packageFrame");
   // Click on Link, available in frame 3.
     driver.findElement(By.~~xpath~~ LinkText ("Actions")).click();
   }
}
```

Points for frame handling

→ Whenever object present inside the frame. webdriver cannot identify frame object.

→ In order to perform action on ~~web~~ frame ~~webelement~~ webelement, we have to pass control to frame before performing action

→ There are three ways we can pass control to frame.

~~1) Driver.frame.~~

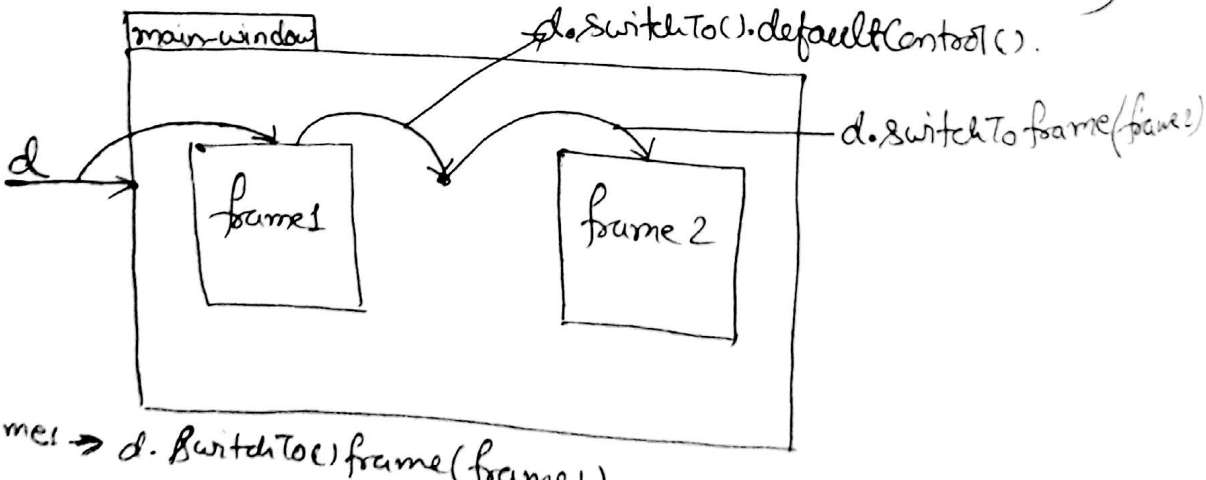
1) Driver.SwitchTo.frame(string)

2) Driver.SwitchTo.frame(int)

3) Driver.SwitchTo.frame(webElement)

→ Basically frame HTML tag will be used to embeded or insert one HTML source code in another HTML page.

→ In order to perform action on multiple frame, we have to pass driver control to frame1 and perform an action, then before passing driver control to frame2 we should get control back to main window then pass control to frame2 from main window (we cannot pass driver control from one frame to another frame)



for Frame1 → d.SwitchTo().frame(frame1)

Q) How to find no of frames available in UI.

→
```
List = d.findElements(By.TagName("frame"))
System.out.println(list.size())
```

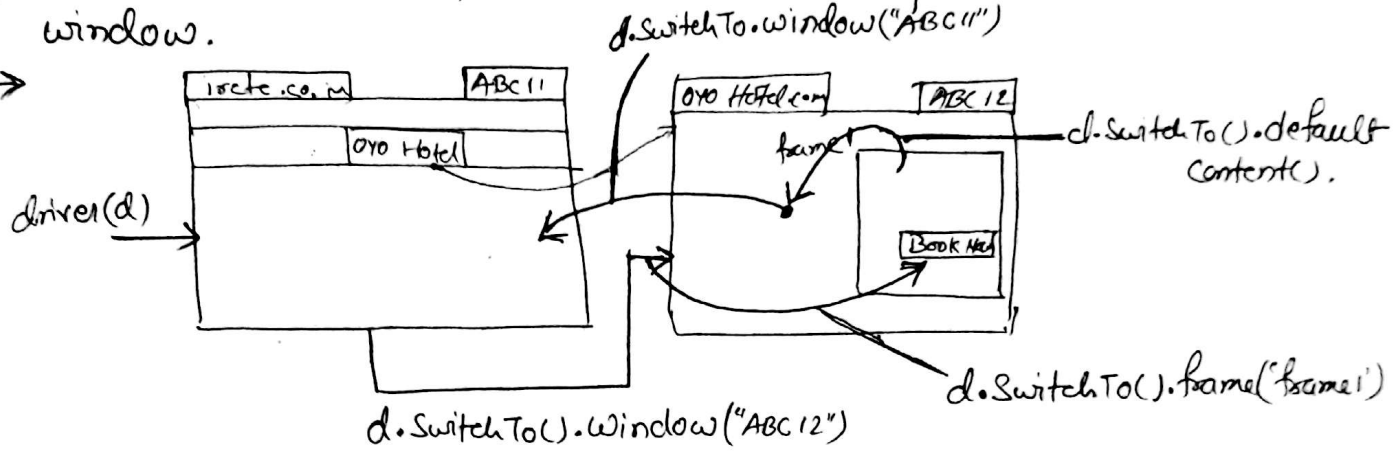
Q) How to work with frame object

→ Pass driver control to frame using driver.SwitchTo().frame and perform an action.

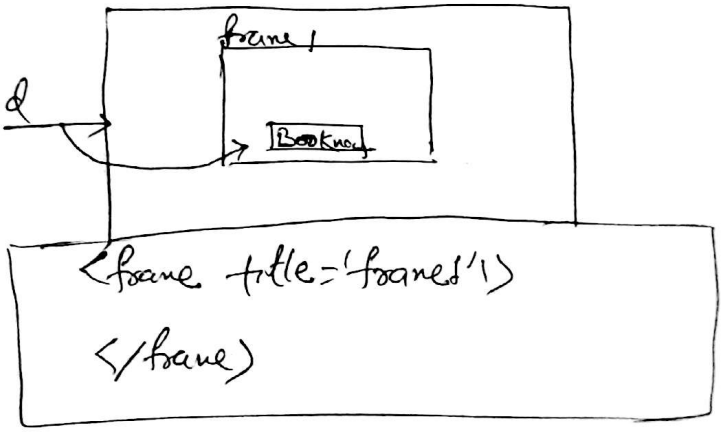
Q) How to work with multiple frame.

Q) How to work with frame, when frame present in child window.

→



Q) How to work with frame, when frame doesnot have unique attribute.



WebElement wbf = d.findElement(By.xpath("//frame[@title='frame1']"));

d.SwitchTo().frame(wbf);

Q) Do we use overloaded method in webdriver.

→ Yes, we have overloaded method in webdriver, while switching to the frame we are using.

Ex. frame(String)
frame(int)
frame(webElement)

Q) How to work with new Window or new Tab.

→ Using getWindowHandle get all the window or Browser id, which returns set of string, then use iterator to capture the window id from the set and pass a control to child or new window using driver.SwitchTo().window(newwindowid)

Q) How to work with Alert.

→ Pass a control to Alert using driver.SwitchTo().Alert() and using Alert methods (Accept, dismiss and getText()) perform action on Alert.

Q) How to work with hidden calendar Popup.

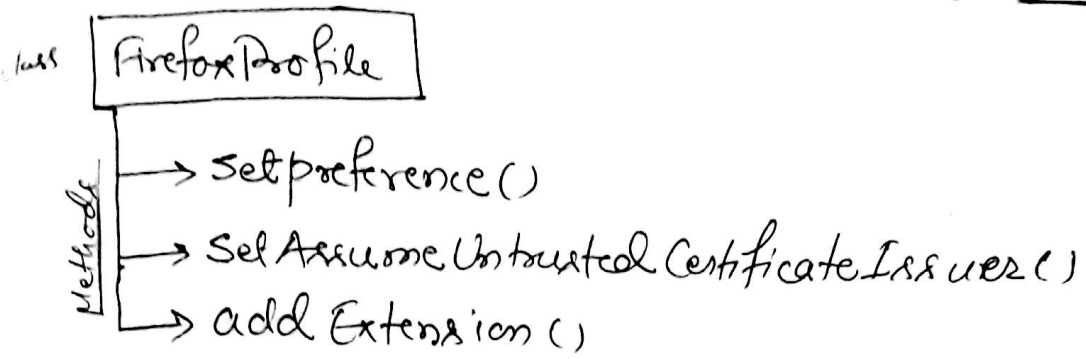
→ Perform action to enable hidden Popup, and then using while loop check for the expected date repetedly, whenever expected date is not available try block will throw an exception and then handle exception using catch block and click on next button/link until unless we get expected date.

Q) How to work with Adevertisement Popup

→ Write webdriver code inside the try block to handle advertisement Popup, because whenever Popup is not available try block will throw an exception and catch exception. Write fail result in catch block.

FILE DOWNLOAD POPUP

→ In order to work with file download popup which are not alert we will use FirefoxProfile class



SetPreference

```
* Public class FileDownload {
    Public static void main (String [ ] args) {
    FirefoxProfile P= new FirefoxProfile();
    P.setPreference ("browser.download.folderList", 2);
    P.setPreference ("browser.helperApps.neverAsk.saveToDisk"
                    , "application/zip");
    P.setPreference ("browser.download.dir", "D:\\Files\\");
    WebDriver driver = new FirefoxDriver(P);
    driver.get ("http://www.seleniumhq.org/download/");
    driver.findElement (By.xpath ("//a[contains(@href,
            'selenium-java')]")).click();
    }
}
```

Points :-

1) FirefoxProfile independent class available in webdriver jar, which is used to handle browser native popups.
2) SetPreference() method is used to change the browser native setting from the backend.

3) In order to enable browser backend keys, go to browser navigation bar and type **about:config** click enter and click on I'll be careful button.

4) In order to disable file download popup we have to set below keys.

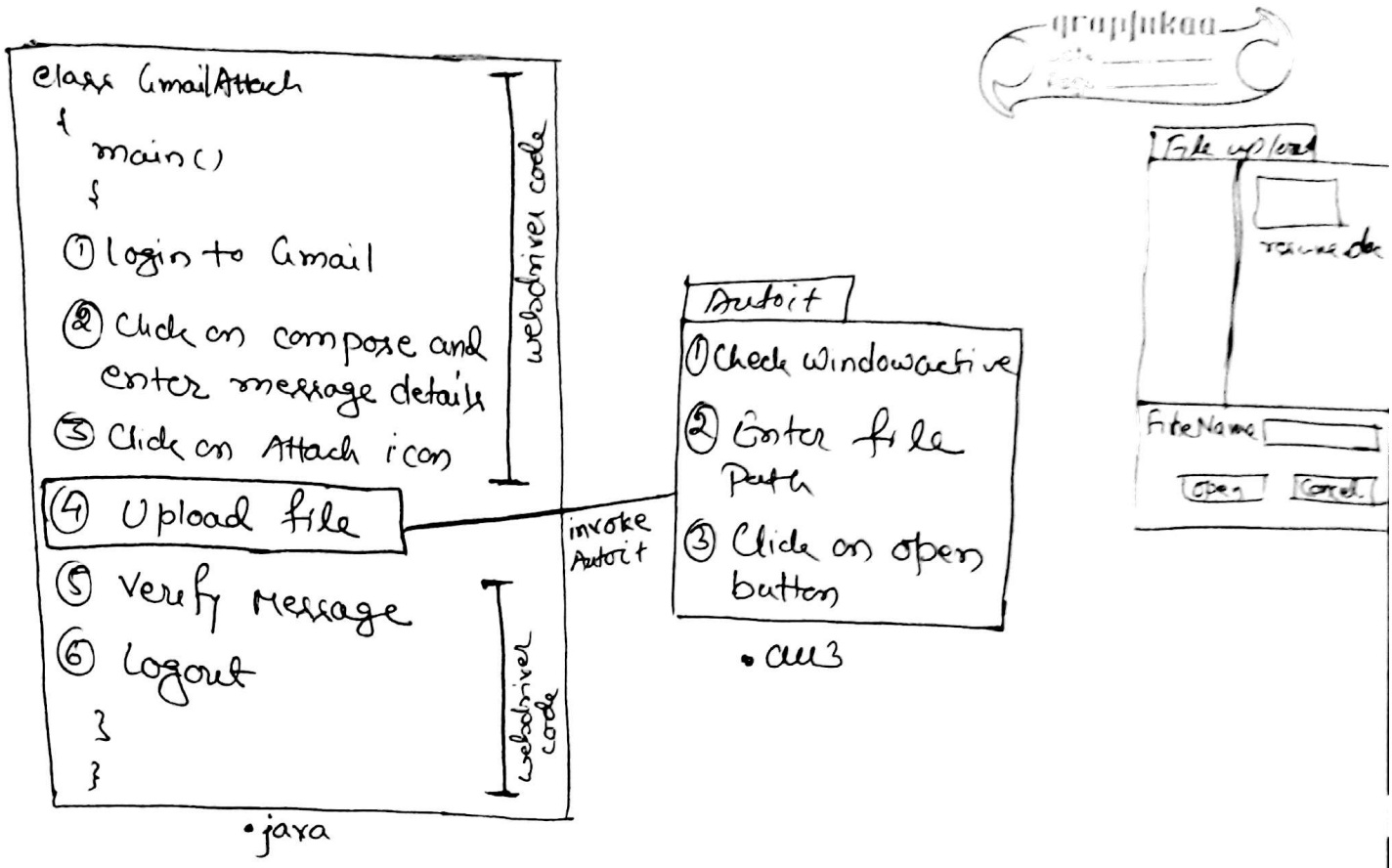
Keys	Value
browser.download.folderList	0 – desktop 1 – download folder 2 – local drive path
browser.helperApps.neverAsk.saveToDisk	mime Type • zip = application/zip • xml = application/xml • png = image/png
browser.download.dir	specify local drive path

5) Browser configuration setting should be done before launching firefox driver and it is applicable only for webdriver launched browser.

6) File download popup is only handled by using firefox browser. (filedownload popup cannot be automatable in i.e, chrome safari browser).

FILE UPLOAD / ATTACHMENT

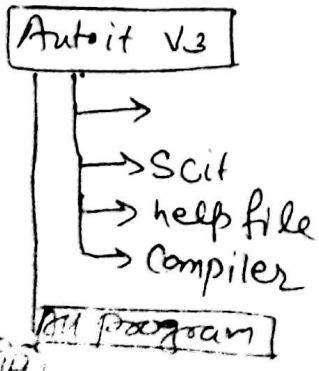
→ File attachment popup cannot be automatable using webdriver, because we have to attach the file from desktop or local system.

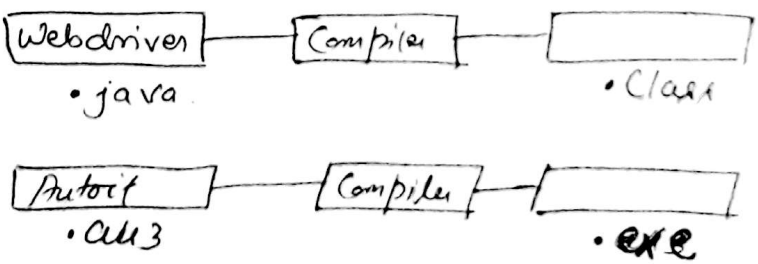


→ Autoit is another automation tool supports only standalone application, in order to handle file attachment popup either go for Autoit or Sikuli

Installation Steps for Autoit

→ Go to google search for download Autoit
→ Click on first link navigate to Autoit community.
→ Scroll down and click on ~~Aut~~ download Autoit image in the ~~first~~ row.
→ After downloading double click on exe file to instal.
→ In order to open Autoit tool go to window start menu click on All programs find Autoit v3 folder and expand





Autoit code

```
winActivate("File Upload")
controlSetText("File Upload", "", "Edit1",
                "C:\Users\oar16\Desktop\file.txt")
controlClick("File Upload", "", "Button1")
```

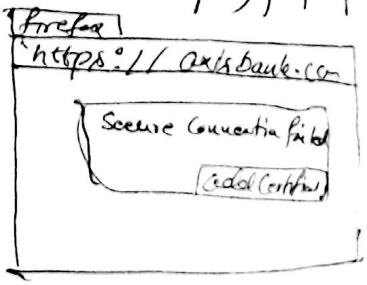
How to run Autoit Code in Eclipse

```
Package Pac
Public class Runtime {
  Public static void main(String[] args) {
    System.out.println("write webdriver code till click on attach");
    Runtime.getRuntime().exec("C:\\Users\\oar16\\Desktop\\
                              file Upload.exe");
  }
}
```

SSL Popup

Date: 20/05/16

* Write a code to handle SSL (secure socket layer) popup




```
Public class SecurityPopup{
  Public static void main(String[]args){
   FirefoxProfile p= new FirefoxProfile();
  P.setAssumeUntrustedCertificateIssuer(false);
  WebDriver driver = new FirefoxDriver(p);
   driver.get("https://axisbank.com");
  }
}
```

* Write a program to add firebug plugin to webdriver launched browser.

