

# Selenium Handbook

A basic referential guide to Selenium

By

**Suresh**

## **Selenium- Course Outline**

### **Introduction to Automation**

- What is automation testing
- Advantages of Automation Testing
- How to learn any automation tool
- Types of Automation tools

### **Introduction to Selenium**

- What is Selenium
- Use of Selenium
- Features of selenium
- Difference between Selenium and QTP

### **Selenium Components**

- Selenium IDE
- Selenium Core
- Selenium RC
- Selenium Grid
- Selenium 2.0 - Web Driver

### **Selenium IDE**

- Selenium Overview
- Selenium IDE Introduction
- Downloading and Installing Selenium IDE
- Recording and Running a Simple Test
- Selenium IDE – Features
- Installing Useful Tools for Writing Tests
- Selenium Concepts
- Selenium Commands
- Verifying Page Elements – Assertions and Verifications
- Wait Commands
- Object Identification
- Element Locators
- Regular Expression patterns
- Selenium Test Runner
- Using Regular Expressions in Selenium IDE
- Using Java script functions in Selenium IDE
- Creating Selenium Test Suites
- How to run the recorded script against other browsers
- Why companies are not using recording tools
- Limitations of Selenium IDE

### **Selenium Core**

- Selenium Core Overview
- Installing Selenium Core
- Running Selenium Core Test Suites

## **Core Java Fundamentals**

### **Language Fundamentals**

- History of Java
- Features of java
- Java Programming Language Keywords
- Class and Object
- Data Types
- Array Declaration, Construction and Initialization
- Encapsulation
- Inheritance
- Polymorphism

### **Flow Control, Exceptions, and Assertions**

- Writing Code Using if and switch
- Statements
- Writing Code Using Loops
- Handling Exceptions
- Working with the Assertion Mechanism

### **Using the java.lang.String Class**

- Using the java.lang.Math Class
- Using Wrapper Classes
- Using the equals() Method with
- Strings and Wrappers and Objects

## **Inner Classes**

- Method-Local Inner Classes
- Anonymous Inner Classes
- Static Nested Classes

## **Defining, Instantiating, and Starting Threads**

- Preventing Thread Execution
- Synchronizing Code
- Thread Interaction

## **Object Orientation, Overloading and Overriding, Constructors**

- Benefits of Encapsulation
- Overridden and Overloaded Methods

## **About Eclipse**

- Installing Eclipse
- Creating Simple Project in eclipse
- Eclipse and Selenium together
- Importing and Exporting
- Debugging using Eclipse
- Exploring Eclipse – Basic
- Exploring Eclipse – Advanced

## **Fire Bug, Xpath and CSS**

- Introduction to Firebug
- Downloading and installing of Firebug
- Downloading and installing of xpath
- How to identify the xpath for an particular element
- Identifying objects using CSS

## **Selenium RC**

- Installing Selenium RC
- Selenium RC Overview
- Starting and Stopping Selenium Server
- Creating the generic scripts in selenium
- Creating the scripts by using functions
- Selenium Client Libraries
- Browser commands with examples
- Interactive commands with examples
- Information commands with examples
- Validation commands with examples
- How to take data from excel sheets
- Why should we use excel sheets
- How to take data from DB
- Debugging the scripts
- Maintaining the synchronization points
- How to handle Pop-up's and alert messages

## **How to use TestNG and Junit in Selenium**

- Introduction to TestNG
- Why TestNG
- Setting up TestNG
- Working with TestNG
- Advantages of TestNG over Junit
- Exploring TestNG Features
- How to Use TestNG Annotations
- Data Driven Testing TestNG
- TestNG Execution Report
- TestNG Results output folder walkthrough
- TestNG Reporting features

## **Sikuli Tool for handling windows**

- Installing Sikuli
- Sikuli Overview
- Why Sikuli?
- Sikuli Script Examples
- Compiling Sikuli scripts
- How To Use Sikuli scripts in Selenium WebDriver

## **Selenium Grid**

- Introduction Selenium Grid
- Advantages of Selenium Grid

## Advanced - Selenium 2.0 - WebDriver

- Introduction to selenium 2.0
- Advantages of web driver
- Web Driver v/s RC
- Architecture of Web Driver and RC
- Installation / Configuring Eclipse for Web Driver
- Identifying the elements in Web Driver Using Id, Name, Xpath ,Dom and CSS
- Working with Different drivers like HtmlUnit driver, Firefox Driver, Chrome Driver, Android Driver etc...
- Creating the generic scripts in Web Driver
- Creating the scripts by using functions
- Web Driver Client Libraries
- Web Driver commands with examples
- Working with excel sheets using Web Driver
- Handling Pop-up's and alert messages
- Working with Dropdown and page back commands
- Working with frames
- Web Driver with TestNG / Junit

## Android Mobile Automation Testing with Selenium WebDriver

- Introducing Android WebDriver
- Webdriver Bascis
- Installation of Android SDK
- Setup the Environment
- Setup the Emulator
- Setup the android Virtual Device
- Using the RemoteServer
- Install webdriver APK in Android Device
- Running the tests
- Running Gmail testcase in android device
- Web Driver commands with examples
- Generating TestResult reports using TestNG /Junit

## Automation Framework

- What is Framework
- Types of Frameworks
- What is modular framework
- What is Data Driven framework
- What is Keyword driven framework
- What is Hybrid framework
- Use of Framework
- How to develop the framework
- Integration of the framework
- How to execute the scripts from framework

### What is mean by Testing?

- The process of exercising software to verify that it satisfies specified requirements and to detect errors.
- The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs), and to evaluate the features of the software item.
- The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component.

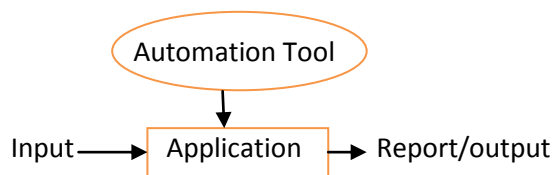
### Disadvantages of Manual Testing

- Manual tests can be very time consuming
- For every release you must rerun the same set of tests which can be time consuming
- Requires heavy investment
- Requires more number of human resources

## Automation

**Automation Testing:** Testing which is done by any other third party tool

**Test automation** is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions. Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.



### Advantages:

**Reliable:** Tests perform precisely the same operations each time they are run, thereby eliminating human error

**Repeatable:** You can test how the software reacts under repeated execution of the same operations.  
**Programmable:** You can program sophisticated tests that bring out hidden information from the application.

**Comprehensive:** You can build a suite of tests that covers every feature in your application.

**Reusable:** You can reuse tests on different versions of an application, even if the user interfaces changes.

**Better Quality Software:** Because you can run more tests in less time with fewer resources

**Fast:** Automated Tools run tests significantly faster than human users.

**Cost Reduction:** As the number of resources for regression test are reduced.

**Reporting:** Customized reporting of application defects

### Disadvantages of Automation Testing

- Proficiency is required to write the automation test scripts.
- Debugging the test script is major issue. If any error is present in the test script, sometimes it may lead to deadly consequences.
- Test maintenance is costly in case of playback methods. Even though a minor change occurs in the GUI, the test script has to be rerecorded or replaced by a new test script.
- Maintenance of test data files is difficult, if the test script tests more screens.

### Functional Testing Tools

Open Source	Commercial
Selenium-1.0	QTP
Web driver 2.0	Test Partner
Sahi	Test complete
Bad Boy	RFT
Ruby	Silk
Watir	

### Selenium vs. QTP

Selenium	QTP
Open source	Paid tool
Works on all OS (Windows, OS X, Linux, Solaris)	Works on Windows
Tests only Web applications	Tests web and desktop applications
Works on almost all browsers(IE, Firefox, Safari, Opera)	Works on Firefox 3.5.x and IE
Code can be made in any one of languages such as Java, C#, Ruby, Python, perl, php etc	VB Script
Html ID, Xpath, CSS, DOM, Link text	Object properties, Repository objects
There is no option, can record script in Selenium IDE, can spy objects using IE developer tool bar, Firebug and also using <a href="http://saucelabs.com/builder">http://saucelabs.com/builder</a>	GUI Spy
IDE sometimes does not record some events	Recording is a little reliable
Set of Libraries, around 20MB (Need to include other supporting software)	Around 1.5GB
Saucelabs.com, Element34 , Commercial Support	From HP

### History of selenium

- In 2004 invented by Jason Huggins and team
- Originally name is JavaScript Functional Tester (JSFT)
- Open source browser based integration framework built originally by Thought Works
- 100% JavaScript and HTML
- Web Testing Tool
- That supports testing web 2.0 applications
- Supports for cross –Browser testing (On Multiple browsers)
- Supports multiple operating systems

## Selenium Components

- Selenium IDE (Integrated Development Environment) - Record & Play back
- Selenium Core –Runs the test suits on the web server where the web application is deployed
- Selenium RC 1.0 – Server /Client (Selenium. Start /Selenium. Stop)
- Selenium Grid – Performance Tool
- Selenium 2.0 (Known as selenium Web Driver)

## Selenium IDE

### Introduction

The selenium IDE is the tool you use to develop your selenium test cases by using record and play back.

### Advantages

- It's an easy –to-use
- It's just Firefox plug in and is generally the most efficient way to develop test cases
- Very useful tool for beginners
- Firefox extension which allows record/Play testing paradigm
- Creates the simplest possible locator based on selenes
- Look at various possible commands in the dropdown
- Records a test at HTML file
- We can export the test as Java /Ruby etc...

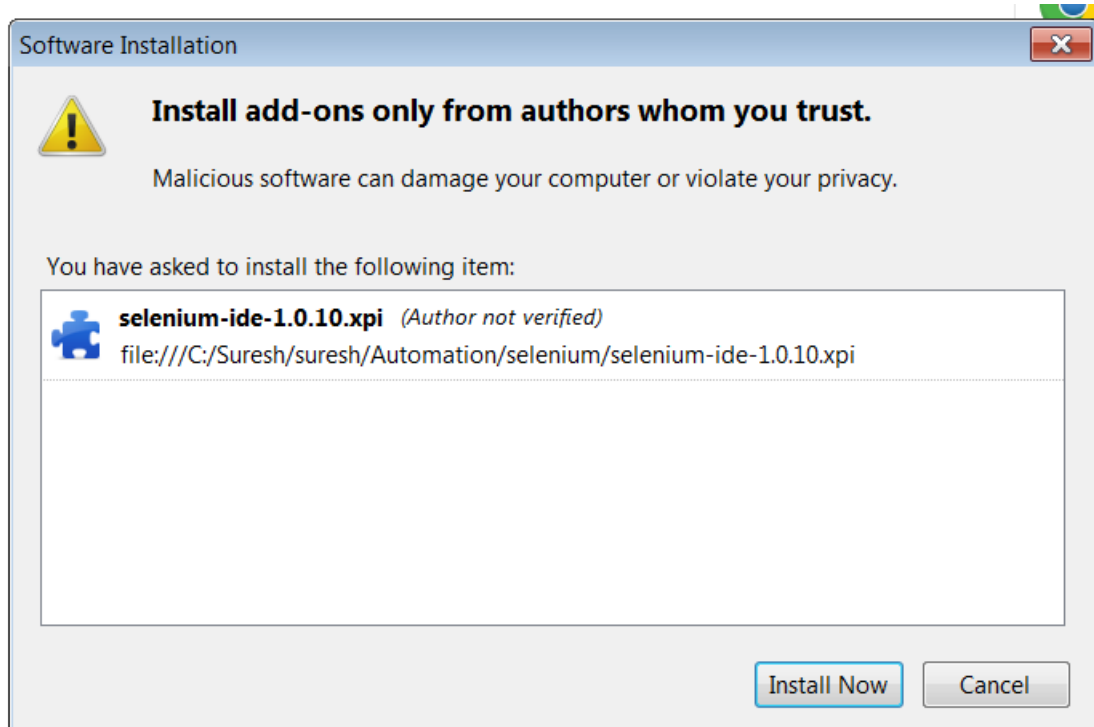
### Disadvantages

#### Selenium-IDE does not directly support:

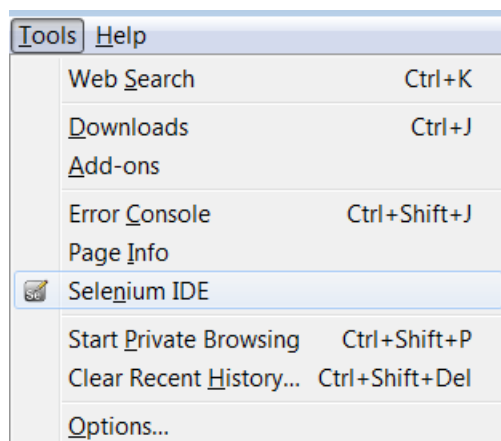
- Condition statements
- Iteration or looping
- Logging and reporting of test results
- Error handling, particularly unexpected errors
- Database testing
- Test case grouping
- Re-execution of failed tests
- Test case dependency
- Capture screenshots on test failures
- Results Report generations

## Installation

- Download Selenium IDE plug in from seleniumhq.org - <http://seleniumhq.org/download/>
- Open Firefox browser and do drag & drop

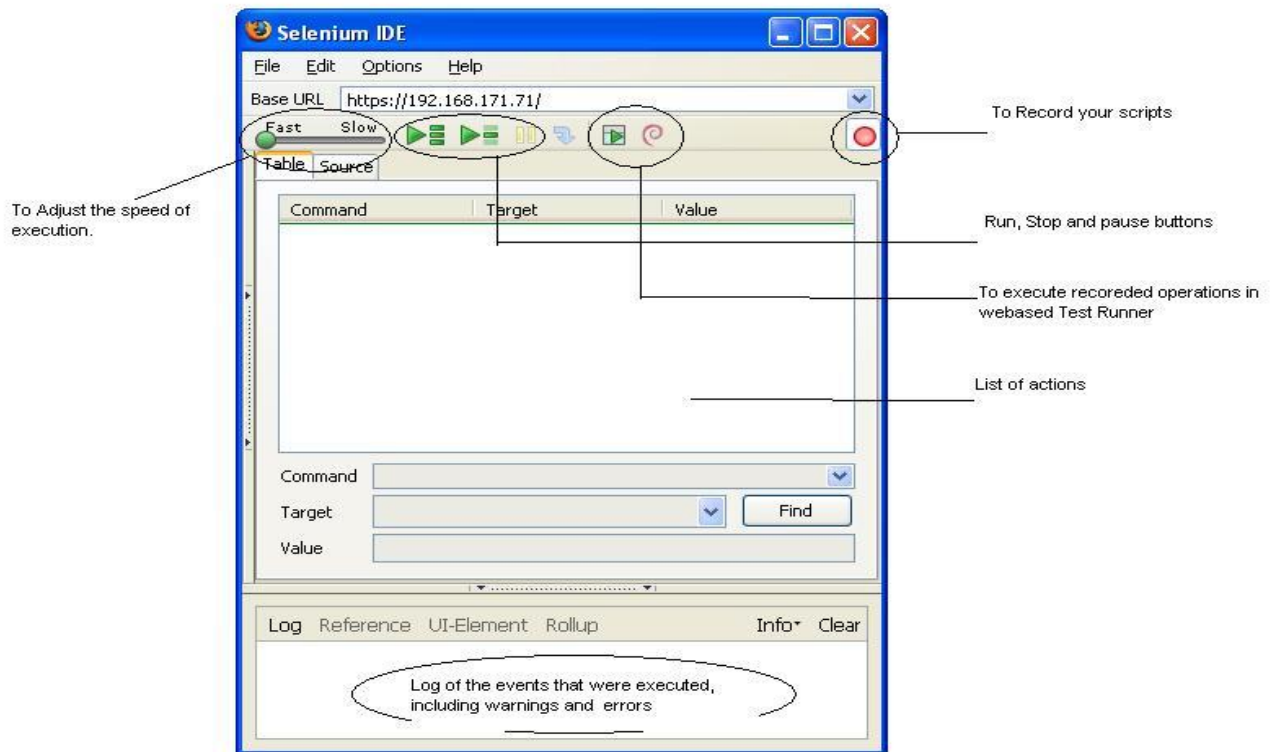


- Click on Install Now button
- Restart Firefox
- Navigate to -> tools option here we can find IDE





- Selenium IDE looks like below



## Recording

- Recording user actions with respect to user objects
- In selenium we have two types of recording options
  - i) Remember base URL
  - ii) Record absolute URL

## Selenium IDE Concepts

- Element Locators
- Actions
- Asserts
- Assessors
- Pattern matches /Regular expressions

## Element Locators

Selenium will identify the objects by using below locators

- By Using "ID "
- By Using "Name "
- By Using "Link "
- By Using "Xpath "
- By Using "Dom "
- By Using "CSS "

## Basic Selenium Actions

- Open - Opening web page
- Type - typing user name
- Click - Click on
- Select - Selecting particular item from drop down
- Check
- Uncheck

**Fire Bug –Plug in** (Used for locating the elements)

## Installation

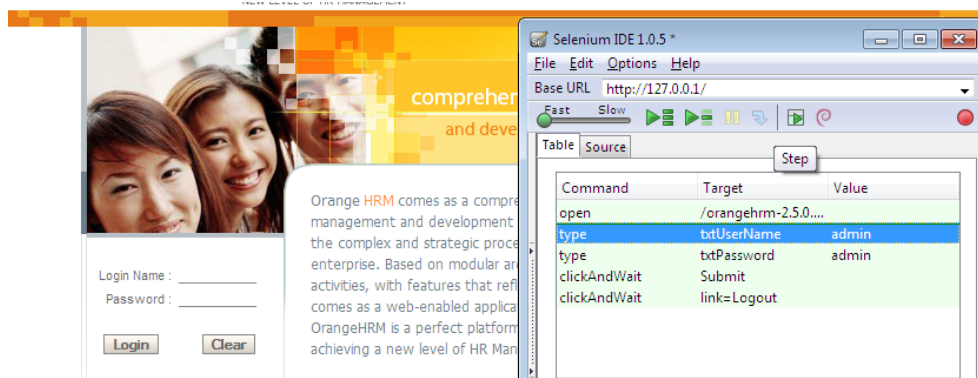
- Download firebug plug-in
- Open Firefox
- Do drag and drop
- Restart Firefox
- Then in the Firefox right downside we can find firebug icon
- Click on that icon below screen will be shown



## TC 1- Login

1. Open Web page
2. Type User name
3. Type Password
4. Click on Login button
5. Click on Logout button

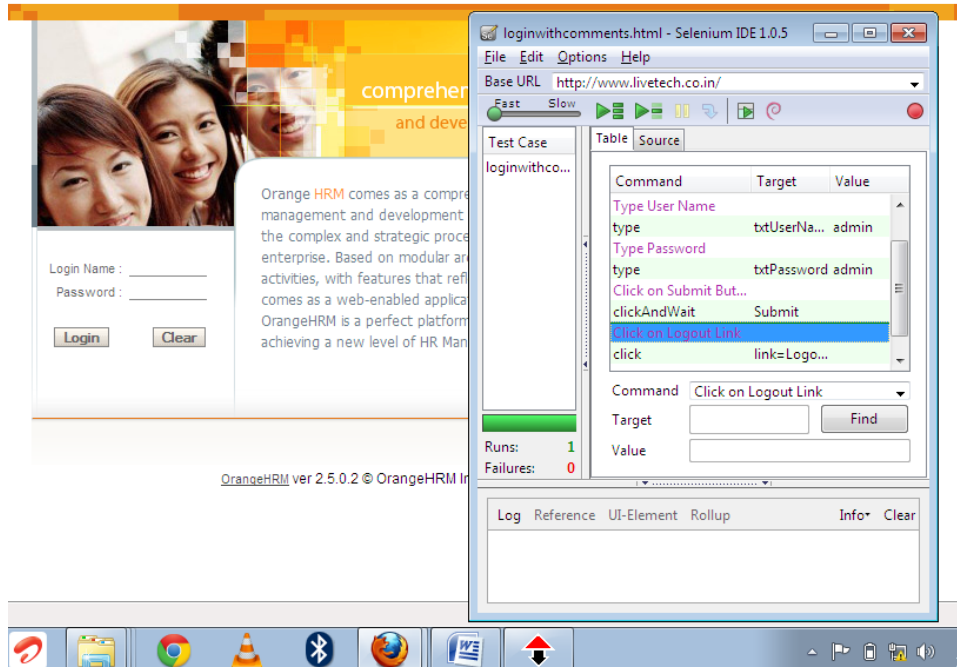
Record above steps in IDE and indentify the objects using name



## TC-2: Execute same test case by Inserting Comments in IDE

- For inserting comments in IDE do right click and insert Comments

Repeat TC1 by using comments for every step



## TC-Add Employee

**LiveHRMS**  
NEW LEVEL OF HR MANAGEMENT

The screenshot shows the LiveHRMS application interface. The top navigation bar includes links for ADMIN, PIM, LEAVE, TIME, BENEFITS, RECRUITMENT, PERFORMANCE, REPORTS, BUG TRACKER, and HELP. The user is logged in as 'rajesh'. The main content area displays the 'PIM : Add Employee' form, which includes fields for Code (0004), Last Name, First Name, Middle Name, Nick Name, and Photo. There are 'Save' and 'Reset' buttons at the bottom. A note states: 'Fields marked with an asterisk \* are required.' To the right, the Selenium IDE test case 'addempl...' is visible, showing a series of commands: open url, open, type user name, type password, type txtUserName, type txtPassword, click on submit button, clickAndWait, click on Add employ..., click, selectFrame, type last name, type first name, click on save, clickAndWait, selectFrame, click on logout link, and click.