```
Public class Firebug{xt {
Public static void main(String[] args){
  FirefoxProfile P= new FirefoxProfile();
  File f = new File("C:\\User\\oar16\\Desktop\\firebug-2.0.16-fx.xpi");
```

```
P. addExtension(f);
P. setPreference("extensions.firebug.currentVersion",
                 "2.0.16");
WebDriver driver = new FirefoxDriver(p);
 driver.get("https://gmail.com");
 }
}
```

TEST NG

→ TestNG is a unit testing framework tool.

Unit Testing Tool

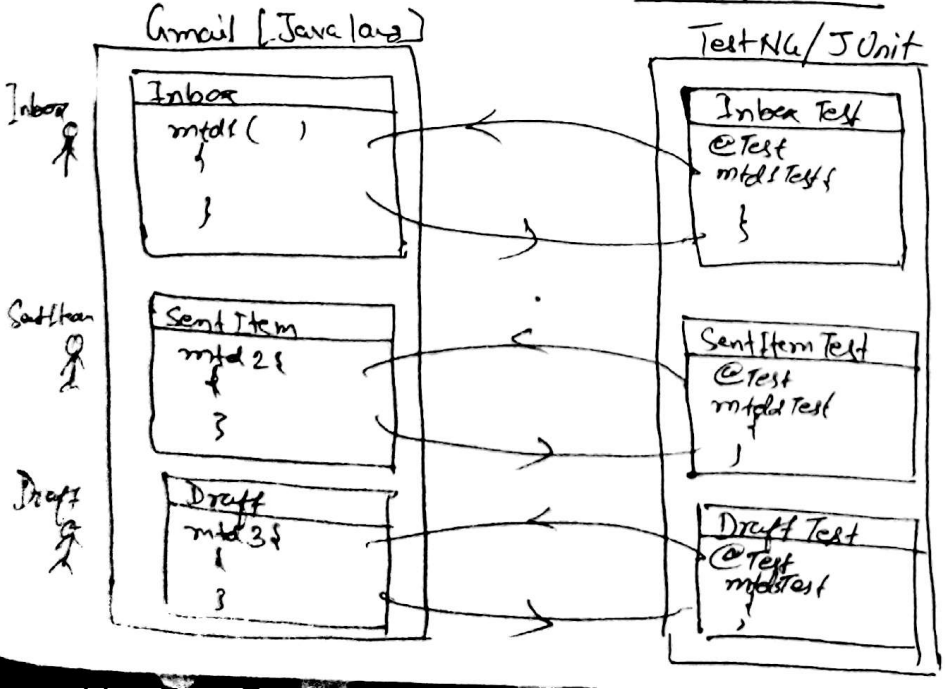
- → JUnit → Java
- → NUnit → .Net
- → TestNG → Java, .net
- → P.ydev → Python
- → Rspec → Ruby

→ All Unit testing tools is implemented as plugin for Eclipse

→ Test Next Generation is advanced unit testing tool which supports both Java and .net.

→ JUnit is a default plugin for Eclipse, no need to install to use it.

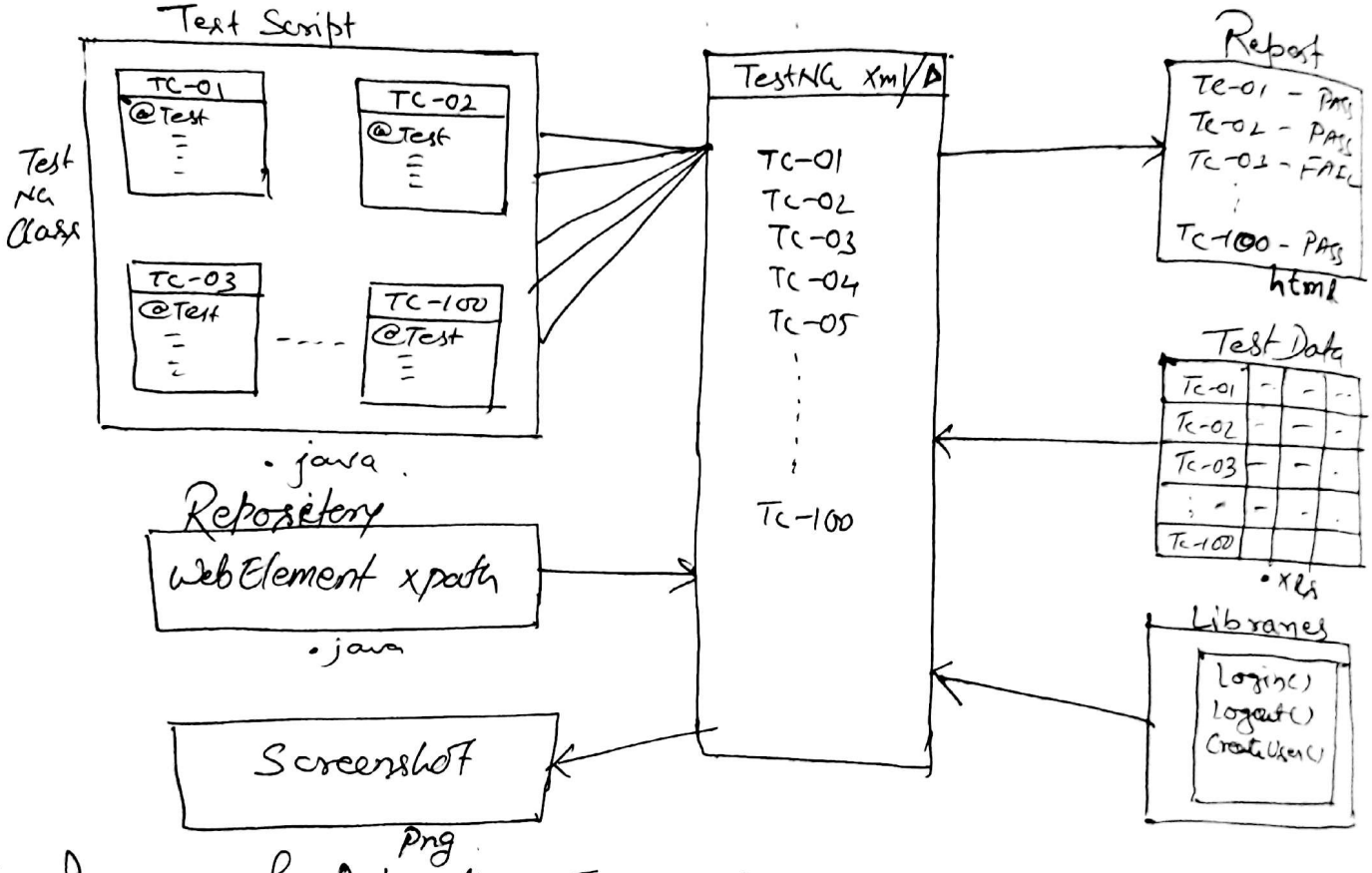
TestNG Uses in case of Development



→ In case of development TestNG will be used to develop unit test cases, Each unit test case will go and test business logic of the source code.

→ Maven/Ant is a build testing tool which is used to check the compilation issue in entire source code and create an executable build like .exe, .war, .jar.

* TestNG uses in case of Automation.



→ In case of Automation TestNG will be used to achieve Batch execution, In order to achieve batch execution Selenium test script should be implemented within TestNG class.

→ TestNG is used to handle framework component and achieve execution without any manual interaction.

→ TestNG is inspired from JUnit and NUnit but introducing some new functionality that makes TestNG more powerful.

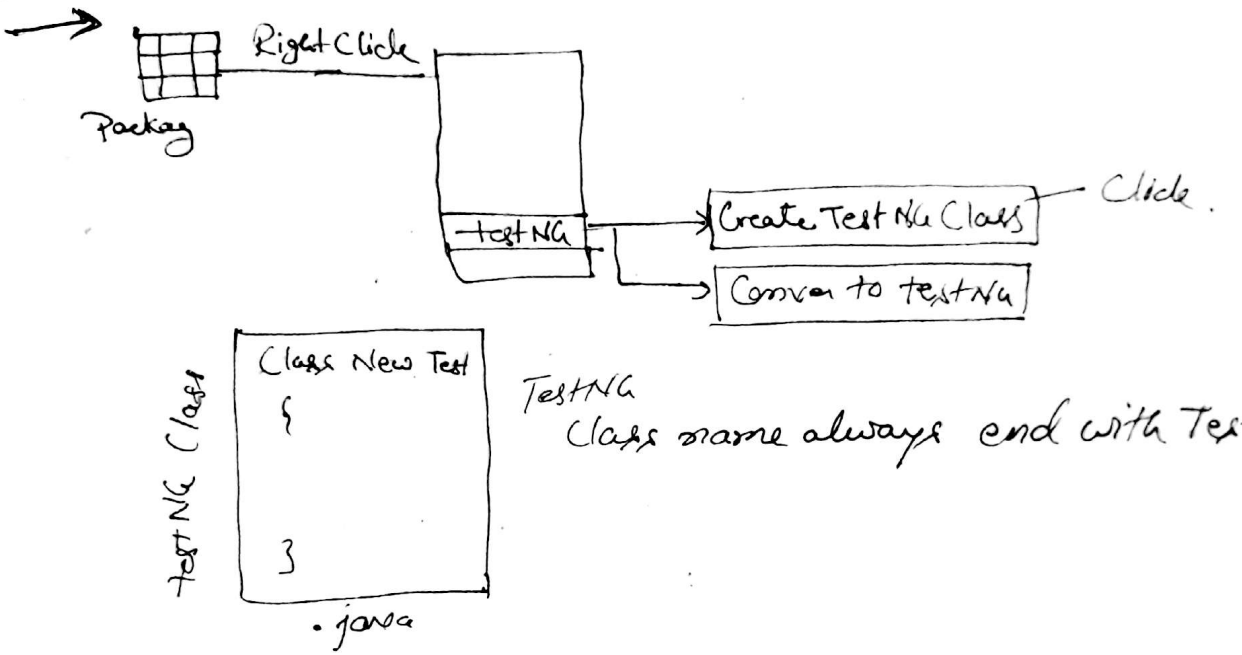
→ TestNG is an Open source Unit Testing Framework tool.

Additional functionality of TestNG are:-

1) HTML Report
2) Grouping execution
3) Parallel execution
4) Parameterization
5) Batch execution. is easier.

* Writing TestNG test script typically uses 4 steps.

Step 1) Create TestNG class



TestNG
Class name always end with Test.

Step 2) Create TestNG testcase within a class.

```
Package Pack1;
Public class NewTest {
 @Test
 Public void createCustomerTest() {
   System.out.println("execute createCustomer Test");
 }
}
```

Step 3 Run TestNG class

Step4) Verify TestNG Result

TestNG generate 3 types of reports

(1) Eclipse console report
(2) TestNG console report
(3) HTML Report.

→ In order to verify HTML report, we have to refresh the project after the TestNG class execution.

→ After refreshing automatically we will get test-output folder within same project.

→ Expand test-output folder and select emailable-report.html right click open with web browser. we will get HTML report

Note:- In order to enable Eclipse default windows like result console, Package Explorer etc go to Eclipse Window → Show View → Select expected feature.

* ANNOTATION

→ Annotation is a metadata which provides information to TestNG compiler @ at the time of execution.

List of annotations:-

→ @Test
→ @Before method
→ @After method
→ @Before class
→ @After class
→ @Before group
→ @After group
→ @Before Suite
→ @After Suite
→ @Data provider.

* @Test

→ Whenever we execute TestNG class, TestNG compiler always looks for @Test annotation method to start the class execution.

→ @Test method act like a main method.

→ In TestNG class we can create multiple TestNG test case, but each Test method should have @Test annotation before the method signature.

→ As per coding standard TestNG method name should be manual Testcase name and name should end with Test, and TestNG class name should also end with Test.

→ TestNG test method return type should be void and Access specifier should be public.

→ In order to execute TestNG class, atleast we should have one @Test annotion method.

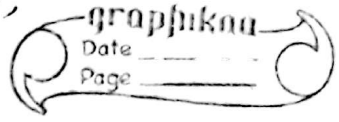
→ As per the Rule one Annotion (@Test) method is used to automate one testcase only.

Ex:-
```
package pack1;
    Public class NewTest {
    @Test
    Public void createCustomerTest() {
        webdriver code for login;
        webdriver code for create customer;
        webdriver code for verify customer;
        webdriver code for logout;
    }
}
```

* @Before method & @After method.

Ex:-
```
Public class NewTest {
    @Test
    Public void createCustomerTest() {
```

```
System.out.println("CreateCustomer Test");
}
```




```
@BeforeMethod
  Public void configBeforeMtd(){
    System.out.println("execute login mtd");
  }

@AfterMethod
  public void configAfterMtd(){
    System.out.println("execute logout mtd");
  }
```

O/P:- execute login mtd.
CreateCustomer Test
execute logout mtd.

actiTime Application test Case

Date:- 24/05/16

Project and Customer

Create Customer
- ① Launch browser
- ② Login
- ③ Create Customer
- ④ Verify Customer
- ⑤ Logout
- ⑥ Close browser

Modify Customer
- ① Launch browser
- ② Login
- ③ Create & modify Customer
- ④ Verify Customer
- ⑤ Logout
- ⑥ Close browser

Points:→ Before annotation method will be executed before executing each test case in a class

→ After annotation method will be executed after executing each test case in a class.

→ Befor & After method will be used to automate pre and Post condition of the test condition.

→ Without @Test we cannot execute before and after method annotation.

@Before class & @After class

```
→ public class ProjectAndCustomerTest {
   @BeforeClass
   Public void configBeforeClass() {
     System.out.println("launch browser, object/variable initialization");
   }

  @BeforeMethod
   public void configBeforeMtd() {
     System.out.println("login");
   }
  @Test
    Public void createCustomerTest() {
    System.out.println("Create Customer Test");
    }

  @Test
    Public void ModifyCustomerTest() {
    System.out.println("Modify customer Test");
    }

  @AfterMethod
    Public void configAfterMtd() {
     System.out.println("logout");
    }

  @AfterClass
    Public void configAfterClass() {
     System.out.println("Close Browser");
     }
```

Output

→ launch browser, object/variable initialization
→ login
→ Create Customer Test
→ logout
→ login
→ Modify Customer Test
→ logout
→ Close Browser.

```
Class ProjectAndCustomerTest
int i
webDriver d;

@BeforeClass
  i = 10;
  d = new FirefoxDriver();

@Test
Public void CreateCustomerTest()
{ Sopln(i)
} d.get()

@Test
Public void ModifyCustomerTest()
{
  Sopln(i)
  d.get()
}
```

Points

→ @BeforeClass Annotation method will be executed before executing first test in a class

→ @AfterClass Annotation method will be executed After executing all the test in a class.

→ BeforeClass & AfterClass Annotation method will be used to implement global configuration steps like launch browser, variable/Object initialization etc

→ As per Automation rule one testcase shouldnot have a dependency to other test case, because if one test case fails all dependent test case are going to fail (Each test case should be unique).

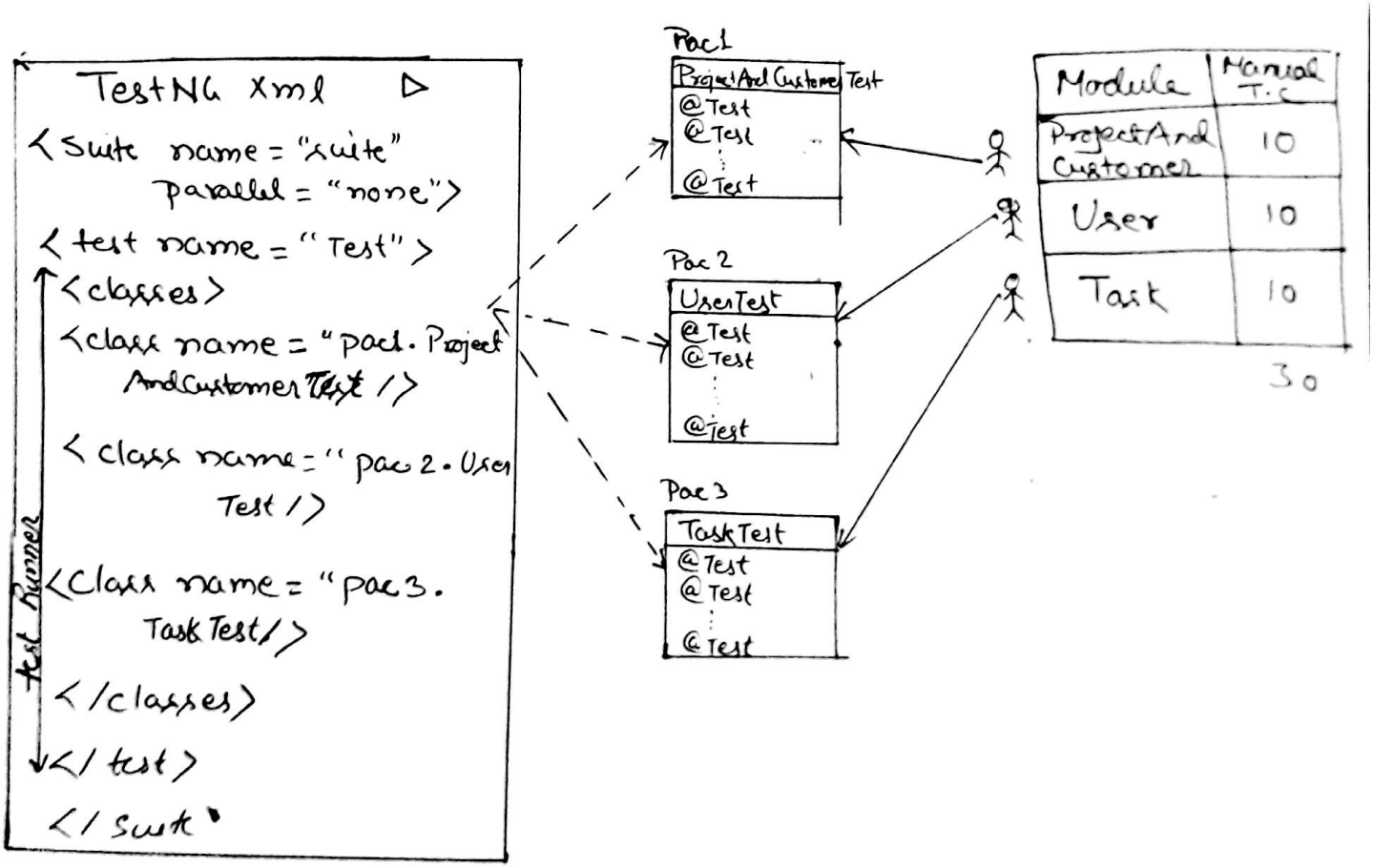
BATCH EXECUTION

>How to create TestNG xml file automatically through Eclipse?

→ Select all the package, right click and select TestNG option & click on ~~connect~~ convert to TestNG & then click on finish.

→ Automatically then xml file will be created in the same project.

→ Whenever we double click/open xml file we might see xml in table format, to open source code click on source button.



→ Collection of multiple testcase is called Test suite

→ Execute multiple testcase through TestNG.xml file is called Batch Execution.

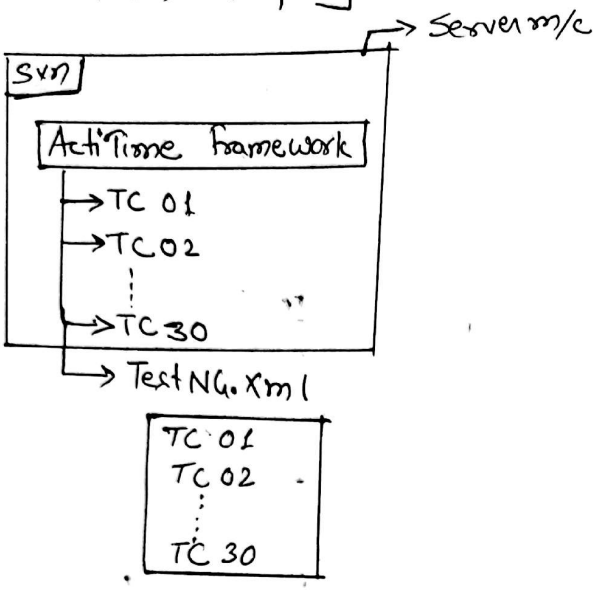
→ In order to achieve batch execution, all the test case should be implemented using TestNG classes and TestNG annotations.

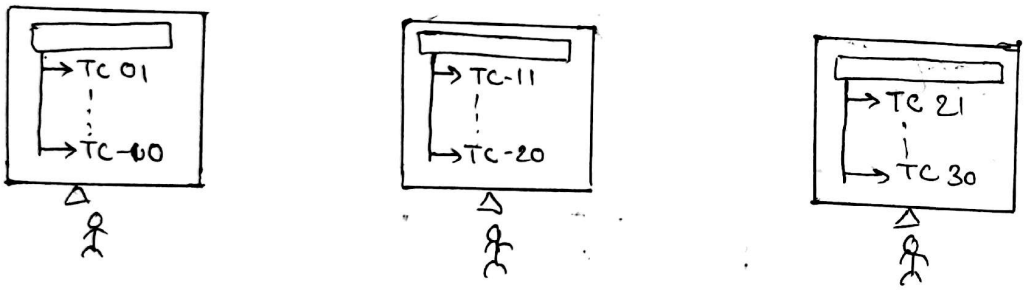
→ In TestNG Batch Execution can be done by xml & xml file always start with suite test & classes xml tags.

→ One TestNG suite can have multiple Test Runners.

→ In order to execute n-no of TestNG classes specify the TestNG class Name along with package name in class xml tag.

→ In one xml file we can invoke n-no of testNG classes but class should be within a project.





→ TestNG always execute the test based on alphabatical order

→ In order to change the order of execution either provide priority or depends on method

→ Whenever we execute TestNG class all the test case will be executed.

→ In order to skip or disabled few test cases provide (enable = false) attribute along with annotation.

→ In order to execute same test case multiple times provide (invocation count = 10) attribute along with test annotation.

→ To preserve order of packages in TestNG xml file.

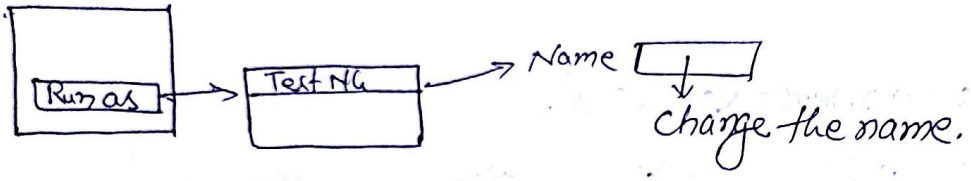
<suite name = "suite" parallel = "none"
preserve order = "true">

* Suite xml tag contains name & parallel mandatory attributes and few additional/optional attributes like preserve order and thread count.

Preserve Order :- Using preserve order attribute we can execute all the test cases in order which choose our specified way in xml file.

* In order to achieve the Batch Execution
Select TestNG file → right click → Run as → TestNG suite.

* We can create more than one TestNG class but name should be different.



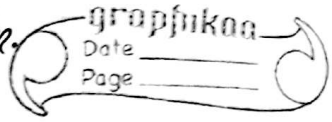
Dependent Methods:-

Eg:-
```
package pac;
public class OrderTest
{
 @Test(Priority=1)
public void createSalesOrderTest()
 {
 System.out.println("Login, CreateSO, Verify, Logout");
 }
 @Test (Priority=2)
  public void CreateBillingTest().
   {
    System.out.println("Login, CreateBilling, Verify, Logout");
   }
```

Output

Login, CreateSO, Verify, Logout
Login, CreatBilling, Verify, Logout

suppose we won't give priority at @Test method TestNG execute based on alphabetical order.



Then Output will be

Login, CreateBilling, Verify, Logout
Login, CreateSO, verify, Logout.

→ TestNG always executes the @Test method based on alphabetical order.

→ In order to execute the test case in one specified order either go for Priority or Dependson Methods.

→ "0" is the topmost priority for the Test.
"-ve" values can also be allowed in priority then that will have the highest priority.

-20 → Highest priority.
-10
0
+10
+20

→ If depends on methods used.

@Test (depends on methods = "createSalesOrderTest")

First it execute createSalesOrderTest after that it executes createBilling Test Method.

→
```
@Test
Public void CreateSalesOrderTest() {
 System.out.println("Login, CreateSO, Verify, Logout");
  }
@Test (depends on method = "createSalesOrderTest")
 Public void CreateBillingTest() {
  System.out.println("Login, CreateBilling, Verify, Logout);
  }
```

* How to disable the test case in TestNG class

→
```
@Test (enabled = "false")
Public void createSalesOrderTest() {
 System.out.println("Login, CreateSO, Verify, Logout");
 }
@Test
```

```
public void createBillingTest() {
System.out.println("Login, CreateBilling, Verify, Logout");
}
```

* How to execute same Testcase multiple times

→
```
@Test (invocationCount = 10)
  public void CreateSalesOrderTest() {
  System.out.println("Login, CreateSO, Verify, Logout");
  }
@Test
  public void CreateBillingTest() {
  System.out.println("Login, CreateBilling, Verify, Logout");
  }
```

* Priority and InvocationCount can be given together.

* Grouping Execution

Group Execution:- Execute group of similar test case is called group execution. In order to execute test cases of one group we should invoke group key in xml file and each each and every test case should have group name along with test annotation.

→ One test case can have multiple group name

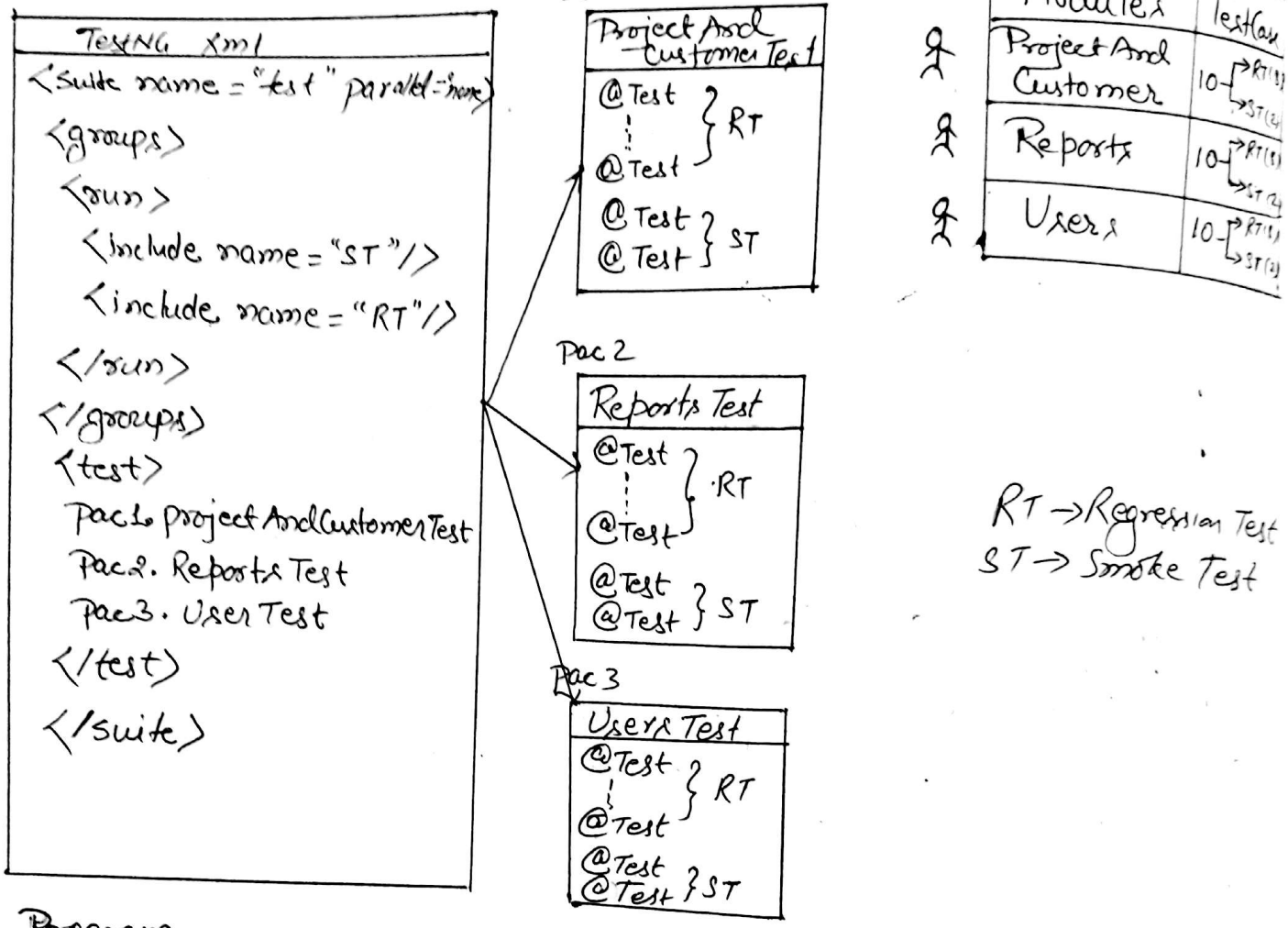
→ To execute smoke group related test case we should use include xml tag within a group

```
<groups>
  <run>
    <include name="smokeTest"/>
  </run>
</groups>
```

→ To skip smoke group related test case we should use exclude xml tag within a group

```
<groups>
  <run>
    <exclude name="smokeTest"/>
  </run>
</groups>
```

→ Collection of similar Test cases
One test case have multiple groupname.



Program

```
Package pac1;
Public class ProjectAndCustomerTest{
@Test (groups = {"Smoke Test"})
   Public void createCustomerTest(){
    System.out.println ("execute create Customer");
    }.
@Test(groups = {"Regression Test"})
   Public void modifyCustomerTest(){
    System.out.println("execute modify Customer");
    }
  }
  Package pac3;
  Public class UserTest{
@Test (groups = {"Smoke Test"})
   Public void createUserTest(){
```

```
System.out.println("execute create user");
}
@Test(groups = {"Regression Test"})
public void modify userTest(){
System.out.println("execute modify Users");
}
}
```

```
package pac2;
Public class ReportsTest {
@Test(groups = {"smoke Test"})
public void create ReportTest(){
System.out.println("execute create Report");
}
@Test(groups = {"Regression Test"})
public void modify ReportTest(){
System.out.println("execute modify Reports");
}
}
```

TestNG Xml

```
<Suite name="suite" parallel="none">
<groups>
<run>
<include name = smoke Test>
</run>
</groups>
<test name="test">
<classes>
<class Name = 'pac1.projectAndCustomerTest'/>
<class Name = 'pac2.ReportsTest />
<class Name = 'pac3.UserTest/>
</classes
</test)
</Suite)
```

o/p:- execute Create Customer
execute create Reports
execute create Users.

```
<groups>
  <run>
    <include name="Regression Test"/>
  </run>
</groups>
```

o/p:- execute modify Customer
execute modify Reports
execute modify Users.

Note:- Execute collection of similar test cases is called grouping execution, in order to invoke grouping execution insert group name in xml file, group name should be inserted before test and after suite tag.

Syntax:

```
<suite>
  <groups>
    <run>
      <include name='groupname'/>
    </run>
  </groups>
</suite>
```

* In one xml file multiple group invocation is allowed.

```
<suite>
  <groups>
    <run>
      <include name='smoketest'/>
      <include name='regressiontest'/>
    </run>
  </groups>
</suite>
```

* In order to skip similar group test case

```
<suite>
  <groups>
    <run>
      <exclude name = 'smoketest' />
    </run>
  </groups>
 <test>
```

* In order to achieve group execution each & every @Test method should have group name along with annotation (@Test), one testcase can have multiple group name.

Eg:-

```
@Test (groups = {"regressionTest", "smokeTest"})
    Public void modifyCustomerTest()
     { System.out.println("execute modifyCustomer");
     }
```

→ While grouping no both include and exclude html tag is included while executing.

→ If Before method is provided it wont execute for grouping Hence we have to provide group name.

* In grouping execution configuration method will not involved in execution without group name.

Eg:-

```
Public class
 {
 @Before Method (groups {"smokeTest", "regressionTest"})
    Public void configMtd()
     { System.out.println("before");
     }
 @Test (groups {"smokeTest"})
    Public void createCustomerTest()
     { System.out.println("execute createCustomer");
     }
```

```
@Test (groups={"regression Test"})
public void modifyCustomerTest()
{
System.out.println("execute modify Customer");
}
```

graphikaa
Date
Page

Output

before
execute create Customer

before
execute modify Customer.

→ If Before/After → class/Method to be execute in grouping execution we should provide group name.

Eg
```
<groups>
  <run>
    <include name='smoke Test'/>
    <include name='regression Test'/>
  </run>
</groups>
```

→ whenever Before method fails — all Test Method skips.

* Parallel Execution

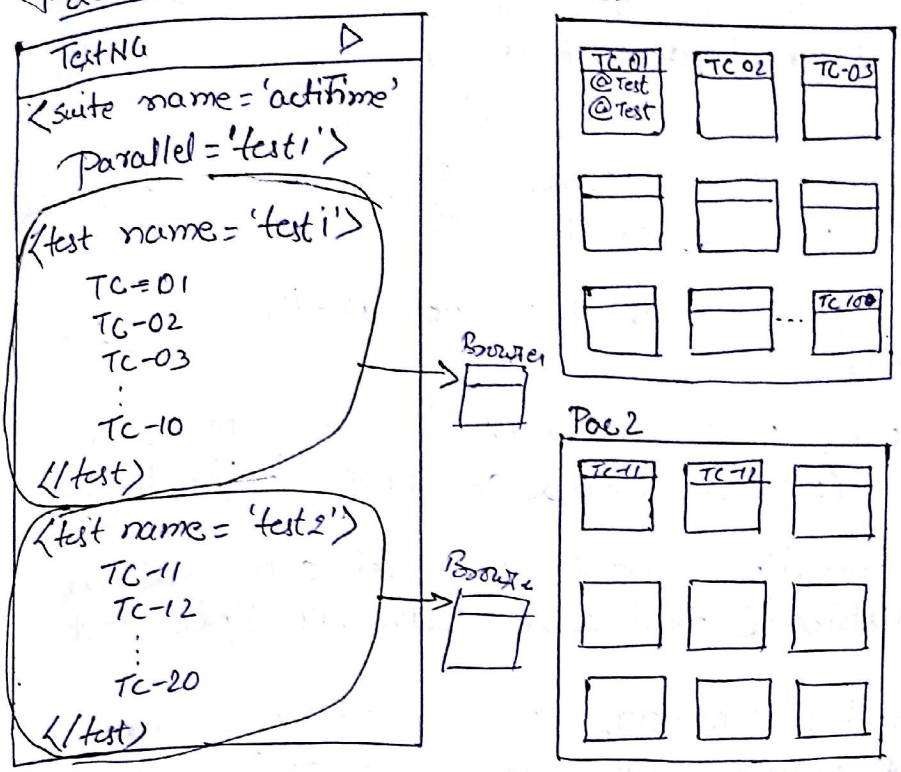
Distribute and execute multiple test case with multiple browser concurrently is called parallel execution. In order to achieve parallel execution we should enable parallel and thread count attribute in suite xml tag, and also we should create multiple test runners and distribute the test case across the test runner.

→ Thread count should be same as no of test runner

→ TestNG supports maximum 5 test runners.

~~Exceptions~~

Parallel Execution



ActiTime	
Modules	manual testCase
ProjectAnd Customer	100
User	100

200
4hr
(Sequential Execution)
2hr
(Parallel Execution)

thread count → (No of thread used to launch browser for parallel execution).
↓
maximum of 5 threads.

TestNG xml file Pallel Execution E.g

```
<Suite name="Suite" parallel="tests" thread-count="2">
  <test name="Test1">
    <classes>
      <class name="pac1.userTest"/>
    </classes>
  </test>
    <test name="Test2">
      <classes>
        <class name="pac2.ReportsTest"/>
      </classes>
    </test>
</Suite>
```

login firefox

login Chrome

If thread count = '1' Sequential execution will happen.

→ So for parallel execution, first enable the Parallel field in xml tag of suite by assigning a value "tests" & also thread count should be assigned (default is 5)
↓
maximum

Note

* In order to execute multiple testcase with multiple browsers parallelly (concurrently) we go for parallel execution.
* In order to get the result in short period of time we go for parallel execution.
* Execute same test case with different browser is called cross-browser testing (or) also called as compatability testing.
* To achieve parallel execution, enable parallel attribute in suite xml tag & distribute test case with multiple test runners.
* Thread count should be depending on No. of test runner and maximum test runner we can go till 5 (Real time theoritically 2 is maximum).

ASSERTIONS

Eg: Login page verification

① Verify invalid msg
- → login to actiTime with invalid data
- → Verify the error msg.

ER → "invalid msg should be displayed"

② Verify Logo.
- → navigate to actiTime
- → Verify Login page Logo

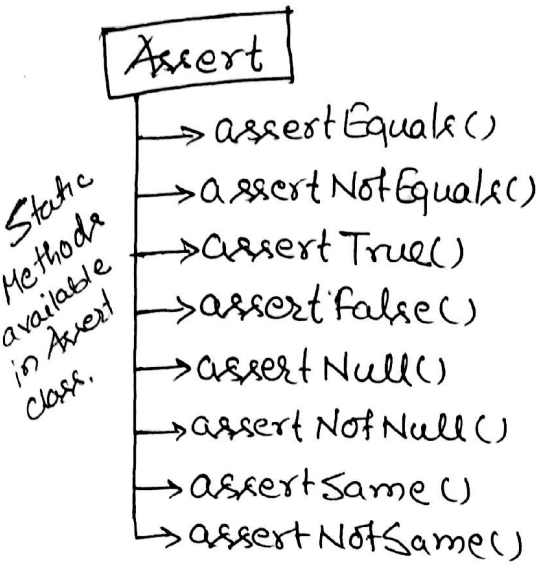
ER → "Logo Image should be displayed.

if (expectedval.equals (actval))

true means execute if statement
false means execute else statement.

→ but it fails also it will not show count of fails but shows it as a pass.

→ In testNG, if we use if else statement in java then there will be no failure of test case so we go for Assertions (means failure count can be evaluated).



assertEquals()

Syntax:- Assert.assertEquals("actval", "Expectedval", "ErrorMsg")

Mandatory — Optional

assertTrue()

Syntax:- Assert.assertTrue(condition, "ErrorMsg")

Program:-