### How Selenium is Identifying the Objects?:

Selenium is identifying the objects by using following Locator strategies.

1. Id = @Id
2. Name = @name
3. Identifier = @id
4. Dom = JavaScript expression
5. Xpath = Xpath expression
6. Link =Text pattern
7. css = css Selector Index

#### Locating by ID:

For instance, your page source could have id and name attributes as follows:

```
1 <html>
2 <body>
3 <form id="loginForm">
4 <input name="username" type="text" />
5 <input name="password" type="password" />
6 <input name="continue" type="submit" value="Login" />
7 </form>
8 </body>
9 </html>
```

The following locator strategies would return the elements from the HTML snippet above indicated by line number:

id=loginForm (3)

#### Locating by Name

- The name locator type will locate the first element with a matching name attribute.
- If multiple elements have the same value for a name attribute, then you can use filters to further refine your location strategy.
- The default filter type is value (matching the value attribute).

```

2 <html>
3 <body>
4   <form id="loginForm">
5     <input name="username" type="text" />
6     <input name="password" type="password" />
7     <input name="continue" type="submit" value="Login" />
8     <input name="continue" type="button" value="Clear" />
9   </form>
10 </body>
10 </html>

```

- name=username (5)
- name=continue value=Clear (8)
- name=continue Clear (8)
- name=continue type=button (8)

### Locating by Xpath

- Xpath is the language used for locating nodes in an XML document.
- We are using Xpath is when you don't have a suitable id or name attribute for the element you wish to locate.

```

2 <html>
3 <body>
4   <form id="loginForm">
5     <input name="username" type="text" />
6     <input name="password" type="password" />
7     <input name="continue" type="submit" value="Login" />
8     <input name="continue" type="button" value="Clear" />
9   </form>
10 </body>

```

❑ Xpath=/html/body/form[1] (4) - Absolute path (would break if the HTML was changed only slightly)

- //form[1] (3) - First form element in the HTML
- xpath=//form[@id='loginForm'] (4) - The form element with attribute named 'id' and the value 'loginForm'

❑ Xpath=//form[input/@name='username'] (5) - First form element with an input child element with attribute named 'name' and the value 'username'

❑ //input[@name='username'] (5) - First input element with attribute named 'name' and the value 'username'

❑ //form[@id='loginForm']/input[1] (5) - First input child element of the form element with attribute named 'id' and the value 'loginForm'

❑ //input[@name='continue'][@type='button'] (8) - Input with attribute named 'name' and the value 'continue' and attribute named 'type' and the value 'button'

❑ //form[@id='loginForm']/input[4] (8) - Fourth input child element of the form element with attribute named 'id' and value 'loginForm'

### Example for Xpath:

- Let us take the source of google.com to make our task simpler.
- The first three locator strategies state that, if any element having the name, id or identifier attribute
- Observe the source code of below. After we click on the Google Search button we move to the results page, now let us understand how the first link is identified from the below source code.

```
<div id="res" class="med">
```

```
<h2 class="hd">Search Results</h2>
```

```
<div>
```

```
<ol>
```

```
<di
```

```
v/>
```

```
<di
```

```
v/>
```

```
<li class="g
```

```
w0"> <h3
```

```
class="r">
```

```
<a class="l" onmousedown="return rwt(this,',','res','1','AFQjCNFp1UrvQWdtIVY_6kO-  
kEDVmdSnig','&sig2=HRDAzRP8Pz96kL3VNCyX1A')" href="http://www.testinggeek.com/in  
dex.php/testing-tools/test-execution/97-selenium-ide-introduction">
```

```
<em>Selenium  
IDE</em> - Introduction
```

```
</a>
```

```
</h3>
```

- As you see above the anchor tag does not have anything that can identify the element uniquely, hence we take the Xpath here.
- For that we need to see which tag of the upper hierarchy of the element has a unique attribute to identify it uniquely. We deduct the Xpath as

```
//div[@id='res']/div[1]/ol/li[1]/h3/a
```

The Xpath should always start from a hierarchy from where the element can be identified uniquely.

For example there is an element which has the attributes as follows

<input class="username" type=""> and say the class attribute is unique to this tag then the element locator would be //input[@class="username"]. This is also Xpath but it is at the same level.

### Locating Hyperlinks by Link Text

- This is a simple method of locating a hyperlink in your web page by using the text of the link. If two links with the same text are present, then the first match will be used.

```
<html>  
<body>  
<p>Are you sure you want to do this?</p>  
<a href="continue.html">Continue</a>  
<a href="cancel.html">Cancel</a>  
</body>  
</html>
```

link=Continue (4)

link=Cancel (5)

## Locating by DOM

- The Document Object Model represents an HTML document and can be accessed using JavaScript.
- This location strategy takes JavaScript that evaluates to an element on the page, which can be simply the element's location using the hierarchical dotted notation.
- Since only dom locators start with "document", it is not necessary to include the dom= label when specifying a DOM locator.

```
<html>
<body>
<form id="loginForm">
<input name="username" type="text" />
<input name="password" type="password" />
<input name="continue" type="submit" value="Login" />
<input name="continue" type="button" value="Clear" />
</form>
</body>
</html>
```

---

?	dom=document.getElementById('loginForm') (3)
?	dom=document.forms['loginForm'] (3)
?	dom=document.forms[0] (3)
?	document.forms[0].username (4)
?	document.forms[0].elements['username'] (4)
?	document.forms[0].elements[0] (4)
?	document.forms[0].elements[3] (7)

## Locating by CSS

- CSS (Cascading Style Sheets) is a language for describing the rendering of HTML and XML documents.
- CSS uses Selectors for binding style properties to elements in the document.

```
2 <html>
3 <body>
4 <form id="loginForm">
5   <input class="required" name="username" type="text" />
6   <input class="required passfield" name="password" type="password" />
7   <input name="continue" type="submit" value="Login" />
8   <input name="continue" type="button" value="Clear" />
9 </form>
10 </body>
10 </html>
```

- css=form#loginForm (4)
- css=input[name="username"] (5)
- css=input.required[type="text"] (5)
- css=input.passfield (6)
- css=#loginForm input[type="button"] (8)

### Commonly used selenium commands

Command	Description
open	tells the browser to open a specific page
verifyTextPresent	Verifies that text appears on the page
verifyValue	Verifies the value of an input field
Verify Title /Assert Title	Verifies expected test is somewhere on the page
Verify Element-presents	verify an expected UI element, as defined by its HTML tag is present on the page
Verify Text	Verifies expected text and its corresponding HTML tag are present on the page
Verify table	Verifies a table expected contents
Wait for page load	Pauses execution until an expected new page loads
Wait For element present	Pauses execution until an expected UI element, as defined by its HTML tag, is present on the page.
click	tell the test to click something on the page
clickAndWait	tell the test to click something on the page and wait for some time
select	select an option in a select box by providing the id of the select box and the option value
pause	Tell the test to pause for a while
storeAttribute	Selenium IDE can store variables and allows you to reference them in the same test case or another test case

### Some more commands / methods

assignId("Locator","String")	Temporarily sets the "id" attribute of the specified element
capture Screenshot ("File name")	Captures a PNG screenshot to the specified file.
Check("Locator")	Check a toggle-button(checkbox/radio)
click("Locator")	Clicks on a link, button, checkbox or radio button.
clickAt("Locator","Coordinate String")	Clicks on a link, button, checkbox or radio button.
close()	Simulates the user clicking the "close" button in the title bar of a popup window or tab.
doubleClick("Locator")	Double clicks on a link, button, checkbox or radio button.
doubleClickAt("Locator","Coordinate String")	Double clicks on a link, button, checkbox or radio button.
getAlert()	Retrieves the message of a JavaScript alert generated during the previous action, or fail if there were no alerts.
getAllButtons()	Returns the IDs of all buttons on the page.
getAllFields()	Returns the IDs of all input field son the page.
getAllLinks()	Returns the IDs of all links on the page.
getAllWindowIds()	Returns the IDs of all windows that the browser knows about.
getAllWindowNames()	Returns the names of all windows that the browser knows about.
getAllWindowTitles()	Returns the titles of all windows that the browser knows about.
getAttribute("Attribute Locator")	Gets the value of an element attribute.
getBodyText()	Gets the entire text of the page.



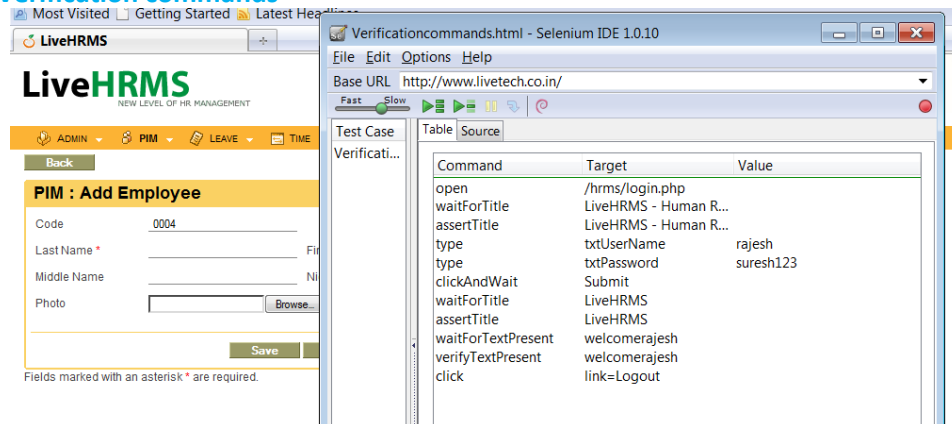
getConfirmation()	Retrieves the message of a JavaScript confirmation dialog generated during the previous action.
getCookie()	Return all cookies of the current page under test.
getElementHeight("Locator")	Retrieves the height of an element
getElementPositionLeft("Locator")	Retrieves the horizontal position of an element
getElementPositionTop("Locator")	Retrieves the vertical position of an element
getElementWidth("Locator")	Retrieves the width of an element
getEval("JS Expression")	Gets the result of evaluating the specified JavaScript snippet.
getLocation()	Gets the absolute URL of the current page.
getMouseSpeed()	Returns the number of pixels between "mouse move" events during dragAndDrop commands (default=10).
getPrompt()	Retrieves the message of a JavaScript question prompt dialog generated during the previous action.
getSelectedId("Select Locator")	Gets option element ID for selected option in the specified select element.
getSelectedIds("Select Locator")	Gets all option element IDs for selected options in the specified select or multi-select element.
getSelectedIndex("Select Locator")	Gets option index (option number, starting at 0) for selected option in the specified select element.
getSelectedIndexes("Select Locator")	Gets all option indexes (option number, starting at 0) for selected options in the specified select or multi-select element.
getSelectedLabel("Select Locator")	Gets option label (visible text) for selected option in the specified select element.
getSelectedLabels("Select Locator")	Gets all option labels (visible text) for selected options in the specified select or multi-select element.
getSelectedValue("Select Locator")	Gets option value (value attribute) for selected option in the specified select element.
getSelectedValues("Select Locator")	Gets all option values (value attributes) for selected options in the specified select or multi-select element.
getSelectOptions("Select Locator")	Gets all option labels in the specified select drop-down.
getSpeed()	Get execution speed (i.e., get the millisecond length of the delay following each selenium operation).
getTable("Table Cell Address")	Gets the text from a cell of a table.
getText("Locator")	Gets the text of an element.
getTitle()	Gets the title of the current page.
getValue("Locator")	Gets the (whitespace-trimmed) value of an input field (or anything else with a value parameter).
get Whether This Frame MatchFrameExpression("Current rame","Target")	Determine whether current/locator identify the frame containing this running code
get Whether This Window MatchWindowExpression("CurrentWindow", "Target")	Determine whether currentWindow String plus target identify the window containing this running code.
goBack()	Simulates the user clicking the "back" button on their browser.
highlight("Locator")	Briefly changes the backgroundColor of the specified element yellow.
isAlertPresent()	Has an alert occurred?

isChecked("Locator")	Gets whether a toggle-button (checkbox/radio) is checked.
isConfirmationPresent()	Has confirm() been called?
isEditable("Locator")	Determines whether the specified input element is editable, ie hasn't been disabled.
isElementPresent("Locator")	Verifies that the specified element is somewhere on the page.
isPromptPresent()	Has a prompt occurred?
isSomethingSelected("Locator")	Determines whether some option in a drop-down menu is selected.
isTextPresent("Pattern")	Verifies that the specified text pattern appears somewhere on the rendered page shown to the user.
isVisible("Locator")	Determines if the specified element is visible.
open("URL")	Opens an URL in the test frame.
openWindow("URL","WindowID")	Opens a popup window (if a window with that ID isn't already open).
refresh()	Simulates the user clicking the "Refresh" button on their browser.
removeAllSelections("Locator")	Unselects all of the selected options in a multi-select element.
removeSelection("Locator","Option Locator")	Remove a selection from the set of selected options in a multi-select element using an option locator.
select("Select Locator","Option Locator")	Select an option from a drop-down using an option locator.
selectFrame("Locator")	Selects a frame within the current window.
selectWindow("WindowID")	Selects a popup window; once a popup window has been selected, all commands go to that window.
setSpeed("Value")	Set execution speed (i.e., set the millisecond length of a delay which will follow each selenium operation).
setTimeout("Time")	Specifies the amount of time that Selenium will wait for actions to complete.
start()	Launches the browser with a new Selenium session
stop()	Ends the test session, killing the browser
submit("Form Locator")	Submit the specified form.
type("Locator","Value")	Sets the value of an input field, as though you typed it in.
unCheck("Locator")	Uncheck a toggle-button (checkbox/radio)
waitForCondition("JavaScript","Timeout")	Runs the specified JavaScript snippet repeatedly until it evaluates to "true".
waitForFrameToLoad("Frame Address","Timeout")	Waits for a new frame to load.
waitForPageToLoad("Timeout")	Waits for a new page to load.
waitForPopUp("WindowID","Timeout")	Waits for a popup window to appear and load up.
windowFocus()	Gives focus to the currently selected window
windowMaximize()	Resize currently selected window to take up the entire screen

## Asserts (Verification commands)

Waiting commands	Verify Commands(Continues execution after failure)	Assert commands(Stops execution after failure)
Wait For Title	Verify Title	Assert Title
Wait for text present	Verify text present	Assert text present
Wait for values	Verify value	Assert value
Wait for text	Verify text	Assert text
Wait for element present	Verify element present	Assert element present
Wait for table	Verify table	Assert table

## TC-Using Verification commands



## Useful Selenium Tools

- XPATHER
- FIREBUG
- XPATH CHECKER
- Firepath
- Web Developer
- IE Developer Toolbar
- HTML validator
- DOM Inspector

## Assessors (Storing commands)

- Store title
- Store text present- True/false
- store Text(stores from element / location)
- store value(stores from object)
- store element present-True/false
- store table

## Storing constant values in to variable

```
Store admin v1
Echo ${v1}
```

**Store Value:** This stmt generated from get value. It is going to get the value from object and it stores in the variable.

```
Store value txtusername v2
Echo ${v2}
```

**Store text:** This stmt is generated from locator. It will get the values from element and stores in the variable

**Store text present:** this stmt generated from isTextpresent.

**Store element present:** This stmt generated from isElementPreset. If element is presented it returns true else returns false.

**Select Frame:** selecting the frame with in the current window

## TC – Select frame & Verification commands & storing a constant in to variable

The screenshot shows the Selenium IDE interface with a test case titled "selectframe and storing variables.html". The test case is loaded with the following commands:

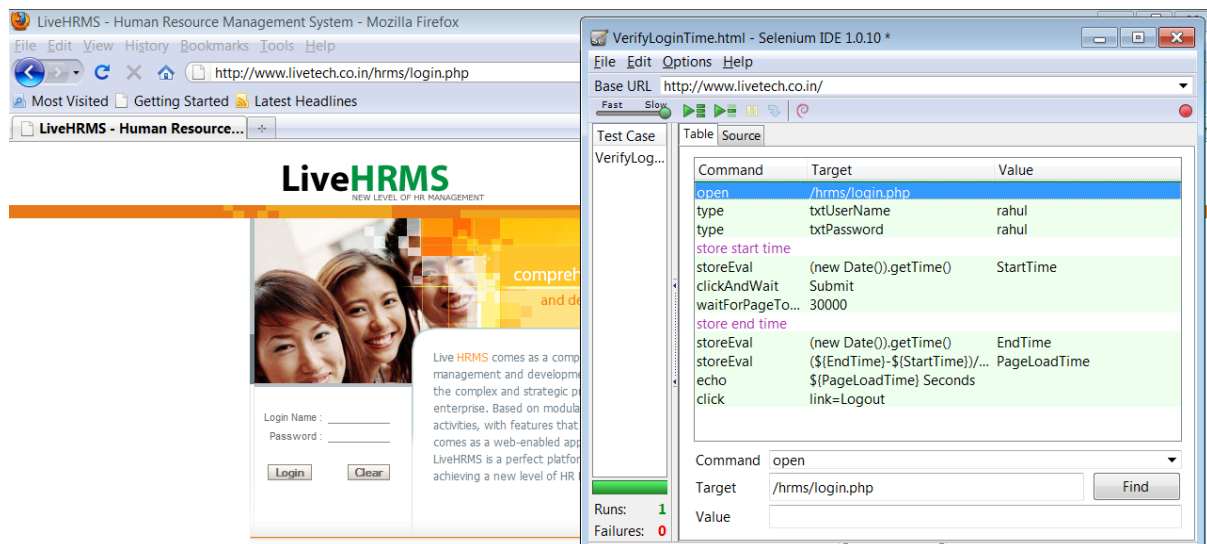
Command	Target	Value
open	/hrms/login.php	
waitForTitle	LiveHRMS - Human R...	
assertTitle	LiveHRMS - Human R...	
store	rajesh	username
store	suresh123	password
type	txtUserName	\${username}
type	txtPassword	\${password}
clickAndWait	Submit	
waitForTitle	LiveHRMS	
assertTitle	LiveHRMS	
waitForTextPresent	welcome\${username}	
verifyTextPresent	welcome\${username}	
click	//li[@id='pim']/ul/li[2]...	
waitForText	//form[@id='frmEmp'...] PIM : Add Employee	
verifyText	//form[@id='frmEmp'...] PIM : Add Employee	
selectFrame	rightMenu	
type	//input[@id='txtEmpF...' madhu	
type	txtEmpLastName	babu
clickAndWait	btnEdit	
selectFrame	relative=up	
click	link=Logout	

### TC-1: Verify login time

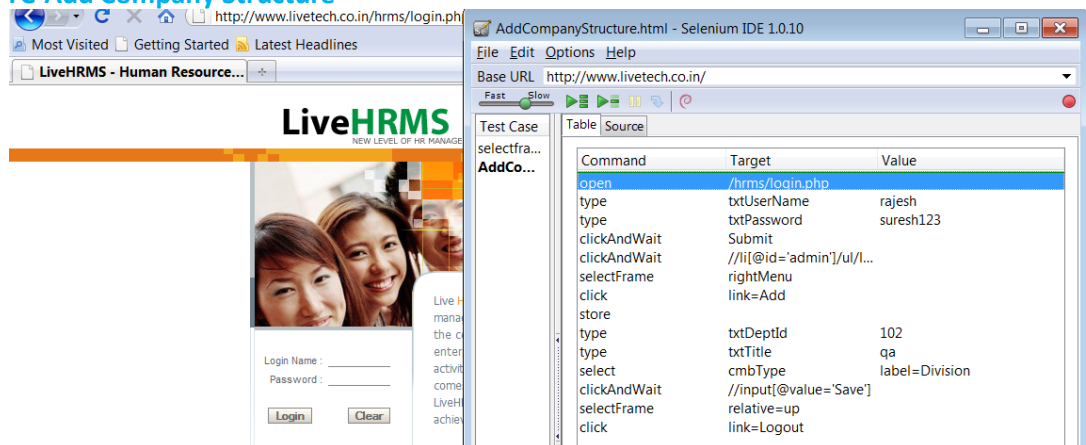
**storeEval:** This stmt generated from getEval. This command we are using to run off Java script snippet (Snippet means-Java script functions)

**eg:** storeEval new Date().getTime() v1  
echo \${v1}

**eg:2** store JavaScript{new Date().getTime()} v1  
echo \${v1}



### TC-Add Company Structure

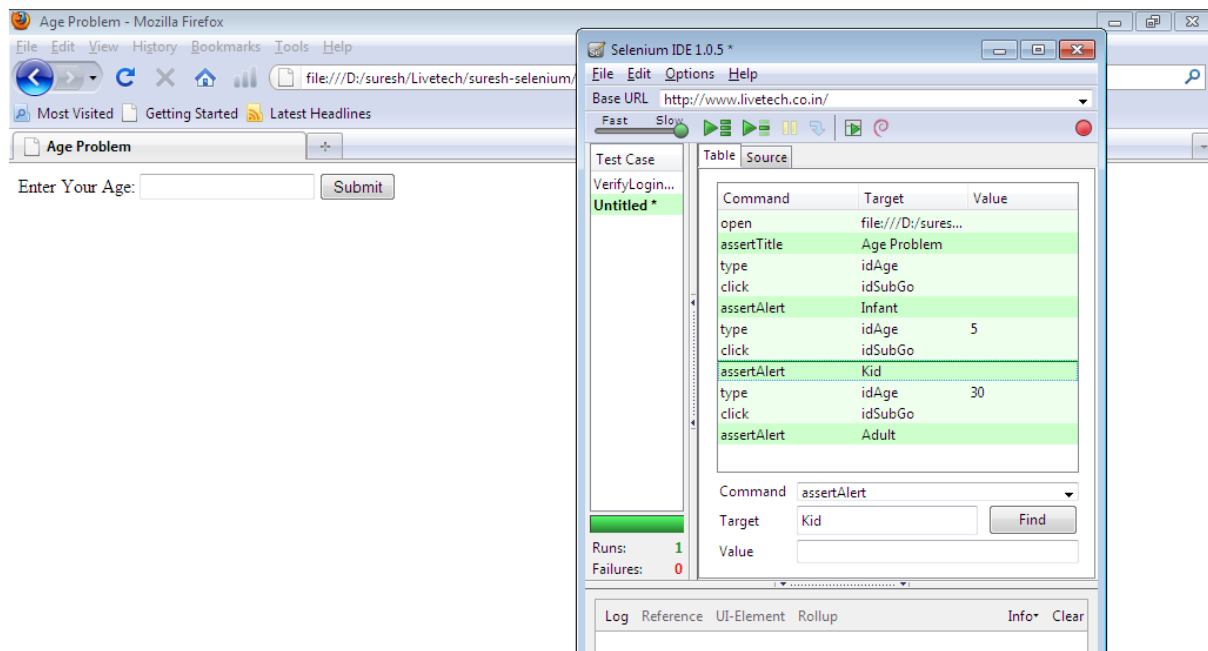


### Verify Alerts

**assertAlert:** By using this command we can handle JavaScript alerts

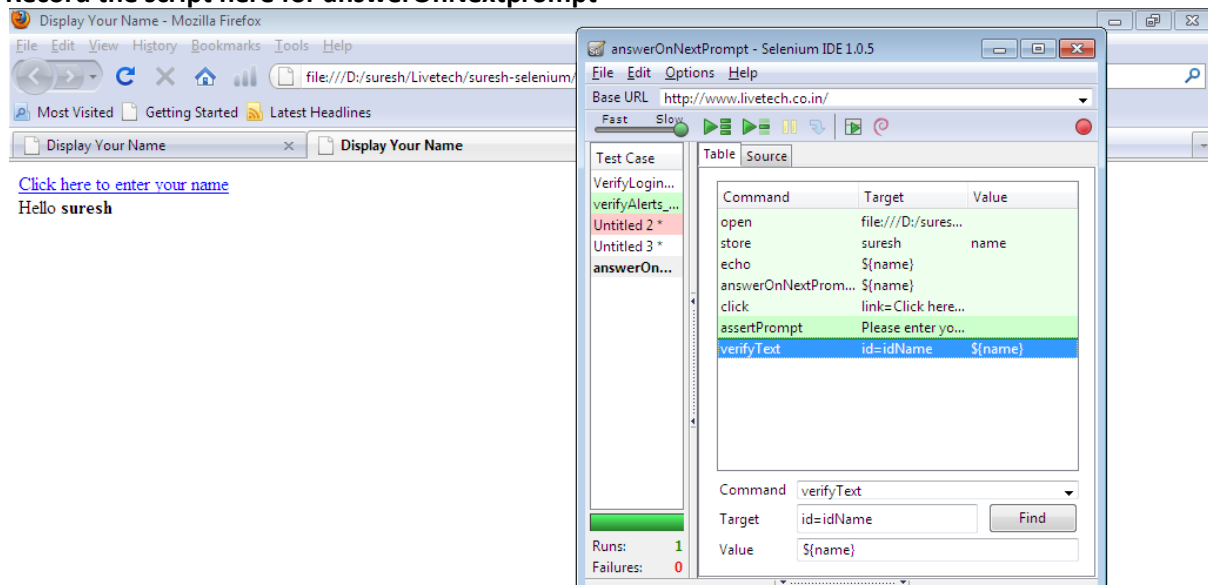
- This stmt generated from getAlert
- Getting an alert has the same effect as manually clicking OK.

**Record the script here for assert alert.**



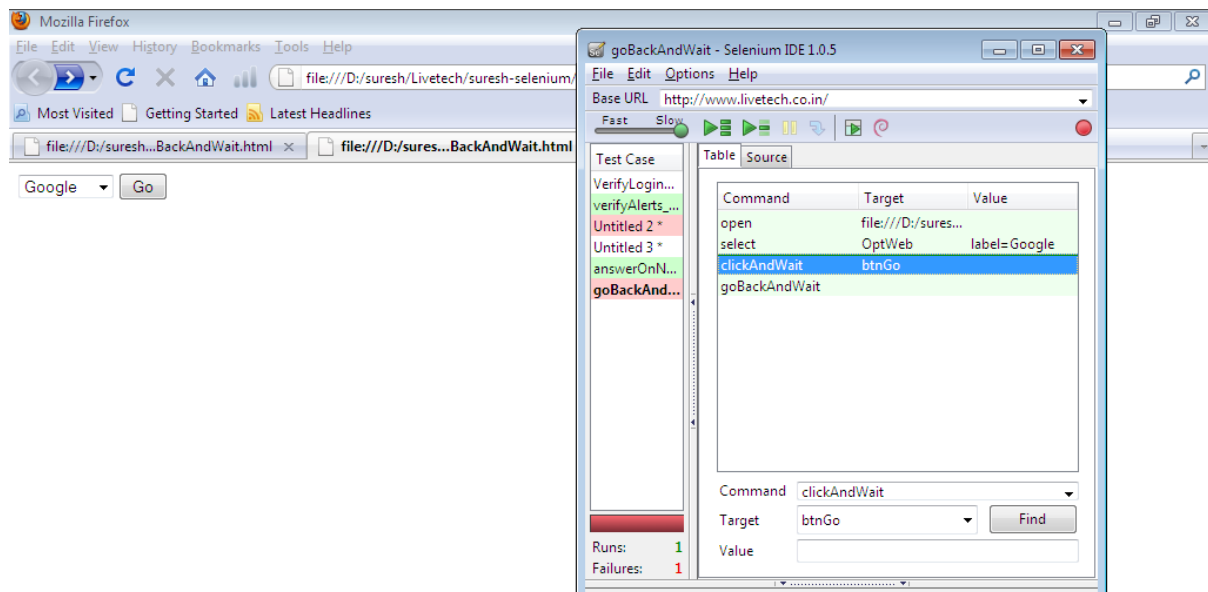
**answerOnNextprompt:** The answer to give in response to the prompt pop-up.

**Record the script here for answerOnNextprompt**



**goBack :** Simulates the user clicking the back button on the browser.

**goBackandWait:** generates from goBack()



### Get Xpath count:

storeXPathCount(xpath,VariableName)

Generated from get xpath count(xpath)

**Xpath:** the xpath expression to evaluate .do not wraps this expression in a count () function. We will do that for you.

**Returns:** the number of nodes that matches the specified xpath.

**Verify Table:** Verify table(tableCelladdress ,pattern)

Generated from getTable(tableCelladdress)

**Arguments:**

→Table cellAddress

**Returns:**

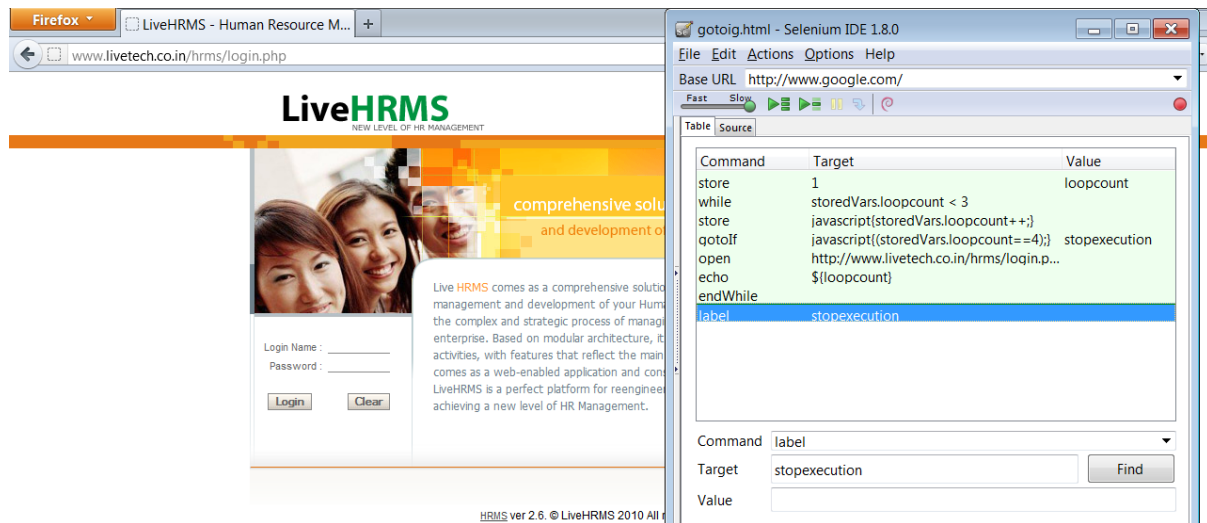
The text from the specified cell. Gets the text from a cell of a table.

The cell address **Syntax:** table locator.row.column

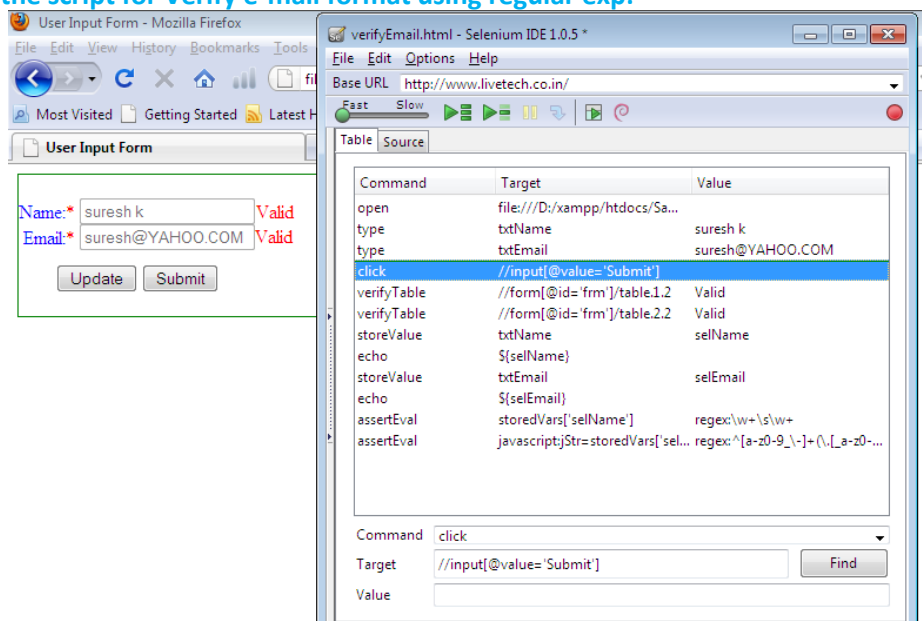
S#	Course Name	Instructor Name	Start Date
1	C++	James	1/2/2009
2	Pascal	John	2/2/2009
3	Cobol	Raja	3/3/2009
4	Selenium	Kangs	4/4/2009
5	Perl	Keith	5/5/2009
6	Python	Michell	6/6/2009

## Adding .js files to selenium IDE(for using While loop and if condition in IDE)

**Navigation:** Select options in IDE ->Focus on Selenium Core Extension ->Browse the .js file ->Add it->Restart Firefox



## TC-Record the script for Verify e-mail format using regular exp:





# Selenium Core

- Selenium core is a test tool for web application
- Selenium core tests run directly in a browser just as real users do and they run in internet explorer, Mozilla and Firefox on windows, linux, and Macintosh.
- Selenium uses java script and iframes to embed a test automation engine in browser.
- Selenium was designed specifically for the acceptance testing requirement of the agile teams.

## Selenium core concepts:

- Cross browser and cross platform compatibility testing.
- Once can test and verify whether the application works correctly on different browsers and operating systems.
- the same script can run on any selenium platform

## For doing execution in selenium core we have 2 modes

1. HTA Mode->its only for IE
  2. HTML mode ->Its supports for all the browsers
- Selenium core will not run directly .its require web server to run

## Installing selenium core

Installing selenium core is a 2 step process

- First install any web server(eg:Apache)
- Place the selenium core under the web server(htdocs folder)
- You need to install the selenium core within the web server, where AUT are deployed.

### 1. Installing Apache web server

- Goto <http://httpd.apache.org/download.cgi>
- download win32 binary without crypto(no\_mod\_ssl)(MSI Installer)
- apache\_2.2.11-win32-x86-no-ssl.msi
- double click the downloaded file and install it

To Check apache works fine do the below steps

- open your windows explorer
- **go to c:\ProgramFiles\ApacheSoftwareFoundation\Apache2.2\htdocs**
- check index.html file is there
- open index.html check whether it will display “it works” message or not-If it displays “It works “ message then Apache is installed successfully

### 2. Installing selenium Core

- Go to C:\ProgramFiles\ApacheSoftwareFoundation\Apache2.2\htdocs folder and create new folder and Name the folder as “Selenium Core”
- Go to <http://seleniumhq.org/download/>
- Right click on the download link under selenium core and download the file under the newly create folder (“Selenium Core”) in htdocs
- Save selenium-core-1.0.bea-2.zipfile under the C:\ProgramFiles\ApacheSoftwareFoundation\Apache2.2\htdocs\SeleniumCore
- Right click on the zip file and unzip by selecting Unzip (Select extract to here)
- After doing unzip open Selenium Core folder ->Open Core folder
- Here if you see Test Runner File then installation is complete successfully.

### Navigation for creating Test suits:

- Open Selenium IDE
- Create TC1 and save as TC1 in the path of (C:\ProgramFiles\ApacheSoftwareFoundation\Apache2.2\htdocs\SeleniumCore\core)
- Create TC2 and save as TC2 in the path of (C:\ProgramFiles\ApacheSoftwareFoundation\Apache2.2\htdocs\SeleniumCore\core)
- In selenium IDE Create New Test Suite and Add created test cases to this suite
- Next save the test suite in the path of (C:\ProgramFiles\ApacheSoftwareFoundation\Apache2.2\htdocs\SeleniumCore\core)

### Navigation to execute HTML suite through selenium Core:

- Open Apache server
- Open Htdocs folder
- Open Selenium core
- Again open selenium core
- Open test Runner
- Give Test Suite path
- Select check box AUT in Separate window
- Select Run automatically
- Select close afterward
- Select save to file
- Provide Result file path

## Core Java

### concepts

- History
- Overview
- Environment Setup
- Basic Syntax
- Object & Classes
- Variable Types
- Modifier Types
- Basic Operators
- Loop Control
- Decision Making
- Numbers
- Characters
- Strings
- Arrays
- Date & Time
- Methods
- Exceptions

### Java Object Oriented

- Inheritance
- Overriding
- Polymorphism
- Abstraction
- Encapsulation
- Interfaces
- Packages

## History of JAVA

JAVA is a distributed technology developed by James Gosling, Patric Naughton, etc., at Sun Micro System has released lot of rules for JAVA and those rules are implemented by JavaSoft Inc, USA (which is the software division of Sun Micro System) in the year 1990. The original name of JAVA is OAK (which is a tree name). In the year 1995, OAK was revised and developed software called JAVA (which is a coffee seed name).

JAVA released to the market in three categories J2SE (JAVA 2 Standard Edition), J2EE (JAVA 2 Enterprise Edition) and J2ME (JAVA 2 Micro/Mobile Edition).

- i. J2SE is basically used for developing client side applications/programs.
- ii. J2EE is used for developing server side applications/programs.
- iii. J2ME is used for developing server side applications/programs.

If you exchange the data between client and server programs (J2SE and J2EE), by default JAVA is having on internal support with a protocol called http. J2ME is used for developing mobile applications and lower/system level applications. To develop J2ME applications we must use a protocol called WAP (Wireless Applications Protocol).

### FEATURES of java

1. Simple
2. Platform independent
3. Architectural neutral
4. Portable
5. Multi threading
6. Distributed
7. Networked
8. Robust
9. Dynamic
10. Secured
11. High performance
12. Interpreted
13. Object Oriented Programming Language

**1. Simple:** JAVA is simple because of the following factors:

- JAVA is free from pointers hence we can achieve less development time and less Execution time [whenever we write a JAVA program we write without pointers and Internally it is converted into the equivalent pointer program].
- Rich set of API (application protocol interface) is available to develop any complex Application.
- The software JAVA contains a program called garbage collector which is always used to Collect unreferenced (unused) memory location for improving performance of a JAVA Program. [Garbage collector is the system JAVA program which runs in the background Along with regular JAVA program to collect unreferenced memory locations by running At periodical interval of times for improving performance of JAVA applications.
- JAVA contains user friendly syntax's for developing JAVA applications.