```
Public class ProjectAndCustomerTest
{
  @Test
  public void VerifyInvalidMsgTest()
  {
   WebDriver driver = new FirefoxDriver();
   driver.get("http://deepu-pc/login.do");
   String expectedMsg = "Username or Password is invalid";
   driver.findElement(By.id("loginButton")).click();
   String actMsg = driver.findElement(By.xpath("//span[
                contains(text(),'Pa')]")).getText();
   Assert.assertEquals(actMsg, ExpectedMsg, "expected
                msg is not displayed == FAIL");
   System.out.println("msg is displayed == PASS");
   driver.quit();
  }
```

```
@Test
Public void verifyLogoTest()
{
 WebDriver driver = new FirefoxDriver();
 driver.get("http://deepa-pc/login.do");
 boolean flag = driver.findElement(By.xpath("//img[contains
                (@src,'time.gif')]")).isDisplayed();
 Assert.assertTrue(flag, "image is not displayed==FAIL");
 System.out.println("image is displayed==PASS");
 driver.quit();
}
}
```

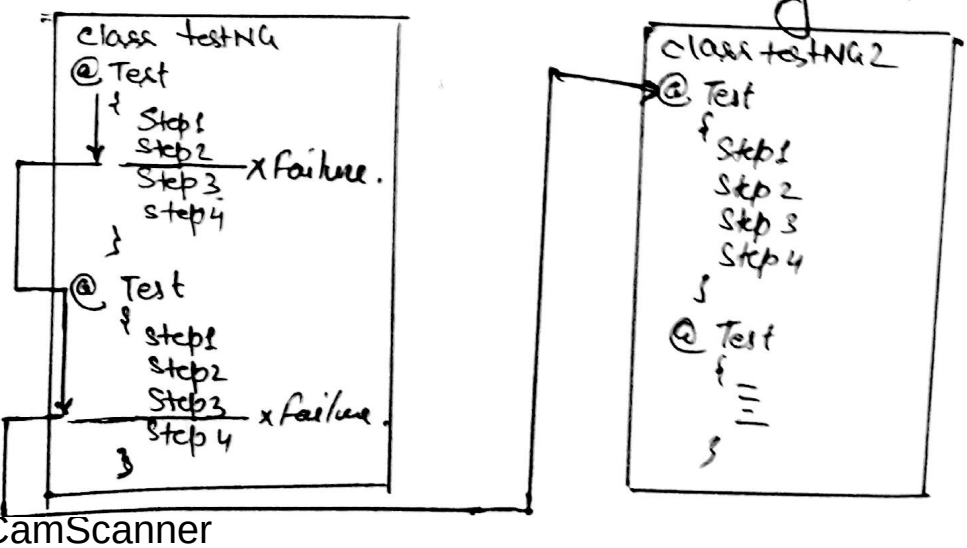
Output

run: 2 Failure: 1

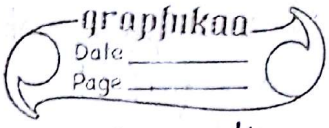
Assertion Error Exception:- actual with expected result at what line is error.

NOTE

* Assert is a TestNG class which is used to verify the expected result of the test case.
* All the methods available in Assert class are static methods, those method can be used to verify toString variable, Array, collection.
* As per Automation rule expected result should be verified with assert methods.
* Whenever assert method fails testNG throws assert error exception along with error message and stop current test execution, and continue execution with remaining test script.



* Whenever assert method fails in test suite execution, TestNG stop the current test and continue execution with remaining test, it generate error message along with line number, line number will be use to debug the test script.



Advantage of Assert.

* Assert methods are used to fail the testNG test.
* Assert methods are used for debugging because whenever assert fails TestNG fail the test along with line Number. (To find where test case got failed).
* Using Assert methods we can compare any primitive datatype or collection or Arrays.
  - → Assert.equals() → compare primitive type
    assert.same() → compare object type.

Note
Assert.same() method is used to compare two object or two object array only.

Scenario
1. Login to facebook.
2. Upload Image
3. Verify image is displayed
4. Remove image
5. Verify image should not be available in UI.

```
boolean flag1= driver.findElement(By.xpath("//img")).isDisplayed
Assert.assertTrue(flag1)
boolean flag2 = driver.findElement(By.xpath("//img")).isDisplayed
Assert.assertTrue(flag2); -x→ it fails the test
Assert.assertFalse(flag2); ✓
```

Note:-
* assertFalse() method is always look for false boolean value. If the argument is false pass the test or else fail the test.

* assertTrue() method is always look for true argument. If the argument is true pass the test or else fail the test.

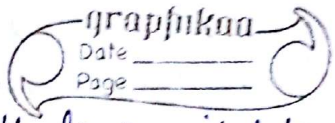
→ Soft Assest → is similar to Verify checkpoint in selenium IDE
In java if-else acts as verify
Verify → is like if-else does not fails the test case
Soft assert fails the testNG test throws exception but continues further execution on the same test.
SoftAssert ⊙ methods are non static.
↓
assertAll() → collect all the error & display in console it is mandatory.

```
Public class UserTest
{
 @Test
  Public void CreateUserTest()
  {
   SoftAssert s = new SoftAssert();
   System.out.println("Step: 1");
   s.assertEquals("A","B","Customer name is not same");
   System.out.println("Step: 2");
   s.assertEquals("100","101","Customer amount is not same");
   System.out.println("Step: 3");
   s.assertEquals("1 cross","1 cross","add is not same");
   System.out.println("Step: 4");
   s.assertAll(); SOP("execute create user");
  }
 @Test
   Public void ModifyUserTest()
   {
    System.out.println("execute modify user");
   }
}
```

O/P test run: 2 failure: 1 Skip: 0
Step: 1
Step: 2 — Assert Error Exception
Step: 3 — Customer name is not same
Step: 4 — Customer amount is not same
execute Create User
execute modify User.

Diff between Soft Assert and Hard Assert.

Soft Assert	Hard Assert
* All the methods available in soft Assert are non static	* All the methods available in Hard Assert are static.
* Whenever soft Assert fails TestNG fail the testNG test & continue remaining steps execution.	* Whenever Hard Assert fails TestNG fail the testNG test & generate error msg & stop the current test execution.
* assertAll() method should be inserted at the end of the test, to collect all the assert error.	* assertAll() method is not required (not available only)



Interview Question

Q) What is TestNG ?

→ Unit testing framework

Q) Why TestNG ? why not Junit?

→ To achieve batch execution we go for TestNG.
Junit does not support.
Parallel execution
Grouping execution
html reports
So we go for TestNG

← Junit is similar to TestNG but no these feature & also to generate report we have to use java.

Q) What is Annotation and Explain all the annotation with e.g?

Q) We have 100 testNG class how to execute 10 testNG class.

→ In xml file invoke only 10 test class.

Q) We have Smoke test script in all testNG classes how to execute only smoke test case.

→ By using grouping.

Q) How to achieve batch execution in TestNG?

→ Create xml file & include all the classes & execute.

Q) What is grouping execution & How to achieve?

→ Execute similar test case.

Q) What is dependent test case, how to execute ~~group~~ test case in custom order.

→ Either use priority or dependecon.

Q) How to execute same test case n number of times

→ invocation count.

Q) What is parallel execution, and how to achieve parallel execution?

Q) What is assert, explain all the assert method with real time example.

Q) Difference between soft assert and hard assert?

Q) What is listners annotation and where to use?
Keep on monitoring execution event at run time & if bug exists take a screenshot.

→ Listener is an annotation in TestNG which is use to monitor the execution events happening in runtime. if any test case fails using listners we can take a screenshot.

Q) Whenever we execute 100 of test case through batch execution, how to execute only failed test case.

→ In order to execute failed test case invoke(execute) testing-failed.xml file available in test-output folder.

Q) I have 10 test case in testNG class how to execute only one test.

→ There are two ways.

① In 9 test case we have to put enabled=false.

②
```
<Suite name="suite">
    <test name="Test">
      <classes>
        <class name="com.actiTime.tasks.STest">
          <methods>
            <include name="verifyLogo"/>
          </methods>
        </class>
      </classes>
    </test>
</Suite>
```

Q What is parameterization and How to achieve parameterization?

→ Execute same test case with different test data is called parameterization.

graphikaa
Date
Page

Scenario: Create Account.

1. Login to Bank
2. Create 100 Account.
3. Verify all Account
4. Logout

Inorder to achieve parameterization in TestNG we use @dataprovider annotation.

```
Class Banking.
{
@Test.
Public void CreateAccount(Retc)
{ 1
  2
  3
  4
}
@dataprovider
Public void getdata()
{
}
```

Excel.

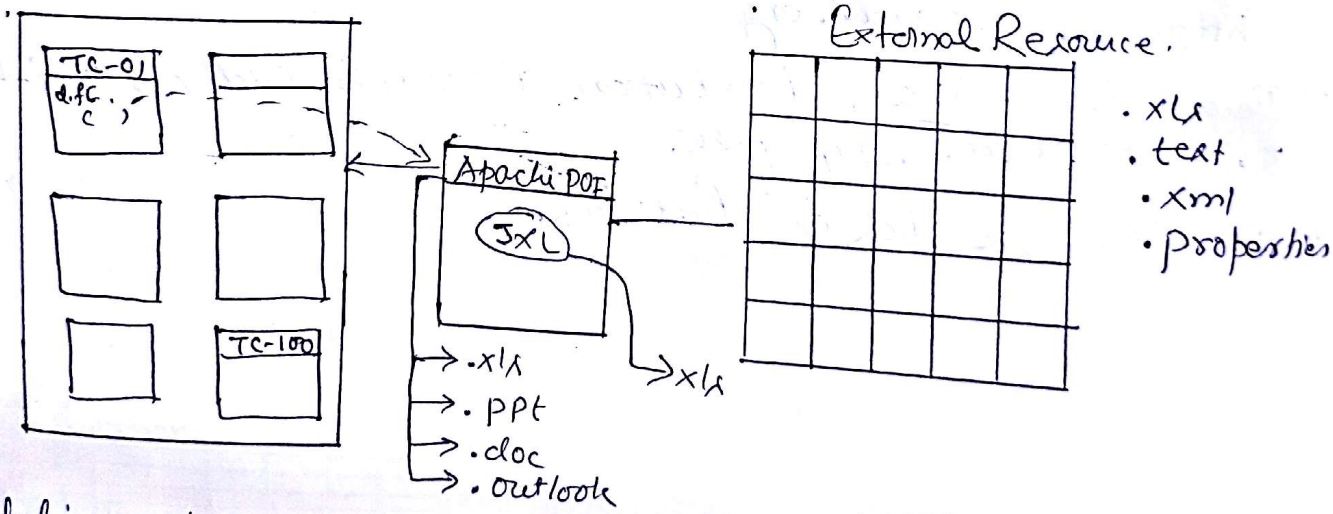
Acc No.	Pwd.
Acc 1	Pwd 1
Acc 2	Pwd 2
Ac 100	Pwd 100

File Handeling

Data fixed in Test Case → Hard coded data.

If 1000 test cases we have to change data, it will be tedious process, so we do not hard code test data in test case.

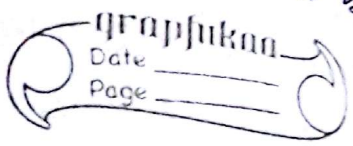
We use External Resource to provide ~~to provide~~ test data or get Test Data.



Webdriver does not support Excel sheet (since it is a standalone application).

So we take help of "Apache POI" to support microsoft document to webDriver.

Note:- As per the Automation rule, test data should not be hard coded within the test script. instead we should get data from external resource & run the test.



* External Resource might be
  - xml
  - xlsx
  - ppt
  - db
  - txt

* WebDriver does not support xls (or) any other documents like .properties, .txt etc so in order to get data from .xls file we should take help of third party tool like Apachi POI / Jxl

* Apachi POI is an open source tool available in Apachi community which supports all microsoft documents.

* Apachi POI is not a UI based tool, it is just a collection of jars.

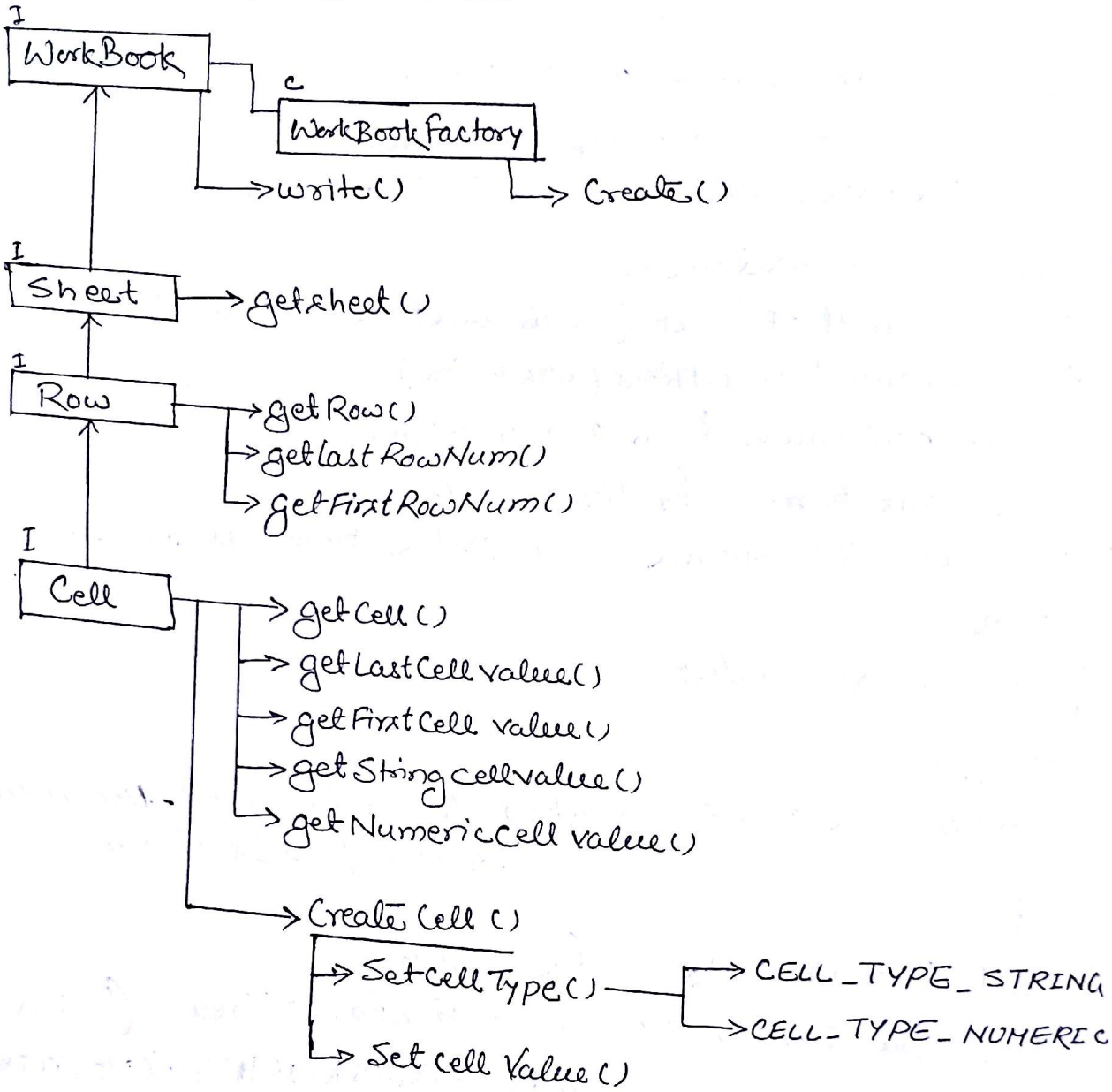
Installation steps of Apachi POI

1) Go to google type download apachi POI
2) Click on first link and navigate to apachi community https://POI.apachi.org
3) Below the Binary distribution division & click on poi-bin-3.13-20150429.zip file.
4) In other page click on first link.

Apachi POI Architecture

C FileInputStream

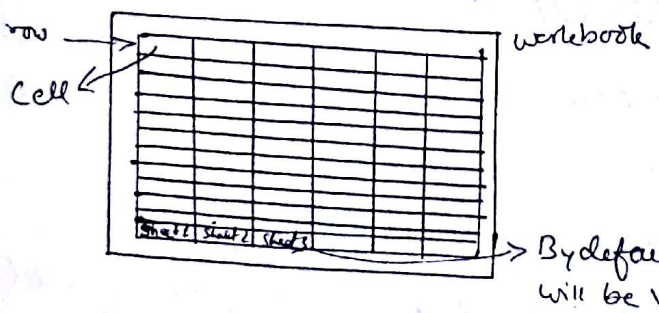
C FileOutputStream



JXL is old tool and only support xls. Apachi POI is a very powerful tool which supports all microsoft documents.

Excel → microsoft

Bottom up approach



1. Cell
2. Row
3. Sheet
   ↓
   maximum of 256 sheet can be created
   ↓
   workbook

→ Window 7
extension → .xlsx

TestName	TestData	arg0	arg1	arg2
TC01	userTest	admin	manager	Test

Sheet1

① Get location of the file using java API's (to read/write)
FileInputStream → It support Notepad
For Excel we use Apache POI Interface.

② Open Excel in Read mode.
③ Get the control of 'sheet' [which sheet is Data written]
④ Get the control of 1st Row (which Row)
⑤ Get the cell value from 2, 3, 4 columns.
→ Index starts from 0 for Row & column
→ All Apache POI should be imported from xs model.

Program

```
Public class UserTest
{
  @Test
  Public void createUserTest() throws EncryptedDocumentException,
                                      InterruptedException
  {
```

Step:1 Get the workbook file location

```
FileInputStream fis = new FileInputStream("C:\\Users\\
                      DEEPU\\DESKTOP\\TestData.xlsx");
```

Step:2 Open Workbook in read mode

```
Workbook wb = WorkbookFactory.create(fis);
```

Step:3 Get the control of the "SHEET1"

```
Sheet sh = wb.getSheet("Sheet1");
```

Step:4 Get the control of 1st Row

```
Row row = sh.getRow(1);
```

Step:5 Get the cell data from 2, 3, & 4 column.