**2. Platform Independent:**

A program or technology is said to be platform independent if and only if which can run on all available operating systems.

The languages like C, Cpp are treated as platform dependent languages since these

languages are taking various amount of memory spaces on various operating systems [the operating system dos understands everything in the form of its native format called Mozart (MZ) whereas the operating system Unix understands everything in its native format called embedded linking format (elf). When we write a C or Cpp program on dos operating and if we try to transfer that program to Unix operating system, we are unable to execute since the format of these operating systems are different and more over the C, Cpp software does not contain any special programs which converts one format of one operating system to another format of other operating system].

The language like JAVA will have a common data types and the common memory spaces on all operating systems and the JAVA software contains the special programs which converts the format of one operating system to another format of other operating system. Hence JAVA language is treated as platform independent language.

### 3. Architectural Neutral:

A language or technology is said to be architectural neutral which can run on any available processors in the real world. The languages like C, Cpp are treated as architectural dependent. The language like JAVA can run on any of the processor irrespective of their Vendor.

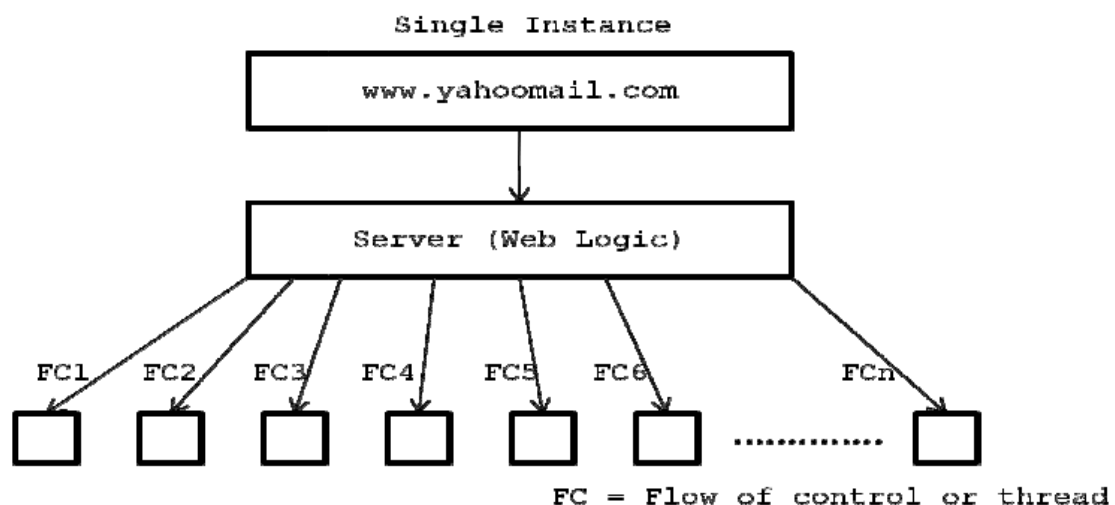
Machine1		Machine2
DOS	$\neq$	UNIX
P3	$\neq$	AMD
JDK	$=$	JDK

**Portability = platform independent + architecture neutral**

### 4. Portable:

A portable language is one which can run on all operating systems and on all processors irrespective their architectures and providers. The languages like C, Cpp are treated as non portable languages whereas the language JAVA is called portable language.

### 5. Multi Threading:

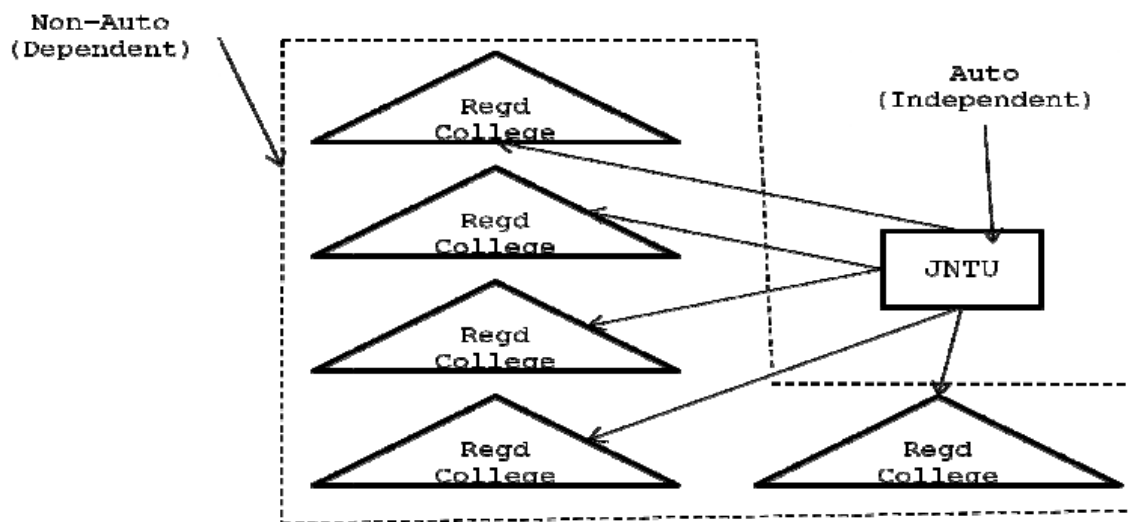


- A flow of control is known as thread
- A multi threaded program is one in which there exists multiple flow of controls i.e. threads

- A program is said to be multi threaded program if and only if there exists n number of sub-programs. For each and every sub sub-program there exists a separate flow of control. All such flow of controls are executing concurrently. Such flow of controls is known as threads .Such type of applications type of applications is known as multithreading applications
- The languages like C, Cpp are treated as single threaded modeling languages (STML). SMTL are those in which there exists single flow of control
- The languages like JAVA and DOT NET are treated as multi threaded modeling languages(MTML). MTML are those in which there exist multiple flows of controls

#### 6. Distributed:

A service is said to be a distributed service which runs in multiple servers and that service can be accessed by n number of clients across the globe. In order to develop distributed applications we must require architecture called trusted network architecture. To develop these applications we require a technology called J2EE. Distributed applications are preferred by larger scale organizations.

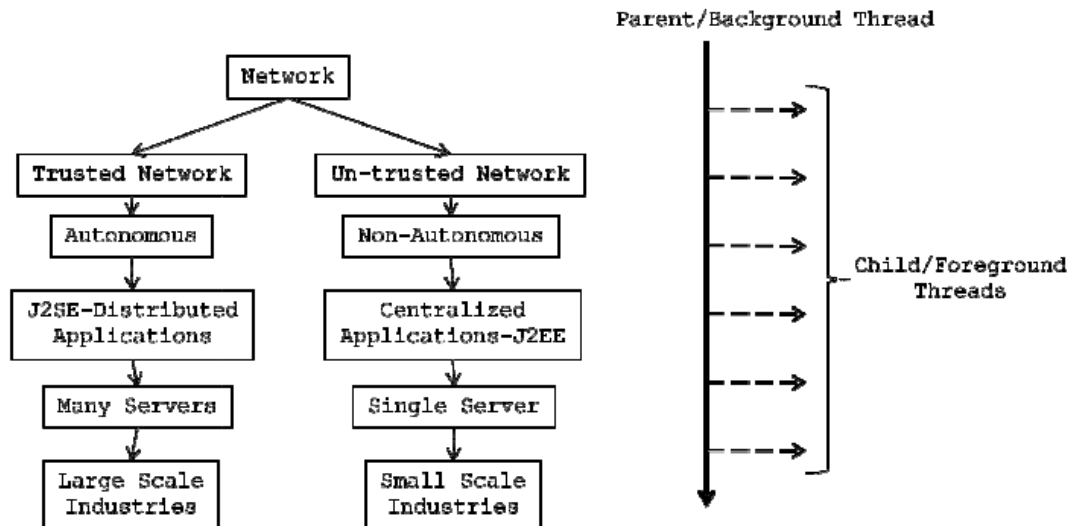


#### 7. Networked:

In real world we have two types of networks. They are un-trusted network and trusted networks.

##### Un-trusted networks:

A network is said to be un-trusted network in which there exists n number of inter connected non-autonomous architecture Un-trusted network is also known as LAN. Using this network architecture, we develop centralized applications. A centralized application is one which runs on single server and it can be access in limited graces. In order to develop centralized applications we may use a technology called J2SE and these kinds of applications are preferred by small scale organization.



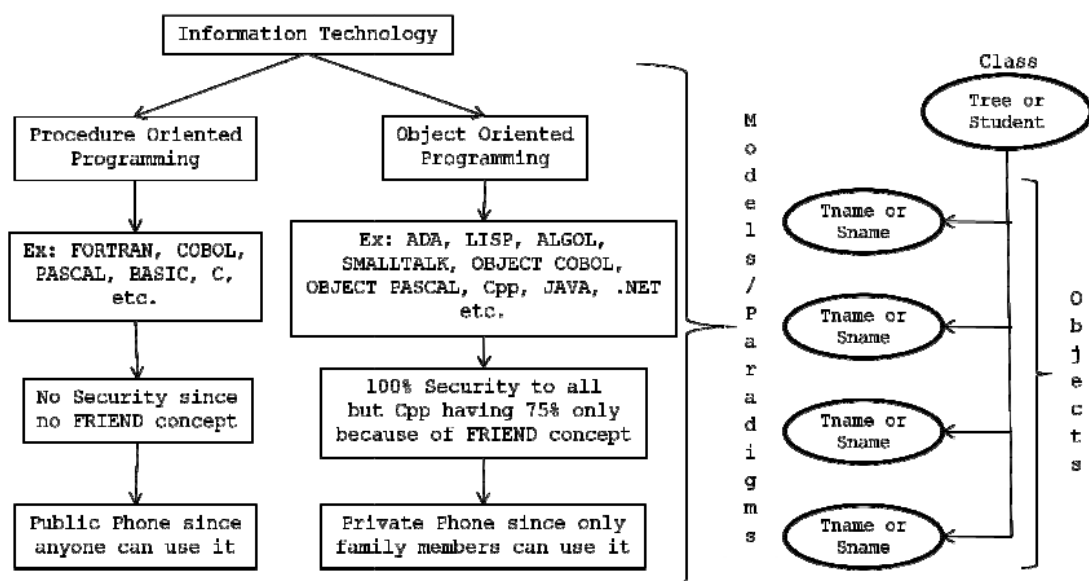
### Trusted network:

A network is said to be trusted network in which there exists n number of inter connected autonomous architecture. Trusted network Trusted network is also know as WAN Using this network, we can develop distributed applications .A distributed application is one which runs on multipul servers and it can be access in unlimited graces. In order to develop distributed applications we may use a technology called J2EE and these kinds of applications are preferred by large scale organizations.

### 8.Java is OBJECT ORIENTED PROGRAMMING

In an IT we have two types of programming models (pa paradigms) are available. They are procedure oriented programming language and object oriented programming language

If we represent the data using procedural oriented programming languages then there is no security for the data which we represent . For example when we represent the data of a student in C language using structures concept, the student data can be accessed by all the functions which we write as a part of C program. If one of the functions manipulates or damages the data then we are loosing correction-less (integrity) of the data. Examples of procedure oriented programming languages are FORTRON, COBOL, PASCAL, BASIC, C, etc.



When we represent the data in object oriented programming language we get the security.Examples of object oriented programming languages are LISP, ADA, ALGOL,

SMALLTALK, OBJECT, COBOL, OBJECT PASCAL, Cpp, JAVA, DOT NET, etc. In order to say any language is an object oriented programming language.

### OOPs Principles

1. Class
2. Object
3. Abstraction
4. Encapsulation
5. Inheritance
6. Polymorphism
7. Interfaces
8. Packages

### **Setup-Environment variables**

- Right Click on my computer icon of desktop of your computer
- Select properties
- Select Advanced tab
- click on environment variables button
- Click on New/Edit to create a variable

Here the variable is a pair of values name & value

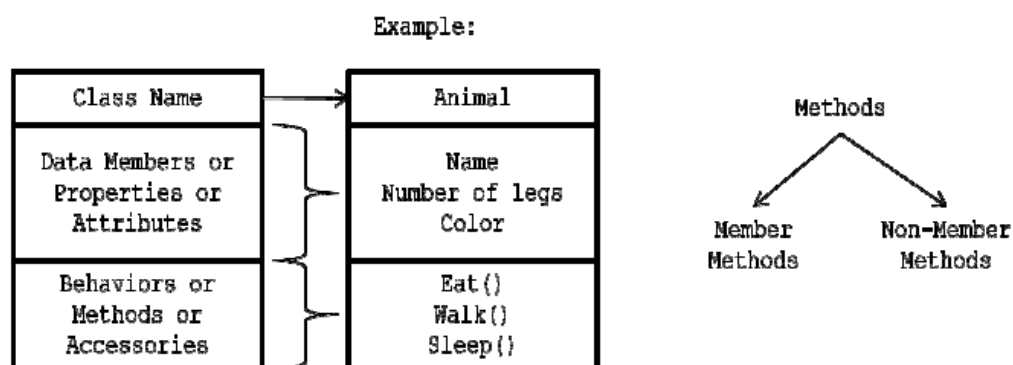
- Path: c:\programfiles\java\JDK1.6.0.1\bin
- ClassPath(JavaFiles): %classpath%path for java files not spaces

### **Eclipse**

It is one of the editors for writing the Java code.

**Class :** A class can be defined as a template/ blue print that describe the behaviors/states that object of its type support. (or) "A class is a way of binding the data and associated methods in a single unit

*Class diagram for defining a class:*



### **Syntax for defining a CLASS:**

```
Class <clsname>
{
Variable declaration;
Methods definition;
};
```

Here, class is a **keyword** which is used for developing or creating **user defined data types**. Clnsname represents a JAVA valid variable name and it is treated as name of the class. Class names are used for creating **objects**.

Class contains two parts namely **variable declaration** and **method definitions**. Variable Declaration represents what type of **data members** which we use as a part of the class. Method definition represents the type of methods which we used as the path of the class to perform an operation.

By making use of the variables, which are declared inside the class? Every operation in JAVA must be defined with in the class only i.e. outside definition is not possible.

#### First Java Program to print Hello World

```
public class MyFirstJavaProgram
{
    /* This is my first java program.
     * This will print 'Hello World' as the output
     */

    public static void main(String []args)
    {
        System.out.println("Hello World"); // prints Hello World
    }
}
```

**Lets look at how to save the file, compile and run the program. Please follow the steps given below:**

1. Open notepad and add the code as above.
2. Save the file as : MyFirstJavaProgram.java.
3. Open a command prompt window and go o the directory where you saved the class. Assume its C:\.
4. Type ' javac MyFirstJavaProgram.java ' and press enter to compile your code. If there are no errors in your code the command prompt will take you to the next line.( Assumption : The path variable is set).
5. Now type ' java MyFirstJavaProgram ' to run your program.
6. You will be able to see ' Hello World ' printed on the window.

```
C : > javac MyFirstJavaProgram.java
C : > java MyFirstJavaProgram
Hello World
```

#### Program for taking the inputs from key board(Addition of 2 numbers)

```
import java.util.Scanner;
class AddNumbers
{
    public static void main(String args[])
    {
        int x, y, z;
        System.out.println("Enter two integers to calculate their sum ");
        Scanner in = new Scanner(System.in);
        x = in.nextInt();
        y = in.nextInt();
        z = x + y;
        System.out.println("Sum of entered integers = "+z);
    }
}
```



**About Java programs, it is very important to keep in mind the following points.**

**Case Sensitivity** - Java is case sensitive which means identifier Hello and hello would have different meaning in Java.

**Class Names** - For all class names the first letter should be in Upper Case.

If several words are used to form a name of the class each inner words first letter should be in Upper Case. Example class MyFirstJavaClass

**Method Names** - All method names should start with a Lower Case letter.

If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case. Example public void myMethodName()

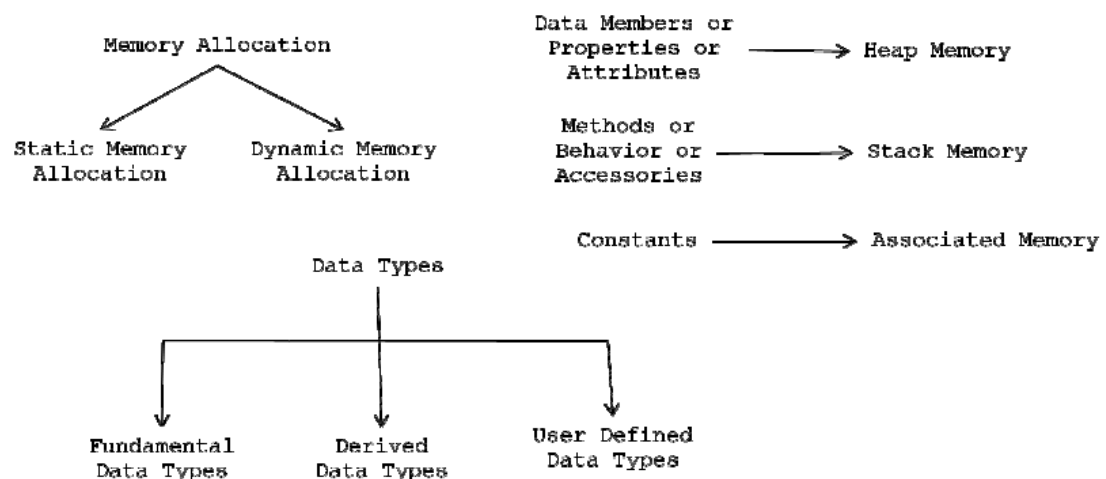
**Program File Name** - Name of the program file should exactly match the class name.

Example : Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as 'MyFirstJavaProgram.java'

**public static void main(String args[])** - java program processing starts from the main() method which is a mandatory part of every java program..

**Object** - Objects have states and behaviors. Example: A dog has states-color, name, breed as well as behaviors -wagging, barking, eating. An object is an instance of a class. (or)

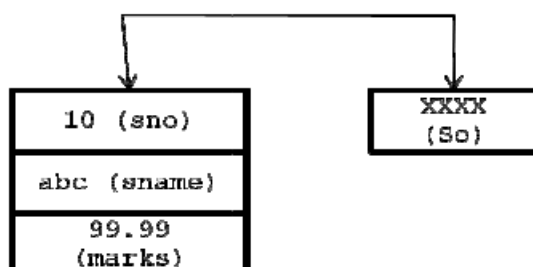
**Instance** (instance is a mechanism of allocating sufficient amount of memory space for data members of a class) of a class is known as an object.



**Syntax-1 for defining an OBJECT:**

```

<Clname> objname = new <clname ()>
Student so = new student();
  
```



```

student So; //student object declaration//
So = new student(); //student object referencing//
  
```

**Example of creating an object is given below:**

```

class Puppy{
    public Puppy(String name)
    {
        System.out.println("Passed Name is :" + name );
    }
    public static void main(String []args){
        // Following statement would create an object myPuppy
        Puppy myPuppy = new Puppy( "tommy" );
    }
}

```

If we compile and run the above program then it would produce following result:

```
Passed Name is :tommy
```

**Main Class:** It is important section for all programming .it is defined on a class which as main method is called main class.

#### Syntax:

```

Public static void main (String args[])
{
}

```

#### Data Types

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals, or characters in these variables.

**There are two data types available in Java:**

1. Primitive Data Types
2. Reference/Object Data Types

#### Primitive Data Types:

There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named by a key word. Let us now look into detail about the eight primitive data types.