```
String userName = row.getCell(2).getStringCellValue();
```

```
String password = row.getCell(3).getStringCellValue();
String user = row.getCell(4).getStringCellValue();
System.out.println(userName);
System.out.println(password);
System.out.println(user);
WebDriver driver = new FirefoxDriver();
   driver.findElement(By.name("username")).sendkeys
                                          (userName);
   driver.findElement(By.name("pwd")).sendkeys(password);
   driver.findElement(By.id("loginButton")).click();
   driver.quit();
   }
  }
```

Output

```
admin
manager
user1-RAM
Passed: createUserTest
Test run: 1  Failure: 0  Skip: 0
```

TC-02	CreateUser	admin	manager	RAM; SAM

```
String user = row.getCell(4).getStringCellValue()
 String[] arr = user.split(";")
 String user1 = arr[0];
 String newuser = arr[1];
   Sop(arr[0]);
   Sop(arr[1]);
```

Output

```
RAM
SAM
```

→ Write a Sample code to write data back to Excel.

To write data back to Excel

① We should close the Excel if already opened.

② First to write
open Excel sheet in read mode
& then open Excel sheet in write mode
Save & close the sheet
(write)

Program

```
Public class CreateTest
@Test
Public void createUserTest() throws EncryptedDocumentException,
                                    InvalidFormatException, IOException.

String filePath = "C:\\Users\\DEEPU\\Desktop\\TestData.xlsx";
// Open workbook
FileInputStream fis = new FileInputStream(filePath);
Workbook wb = WorkbookFactory.create(fis);
    Sheet sh = wb.getSheet("Sheet1");
    Row row = sh.getRow(1);
    Cell cell = row.createCell(5);
    Cell.setCellType(Cell.CELL_TYPE_STRING);
FileOutputStream fos = new FileOutputStream(filePATH);
    Cell.setCellValue("PASS");
    wb.write(fos);
    wb.close();
  }
}
```

O/P

Test_id	Test_Name	arg 0	arg 1	arg 2	STATUS
TC_01	Create Test	admin	manager	use1 - RAM	PASS

Note:-

* While writing WebDriver all the API should import from "org.apache.poi.ss.usermodel" package.
* While providing file location, file extension should be .xlsx
* All the jars available inside the POI folder/library should be imported to java project.
* While writing data back to Excel, already opened workbook should be closed.

FRAMEWORK

Manual test cases

Project And Customer

1. Create Customer
2. Modify Customer
3. Create Project
4. Modify project
5. Create Customer with discription
6. Create Duplicate Customer
7. Create Duplicate Project.

User

1. Create User
2. Modify User
3. Create Duplicate User
4. Create User with Contact Number.

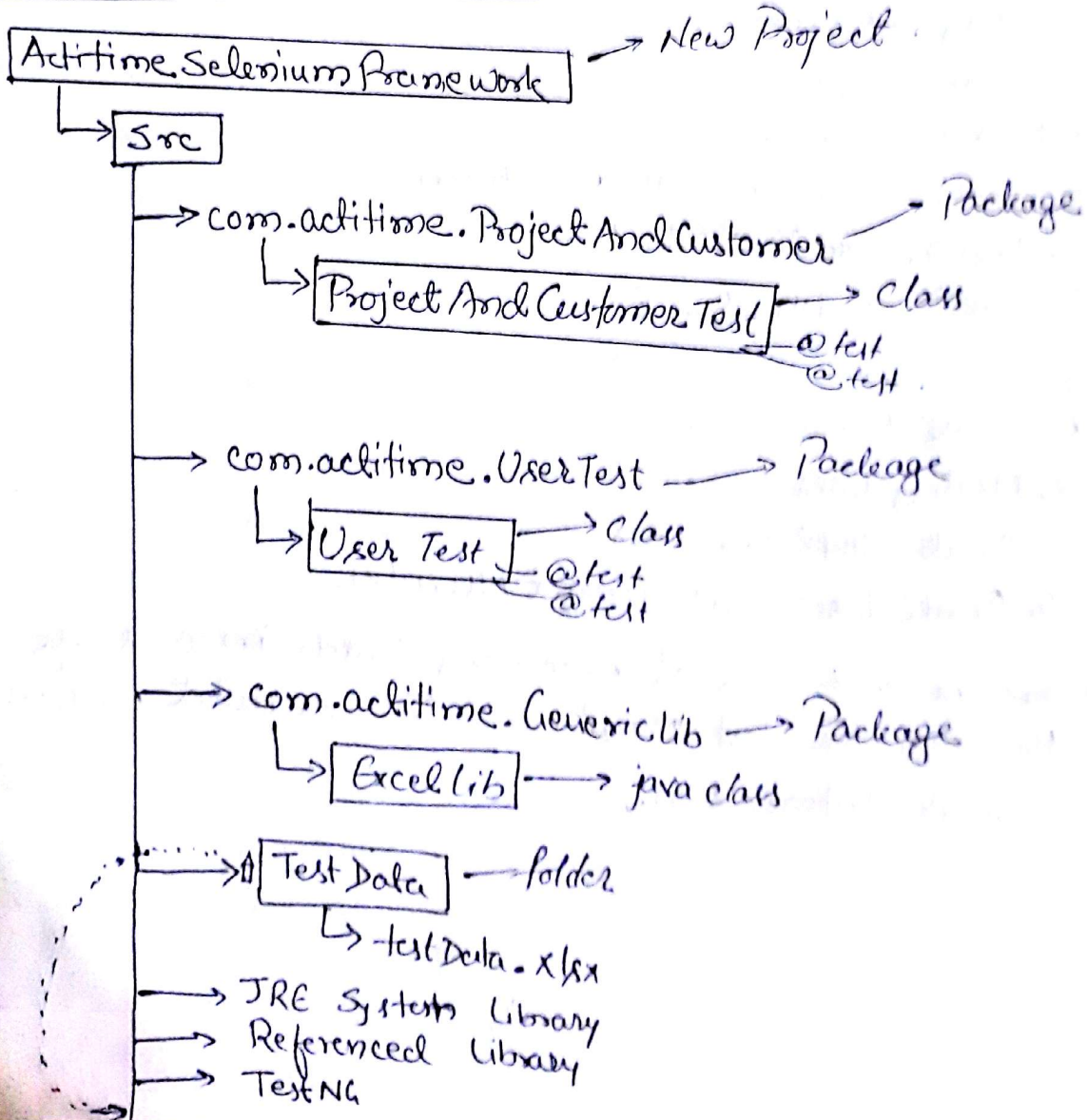
→ Framework is a process of executing batch in a single click, then all the test script will be executed without any manual interaction.

Framework Components are:-

1) testNG test script
2) test Data
3) Libraries
   - Generic Lib
   - Page Lib
4) Object Repository
5) Screenshot
6) Report
7) Batch Runner
8) Resource.

→ In Real time project test script cannot be automated without framework. framework is implemented by senior automation engineer. when framework is developed then using framework component test script should be automated

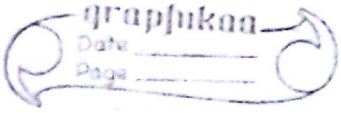
Framework Tree Structure



Framework Architecture

Test_Scripts

Generic Lib



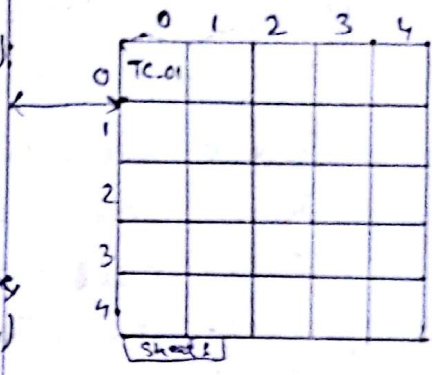
UserTest

```
ExcelLib elib = new ExcelLib();

@Test
public void CreateUserTest()
  {
  String usr = elib.getExcelData
          ("Sheet1", 1, 2);
  String pwd = elib.getExcelData
          ("Sheet1", 1, 3);
  }

@Test
Public void modifyUserTest()
  {

  }
```

ExcelLib

```
getExcelData (String
  SheetName, int row, int col)
  String
 {

 }
setExcelData(String SheetName,
  int row, int col, String data)
   : void
 {
 }
 getRowCount (String)
    : int
  {
  }
```

Test data



ExcelLib

```
Public class ExcelLib
 {
 String filepath = "C:\\User\\DEEPU\\workspace-OCM30\\Framework\\
                    TestData\\testdata.xlsx.

Public String getExcelData (String SheetName, int rowNum,
                            int colNum) throws EncryptedDocumentException IOException,
                            InvalidException.
 {
  FileInputStream fis = new FileInputStream (filepath);
    Workbook wb = WorkbookFactory.create (fis);
      Sheet sh = wb.getSheet (sheetName);
      Row row = sh.getRow (rowNum);
      String data = row.getCell (colNum).getStringCellValue();
      return data;
  }
```

```
public void setExcelData(String sheetName, int rowNum, int colNum,
    String data) throws EncryptedException,
    IOException, InvalidException
{
 FileInputStream fis = new FileInputStream(filepath)
 Workbook wb = WorkbookFactory.create(fis);
 Sheet sh = wb.getSheet(sheetName);
  Row row = sh.getRow(rowNum);
  Cell cell = row.createCell(colNum);
  Cel.setCellType(Cel.CELL_TYPE_STRING);
FileOutputStream fos = new FileOutputStream(filepath)
   cel.setCellvalue(data);
   wb.write(fos)
   wb.close();
  }
}
```

UserTest.java

```
Public class UserTest
{
 ExcelLib eLib = new ExcelLib();
 @Test
  Public void CreateUserTest()
   {
    //get test data.
    String userId = eLib.getExcelData("Sheet1", 1, 2);
    String password = eLib.getExcelData("Sheet1", 1, 3);
    String userName = eLib.getExcelData("Sheet1", 1, 4);
    //Step1: Login to app
     Webdriver driver = new FirefoxDriver();
       driver.get("http://actitime");
       driver.findElement(By.name("username")).sendkeys
                                              (userID);
       driver.findElement(By.name("pwd").sendkeys(password)
```

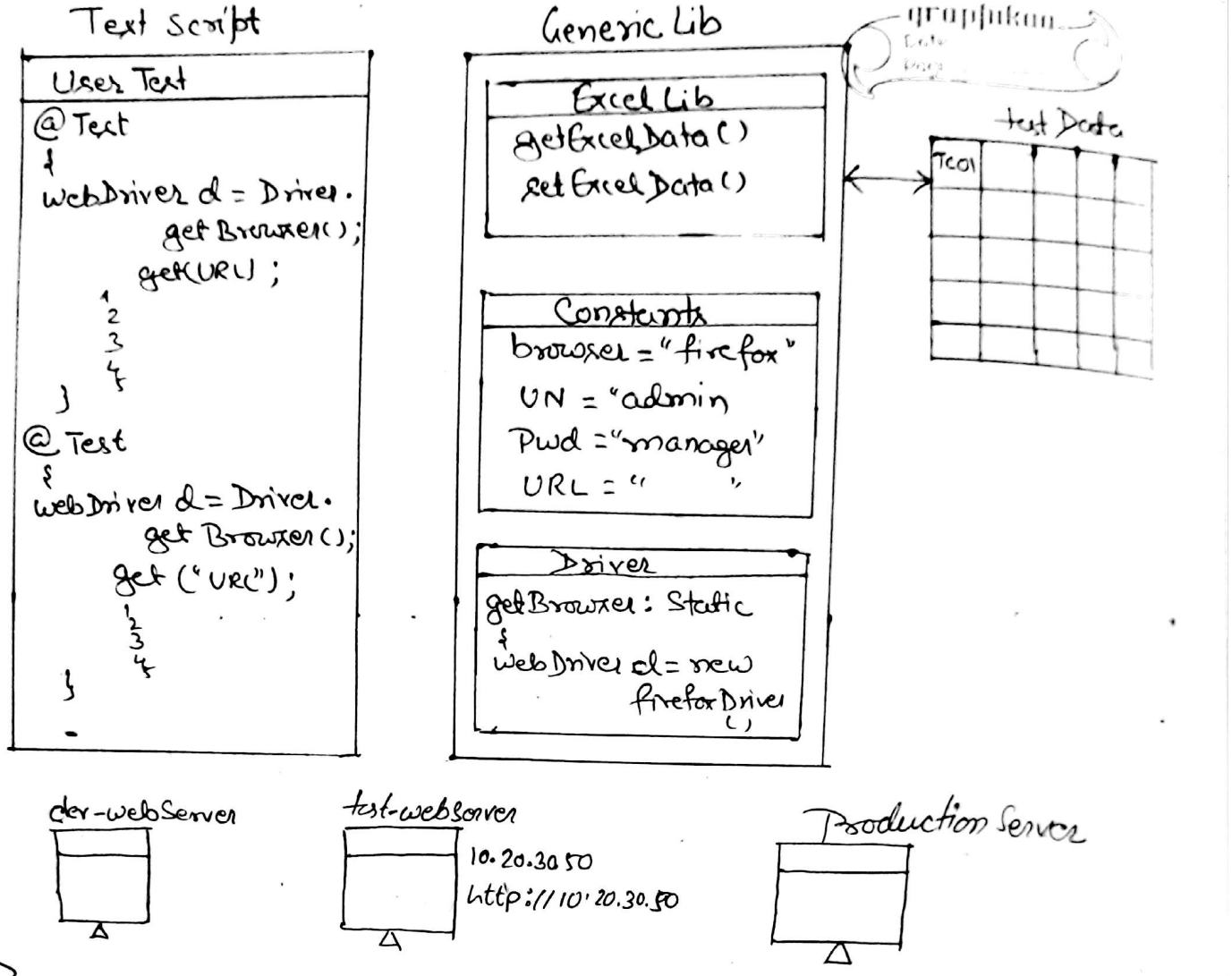
```
driver.findElement(By.id("LoginButton")).click();
Thread.sleep(2000);
driver.findElement(By.id("logout link")).click();
driver.quit();
// write data back to excel.
eLib.setExcelData("Sheet1", 1, 5, "PASS");
}
}
```

Output

Test_id	Test_Name	arg 0	arg 1	arg-2	Status
TC_01	UserTest	admin	manager	user-RAM	PASS

→ If multiple test case has to run in different browser. if we declare launching of browser in each test case then it will be a tedious process to change browser in all test case instead we create a class Driver in Generic Library it consist a code to launch browser. So to run in different browser we can only change code in driver class rather than changing in all test cases.

→ In Driver class, every time writing two lines of code to launch a browser instead we create an interface constant to which contains a variable where we will store variable details.
Whenever we want to launch application in different environment (i.e with different URL) instead of changing everytime we can move it to constant interface and also we can ~~move~~ store username & password in constant interface instead getting from excel. Since this are global constant so we store in constants Interface [No user can change only developed person has rights]



Program

Generic Lib

1

```
Public Interface Constants
{
 String browser = "firefox";
 String url = "http://deepu-pc//login.do";
 String userId = "admin";
 String password = "manager";
}
Public class Driver
{
 Public static WebDriver driver;
 Public static WebDriver getBrowser();
 {
  if (Constants.browser.equals("ie"))
  {
   System.setProperty("webdriver.ie.driver", "C:\\Users\\
   DEEPU\\DESKTOP\\IEDriverServer.exe");
```

```
driver = new InternetExplorerDriver();

}else
if(constants.browser.equals("chrome"))
 {
 System.setProperty("webdriver.chrome.driver", "C:\\Users\\
                    DEEPU\\DESKTOP\\new chromeDriverServer.exe");

  driver = new ChromeDriver();
 }
 else {
    driver = new FirefoxDriver();
    }
 return driver;
```

Or

```
public class Driver
 {
  public static WebDriver driver;
  public static WebDriver getBrowser()
   {if(constant.browser.equals("ie"))
    {System.setProperty("webdriver.ie.driver", "path of IEServer");
     driver = new InternetExplorerDriver();
    }
   else if(Constant.browser.equals("chrome"))
     {
     System.setProperty("webdriver.chrome.driver", "path of
                                                  chromeServer");

     driver = new ChromeDriver();
     }
    else
      {
       driver = new FirefoxDriver();
      }
     return driver;
   }
```

User Test Library

```
Public class UserTest
{
 Excellib elib = new Excellib();
@Test
 public void createUserTest()
 {
  // get test data
   String userName = elib.getExcelData("sheet1", 1, 2);