##### byte:

- Byte data type is a 8-bit signed two's complement integer.
- Default value is 0
- Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an int.
- Example : byte a = 100 , byte b = -50

##### short:

- Short data type is a 16-bit signed two's complement integer.
- Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an int
- Default value is 0.
- Example : short s= 10000 , short r = -20000

**int:**

- Int data type is a 32-bit signed two's complement integer.
- Int is generally used as the default data type for integral values unless there is a concern about memory.
- The default value is 0.
- Example : int a = 100000, int b = -200000

**long:**

- Long data type is a 64-bit signed two's complement integer.
- This type is used when a wider range than int is needed.
- Default value is 0L.
- Example : int a = 100000L, int b = -200000L

**float:**

- Float data type is a single-precision 32-bit IEEE 754 floating point.
- Float is mainly used to save memory in large arrays of floating point numbers.
- Default value is 0.0f.
- Float data type is never used for precise values such as currency.
- Example : float f1 = 234.5f

**double:**

- double data type is a double-precision 64-bit IEEE 754 floating point.
- This data type is generally used as the default data type for decimal values. generally the default choice.
- Double data type should never be used for precise values such as currency.
- Default value is 0.0d.
- Example : double d1 = 123.4

**boolean:**

- boolean data type represents one bit of information.
- There are only two possible values : true and false.
- This data type is used for simple flags that track true/false conditions.
- Default value is false.
- Example : boolean one = true

**char:**

- char data type is a single 16-bit Unicode character.
- Minimum value is '\u0000' (or 0).
- Maximum value is '\uffff' (or 65,535 inclusive).
- Char data type is used to store any character.
- Example . char letterA ='A'

**Reference Data Types:**

- Reference variables are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed. For example, Employee, Puppy etc.
- Class objects, and various type of array variables come under reference data type.
- Default value of any reference variable is null.

- A reference variable can be used to refer to any object of the declared type or any compatible type.
- Example : `Animal animal = new Animal("giraffe");`
- Java language supports few special escape sequences for String and char literals as well. They are:

Notation	Character represented
<code>\n</code>	Newline (0x0a)
<code>\f</code>	Formfeed (0x0c)
<code>\b</code>	Backspace (0x08)
<code>\s</code>	Space (0x20)
<code>\t</code>	tab
<code>\"</code>	Double quote
<code>\'</code>	Single quote
<code>\\</code>	backslash
<code>\ddd</code>	Octal character (ddd)
<code>\uxxxx</code>	Hexadecimal UNICODE character (xxxx)

Eg:

```
Public class example1
{
Public static void main (string args[])
{
Int a=10;
Int b=20;
System.out.println(a+b);
}
}
```

## Variable Types

In Java, all variables must be declared before they can be used. The basic form of a variable declaration is shown here:

```
type identifier [ = value][, identifier [= value] ...] ;
```

The type is one of Java's datatypes. The identifier is the name of the variable. To declare more than one variable of the specified type, use a comma-separated list.

Here are several examples of variable declarations of various types. Note that some include an initialization.

```
int a, b, c;           // declares three ints, a, b, and c.
int d = 3, e, f = 5;   // declares three more ints, initializing
                        // d and f.
byte z = 22;           // initializes z.
double pi = 3.14159;   // declares an approximation of pi.
char x = 'x';          // the variable x has the value 'x'.
```

### There are three kinds of variables in Java:

1. Local variables
2. Instance variables
3. Class/static variables

#### Local variables :

- Local variables are declared in methods, constructors, or blocks.
- Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor or block.
- Access modifiers cannot be used for local variables.
- Local variables are visible only within the declared method, constructor or block.
- Local variables are implemented at stack level internally.
- There is no default value for local variables so local variables should be declared and an initial value should be assigned before the first use.

#### Example:

Here age is a local variable. This is defined inside pupAge() method and its scope is limited to this method only.

```
public class Test{
    public void pupAge(){
        int age = 0;
        age = age + 7;
        System.out.println("Puppy age is : " + age)
    }

    public static void main(String args[]){
        Test test = new Test();
        Test.pupAge();
    }
}
```

This would produce following result:

```
Puppy age is: 7
```

#### Instance variables:

- Instance variables are declared in a class, but outside a method, constructor or any block.
- When a space is allocated for an object in the heap a slot for each instance variable value is created.
- Instance variables are created when an object is created with the use of the key word 'new' and destroyed when the object is destroyed.
- Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present through out the class.
- Instance variables can be declared in class level before or after use.

- Access modifiers can be given for instance variables.
- The instance variables are visible for all methods, constructors and block in the class. Normally it is recommended to make these variables private (access level). However visibility for subclasses can be given for these variables with the use of access modifiers.
- Instance variables have default values. For numbers the default value is 0, for Booleans it is false and for object references it is null. Values can be assigned during the declaration or within the constructor.
- Instance variables can be accessed directly by calling the variable name inside the class. However within static methods and different class (when instance variables are given accessibility) they should be called using the fully qualified name . ObjectReference.VariableName.

#### Example:

```
import java.io.*;

class Employee{
    // this instance variable is visible for any child class.
    public String name;

    // salary variable is visible in Employee class only.
    private double salary;

    // The name variable is assigned in the constructor.
    public Employee (String empName){
        name = empName;
    }

    // The salary variable is assigned a value.
    public void setSalary(double empSal){
        salary = empSal;
    }

    // This method prints the employee details.
    public void printEmp(){
        System.out.println("name : " + name );
        System.out.println("salary : " + salary);
    }

    public static void main(String args[]){
        Employee empOne = new Employee("Ransika");
        empOne.setSalary(1000);
        empOne.printEmp();
    }
}
```

This would produce following result:

```
name : Ransika
salary :1000.0
```

#### Class/static variables:

- Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
- There would only be one copy of each class variable per class, regardless of how many objects are created from it.

- Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final and static. Constant variables never change from their initial value.
- Static variables are stored in static memory. It is rare to use static variables other than declared final and used as either public or private constants.
- Static variables are created when the program starts and destroyed when the program stops.
- Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.
- Default values are same as instance variables. For numbers the default value is 0, for Booleans it is false and for object references it is null. Values can be assigned during the declaration or within the constructor. Additionally values can be assigned in special static initializer blocks.
- Static variables can be accessed by calling with the class name . ClassName.VariableName.
- When declaring class variables as public static final, then variables names (constants) are all in upper case. If the static variables are not public and final the naming syntax is the same as instance and local variables.

#### Example:

```
import java.io.*;

class Employee1{
    // salary variable is a private static variable
    private static double salary;

    // DEPARTMENT is a constant
    public static final String DEPARTMENT = "Development";

    public static void main(String args[]){
        salary = 1000;
        System.out.println(DEPARTMENT+"average salary:"+salary);
    }
}
```

This would produce following result:

```
Development average salary:1000
```

#### Java Access Modifiers

The Java language has a wide variety of modifiers, including the following:

- Java Access Modifiers
- Non Access Modifiers

Java provides a number of **access modifiers** to set access levels for classes, variables, methods and constructors. The four access levels are:

1. Visible to the package. the **default**. No modifiers are needed.
2. Visible to the class only (**private**).
3. Visible to the world (**public**).
4. Visible to the package and all subclasses (**protected**).

Java provides a number of **non-access modifiers** to achieve many other functionality.

- The **static** modifier for creating class methods and variables
- The **final** modifier for finalizing the implementations of classes, methods, and variables.
- The **abstract** modifier for creating abstract classes and methods.
- The synchronized and volatile modifiers, which are used for threads.

## Operators

Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups:

- Arithmetic Operators(- + \* / % ++ --)
- Relational Operators (> < >= <= == !=)
- Bitwise Operators(& | ^ >> >>>)
- Logical Operators (&& || & | ! ^)
- Assignment Operators( = , +=)
- Misc Operators ( ? : )

### The Arithmetic Operators:

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:

Assume integer variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
+	Addition - Adds values on either side of the operator	A + B will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	A - B will give -10
*	Multiplication - Multiplies values on either side of the operator	A * B will give 200
/	Division - Divides left hand operand by right hand operand	B / A will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	B % A will give 0
++	Increment - Increase the value of operand by 1	B++ gives 21
--	Decrement - Decrease the value of operand by 1	B-- gives 19

```
class Test {
public static void main(String args[]) {
    int a = 10;
    int b = 20;
    int c = 25;
    int d = 25;
    System.out.println("a + b = " + (a + b) );
    System.out.println("a - b = " + (a - b) );
    System.out.println("a * b = " + (a * b) );
    System.out.println("b / a = " + (b / a) );
    System.out.println("b % a = " + (b % a) );
    System.out.println("c % a = " + (c % a) );
    System.out.println("a++ = " + (a++) );
    System.out.println("b-- = " + (a--) );
    // Check the difference in d++ and ++d
    System.out.println("d++ = " + (d++) );
    System.out.println("++d = " + (++d) );
}
}
```

This would produce following result:



```

a + b = 30
a - b = -10
a * b = 200
b / a = 2
b % a = 0
c % a = 5
a++    = 10
b--    = 11
d++    = 25
++d    = 27

```

### The Relational Operators:

There are following relational operators supported by Java language  
Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

#### Example :

```

class Test {
public static void main(String args[]) {
    int a = 10;
    int b = 20;
    System.out.println("a == b = " + (a == b) );
    System.out.println("a != b = " + (a != b) );
    System.out.println("a > b = " + (a > b) );
    System.out.println("a < b = " + (a < b) );
    System.out.println("b >= a = " + (b >= a) );
    System.out.println("b <= a = " + (b <= a) );
}
}

```

This would produce following result:

```

a == b = false
a != b = true

```

```

a > b = false
a < b = true
b >= a = true
b <= a = false

```

### The Bitwise Operators:

Java defines several bitwise operators which can be applied to the integer types, long, int, short, char, and byte.

Bitwise operator works on bits and perform bit by bit operation. Assume if a = 60; and b = 13; Now in binary format they will be as follows:

a = 0011 1100

b = 0000 1101

-----

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

The following table lists the bitwise operators:

Assume integer variable A holds 60 and variable B holds 13 then:

Show Examples

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
	Binary OR Operator copies a bit if it exists in either operand.	(A   B) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A ) will give -60 which is 1100 0011
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111
>>>	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>>2 will give 15 which is 0000 1111

### Example

```

class Test {
public static void main(String args[]) {
    int a = 60;    /* 60 = 0011 1100 */
    int b = 13;    /* 13 = 0000 1101 */
    int c = 0;
}
}

```

```
c = a & b;          /* 12 = 0000 1100 */
System.out.println("a & b = " + c );

c = a | b;          /* 61 = 0011 1101 */
System.out.println("a | b = " + c );

c = a ^ b;          /* 49 = 0011 0001 */
System.out.println("a ^ b = " + c );

c = ~a;             /* -61 = 1100 0011 */
System.out.println("~a = " + c );

c = a << 2;          /* 240 = 1111 0000 */
System.out.println("a << 2 = " + c );

c = a >> 2;          /* 215 = 1111 */
System.out.println("a >> 2 = " + c );

c = a >>> 2;         /* 215 = 0000 1111 */
System.out.println("a >>> 2 = " + c );
}
}
```

This would produce following result:

```
a & b = 12
a | b = 61
a ^ b = 49
~a = -61
a << 2 = 240
a >> 15
a >>> 15
```

**The Logical Operators:**

The following table lists the logical operators:  
Assume boolean variables A holds true and variable B holds false then:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is false.
	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A    B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

**Example**

```
class Test {
public static void main(String args[]) {
    int a = true;
```

```

int b = false;

System.out.println("a && b = " + (a&&b) );

System.out.println("a || b = " + (a||b) );

System.out.println("!(a && b) = " + !(a && b) );
}
}

```

This would produce following result:

```

a && b = false
a || b = true
!(a && b) = true

```

## The Assignment Operators:

There are following assignment operators supported by Java language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assigne value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator	C ^= 2 is same as C = C ^ 2
=	bitwise inclusive OR and assignment operator	C  = 2 is same as C = C   2

Example :

```

class Test {
public static void main(String args[]) {
    int a = 10;
    int b = 20;
    int c = 0;

```

```

c = a + b;
System.out.println("c = a + b = " + c );

c += a ;
System.out.println("c += a  = " + c );

c -= a ;
System.out.println("c -= a = " + c );

c *= a ;
System.out.println("c *= a = " + c );

a = 10;
c = 15;
c /= a ;
System.out.println("c /= a = " + c );

a = 10;
c = 15;
c %= a ;
System.out.println("c %= a  = " + c );

c <<= 2 ;
System.out.println("c <<= 2 = " + c );

c >>= 2 ;
System.out.println("c >>= 2 = " + c );

c >>= 2 ;
System.out.println("c >>= a = " + c );

c &= a ;
System.out.println("c &= 2  = " + c );

c ^= a ;
System.out.println("c ^= a   = " + c );

c |= a ;
System.out.println("c |= a   = " + c );
}
}

```

This would produce following result:

```

c = a + b = 30
c += a  = 40
c -= a = 30
c *= a = 300
c /= a = 1
c %= a  = 5
c <<= 2 = 20
c >>= 2 = 5
c >>= 2 = 1
c &= a  = 0
c ^= a   = 10
c |= a   = 10

```

## Misc Operators

There are few other operators supported by Java Language.

### Conditional Operator ( ? : ):

Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide which value should be assigned to the variable. The operator is written as :

```
variable x = (expression) ? value if true : value if false
```

Following is the example:

```
public class Test {
    public static void main(String args[]){
        int a , b;
        a = 10;
        b = (a == 1) ? 20: 30;
        System.out.println( "Value of b is : " + b );

        b = (a == 10) ? 20: 30;
        System.out.println( "Value of b is : " + b );
    }
}
```

This would produce following result:

```
Value of b is : 30
Value of b is : 20
```

## Loops:

There may be a situation when we need to execute a block of code several number of times, and is often referred to as a loop.

Java has very flexible three looping mechanisms. You can use one of the following three loops:

- while Loop
- do...while Loop
- for Loop

As of java 5 the enhanced for loop was introduced. This is mainly used for Arrays.

### The while Loop:

A while loop is a control structure that allows you to repeat a task a certain number of times.

#### Syntax:

The syntax of a while loop is:

```
while(Boolean_expression)
{
    //Statements
}
```

When executing, if the `boolean_expression` result is true then the actions inside the loop will be executed. This will continue as long as the expression result is true. Here key point of the while loop is that the loop might not ever run. When the expression is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

**Example:**

```
public class Test {
    public static void main(String args[]){
        int x= 10;

        while( x < 20 ){
            System.out.print("value of x : " + x );
            x++;
            System.out.print("\n");
        }
    }
}
```

This would produce following result:

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
```

**The do...while Loop:**

A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time.

Syntax:

The syntax of a do...while loop is:

```
do
{
    //Statements
}while(Boolean_expression);
```

Notice that the Boolean expression appears at the end of the loop, so the statements in the loop execute once before the Boolean is tested.

If the Boolean expression is true, the flow of control jumps back up to do, and the statements in the loop execute again. This process repeats until the Boolean expression is false.

**Example:**

```
public class Test {
    public static void main(String args[]){
        int x= 10;

        do{
```

```

        System.out.print("value of x : " + x );
        x++;
        System.out.print("\n");
    }while( x < 20 );
}

```

This would produce following result:

```

value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19

```

### The for Loop:

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

A for loop is useful when you know how many times a task is to be repeated.

### Syntax:

The syntax of a for loop is:

```

for(initialization; Boolean_expression; update)
{
    //Statements
}

```

### Here is the flow of control in a for loop:

The initialization step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.

Next, the Boolean expression is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement past the for loop.

After the body of the for loop executes, the flow of control jumps back up to the update statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the Boolean expression.

The Boolean expression is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then update step, then Boolean expression). After the Boolean expression is false, the for loop terminates.

### Example:

```

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

        for(int x = 10; x < 20; x = x+1){
            System.out.print("value of x : " + x );

```



```

        System.out.print("\n");
    }
}

```

This would produce following result:

```

value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19

```

### Enhanced for loop in Java:

As of java 5 the enhanced for loop was introduced. This is mainly used for Arrays.

#### Syntax:

The syntax of enhanced for loop is:

```

for(declaration : expression)
{
    //Statements
}

```

- **Declaration.** The newly declared block variable, which is of a type compatible with the elements of the array you are accessing. The variable will be available within the for block and its value would be the same as the current array element.
- **Expression.** This evaluate to the array you need to loop through. The expression can be an array variable or method call that returns an array.

#### Example:

```

public class Test {
    public static void main(String args[]){
        int [] numbers = {10, 20, 30, 40, 50};

        for(int x : numbers ){
            System.out.print( x );
            System.out.print(",");
        }
        System.out.print("\n");
        String [] names ={"James", "Larry", "Tom", "Lacy"};
        for( String name : names ) {
            System.out.print( name );
            System.out.print(",");
        }
    }
}

```

**This would produce following result:**

```
10,20,30,40,50,  
James,Larry,Tom,Lacy,
```

### The break Keyword:

The break keyword is used to stop the entire loop. The break keyword must be used inside any loop or a switch statement.

The break keyword will stop the execution of the innermost loop and start executing the next line of code after the block.

### Syntax:

The syntax of a break is a single statement inside any loop:

```
break;
```

### Example:

```
public class Test {  
    public static void main(String args[]){  
        int [] numbers = {10, 20, 30, 40, 50};  
  
        for(int x : numbers ){  
            if( x == 30 ){  
                break;  
            }  
            System.out.print( x );  
            System.out.print("\n");  
        }  
    }  
}
```

This would produce following result:

```
10  
20
```

### The continue Keyword:

The continue keyword can be used in any of the loop control structures. It causes the loop to immediately jump to the next iteration of the loop.

In a for loop, the continue keyword causes flow of control to immediately jump to the update statement.

In a while loop or do/while loop, flow of control immediately jumps to the Boolean expression.

### Syntax:

The syntax of continue is a single statement inside any loop:

```
continue;
```

### Example:

```
public class Test {
    public static void main(String args[]){
        int [] numbers = {10, 20, 30, 40, 50};

        for(int x : numbers ){
            if( x == 30 ){
                continue;
            }
            System.out.print( x );
            System.out.print("\n");
        }
    }
}
```

This would produce following result:

```
10
20
40
50
```

### Conditions:

There are two types of decision making statements in Java. They are:

- if statements
- switch statements

#### The if Statement:

If statement consists of a Boolean expression followed by one or more statements.

#### Syntax:

The syntax of if statement is:

```
if(Boolean expression)
{
    //Statements will execute if the Boolean expression is true
}
```

If the Boolean expression evaluates to true then the block of code inside the if statement will be executed. If not the first set of code after the end of if statement (after the closing curly brace) will be executed.

### Example:

```
public class Test {
    public static void main(String args[]){
        int x = 10;
```

```

        if( x < 20 ){
            System.out.print("This is if statement");
        }
    }
}

```

This would produce following result:

```
This is if statement
```

#### **if...else Statement:**

An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.

#### **Syntax:**

The syntax of a if...else is:

```

if(Boolean expression){
    //Executes when the Boolean expression is true
}else{
    //Executes when the Boolean expression is false
}

```

#### **Example:**

```

public class Test {
    public static void main(String args[]){
        int x = 30;

        if( x < 20 ){
            System.out.print("This is if statement");
        }else{
            System.out.print("This is else statement");
        }
    }
}

```

This would produce following result:

```
This is else statement
```

#### **if...else if...else Statement:**

An if statement can be followed by an optional else if...else statement, which is very usefull to test various conditions using single if...else if statement.

When using if , else if , else statements there are few points to keep in mind.

- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of the remaining else if's or else's will be tested.

#### **Syntax:**

The syntax of a if...else is:

```

if(Boolean expression 1){
    //Executes when the Boolean expression 1 is true
}else if(Boolean_expression 2){
    //Executes when the Boolean expression 2 is true
}else if(Boolean_expression 3){
    //Executes when the Boolean expression 3 is true
}else {
    //Executes when the none of the above condition is true.
}

```

**Example:**

```

public class Test {
    public static void main(String args[]){
        int x = 30;

        if( x == 10 ){
            System.out.print("Value of X is 10");
        }else if( x == 20 ){
            System.out.print("Value of X is 20");
        }else if( x == 30 ){
            System.out.print("Value of X is 30");
        }else{
            System.out.print("This is else statement");
        }
    }
}

```

This would produce following result:

```
Value of X is 30
```

**Nested if...else Statement:**

It is always legal to nest if-else statements, which means you can use one if or else if statement inside another if or else if statement.

**Syntax:**

The syntax for a nested if...else is as follows:

```

if(Boolean expression 1){
    //Executes when the Boolean expression 1 is true
    if(Boolean expression 2){
        //Executes when the Boolean expression 2 is true
    }
}

```

You can nest else if...else in the similar way as we have nested if statement.

**Example:**

```

public class Test {
    public static void main(String args[]){
        int x = 30;
        int y = 10;
    }
}

```

```

    if( x == 30 ){
        if( y == 10 ){
            System.out.print("X = 30 and Y = 10");
        }
    }
}

```

This would produce following result:

```
X = 30 and Y = 10
```

### The switch Statement:

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

### Syntax:

The syntax of enhanced for loop is:

```

switch(expression){
    case value :
        //Statements
        break; //optional
    case value :
        //Statements
        break; //optional
    //You can have any number of case statements.
    default : //Optional
        //Statements
}

```

### The following rules apply to a switch statement:

- The variable used in a switch statement can only be a byte, short, int, or char.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The value for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.
- When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.
- A switch statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

### Example:

```

import java.io.*;

public class SwitchExample{
    public static void main(String[] args) throws Exception{
        int x, y;
        BufferedReader object = new BufferedReader

```

```

    (new InputStreamReader(System.in));
    System.out.println("Enter two numbers for operation:");
    try{
        x = Integer.parseInt(object.readLine());
        y = Integer.parseInt(object.readLine());
        System.out.println("1. Add");
        System.out.println("2. Subtract");
        System.out.println("3. Multiply");
        System.out.println("4. Divide");
        System.out.println("enter your choice:");
        int a= Integer.parseInt(object.readLine());
        switch (a){
            case 1:
                System.out.println("Enter the number one=" + (x+y));
                break;
            case 2:
                System.out.println("Enter the number two=" + (x-y));
                break;
            case 3:
                System.out.println("Enetr the number three="+ (x*y));
                break;
            case 4:
                System.out.println("Enter the number four="+ (x/y));
                break;
            default:
                System.out.println("Invalid Entry!");
        }
    }
    catch (NumberFormatException ne){
        System.out.println(ne.getMessage() + " is not a numeric value.");
        System.exit(0);
    }
}

```

## Number Methods:

Here is the list of the instance methods that all the subclasses of the Number class implement:

SN	Methods with Description
1	<a href="#">xxxValue()</a> Converts the value of <i>this</i> Number object to the xxx data type and returned it.
2	<a href="#">compareTo()</a> Compares <i>this</i> Number object to the argument.
3	<a href="#">equals()</a> Determines whether <i>this</i> number object is equal to the argument.
4	<a href="#">valueOf()</a> Returns an Integer object holding the value of the specified primitive.
5	<a href="#">toString()</a> Returns a String object representing the value of specified int or Integer.
6	<a href="#">parseInt()</a>

	This method is used to get the primitive data type of a certain String.
7	<a href="#"><u>abs()</u></a> Returns the absolute value of the argument.
8	<a href="#"><u>ceil()</u></a> Returns the smallest integer that is greater than or equal to the argument. Returned as a double.
9	<a href="#"><u>floor()</u></a> Returns the largest integer that is less than or equal to the argument. Returned as a double.
10	<a href="#"><u>rint()</u></a> Returns the integer that is closest in value to the argument. Returned as a double.
11	<a href="#"><u>round()</u></a> Returns the closest long or int, as indicated by the method's return type, to the argument.
12	<a href="#"><u>min()</u></a> Returns the smaller of the two arguments.
13	<a href="#"><u>max()</u></a> Returns the larger of the two arguments.
14	<a href="#"><u>exp()</u></a> Returns the base of the natural logarithms, e, to the power of the argument.
15	<a href="#"><u>log()</u></a> Returns the natural logarithm of the argument.
16	<a href="#"><u>pow()</u></a> Returns the value of the first argument raised to the power of the second argument.
17	<a href="#"><u>sqrt()</u></a> Returns the square root of the argument.
18	<a href="#"><u>sin()</u></a> Returns the sine of the specified double value.
19	<a href="#"><u>cos()</u></a> Returns the cosine of the specified double value.
20	<a href="#"><u>tan()</u></a> Returns the tangent of the specified double value.
21	<a href="#"><u>asin()</u></a> Returns the arcsine of the specified double value.
22	<a href="#"><u>acos()</u></a> Returns the arccosine of the specified double value.
23	<a href="#"><u>atan()</u></a> Returns the arctangent of the specified double value.
24	<a href="#"><u>atan2()</u></a> Converts rectangular coordinates (x, y) to polar coordinate (r, theta) and returns theta.
25	<a href="#"><u>toDegrees()</u></a> Converts the argument to degrees
26	<a href="#"><u>toRadians()</u></a> Converts the argument to radians.
27	<a href="#"><u>random()</u></a> Returns a random number.



### Character Methods:

Here is the list of the important instance methods that all the subclasses of the Character class implement:

SN	Methods with Description
1	<a href="#">isLetter()</a> Determines whether the specified char value is a letter.
2	<a href="#">isDigit()</a> Determines whether the specified char value is a digit.
3	<a href="#">isWhitespace()</a> Determines whether the specified char value is white space.
4	<a href="#">isUpperCase()</a> Determines whether the specified char value is uppercase.
5	<a href="#">isLowerCase()</a> Determines whether the specified char value is lowercase.
6	<a href="#">toUpperCase()</a> Returns the uppercase form of the specified char value.
7	<a href="#">toLowerCase()</a> Returns the lowercase form of the specified char value.
8	<a href="#">toString()</a> Returns a String object representing the specified character value that is, a one-character string.

### String Methods:

Here is the list methods supported by String class:

SN	Methods with Description
1	<a href="#">char charAt(int index)</a> Returns the character at the specified index.
2	<a href="#">int compareTo(Object o)</a> Compares this String to another Object.
3	<a href="#">int compareTo(String anotherString)</a> Compares two strings lexicographically.
4	<a href="#">int compareToIgnoreCase(String str)</a> Compares two strings lexicographically, ignoring case differences.
5	<a href="#">String concat(String str)</a> Concatenates the specified string to the end of this string.
6	<a href="#">boolean contentEquals(StringBuffer sb)</a> Returns true if and only if this String represents the same sequence of characters as the specified StringBuffer.
7	<a href="#">static String copyValueOf(char[] data)</a> Returns a String that represents the character sequence in the array specified.
8	<a href="#">static String copyValueOf(char[] data, int offset, int count)</a> Returns a String that represents the character sequence in the array specified.

9	<a href="#"><code>boolean endsWith(String suffix)</code></a> Tests if this string ends with the specified suffix.
10	<a href="#"><code>boolean equals(Object anObject)</code></a> Compares this string to the specified object.
11	<a href="#"><code>boolean equalsIgnoreCase(String anotherString)</code></a> Compares this String to another String, ignoring case considerations.
12	<a href="#"><code>byte getBytes()</code></a> Encodes this String into a sequence of bytes using the platform's default charset, storing the result into a new byte array.
13	<a href="#"><code>byte[] getBytes(String charsetName)</code></a> Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array.
14	<a href="#"><code>void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)</code></a> Copies characters from this string into the destination character array.
15	<a href="#"><code>int hashCode()</code></a> Returns a hash code for this string.
16	<a href="#"><code>int indexOf(int ch)</code></a> Returns the index within this string of the first occurrence of the specified character.
17	<a href="#"><code>int indexOf(int ch, int fromIndex)</code></a> Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
18	<a href="#"><code>int indexOf(String str)</code></a> Returns the index within this string of the first occurrence of the specified substring.
19	<a href="#"><code>int indexOf(String str, int fromIndex)</code></a> Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
20	<a href="#"><code>String intern()</code></a> Returns a canonical representation for the string object.
21	<a href="#"><code>int lastIndexOf(int ch)</code></a> Returns the index within this string of the last occurrence of the specified character.
22	<a href="#"><code>int lastIndexOf(int ch, int fromIndex)</code></a> Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
23	<a href="#"><code>int lastIndexOf(String str)</code></a> Returns the index within this string of the rightmost occurrence of the specified substring.
24	<a href="#"><code>int lastIndexOf(String str, int fromIndex)</code></a> Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.
25	<a href="#"><code>int length()</code></a> Returns the length of this string.
26	<a href="#"><code>boolean matches(String regex)</code></a> Tells whether or not this string matches the given regular expression.
27	<a href="#"><code>boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)</code></a> Tests if two string regions are equal.
28	<a href="#"><code>boolean regionMatches(int toffset, String other, int ooffset, int len)</code></a> Tests if two string regions are equal.

29	<a href="#"><u>String replace(char oldChar, char newChar)</u></a> Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.
30	<a href="#"><u>String replaceAll(String regex, String replacement)</u></a> Replaces each substring of this string that matches the given regular expression with the given replacement.
31	<a href="#"><u>String replaceFirst(String regex, String replacement)</u></a> Replaces the first substring of this string that matches the given regular expression with the given replacement.
32	<a href="#"><u>String[] split(String regex)</u></a> Splits this string around matches of the given regular expression.
33	<a href="#"><u>String[] split(String regex, int limit)</u></a> Splits this string around matches of the given regular expression.
34	<a href="#"><u>boolean startsWith(String prefix)</u></a> Tests if this string starts with the specified prefix.
35	<a href="#"><u>boolean startsWith(String prefix, int toffset)</u></a> Tests if this string starts with the specified prefix beginning a specified index.
36	<a href="#"><u>CharSequence subSequence(int beginIndex, int endIndex)</u></a> Returns a new character sequence that is a subsequence of this sequence.
37	<a href="#"><u>String substring(int beginIndex)</u></a> Returns a new string that is a substring of this string.
38	<a href="#"><u>String substring(int beginIndex, int endIndex)</u></a> Returns a new string that is a substring of this string.
39	<a href="#"><u>char[] toCharArray()</u></a> Converts this string to a new character array.
40	<a href="#"><u>String toLowerCase()</u></a> Converts all of the characters in this String to lower case using the rules of the default locale.
41	<a href="#"><u>String toLowerCase(Locale locale)</u></a> Converts all of the characters in this String to lower case using the rules of the given Locale.
42	<a href="#"><u>String toString()</u></a> This object (which is already a string!) is itself returned.
43	<a href="#"><u>String toUpperCase()</u></a> Converts all of the characters in this String to upper case using the rules of the default locale.
44	<a href="#"><u>String toUpperCase(Locale locale)</u></a> Converts all of the characters in this String to upper case using the rules of the given Locale.
45	<a href="#"><u>String trim()</u></a> Returns a copy of the string, with leading and trailing whitespace omitted.
46	<a href="#"><u>static String valueOf(primitive data type x)</u></a> Returns the string representation of the passed data type argument.

### Arrays :

Java provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables.

This tutorial introduces how to declare array variables, create arrays, and process arrays using indexed variables.

### Declaring Array Variables:

To use an array in a program, you must declare a variable to reference the array, and you must specify the type of array the variable can reference. Here is the syntax for declaring an array variable:

```
dataType[] arrayRefVar;    // preferred way.  
  
or  
  
dataType arrayRefVar[];    // works but not preferred way.
```

Note: The style `dataType[] arrayRefVar` is preferred. The style `dataType arrayRefVar[]` comes from the C/C++ language and was adopted in Java to accommodate C/C++ programmers.

### Example:

The following code snippets are examples of this syntax:

```
double[] myList;           // preferred way.  
  
or  
  
double myList[];           // works but not preferred way.
```

### Creating Arrays:

You can create an array by using the new operator with the following syntax:

```
arrayRefVar = new dataType[arraySize];
```

The above statement does two things:

- It creates an array using `new dataType[arraySize]`;
- It assigns the reference of the newly created array to the variable `arrayRefVar`.

Declaring an array variable, creating an array, and assigning the reference of the array to the variable can be combined in one statement, as shown below:

```
dataType[] arrayRefVar = new dataType[arraySize];
```

Alternatively you can create arrays as follows:

```
dataType[] arrayRefVar = {value0, value1, ..., valuek};
```

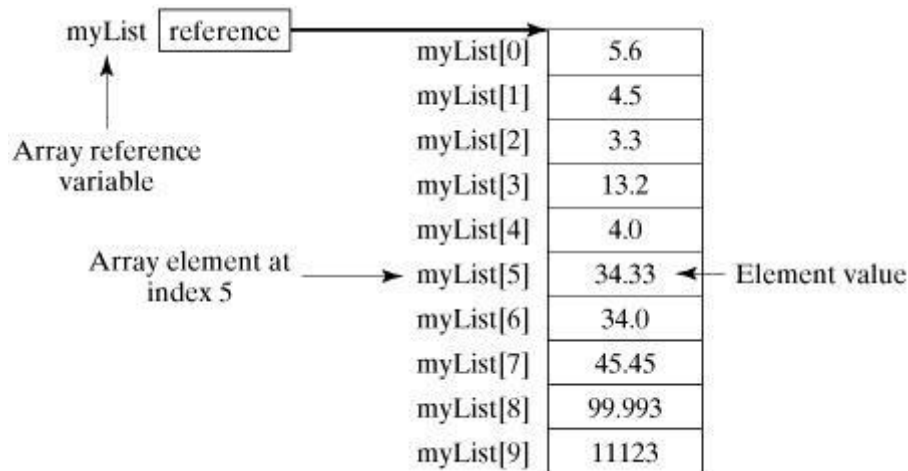
The array elements are accessed through the index. Array indices are 0-based; that is, they start from 0 to `arrayRefVar.length-1`.

### Example:

Following statement declares an array variable, `myList`, creates an array of 10 elements of double type, and assigns its reference to `myList`:

```
double[] myList = new double[10];
```

Following picture represents array myList. Here myList holds ten double values and the indices are from 0 to 9.



### Processing Arrays:

When processing array elements, we often use either for loop or foreach loop because all of the elements in an array are of the same type and the size of the array is known.

### Example:

Here is a complete example of showing how to create, initialize and process arrays:

```
public class TestArray {
    public static void main(String[] args) {
        double[] myList = {1.9, 2.9, 3.4, 3.5};

        // Print all the array elements
        for (int i = 0; i < myList.length; i++) {
            System.out.println(myList[i] + " ");
        }
        // Summing all elements
        double total = 0;
        for (int i = 0; i < myList.length; i++) {
            total += myList[i];
        }
        System.out.println("Total is " + total);
        // Finding the largest element
        double max = myList[0];
        for (int i = 1; i < myList.length; i++) {
            if (myList[i] > max) max = myList[i];
        }
        System.out.println("Max is " + max);
    }
}
```

This would produce following result:

```
1.9
2.9
3.4
```

```
3.5
Total is 11.7
Max is 3.5
```

### The foreach Loops:

JDK 1.5 introduced a new for loop, known as foreach loop or enhanced for loop, which enables you to traverse the complete array sequentially without using an index variable.

#### Example:

The following code displays all the elements in the array myList:

```
public class TestArray {
    public static void main(String[] args) {
        double[] myList = {1.9, 2.9, 3.4, 3.5};

        // Print all the array elements
        for (double element: myList) {
            System.out.println(element);
        }
    }
}
```

This would produce following result:

```
1.9
2.9
3.4
3.5
```

### Java Data & Time

Java provides the **Date** class available in **java.util** package, this class encapsulates the current date and time.

The Date class supports two constructors. The first constructor initializes the object with the current date and time.

```
Date( )
```

### Getting Current Date & Time

This is very easy to get current date and time in Java. You can use a simple Date object with toString() method to print current date and time as follows:

```
import java.util.Date;

class DateDemo {
    public static void main(String args[]) {
        // Instantiate a Date object
        Date date = new Date();

        // display time and date using toString()
        System.out.println(date.toString());
    }
}
```

This would produce following result:

Mon May 04 09:51:52 CDT 2009

### Methods:

```
Public class methods
{
Public void method1()
{
System.out.println("method");
}
Public void method(int a)
{
System.out.println(a);
}
Public void method2(int a,int b)
{
System.out.println(a+b);
}
Public static void main(string args[])
{
    Int a=10;
    Int b=20;
    //create instance for methods
    Method obj = new Methods();
    Obj.method1();
    Obj.method1(a);
    Obj.method2(a,b);
}
```

### Inheritance

Inheritance is the mechanism through which we can derive classes from other classes. The derived class is called as child class or the subclass or we can say the extended class and the class from which we are deriving the subclass are called the base class or the parent class. To derive a class in java the keyword **extends** is used.

The following kinds of inheritance are there in java.

- Simple Inheritance
- Multilevel Inheritance
- Java does not support multiple Inheritance

#### Pictorial Representation of Simple and Multilevel Inheritance



## Simple Inheritance

## Multilevel Inheritance

### Simple Inheritance

When a subclass is derived simply from its parent class then this mechanism is known as simple inheritance. In case of simple inheritance there is only a sub class and its parent class. It is also called single inheritance or one level inheritance.

eg.

```
class D
{
    public void test()
    {
        System.out.println("Hai...");
    }
}

public class SingleInheritance extends D
{
    public static void main(String[] args)
    {
        SingleInheritance s= new SingleInheritance();
        s.test();
    }
}
```

Eg2:

```
class A
{
    protected int a;
    protected int b;
    public void method1()
    {
    }
}

class B extends A
{
    private int c;
    public void method2()
    {
    }
}
```

### Multilevel Inheritance

It is the enhancement of the concept of inheritance. When a subclass is derived from a derived class then this mechanism is known as the multilevel inheritance. The derived class is called the subclass or child class for it's parent class and this parent class works as the child class for it's just above ( parent ) class. Multilevel inheritance can go up to any number of level.

e.g.

```
//package com.nisum;
class A {
    int x;
    int y;
    int get(int p, int q) {
        x=p; y=q; return(0);
    }
    void Show() {
        System.out.println(x+y);
    }
}
```



```

    }
    class B extends A{
        static int a=10;
        static int b=20;
        void Showb() {

            System.out.println("showb method is executed");
            System.out.println(x+y);
        }
    }

    public class MultiLevel extends B{
        public static void display(){
            System.out.println("display method executed");
            System.out.println(a+b);
        }
        public static void main(String args[]){
            MultiLevel a = new MultiLevel();
            a.get(5,6);
            a.Show();
            a.Showb();
            // display();
            a.display();
        }
    }

```

**Polymorphism :** It means one name with many forms.

These are 2 types :

- 1) Method Over loading
- 2) Method Over riding

**Method Overloading** (Writing two or more methods in the same class in such way that each method has same name but with different method signatures)

As we know that a method has its own signature which is known by method's name and the parameter types. Java has a powerful feature which is known as method overloading. With the help of this feature we can define two methods of same name with different parameters. It allows the facility to define that how we perform the same action on different type of parameter. Here is an example of the method overloading. Create a class "mol" and subclass "Test". Now we create two methods of same name "square" but the parameter are different. When the main class call the method the method executes on behalf of the parameter. In the given example we are trying to illustrate the same technique.

**Here is the Code of the Example:**

```

public class OverLoad
{
    public static void main(String args[])
    {
        //create sample class object
        Sample s = new Sample();
        //Call add() and pass two values
        s.add(10,15); //This call is bound with first method
        //call add() and pass three values
        s.add(10,15,20); //this is bound with second method
    }
}

```

```

class Sample
{
    //method to add two values
    public void add(int a,int b)
    {
        System.out.println("Sum of two="+ (a+b));
    }
    public void add(int a,int b,int c)
    {
        System.out.println("Sum of three="+ (a+b+c));
    }
}

```

### Example 2:

```

package com;
import java.lang.*;

public class MethodOverloading
{
    public static void main(String args[])
    {
        Test a= new Test();
        //a.doOverLoad();
        System.out.println(a.square(2));
    }
}
class Test{
    public void doOverLoad() {
        int x = 7;
        double y = 11.4;
        System.out.println(square(x) + "\n" + square(y));
    }
    public int square(int y) {
        return y*y;
    }
    public double square(double y) {
        return y*y;
    }
}

```

**Overriding (Writing two or more methods in super and sub classes such that the methods have same name and same signature)**

Below example illustrates method overriding in java. Method overriding in java means a subclass method overriding a super class method. Super class method should be non-static. Subclass uses extends keyword to extend the super class. In the example class B is the sub class and class A is the super class. In overriding methods of both subclass and super class possess same signatures. Overriding is used in modifying the methods of the super class. In overriding return types and constructor parameters of methods should match.