  // Login to application.
   WebDriver driver = Driver.getBrowser();
    driver.get(constant.url);
    driver.findElement(By.name("username")).sendkeys(constant.
                                                    userId);
    driver.findElement(By.name("pwd")).sendkeys(constant.password);
    driver.findElement(By.id("loginButton")).click();
    driver.quit();
    elib.setExcelData("sheet1", 1, 3, "PASS");
  }
}
```

→ In every test case when there is need of navigating from one page to another page we need of using implicit wait / Explicit wait, instead of writing or remembering the syntax we can use a class WebDriverCommonLib in Generic Lib which writes syntax of wait statement and use whenever required.

Uses

1. Re-usability
2. Can used to change the time.

WebDriver Common Lib

→ waitForPageToLoad() → Implicit wait

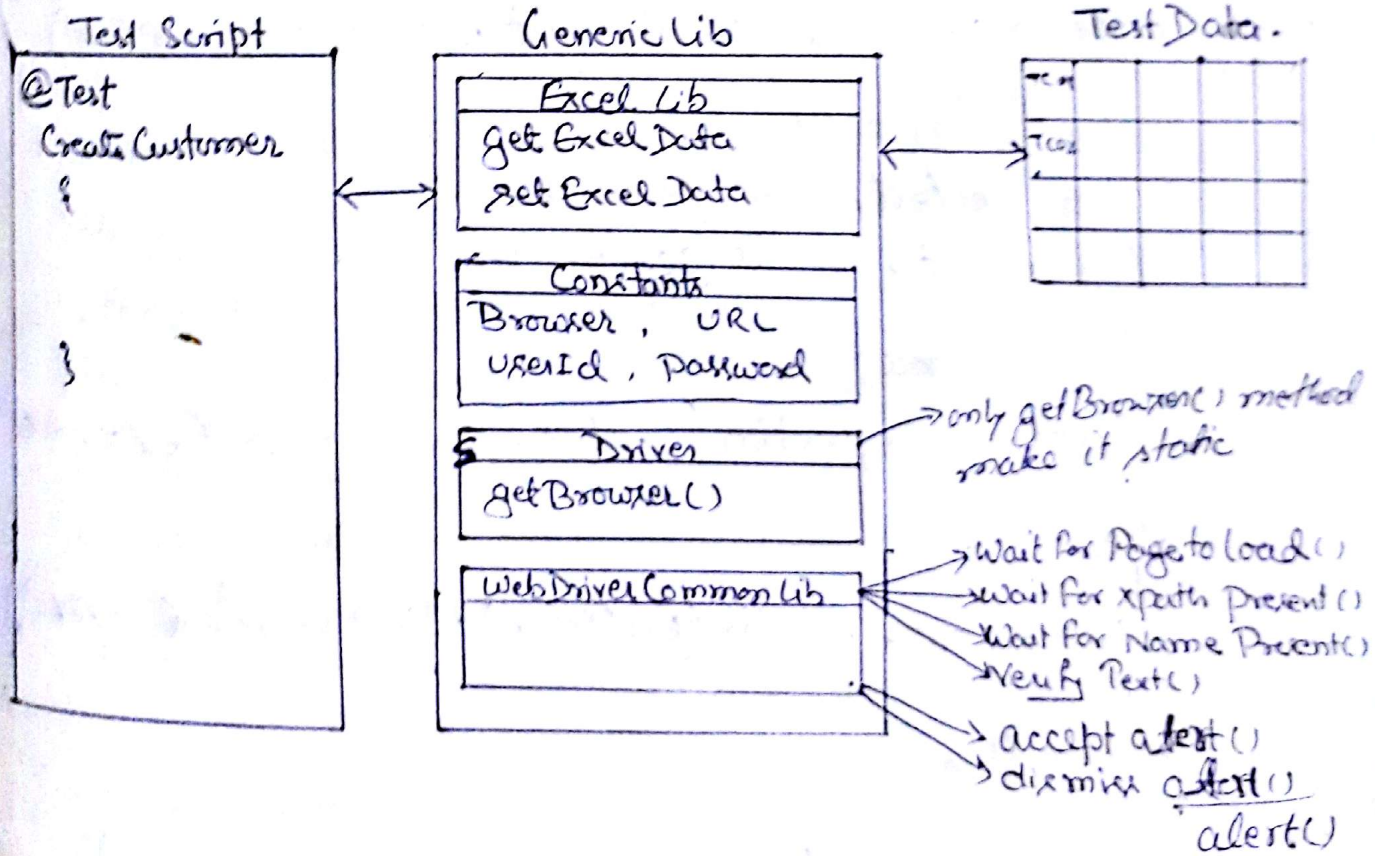
→ waitForXpathPresent() → Explicit wait

Instead of remembering syntax we can pass xpath & wait explicitly by calling this method.

→ VerifyText() → to verify the expected result with actual result instead of writing if else in each test case, use this method to verify the text by passing webElement & expectedmsg.

→ waitforNamePresent() } In Explicit wait, we use to wait
→ waitforIdPresent() } untill xpath present if we want to wait until the element present by name or id use this method.

* class WebDriver Common Lib is not used for failing test case so we use if else statement in this class & return true/false In actual Test case @Test we can use assert to fail the test case.



Program: Generic Lib

Generic Lib. webDrivercommonLib. java

```
Public class WebDriverCommonLib
{
 Public void waitforPageToLoad()
 {
  Driver.driver.manage().timeouts().implicitlywait(20,
                                        TimeUnits.Seconds);
 }
Public void WaitforXpathPresent (String wbXpath)
 {
  WebDriverWait wait = new WebDriverwait (Driver.driver, 20);
  wait.until (Expected conditions.presence of All Elements
              Located By (By..xpath(wbXpath)));
 }
Public void WaitforNamePresent (String wbName)
  {
   WebDriverWait wait = new WebDriverWait (Driver.driver, 20);
   wait.until (Expected Conditions. Presence of All Elements
              Located By (By.name (wbName));
  }
Public boolean Verify Text (WebElement wb, String ExpText)
  {
    boolean flag = false;
    String acttext= wb.getText();
    if (ExpText.equals (acttext)
     {
      flag = true;
      System.out.println (ExpText + "data is verified==PASS");
     }
    else
     {
      System.out.println(ExpText + " is not verified==FAIL");
     }
    return flag;
  }
Public void acceptAlert()
```

```
{
Alert alt = Driver.driver.switchTo().alert();
alt.accept();
}
public void CancelAlert()
{
Alert alt = Driver.driver.switchTo().alert();
alt.dismiss();
}
```

Test Scripts Library

UserTest.java

```
public class UserTest
{
ExcelLib eLib = new ExcelLib();
WebDriverCommonLib wLib = new WebDriverCommonLib();
@Test
public void CreateUserTest()
{
// Get test data
String username = eLib.getExcelData("Sheet1", 1, 2);
String password = eLib.getExcelData("Sheet1" 1, 3);
String firstName = eLib.getExcelData("Sheet1" 1, 4);
String lastName = eLib.getExcelData("Sheet1", 1, 5);
String emailId = eLib.getExcelData("Sheet1" 1, 6);
String expectedMsg = eLib.getExcelData("Sheet1', 1, 7);
// Step1: Logon to app.
WebDriver driver = Driver.getBrowser();
driver.get(constants.url);
driver.findElement(By.name("username")).sendkeys(constants.
                                                  userId);
driver.findElement(By.name("pwd")).sendkeys(constants.password);
driver.findElement(By.id("loginButton")).click();
wLib.waitForXpathPresent("//div[text()='Users']");
```

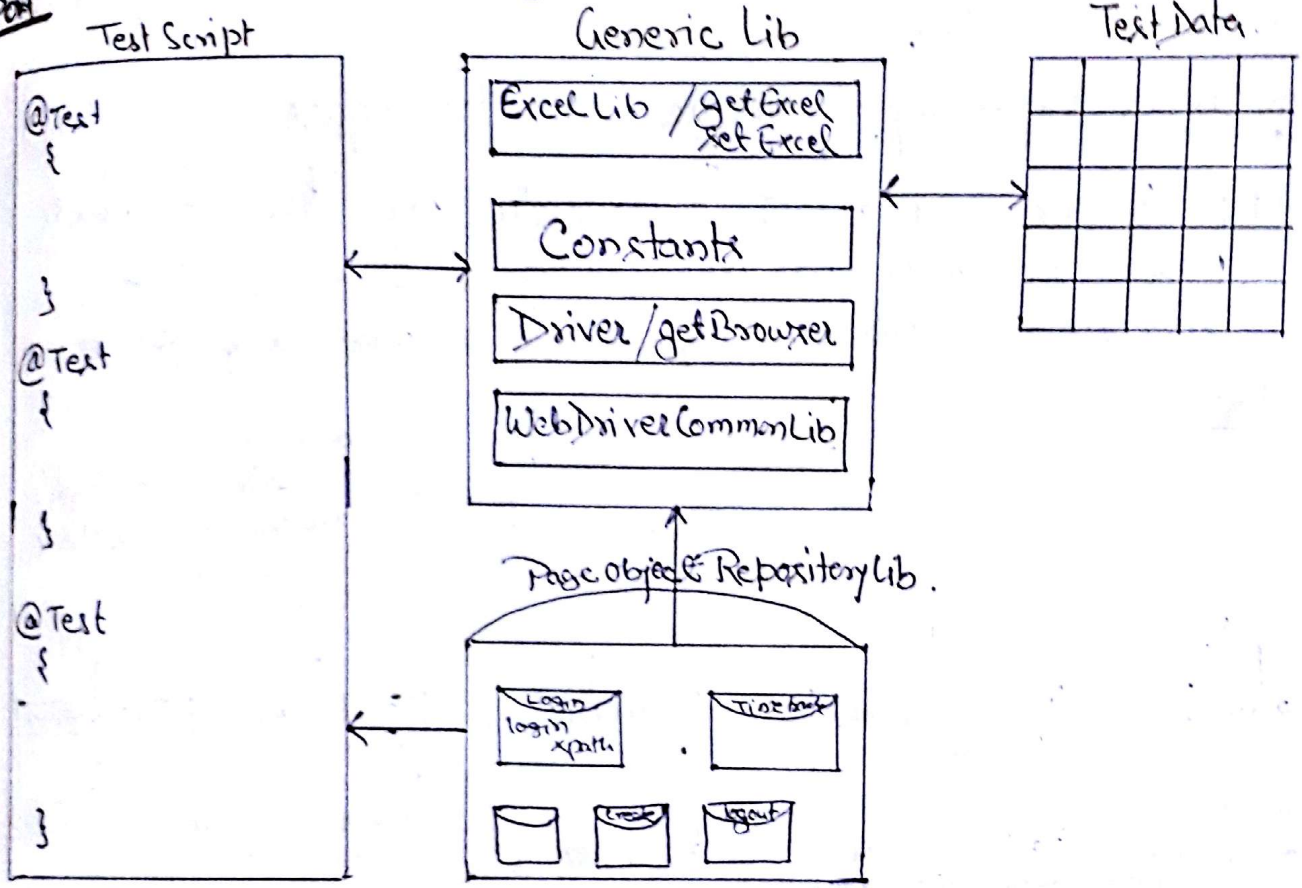
```
// Step2: Navigate to User Page
driver.findElement(By.xpath("//div[text()="
        "Users"]")).click();
wLib.waitForPageToLoad();
// Step3: Create New User
driver.findElement(By.xpath("//span[text()='Create New User']")).
                                                    click();
wLib.findElement
wLib.waitForNamePresent("username");
driver.findElement(By.name("username")).sendkeys(userName);
driver.findElement(By.name("firstName")).sendkeys(firstName);
driver.findElement(By.name("lastName")).sendkeys(lastName);
driver.findElement(By.name("email")).sendkeys(emailID);
driver.findElement(By.name("passwordText")).sendkeys(password);
driver.findElement(By.name("passwordTextRetype")).sendkeys
                                                    (password);
driver.findElement(By.xpath("//input[contains(@value,'createUser')]
                                                    )).click();
// Step4: Verify user created successfully msg.
WebElement textwb = driver.findElement(By.xpath("//span
                    [@class='successmsg']"));
boolean status = wLib.verifyText(textwb, expectedMsg);
Assert.assertTrue(status, "msg is not verified == Fail");
driver.quit();
// write data back to Excel
eLib.setExcelData("Sheet1", 1, 8, "PASS");
}
```

D/P

Test-Id	Test-Name	arg0	arg1	arg2	arg3	arg4	arg5	arg6
TC-01	UserTest	User-RAM	password	RAM	kaka	Ram@gmail.com	User account has been successfully created	PASS

→ If the Xpath is hard coded and if UI changes may be xpath may change then we need to change in every test case. To overcome this problem we are using POM Concept (Page Object Model).

POM



Generic Lib :-
Generic Lib doesn't contains xpath.
framework developer design it.
Common Library to all project.

Page Object Repository :-
It contains Xpath & Business methods.
Test Engg may also be involved to design this.
It is specific to the project.

Note:- As per Automation rule Xpath should not be hardcoded within a test script, instead we should get the xpath from external resource, External resources might be .xls, .properties, .java.

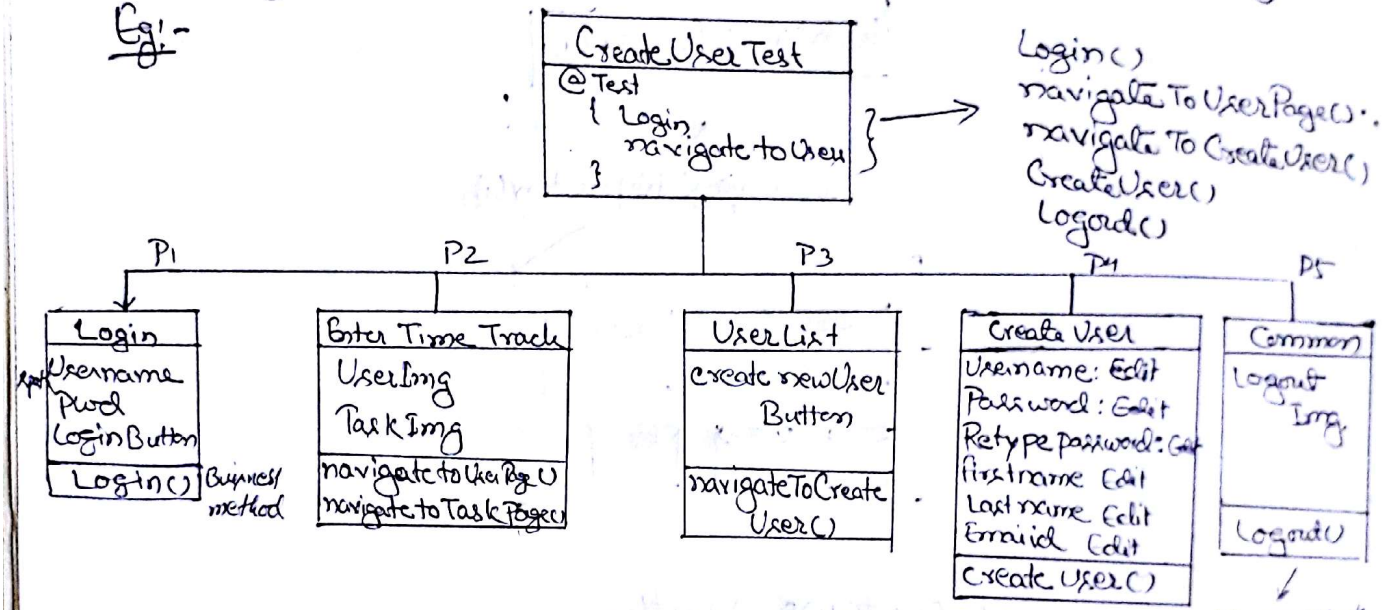
* Page Object Model (or) Page Factory is a java designed pattern prefered by Google to implement object repository.

* Page Object Repository is a collection of webElement xpath & Buisness Lib.

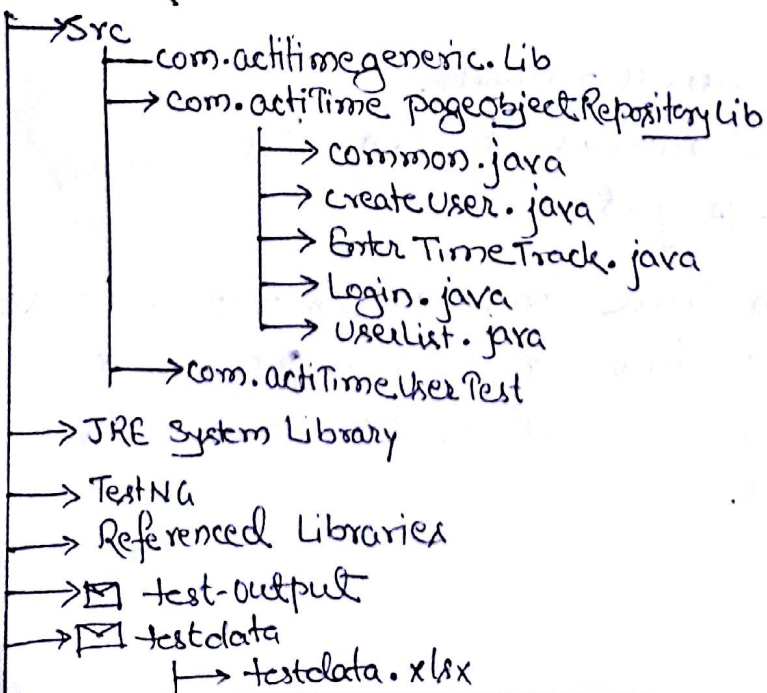
Page Object Repository technique / POM techniques

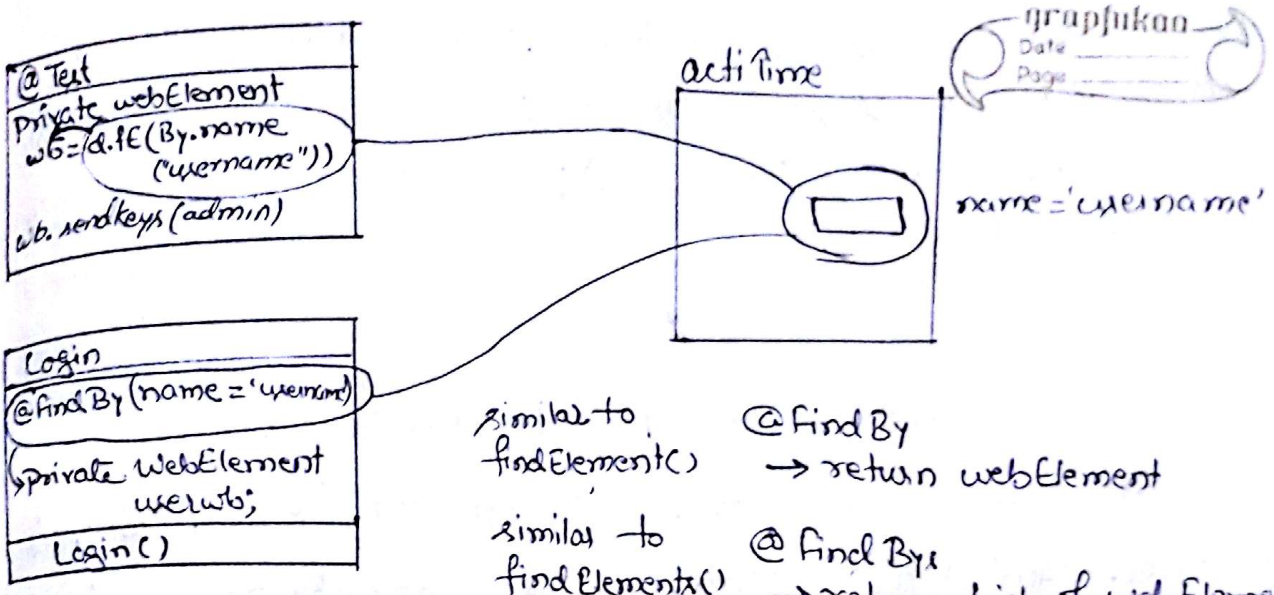
Rule1 :- As per POM create a dedicated java file for each Page available in Application and store page specific webElement (xpath) in a particular page.

Eg:-



Rule2 :- As per POM objec identification can be done by @FindBy & @FindBys





for Page Object Model

Login LoginPage = new Login() ✗

since driver control is in browser we have to pass to page factory.

Login LoginPage = PageFactory.initElements(driver, page.class);
→ e.g. login.class

Rule 3:- In order to create the object to java classes available in page object repository, we should take help of page factory class.

Program:-

PageObjectRepositoryLib