```

class One
{
    //method to calculate square value
    void calculate(double x)
    {
        System.out.println("Square Value="+ (x*x));
    }
}

```

```

}
class Two extends One
{
    //method to calculate sqare root value
    void calculate(double x)
    {
        System.out.println("Square root="+Math.sqrt(x));
    }
}
class OverRide
{
    public static void main(String args[])
    {
        //create sub class object t
        Two t = new Two();
        //Call calculate() method using t
        t.calculate(6);
    }
}

```

### Example 2

```

class A {
    int i;
    A(int a, int b) {
        i = a+b;
    }
    void add() {
        System.out.println("Sum of a and b is: " + i);
    }
}
class B extends A {
    int j;
    B(int a, int b, int c) {
        super(a, b);
        j = a+b+c;
    }
    void add() {
        super.add();
        System.out.println("Sum of a, b and c is: " + j);
    }
}
class MethodOverriding {
    public static void main(String args[]) {
        B b = new B(10, 20, 30);
        b.add();
    }
}

```

**Abstraction:** “Data abstraction is a mechanism of retrieving the essential details without dealing with background details”.

To use use abstraction we need use abstract keyword in the class

```

class Bank
{
    private int accno =123;
    private String name ="Suresh";
    private float balance = 2000 ;
    private float profit;
    private float loan;
    public void displayclerk()
    {

```

```

        System.out.println("Accno="+ accno);
        System.out.println("Name="+name);
        System.out.println("Balance="+balance);

    }

    public static void main(String args[])
    {
        Bank b=new Bank();
        b.displayclerk();

    }
}

```

### Example2

//All the objects need different implementations of the same method  
 abstract class Myclass

```

{
    //this is abstract method
    abstract void calculate(double x);
}
class sub1 extends Myclass
{
    //calculate square value
    void calculate(double x)
    {
        System.out.println("square="+x*x);
    }
}
class sub2 extends Myclass
{
    //calculate sqare root value
    void calculate(double x)
    {
        System.out.println("Square root =" +Math.sqrt(x));
    }
}
class sub3 extends Myclass
{
    //calculate cube value
    void calculate(double x)
    {
        System.out.println("cube="+x*x*x);
    }
}
class Diffirent
{
    public static void main(String args[])
    {
        //create sub class objects
        Sub1 obj1 = new Sub1();
        Sub2 obj2 = new Sub2();
        Sub3 obj3 = new Sub3();
        //let the objects call and use calculate() method
        obj1.calculate(3); //claculate sqare
        obj2.calculate(4); // CALCULATE SQUARE ROOT;
        obj3.calculate(5); //calculate cube value
    }
}

```

```
}  
}
```

**Encapsulation**-Encapsulation is a process of binding or wrapping the data and the codes that operates on the data into a single entity. This keeps the data safe from outside interface and misuse. One way to think about encapsulation is as a protective wrapper that prevents code and data from being arbitrarily accessed by other code defined outside the wrapper.

In other words, encapsulation is the ability of an object to be a container (or capsule) for related properties (ie. data variables) and methods (ie. functions).

Ex:

```
class Person  
{  
    //variable - data  
    private String name = "Suresh";  
    private int age = 26;  
    //method  
    public void talk()  
    {  
        System.out.println("Hello ,Iam"+name);  
        System.out.println("My age is"+age);  
    }  
    public static void main(String args[]){  
        Person p = new Person();  
        p.talk();  
    }  
}
```

**Package:**

This section will be covered in Selenium RC

## Java - Exceptions Handling

An exception is a problem that arises during the execution of a program. An exception can occur for many different reasons, including the following:

- A user has entered invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications, or the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

To understand how exception handling works in Java, you need to understand the three categories of exceptions:

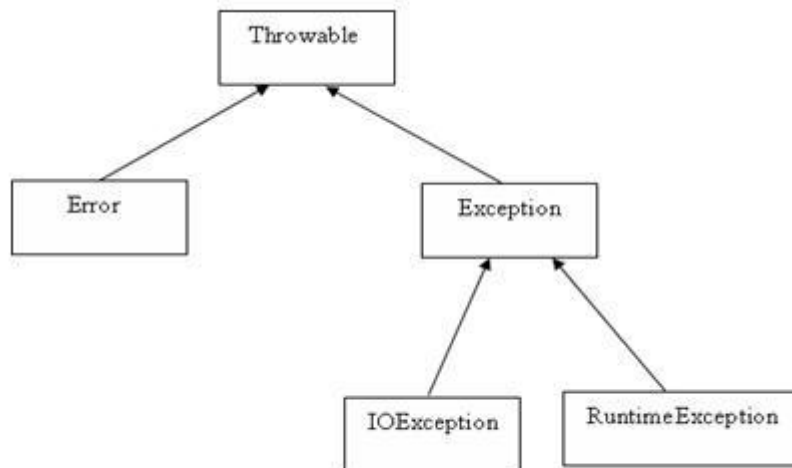
- **Checked exceptions:** A checked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.
- **Runtime exceptions:** A runtime exception is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, runtime exceptions are ignored at the time of compilation.
- **Errors:** These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

## Exception Hierarchy:

All exception classes are subtypes of the `java.lang.Exception` class. The exception class is a subclass of the `Throwable` class. Other than the exception class there is another subclass called `Error` which is derived from the `Throwable` class.

Errors are not normally trapped from the Java programs. These conditions normally happen in case of severe failures, which are not handled by the Java programs. Errors are generated to indicate errors generated by the runtime environment. Example : JVM is out of Memory. Normally programs cannot recover from errors.

The `Exception` class has two main subclasses : `IOException` class and `RuntimeException` Class.



## Catching Exceptions:

A method catches an exception using a combination of the `try` and `catch` keywords. A `try/catch` block is placed around the code that might generate an exception. Code within a `try/catch` block is referred to as protected code, and the syntax for using `try/catch` looks like the following:

```
try
{
    //Protected code
} catch (ExceptionName e1)
{
    //Catch block
}
```

A `catch` statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code, the `catch` block (or blocks) that follows the `try` is checked. If the type of exception that occurred is listed in a `catch` block, the exception is passed to the `catch` block much as an argument is passed into a method parameter.

## Example:

The following is an array is declared with 2 elements. Then the code tries to access the 3rd element of the array which throws an exception.

```
// File Name : ExcepTest.java
import java.io.*;
public class ExcepTest{

    public static void main(String args[]){
        try{
            int a[] = new int[2];
            System.out.println("Access element three :" + a[3]);
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Exception thrown  :" + e);
        }
        System.out.println("Out of the block");
    }
}
```

This would produce following result:

```
Exception thrown  :java.lang.ArrayIndexOutOfBoundsException: 3
Out of the block
```

### Multiple catch Blocks:

A try block can be followed by multiple catch blocks. The syntax for multiple catch blocks looks like the following:

```
try
{
    //Protected code
}catch(ExceptionType1 e1)
{
    //Catch block
}catch(ExceptionType2 e2)
{
    //Catch block
}catch(ExceptionType3 e3)
{
    //Catch block
}
```

The previous statements demonstrate three catch blocks, but you can have any number of them after a single try. If an exception occurs in the protected code, the exception is thrown to the first catch block in the list. If the data type of the exception thrown matches ExceptionType1, it gets caught there. If not, the exception passes down to the second catch statement. This continues until the exception either is caught or falls through all catches, in which case the current method stops execution and the exception is thrown down to the previous method on the call stack.

### Example:

Here is code segment showing how to use multiple try/catch statements.

```
class ex
```

```

{
    public static void main(String[] args)
    {
        try
        {
            //open the files
            System.out.println("open files");
            //do some processing
            int n = 0;
            //System.out.println("n= "+ n);
            int a = 45/n;
            System.out.println("a= "+ a);

            int b[] = {10,20,30};
            b[50] = 100;
        }
        catch (ArithmeticException ae)
        {
            //display the exception details
            System.out.println(ae);
            //display any message to the user
            System.out.println("Please pass data while running this program");
        }
        catch(ArrayIndexOutOfBoundsException aie)
        {
            //display exception details
            aie.printStackTrace();
            //display a message to user
            System.out.println("please see that the array index is within the range");
        }
    }
}

```

### The throws/throw Keywords:

If a method does not handle a checked exception, the method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature.

You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the throw keyword. Try to understand the difference in throws and throw keywords.

The following method declares that it throws a Remote Exception:

```

import java.io.*;
public class className
{
    public void deposit(double amount) throws RemoteException
    {
        // Method implementation
        throw new RemoteException();
    }
    //Remainder of class definition
}

```



A method can declare that it throws more than one exception, in which case the exceptions are declared in a list separated by commas. For example, the following method declares that it throws a Remote Exception and an InsufficientFundsException:

```
import java.io.*;
public class className
{
    public void withdraw(double amount) throws RemoteException,
                                   InsufficientFundsException
    {
        // Method implementation
    }
    //Remainder of class definition
}
```

### The finally Keyword

The finally keyword is used to create a block of code that follows a try block. A finally block of code always executes, whether or not an exception has occurred.

Using a finally block allows you to run any cleanup-type statements that you want to execute, no matter what happens in the protected code.

A finally block appears at the end of the catch blocks and has the following syntax:

```
try
{
    //Protected code
}catch(ExceptionType1 e1)
{
    //Catch block
}catch(ExceptionType2 e2)
{
    //Catch block
}catch(ExceptionType3 e3)
{
    //Catch block
}finally
{
    //The finally block always executes.
}
```

### Example:

```
public class Final{

    public static void main(String args[]){
        int a[] = new int[2];
        try{
            System.out.println("Access element three :" + a[3]);
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Exception thrown  :" + e);
        }
        finally{
            a[0] = 6;
            System.out.println("First element value: " +a[0]);
        }
    }
}
```

```
        System.out.println("The finally statement is executed");
    }
}
```

This would produce following result:

```
Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 3
First element value: 6
The finally statement is executed
```

Note the followings:

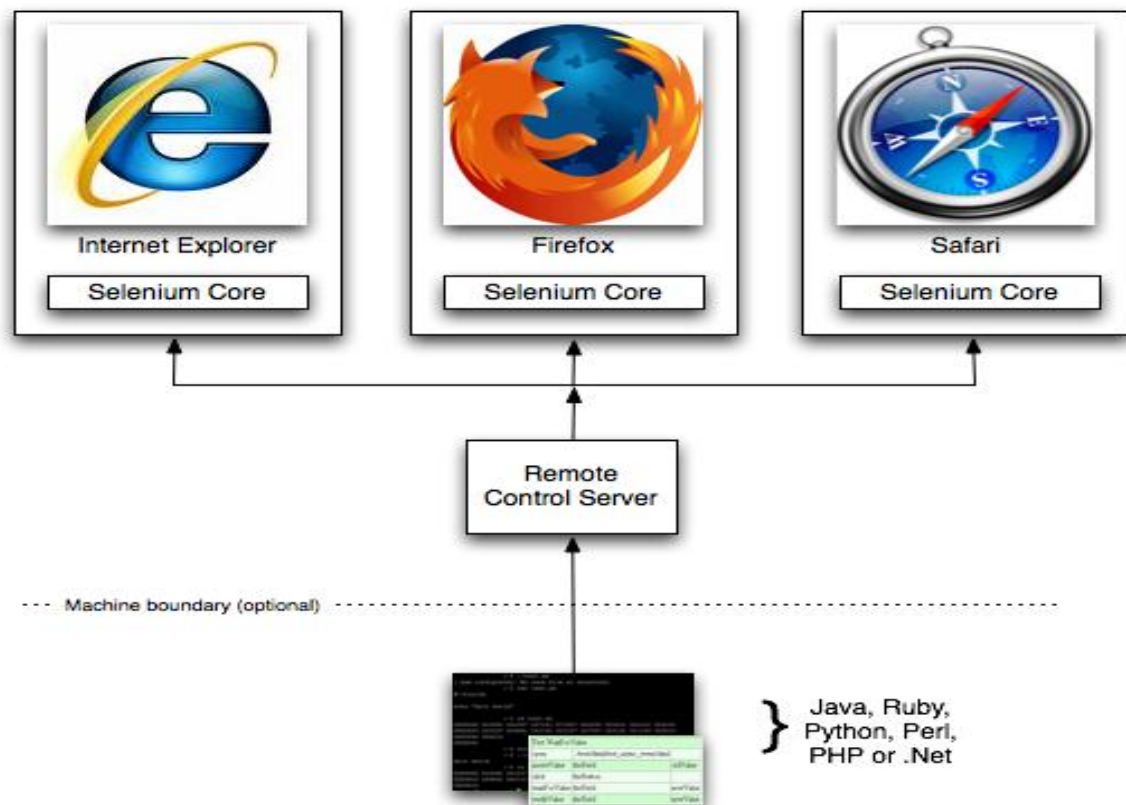
- A catch clause cannot exist without a try statement.
- It is not compulsory to have finally clauses whenever a try/catch block is present.
- The try block cannot be present without either catch clause or finally clause.
- Any code cannot be present in between the try, catch, finally blocks.

## Selenium RC

- Selenium Remote Control is a test tool that allows you to write automated web application in any programming language (Java, .NET, Perl, Python, and Ruby, PHP)
- Selenium Remote Control provides a Selenium Server, which can automatically start/stop/control, any supported browser.
- The Selenium Server communicates directly with the browser using AJAX.
- You can send commands directly to the Server using simple HTTP GET/POST requests;
- Finally, the Selenium Server acts as a client-configured HTTP proxy, to stand in between the browser and your website.
- Selenium RC is the solution for tests that need more than simple browser actions and linear execution
- Selenium RC uses the full power of programming language to create more complex tests like reading and writing files ,querying a database and emailing test results.
- Selenium IDE does not directly support
  - Condition stmt
  - Iteration
  - Logging and reporting of test results
  - Error handling ,particularly unexpected errors
  - Database testing
  - Test case grouping
  - Re-execution of failed tests
  - Test case dependency
  - Screen shot capture of tests failures.
- Although these tasks are not supported by selenium directly, all of them can be achieved by using programming techniques with a language-specific selenium-RC client library.

## Selenium RC –architecture

Windows, Linux, or Mac (as appropriate)...



## RC-Components

- The selenium server which launches and kills browsers, interprets and runs the selenes commands passed from the test program and acts as an HTTP proxy, intercepting and verifying HTTP messages passed between the browser and the AUT.
- Client libraries which provide the interface between each programming language and the selenium-RC server.

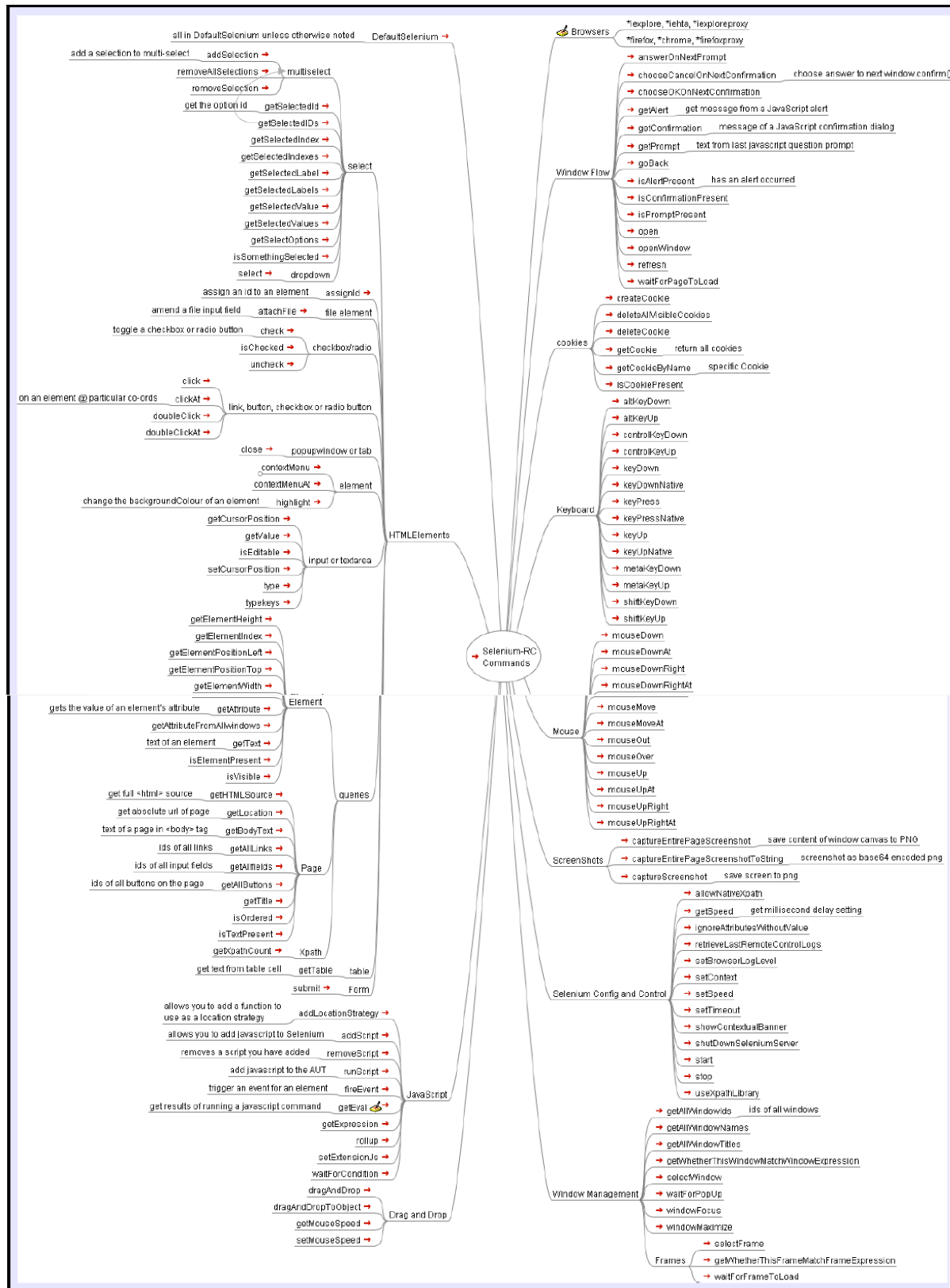
## Selenium –Server

Selenium server will start a browser and kill browser. Selenium server receives selenium commands and it reports back to your program the results of running those tests.

## Client Libraries

- The client libraries provide the programming support that allows you to run selenium commands from a program of your own design
- There is a different client library for each supported language
- A selenium client library provided a programming interface(API)

# Selenium API MindMap



## Installation

- Download selenium server Zip
- extract
- Install JRE(JDK 1.5 ,jse)
- Verify Java version(it should be jdk1.5 or later-to check java installed or not in command prompt type: cd:\ java -version
- open command prompt
- Type cd:\
- type C:\cd Selenium Rc
- type C:\seleniumRC>cd selenium Server
- type C:\seleniumRC>Seleniumserver>Java -jar selenium-server.jar
- Then it will start the selenium server

## Changing the port number

>Java -jar selenium-server.jar -port 1212

To shutdown server: ctrl +C

## How to run selenium RC HTML suits

- we are going to use the HTML suite command of the selenium RC
- The command should look like java -jar selenium-server.jar -htmlsuite<browser><url><path to test suite><where to store the results>

## Creating Batch file

- open notepad /WordPad
- Cut and paste the below code
- Save the batch file name "TestCase.bat"

echo@OFF