```
public class Login
{
 @findBy(name = "username")
 Private WebElement UserNameEdit;

 @findBy(name = "pwd")
 Private WebElement passwordEdit;

 @findBy(id = "LoginButton")
 Private WebElement LoginBtn;
Public void loginToApp(String username, String password,
                       string url)
 {
```

```
Driver.driver.get(url);
Driver.driver.manage().window().maximize();
UserNameEdit.sendkeys(username);
PasswordEdit.sendkeys(password);
loginBtn.click();
}
}
```

* How to Create object to page object repository classes.

```
Login loginPage = PageFactory.initElements.(driver, Login.class);
loginpage.LoginToApp(constants.UserId, constants.password,
                     constants.url);
```

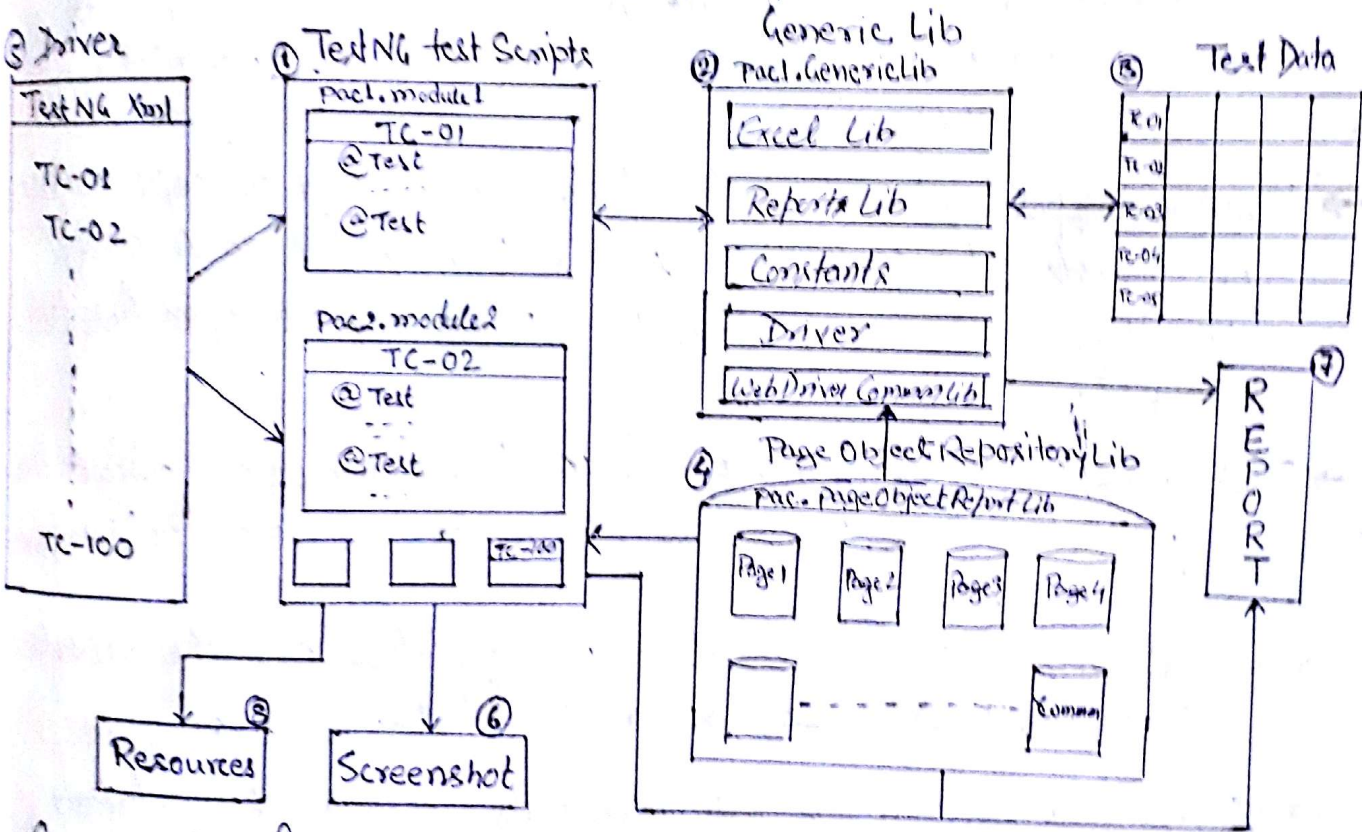
* User Test Script

```
Public class UserTest
{
ExcelLib elib = new ExcelLib();
WebDriver CommonLib wLib = new WebDriver CommonLib();
@Test
Public void createUserTest()
{
// getbrowser object.
WebDriver driver = Driver.getBrowser();
// Login to App.
Login LoginPage = PageFactory.initElements(driver, Loginclass);
LoginPage.LoginToApp(Constants.UserId, Constants.password,
                     Constants.url);
}
}
```

Framework Definitions

* Well organized structure of package, one driver script will take care of entire batch execution without any manual interaction.
* Automation framework is a process followed by different companies based on their requirement to execute the test scripts successfully without any manual interaction.
* Framework is a custom tool developed by framework developer which provides lots of custom reusable methods that makes automation test engineer life easy.

FRAMEWORK ARCHITECTURE



Components of framework are Driver, testNG test scripts, Generic Lib, Page Object Repository Lib, Test Data, Reports, Screenshot & Resources folders.

→ Driver is the brain of the framework which is used to achieve batch execution.

→ TestNG scripts is used to write test scripts according to the module using testNG annotations.

→ Generic Lib consists of all the re-usable methods which is used for all the projects. it contains

Excel Lib → which is used to fetch/retrieve the data from excel using Apache POI.

ReportLib → which is used to take screenshot when test case fails.

Constants → consist of all global parameter like url, browser, username, password.
if we want to test script to execute in different browser then it can be changed here.

Driver → used to launch the browser based on the requirements.

WebDriverCommonLib → consist of all reusable statements in test scripts like waitforPageToLoad, waitforXpath Present.

→ PageObjectRepositoryLib - is developed using POM concept where it consists of xpath & business Library for each page in the application, page object Repository is specific to project.

→ Test Data - All the test data used in the application must be stored in excel. Each row represents the data for each test case.

→ Report - Once the test execution is completed all the reports generated will be stored in this folder.

→ Screenshot - Whenever test case fails, screenshot is taken & stored in this folder.

→ Resources - This folder is used to store all the jars & executable file.

Reporting

Whenever we run test suite with 100's of test scripts, reporting feature might be useful to analyze test script failures.

There are two reasons test scripts might fails,

→ Test script fails because of product issue.

→ Test script fails because of test script issue.

Product issue → raise a defect.

If test script fails because of product issue, login to defect management tool & raise a defect

If test script fails because of test script issue, modify in test scripts

* Reporting are of 2 types :-

① High level reporting.

② Low level reporting.

High level reporting
- html report (TestNG)
- reportNG (jar)
  It is an additional plugin added to TestNG used to generate report in piechart form.

test	Pass	fails	skip	time
20	17	2	1	--

Low level reporting
- → sop x (don't hard code in test script)
- → Reportee [TestNG]
- → Log4J

In real time, we won't use sop statement, instead use reportee class
↓
use to generate low level report.

→ Eg.
```
@Test
Public void Sample-test
{
Reportee.log ("login to app");
Reportee.log ("Create user");
Reportee.log ("Verify");
Reportee.log ("logout");
}
```

When Executed it will not generate in console but in html report when clicked on test case name, low level report will be generated.

If we pass one more argument true then it will generate low level report in html report & also Eclipse console.

Use:-

```
Public class SampleReport
{
@Test
Public void SampleTest()
{
Reportee.log ("login to App", true);
Reportee.log ("Create user", true);
Reportee.log ("Verify", true);
Reportee.log ("logout", true);
}
}
```

Reportee class is an additional feature present in TestNG used to generate low level report along with high level report in html report.

Log4J generate low level report along with date & time..

High level reporting → Customer point of view.

Low level reporting → Test Engg point of view.

Interview Questions

1) Which framework you have used for the project?
→ Hybrid driven framework

2) Explain framework architecture?

3) what is repository?
→ Collection of WebElement xpath in a single place is called repository.

4) What is POM?
→ It is a java designed pattern, we have used to implement object repository.

5) Difference between POM & page factory?
→ POM is a collection of page class which contains xpath & Business library.

→ Page factory is a class available in WebDriver which is used to create an object to page classes to excess WebElements.

6) Advantage of Page Object Repository?
→ Whenever UI is getting changed / requirement changes modification of xpath in repository is very easy.

→ Maintainance will be very easy.

→ Whenever UI getting changed for eg. login page got modified, we know which class to go & directly modify the xpath.

→ In page factory classes we can directly store WebElements, using @FindBy & @FindBys annotations.

7) What kind of reporting feature you have used in Automation.
→ There are two types of reporting.

High level report :- To generate high level report we have used inbuilt emailable report available in testNG.

Low level report :- To generate low level report we have used inbuilt reporter class available in TestNG.

Reporter class is used to insert low level report in html report.

8) What is Generic Library & Advantage of Generic Library.

→ Generic Library contains re-usable methods that can be used to any projects.

Advantage

* Can get data from Excel
* Can get browser instance based on requirements.
* Inbuilt wait statements are available.
* Global parameter are available.

9) What is Business Library?

→ Business Library contains reusable methods that can be used specific to the project.

→ Business Library is a building block of test cases.

Screen Shots

EventFiringWebDriver

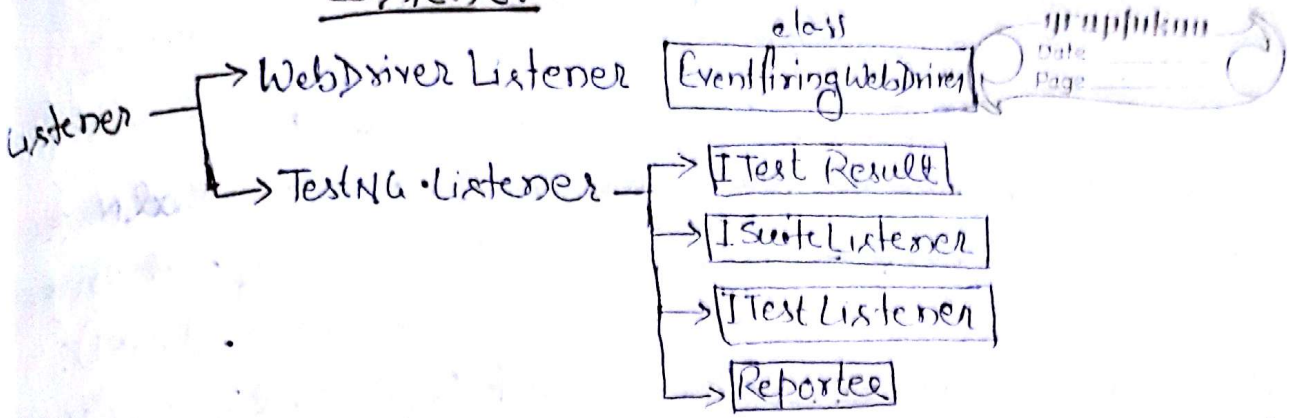
↳ getScreenshots (OutputType.File)

File srcFile = getScreenshotAs (OutputType.File)

Eg:-

```
Public class SampleTest {
  @Test
   Public void samplescreenTest()
   WebDriver d = new FirefoxDriver();
    d.get ("http://application);
  // Create Object to EventFiring web to get Screenshot.
  EventFiring WebDriver eDriver = new EventFiring webDriver
                                      (d);
 // Take Screenshot get screenshot, which return File object.
  File sfile = edriver.getScreenshotAs(OutputType.FILE);
 // Use Apache POI Lib, store Screenshot in Framework.
   File fdest = new file ("C://advanced//ActiTimeSelenium
                          framework//Screenshot//test.png");
  file.Utils.copyfile (sfile, fdest);
  }
}
```

Listener



* Listener is a feature to monitor entire test execution events in the run time.

Eg:- Whenever we run test suite with 100 of test scripts to monitor the entire test execution we go for listener annotation, whenever test fails, listener annotation can raise a request to listener class to take a screenshot in the run time.

* There are two types of listener
  1. WebDriver listener
  2. TestNG Listener.

WebDriver listener:-
Event firing WebDriver class is a WebDriver class which is used to capture the screenshot in runtime.

TestNG listener:-
There are multiple listener class available in TestNG listener

1. ITest Result :- It is used to capture test execution status or test name in run time.
2. ISuite Listener :-
3. Reporter listener :- It is used to generate logs in run time.
4. ITest Listener :- It is an inbuilt interface which is used to webElement custom (or) user-defined listener class.

In order to implement listener feature in our framework we should create custom listener class & implements ISuite Listener (or) ITest Listener.

Public class SampleListeners implements ITestListener {

@Override

Public void onTestFailure (ITestResult result) {

String testName = result.getMethod().getMethodName();

EventFiring WebDriver edriver = new EventFiring WebDriver
(Driver.driver);

File sfile = edriver.getScreenshotAs (OutputType.FILE);

file fdest = new file ("C:\\advanced\\ActiTime_Selenium
Framework\\Screenshot\\" + testName + ".png");

try {

FileUtils.copyFile(sfile, fdest);

} Catch (IOException e) { }

}

@ Override

Public void

unimplemented methods

* @Listeners (com.actiTime.genericLib.SampleListeners.class)

Public class SampleTest

@Test

Public void createUserTest() {

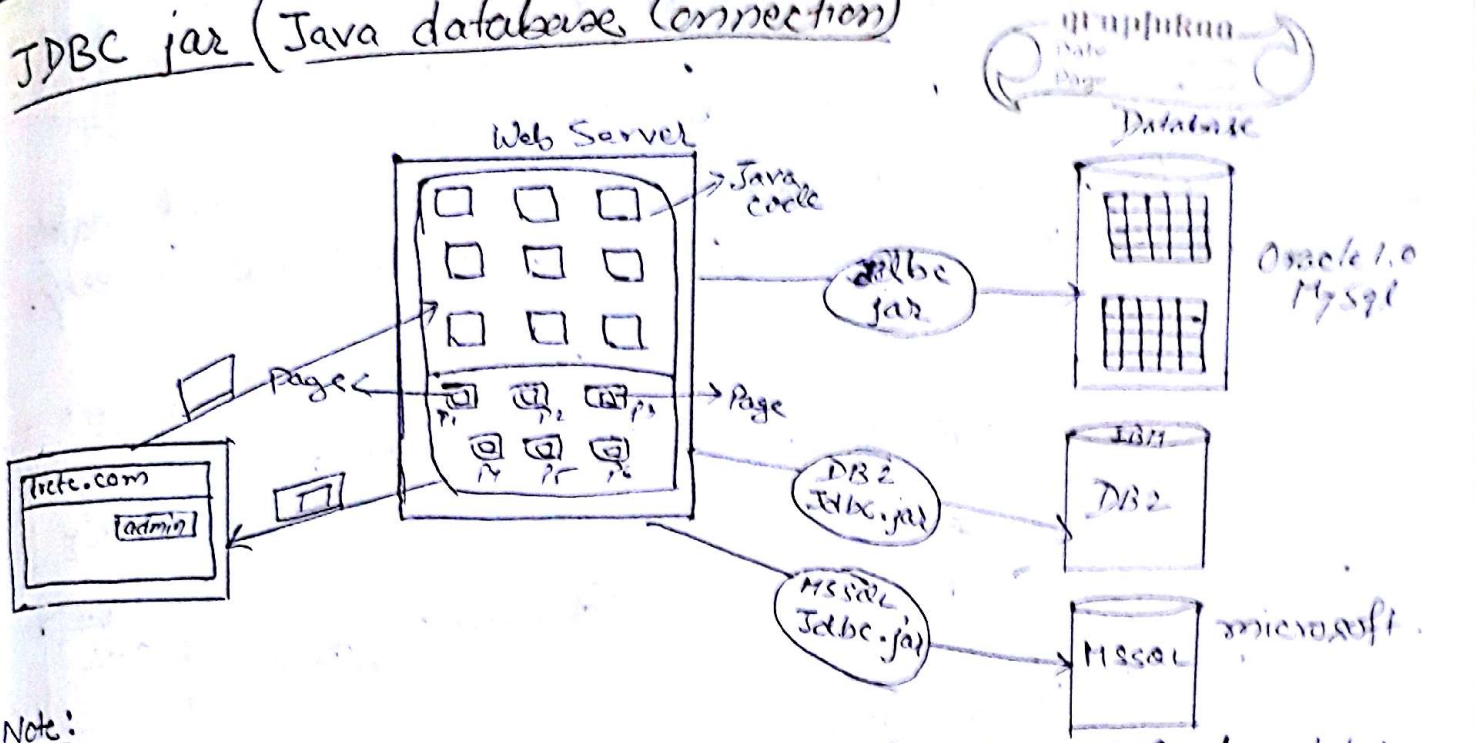
WebDriver d = Driver.getBrowser();

d.get ("http://adv-sel-14/login.do");

Assert.assertEquals ("A", "A");

d.quit();

JDBC jar (Java database Connection)



Note:

* Jdbc.jar is an open source tool implemented by database vendors (oracle, IBM, etc) which is used to establish a connection between Java source code to any database.

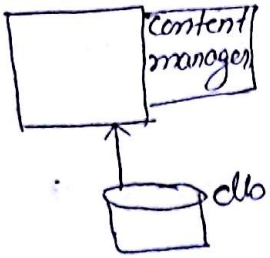
Why Jdbc connection in Automation?

Case1

Pre-Condition :- 200 links should be uploaded [Execute db query to push 200 lines to News table].

1. Navigate to bbc.com
2. Capture all 200 links.

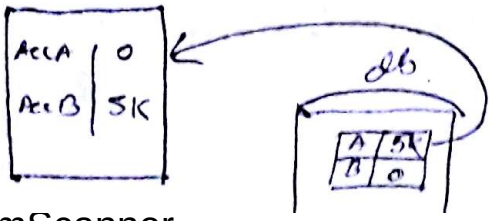
Expected Result :- All the 200 links should be captured.



Case2

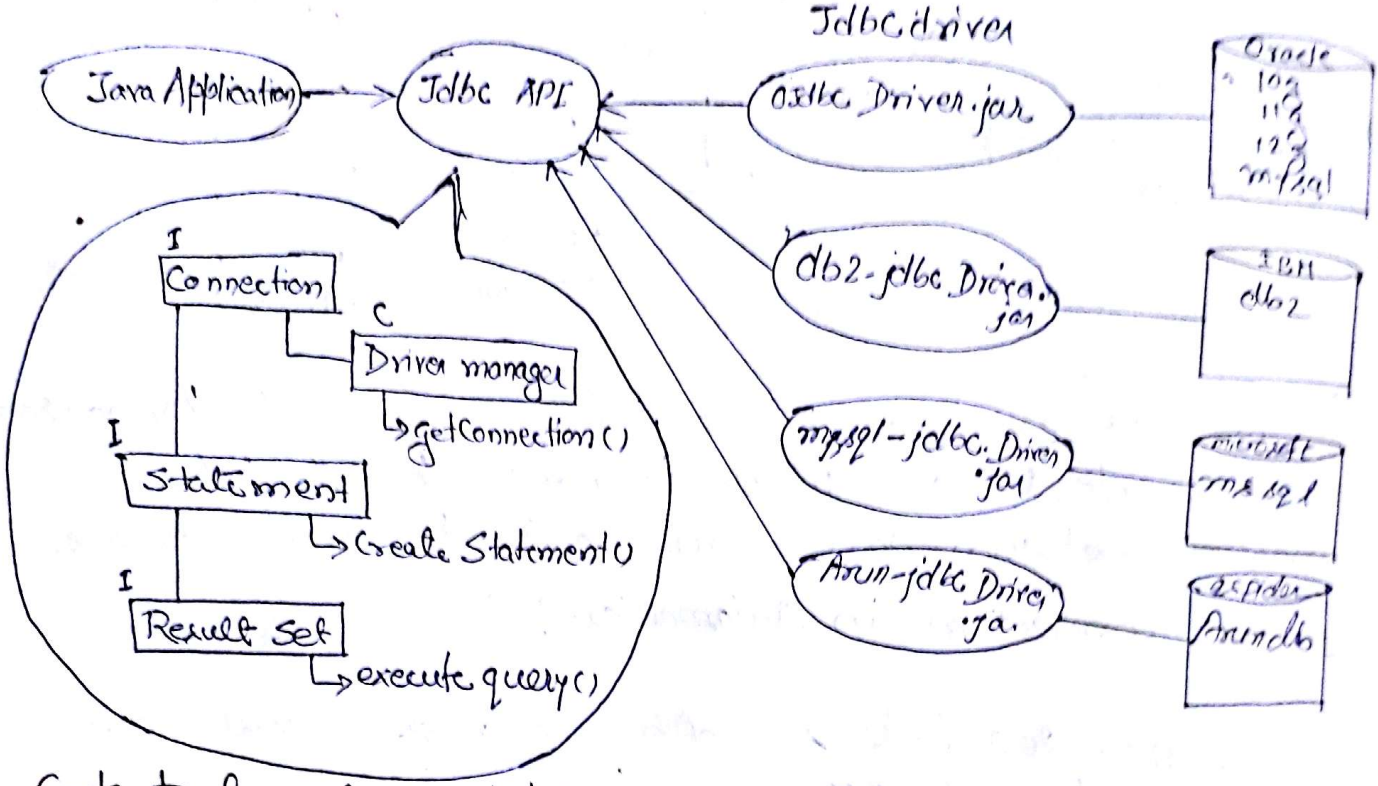
Pre condition :- Account should be have more than 5K.

1. Navigate to Bank app
2. Send 5K amount from Account 'A' to Account 'B'.



In Case1, 200 links must be uploaded by Automation Engg to test the application, so Jdbc.jar is used by them to execute db query.

In case2, Mostly in banking app if UI msg is not matching as in db then to check, Jdbc will be used to verify.



Code to Connect to database

* SampleDbConnection.java

```
Public class SampleDbConnection {
@Test
  Public void SampleTest()
// Register the Oracle db using driver class
   Class.forName("Oracle.jdbc.driver.OracleDriver");
// Create a ref-variable to connection interface and establish a
// connection to DB.
   Connection con = DriverManager.getConnection("jdbc:oracle:
       thin:@127.0.0.1:1521:XE", "system", "Admin123");
// Create a statement through connection reference
   Statement stmt = con.createStatement();
```

//Execute any complex sql query using execute query mtd.

```
Resultset rs = stmt.executeQuery("select * from SYSFILES");
while(rs.next()){
    String tableName = rs.getString(1);
    String tsName = rs.getString(2);
    long blockName = rs.getLong(3);
    System.out.println(tableName + "\t" + tsName +
                       "\t" + blockName);
}
stmt.close();
rs.close();
}
```

CSS Selector :-

→ It is one of the locater used to identify webElement in UI.

```
<div>
  <input id = "next" class = 'rc-button' value = 'login'>
</div>
```

Xpath	CSS-Selector
By.id("next")	By.css-selector(#next)
By.className("rc-button")	By.css-selector(.rc-button)
By.xpath(//input[@value'login'])	By.css-selector(input[value=login])
By.xpath(//div/input)	By.css-selector(div>input

→ css-selector is faster than Xpath because css always identify the object based on id and class.

→ css-selector does not have flexibility to navigate from child to parent, so we cannot the dynamic object using css-selector.

Advantage of Framework

(1) Maintanance is easy.
(2) Reusability
(3) Repeatability
(4) Time Saving
(5) Get better quality of product.
~~Ex: In order~~
(6) Less Coding effort
Ex:- In order to rerun the test case change the data in xls instead of test script.
Ex:- whenever UI is getting changed we can change the Xpath in repository instead of test script.
(7) Cross browser testing
(8) Cross platform testing
(9) Require less resource

Disadvantage

(1) Automation requires coding skill.
(2) selenium doesn't support stand alone application.
(3) Selenium doesn't support image comparision and captcha
(4) Using automation script we cann't audio video related test cases.

Different Types of framework

1. Data driven framework
2. Keyword driven framework
3. Modular driven framework
4. Hybrid framework.

Data driven framework:

→ whenever application deals with huge amount of data we prefer data driven framework.
Eg:- Banking application uses this type of framework as banking app always need to test with different set of data

→ Getting the data from the external resource & run the same test case with different set of data is called data driven testing also called as parametrisation.

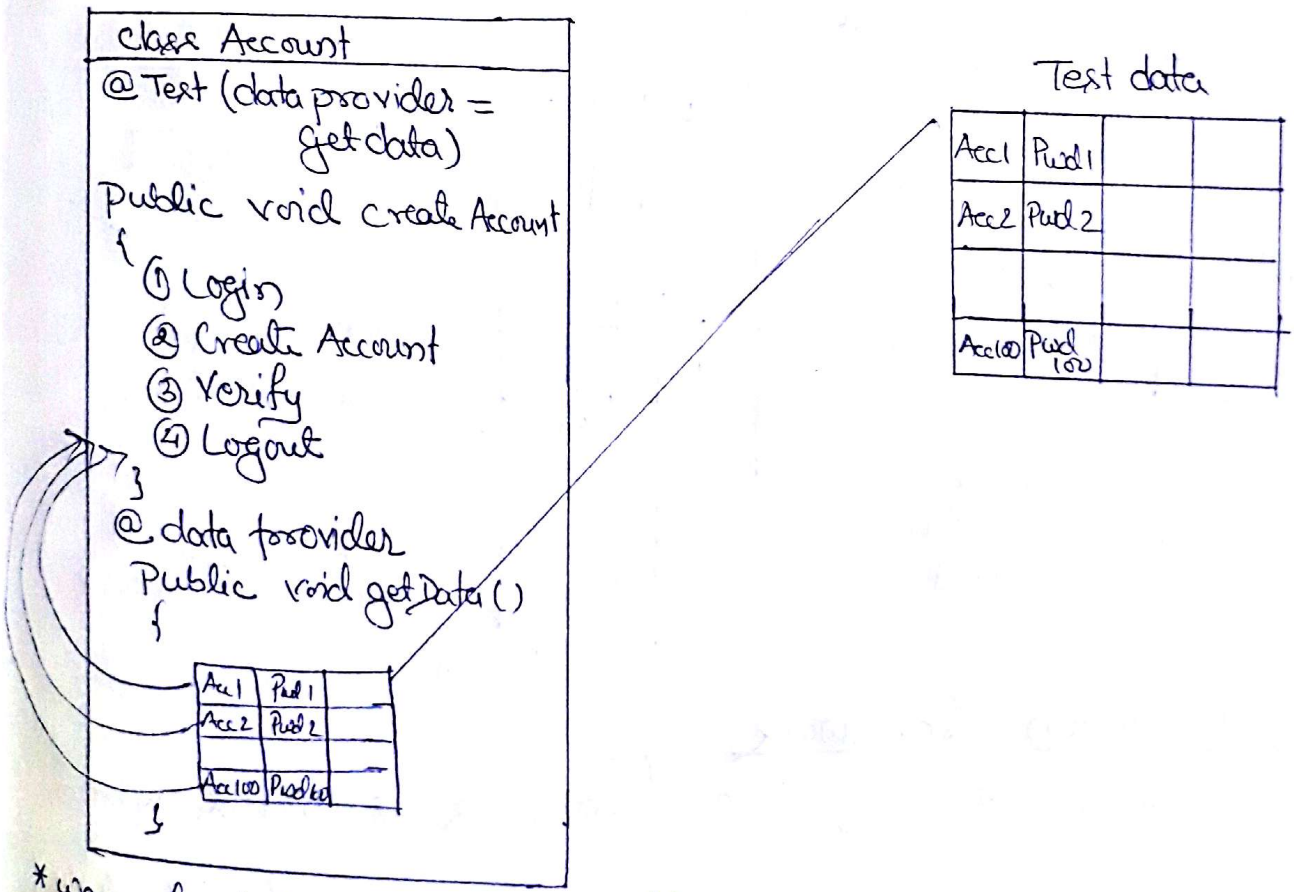
→ In order to achieve parametrization, we should use @dataprovider annotation method for each test in a class.

→ Data provider annotation will be used to rerun the same test case with different set of arguments.

Eg:- Create Account :-

① Login to banking app
② Create 100 Account
③ Verify account
④ Logout

test Script



* way of storing data is different
* One sheet is dedicated for one test case
* Each & every test should have dedicated data provided annotation.

Modular driven framework

(Big Application uses this type of framework like Gmail, CRM)

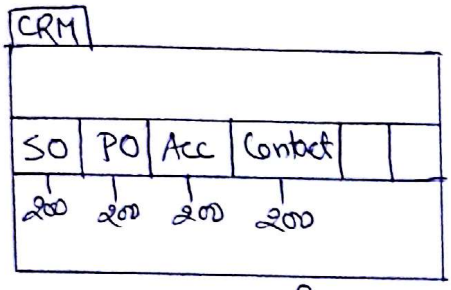
→ Maintaince is difficult.

→ According to modular driven framework each and every component should be maintained seperately.

Disadvantage :- Modular driven framework is very old framework but cannot be used in real time because maintance is difficult.

Advantage :- Components are organised properly.

Eg:-



Modular driven framework

Driver	Test-Script	test data	Page Object Rep.
so.xml	Pac.SO	So.xlsx	
PO.xml	Pac.PO	Po.xlsx	
Acc.xml	Pacc.Acc	Acc.xlsx	

Hybrid driven framework

* Combination of any two framework is called hybrid driven.

Eg:- Our ActiTime framework is combination of data driven & modular driven.

→ Arranging all test case in module wise so modular driven

→ Getting the data from Excel hence it is data driven.

Keyword driven framework

The best example of Keyword driven framework is Selenium IDE tool.

→ Whenever we need use friendly framework we go for Keyword driven.

→ Whenever Test engineer needs to write test script with less knowledge on automation tool and coding skill we prefer keyword driven.

→ As per this framework all the test script should be written in Excel.

→ All the Keywords (Method) like type, click, select is implemented by framework developer.

Driver	Test Scripts	Business Lib

Driver

```
driver
{Parm
}
```

TC01		
get	url	
type		
click		
select		
TC02		
get	url	
type		

Lib.

```
type(xpath,data)
{
}
click()
{
}
select()
{
}
```

SVN

[Sub Version - source control tool]

→ SVN is an open source tool or centralized repository where we can store / maintain entire selenium framework.

→ Technically whenever we developed a framework, framework should ~~not~~ not be available in local system, instead transfer the framework to SVN server machine.

* Uses of SVN Server

→ Entire framework will be available in one place.

→ Sharing files between Team members is easier.

→ SVN server capture history about modification and delete changes.

→ SVN provide merging and backup facility.

→ SVN provide remote access.

* There are two types of tools available in SVN Community.

(1) SVN Server
(2) SVN Client.

SVN Server

→ SVN Server is a s/w or tool it is install in server machine and it can hold entire selenium framework in centralized repository.

→ Go to google, search for download visual SVN Server
→ click on first link navigate to SVN community
→ Click on download link will get .msi file.
→ Double click and install.
→ In order to open SVN Server go to window start menue and search for visual SVN Server manager

SVN Client

→ SVN Client is a software or tool which is installed in all the users/client machine.

→ In order to communicate SVN Server, SVN Client should be installed in client machine.

Installation steps of SVN Client.

→ Go to google search for tortoise SVN.
→ Click on first link and download Tortoise SVN.
→ We will get .msi file.

* Configuration steps of SVN Server.

1) Open SVN Server
2) Create new Repository

Select Repository → Right click → create new repository → give name → next → select single project repository → next → select all subuser read and write → next → get Repository URL → finish.

* How to transfer framework from client to server machine

Select framework → Right click → Select Tortoise SVN → import

* How to get framework from server to client machine
→ Create a new folder
→ Place cursor inside the folder right click → SVN checkout.
→ Copy paste the URL and click on OK.

* How to open existing framework through Eclipse.
→ File → import → Expand General → Existing Project into workspace → click on Browse button → Select the framework folder downloaded from SVN.

* To Update in the Server – SVN Commit.
* To get the update – SVN update
* Show log to get the history.
→ Always batch execution can be done on server machine.

Maven

→ Maven is a build testing tool which is used to check the compilation issue in the entire framework or build

→ Maven is an open source build testing tool implemented as plugin for eclipse.

* Advantage / Uses of Maven.

→ We can get dependency latest jar from internet.
→ Execute selenium test script in command line.
→ Check compilation issue in entire framework.
→ Maven Provide framework folder structure.

* Source Control tool.
  - Github
  - VSS
  - CVS
  - Perforce
  - SVN

* Apache Maven

There are two types of plugin in maven
① Maven plugin for eclipse
② Maven plugin for Command line.

Maven Plugin for Eclipse

→ Maven Eclipse plugin is use to create maven project which provide framework folder structure.

Steps:-

In Eclipse Window File → New → click on others → Expand maven folder → click on maven project → Select both the checkbox → click on next → Give name of the project in GroupId and Artifact Id → click on Next.

maven project

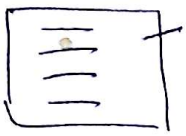
→ Src/main/java
  → Generic Lib → Driver
  → Page Object Repository
  } Libraries.

→ Src/main/resource
  → IE Driver.exe
  → Chrome Driver.exe

→ Src/test/Java
  → Project And Customer Test()
  → User Test()
  } → Test Script.

→ Src/test/Resource
  → test Data

→ Screenshot

→ Pom.xml  maven configuration file.

→ testNG.xml

Note:- Whenever we create maven project automatically we will get pom.xml file within the same project, insert dependency code in pom.xml file to get the latest jar from internet.

Note:- Before writing test script in maven project we should write dependency jar from web, inorder to get jar from web, we should install maven command line plugin.

* Maven Command line Plugin

→ In order to execute test script in command line without eclipse we require maven command line plugin