```
java -jar C:\seleniumRC\JavaServer\Selenium-server.jar -htmlsuite "**chrome"
```

pause

## Selenium .jar File

- jar is nothing but java archive file
- one jar will have n number of package
- Selenium server also have n number of packages

## Major are 2 packages

- org.openqa.selenium.server.\* -server
- com.thoughtworks.selenium.\* -client

## Navigation to add .jar files to project:

- Create new project
- Right click on project
- Select Build Path→Configure build path
- Click on libraries tab
- click on Add external jars button(selenium-server.jar) -we have to add this .jar file to libraries

E.g.:

## TC-Selenium RC basic program

```
import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;
public class basicRCprog
```

```

{
    public static void main(String args[]) throws Exception
    {
        SeleniumServer objserver=new SeleniumServer();
        Selenium objselenium=new
DefaultSelenium("localhost",4444,"*firefox","http://www.gmail.com");
        objserver.start();
        objselenium.start();
        objselenium.windowMaximize();
        objselenium.open("http://127.0.0.1/livehrm-2.5.0.2/login.php");
        objselenium.type("txtUserName","rajesh");
        objselenium.type("txtPassword","suresh123");
        objselenium.click("Submit");
        objselenium.waitForPageToLoad("30000");
        objselenium.click("link=Logout");
        objselenium.stop();
        objserver.stop();
    }
}

```

### Changing the server port number in RC

```

import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;
public class PortChange
{
    public static void main(String args[])throws Exception
    {
        RemoteControlConfiguration rc = new RemoteControlConfiguration();
        rc.setPort(5555);
        SeleniumServer objserver=new SeleniumServer(rc);
        Selenium objselenium=new DefaultSelenium("localhost",5555,"*firefox","http://127.0.0.1");
        objserver.start();
        objselenium.start();
        objselenium.windowMaximize();
        objselenium.open("/orangehrm-2.6/login.php");
        objselenium.type("txtUserName","suresh");
        objselenium.type("txtPassword","suresh123");
        objselenium.click("Submit");
        objselenium.waitForPageToLoad("20000");
        objselenium.click("link=Logout");
        objselenium.stop();
        objserver.stop();
    }
}

```

### TC-VerifyAdminPage

```

import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;
public class verifyadminpage {
    public static void main(String args[])throws Exception
    {
        SeleniumServer objserver=new SeleniumServer();
        Selenium objselenium=new
DefaultSelenium("localhost",4444,"*firefox","http://127.0.0.1");
        objserver.start();
        objselenium.start();
        objselenium.windowMaximize();
        objselenium.open("/livehrm-2.5.0.2/login.php");
        if(objselenium.getTitle().equals("LiveHRM - New Level of HR Management"))
        {

```

```

        System.out.println("Home page Displayed Successfully");
    }
    else
    {
        System.out.println("Do not Displayed");
        return;
    }
    objselenium.type("txtUserName", "rajesh");
    objselenium.type("txtPassword", "suresh123");
    objselenium.click("Submit");
    objselenium.waitForPageToLoad("30000");
    //Verify Admin page
    if(objselenium.getTitle().equals("LiveHRM"))
    {
        System.out.println("Admin page Displayed Successfully");
    }
    else
    {
        System.out.println("Do not Displayed");
    }

    //verify text present
    if(objselenium.isTextPresent("Welcome rajesh"))
    {
        System.out.println("Welcome rajesh Displayed Successfully");
    }
    else
    {
        System.out.println("Do not Displayed");
    }

    objselenium.click("link=Logout");
    objselenium.stop();
    objserver.stop();
}

}

TC-Add Employee
import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;
public class tc2_AddEmpverify {
    static String Url="http://127.0.0.1";
    static String Un="rajesh";
    static String Pwd="suresh123";
    public static void main(String args[])throws Exception
    {
        SeleniumServer objserver=new SeleniumServer();
        Selenium objselenium=new DefaultSelenium("localhost",4444,"*firefox",Url);
        objserver.start();
        objselenium.start();
        objselenium.windowMaximize();
        objselenium.open("/livehrm-2.5.0.2/login.php");
        if(objselenium.getTitle().equals("LiveHRM - New Level of HR Management"))
        {
            System.out.println("Home page Displayed Successfully");
        }
        else

```

```

{
    System.out.println("Do not Displayed");
    return;
}
objselenium.type("txtUserName", Un);
objselenium.type("txtPassword", Pwd);
objselenium.click("Submit");
objselenium.waitForPageToLoad("30000");
if(objselenium.getTitle().equals("LiveHRM"))
{
    System.out.println("Admin page Displayed Successfully");
}
else
{
    System.out.println("Do not Displayed");
}

//verify text present
if(objselenium.isTextPresent("Welcome"+" "+Un))
{
    System.out.println("Welcome"+" "+Un+" Displayed Successfully");
}
else
{
    System.out.println("Do not Displayed");
}

objselenium.click("link=Add Employee");
objselenium.waitForFrameToLoad("rightMenu", "20000");
//verify text
String text=objselenium.getText("//form[@id='frmEmp']/div/div/div[2]/div/h2");
if(text.equals("PIM : Add Employee"))
{
    System.out.println(text+"Displayed Successfully");
}
else
{
    System.out.println("DO not Displayed");
}

//select frame
objselenium.selectFrame("rightMenu");
objselenium.type("txtEmpLastName",Un);
objselenium.type("name=txtEmpFirstName", Pwd);
objselenium.click("btnEdit");
Thread.sleep(5000);
objselenium.selectFrame("relative=up");
//click on emplist
objselenium.click("link=Employee List");
Thread.sleep(5000);
//verify emplist
String Emplist=objselenium.getText("//form[@id='standardView']/div/h2");
if(Emplist.equals("Employee Information"))
{
    System.out.println(Emplist+"Displayed Successfully");
}
else
{
    System.out.println("DO not Displayed");
}

```



```

    }
    //select frame
    objselenium.selectFrame("rightMenu");
    //get row count
    int rc=Integer.parseInt(objselenium.getXpathCount("//table[@class='data-
table']/tbody/tr").toString());
    for(int i=1;i<rc;i++)
    {
        String Empname=objselenium.getText("//table[@class='data-
table']/tbody/tr["+i+"]/td[3]/a");
        if(Empname.equals(Un+" "+Pwd))
        {
            System.out.println(Empname+"Displayed successfully");
            System.out.println("Adding Employee is success with verification");
            break;
        }
    }
    objselenium.selectFrame("relative=up");
    objselenium.click("link=Logout");
    objselenium.stop();
    objserver.stop();

}

}

```

### TC-AnsweronNextPrompt

```

import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;

public class answeronnextprompt
{
    public static void main(String args[]) throws Exception
    {
        SeleniumServer objserver = new SeleniumServer();
        Selenium objselenium = new
        DefaultSelenium("localhost",4444,"*firefox","http://www.google.com/");
        objserver.start();
        objselenium.start();
        objselenium.windowMaximize();
        //open page
        objselenium.open("file:///D:/suresh/Livetech/suresh-
selenium/Selenium_Livetech_Softwares/HTML_Examples/answerOnNextPrompt.html.
html");
        String vName = "suresh";
        objselenium.answerOnNextPrompt(vName);
        objselenium.click("Link=Click here to enter your name");
        Thread.sleep(1000);
        if(objselenium.getPrompt().equals("Please enter your name. "))
        {
            System.out.println("Prompt displayed");
        }
        objselenium.stop();
        objserver.stop();

    }
}

```

### TC-Table

```
import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;

public class table
{
    public static void main(String args[]) throws Exception
    {
        SeleniumServer objserver = new SeleniumServer();
        Selenium objselenium = new
DefaultSelenium("localhost",4444,"*firefox","http://www.gmail.com/");
        objserver.start();
        objselenium.start();
        objselenium.windowMaximize();
        objselenium.open("file:///D:/suresh/Livetech/suresh-
selenium/Selenium_Livetech_Softwares/HTML_Examples/getXpathCountandVerifyTa
ble.html");
        Thread.sleep(2000);
        //verify title
        if(objselenium.getTitle().equals("AssertXpath"))
        {
            System.out.println("Title matched");
        }
        else
        {
            System.out.println("Title not matched");
        }
        //true native xpath
        objselenium.allowNativeXpath("true");
        //get row count
        int row =
Integer.parseInt(objselenium.getXpathCount("//table[@id='idcpurse']/tbody/t
r").toString());
        System.out.println(row);
        for(int i=1;i<=row;i++)
        {
            String data =
objselenium.getText("//table[@id='idcourse']/tbody/tr["+i+"]/td[2]");
            System.out.println(data);
        }
        objselenium.stop();
        objserver.stop();
    }
}
```

### Working with Excel Sheets:

```
import java.io.FileInputStream;
import jxl.*;
public class Excel
{
    public static void main(String args[])
    {
        FileInputStream f1=new FileInputStream("D:\\123.xlsx");
        workbook w1=workbook.getworkbook(f1);
        sheet s1=w1.getsheet(0);
        int i=1;
```

```

String gBrow=s1.getcell(0,i).getcontents();
String gURL=s1.getcell(0,i).getcontents();
String gUN=s1.getcell(0,i).getcontents();
String gPWD=s1.getcell(0,i).getcontents();
String gEMAIL=s1.getcell(0,i).getcontents();
String gPATH=s1.getcell(0,i).getcontents();

System.out.println(gBrow);
System.out.println(gURL);
System.out.println(gUN);
System.out.println(gPWD);
System.out.println(gEMAIL);
System.out.println(gPATH);
}
}

```

### Looping

```

import java.io.FileInputStream;
import jxl.*;
public class Excel
{
    public static void main(String args[])
    {
        FileInputStream f1=new FileInputStream("D:\\123.xlsx");
        workbook objwb=workbook.getworkbook(file);
        sheet s1=objwb.getsheet(0);
        //get row count
        int rows = s1.getRows();
        for(int i=1;i<rows;i++)
        {
            String un=s1.getCell(o,i).getContents();
            String pwd=s1.getCell(1,i).getContents();
            System.out.println(un+" "+pwd);
        }
    }
}

```

### Working with Data Base

#### We have 3 types of databases

- File Data Base –eg:foxpro ,ms-access
- Remote data base-SQL ,Oracle
- index data base-Internet DB

**Drivers:** It s a technology ,this driver has 3 components

- Connection
- Staments(commands)
- Results sets
- By using connection we can connect to the data source and we open database
- Create stmt for executing queries
- Commands are DML commands
- By using results set we can get data from tables.

### How to create DSN:

→Open Control Pane →Administrative Tools→data Sources→click on ADD→Select a driver ("msAccessDriver (\*.mdb,\*.accdb)→click on finish→give dataSourceName(dsn1)→click on select for database→browse create database→click on OK →click on ok

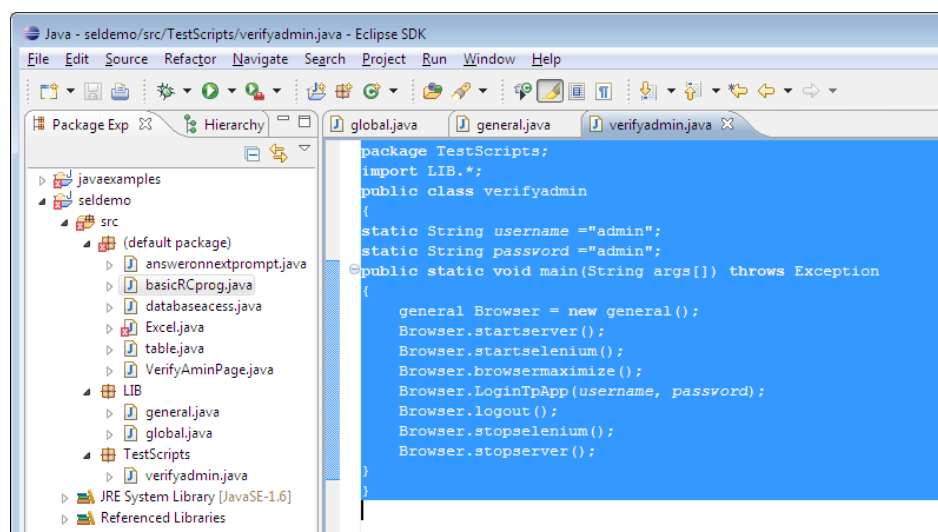
### TC-Creating the data base

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
class databaseaccess
{
    public static void main(String args[])
    {
        try
        {
            Connection con=DriverManager.getConnection("jdbc:odbc:dsn1");
            Statement st=con.createStatement();
            ResultSet rs=st.executeQuery("select * from emptable");
            while(rs.next())
            {
                System.out.println(rs.getString(2)+"\t"+rs.getString(3));
            }
            rs.close();
            st.close();
            con.close();
        }
        catch(Exception e)
        {
            System.out.println("Error:"+e);
        }
    }
}
```

### Package

Group of classes

Eg:



```
package LIB;
import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;
public class global
{

```

```

public SeleniumServer objserver;
public Selenium objselenium;
public static String gBrowser = "*firefox";
public static String gHost = "localhost";
public static int gPort = 4444;
public static String gURL = "http://127.0.0.1/orangehrm-2.5.0.2/login.php/";
}

package LIB;
import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;
public class general extends global
{
//start server
    public void startserver() throws Exception
    {
        objserver = new SeleniumServer();
        objserver.start();
    }
//stop server
    public void stopserver() throws Exception
    {
        objserver.stop();
    }
//start selenium
    public void startselenium()
    {
        objselenium = new DefaultSelenium(gHost, gPort, gBrowser, gURL);
        objselenium.start();
    }
//stop selenium
    public void stopselenium()
    {
        objselenium.stop();
    }
//maximize browser
    public void browsermaximize()
    {
        objselenium.windowMaximize();
    }
//login to the application
    public void LoginTpApp(String username, String password)
    {
        //open page
        objselenium.open("/orange");
        //type username and password
        objselenium.type("txtUserName", username);
        objselenium.type("txtPassword", password);
        //click on submit
        objselenium.click("Submit");
        //wait for page load
        objselenium.waitForPageToLoad("3000");
    }
//log out
    public void logout()
    {
        objselenium.selectFrame("relative=up");
        objselenium.click("Link=Logout");
    }
}

```

```

package TestScripts;
import LIB.*;
public class verifyadmin
{
    static String username ="rajesh";
    static String password ="suresh123";
    public static void main(String args[]) throws Exception
    {
        general Browser = new general();
        Browser.startserver();
        Browser.startselenium();
        Browser.browsermaximize();
        Browser.LoginTpApp(username, password);
        Browser.logout();
        Browser.stopselenium();
        Browser.stopserver();
    }
}

```

## TestNG ( <http://testng.org/doc/index.html>)

### Introduction

TestNG is a testing framework designed to simply a broad range of testing needs, from unit testing (testing a class in isolation of the others) to integration testing (testing entire systems made of several classes, several packages and even several external frameworks, such as application servers).

Writing a test is typically a three-step process:

- Write the business logic of your test and insert TestNG annotations in your code.
- Add the information about your test (e.g. the class name, the groups you wish to run, etc...) in a testng.xml file or in build.xml.
- Run TestNG.

The concepts used in this documentation are as follows:

- A suite is represented by one XML file. It can contain one or more tests and is defined by the <suite> tag.
- A test is represented by <test> and can contain one or more TestNG classes.
- A TestNG class is a Java class that contains at least one TestNG annotation. It is represented by the <class> tag and can contain one or more test methods.
- A test method is a Java method annotated by @Test in your source.

A TestNG test can be configured by @BeforeXXX and @AfterXXX annotations which allows to perform some Java logic before and after a certain point, these points being either of the items listed above.

<b>@BeforeSuite</b> <b>@AfterSuite</b> <b>@BeforeTest</b> <b>@AfterTest</b> <b>@BeforeGroups</b> <b>@AfterGroups</b>	<b>Configuration information for a TestNG class:</b>  <b>@BeforeSuite:</b> The annotated method will be run before all tests in this suite have run. <b>@AfterSuite:</b> The annotated method will be run after all tests in this suite have run.
---	--

<b>@BeforeClass</b> <b>@AfterClass</b> <b>@BeforeMethod</b> <b>@AfterMethod</b>	<b>@BeforeTest:</b> The annotated method will be run before any test method belonging to the classes inside the <test> tag is run. <b>@AfterTest:</b> The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run. <b>@BeforeGroups:</b> The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked. <b>@AfterGroups:</b> The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked. <b>@BeforeClass:</b> The annotated method will be run before the first test method in the current class is invoked. <b>@AfterClass:</b> The annotated method will be run after all the test methods in the current class have been run. <b>@BeforeMethod:</b> The annotated method will be run before each test method. <b>@AfterMethod:</b> The annotated method will be run after each test method.
alwaysRun	For before methods (beforeSuite, beforeTest, beforeTestClass and beforeTestMethod, but not beforeGroups): If set to true, this configuration method will be run regardless of what groups it belongs to. For after methods (afterSuite, afterClass, ...): If set to true, this configuration method will be run even if one or more methods invoked previously failed or was skipped.
dependsOnGroups	The list of groups this method depends on.
dependsOnMethods	The list of methods this method depends on.
enabled	Whether methods on this class/method are enabled.
groups	The list of groups this class/method belongs to.
inheritGroups	If true, this method will belong to groups specified in the @Test annotation at the class level.
<b>@DataProvider</b>	<b>Marks a method as supplying data for a test method. The annotated method must return an Object[][] where each Object[] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.</b>
name	The name of this data provider. If it's not supplied, the name of this data provider will automatically be set to the name of the method.
parallel	If set to true, tests generated using this data provider are run in parallel. Default value is false.
<b>@Factory</b>	<b>Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[].</b>
<b>@Listeners</b>	<b>Defines listeners on a test class.</b>
value	An array of classes that extend <code>org.testng.ITestNGListener</code> .
<b>@Parameters</b>	<b>Describes how to pass parameters to a @Test method.</b>
value	The list of variables used to fill the parameters of this method.
<b>@Test</b>	<b>Marks a class or a method as part of the test.</b>
alwaysRun	If set to true, this test method will always be run even if it depends on a method that failed.
dataProvider	The name of the data provider for this test method.

dataProviderClass	The class where to look for the data provider. If not specified, the data provider will be looked on the class of the current test method or one of its base classes. If this attribute is specified, the data provider method needs to be static on the specified class.
dependsOnGroups	The list of groups this method depends on.
dependsOnMethods	The list of methods this method depends on.
description	The description for this method.
enabled	Whether methods on this class/method are enabled.
expectedExceptions	The list of exceptions that a test method is expected to throw. If no exception or a different than one on this list is thrown, this test will be marked a failure.
groups	The list of groups this class/method belongs to.
invocationCount	The number of times this method should be invoked.
invocationTimeout	The maximum number of milliseconds this test should take for the cumulated time of all the invocationcounts. This attribute will be ignored if invocationCount is not specified.
priority	The priority for this test method. Lower priorities will be scheduled first.
successPercentage	The percentage of success expected from this method
singleThreaded	If set to true, all the methods on this test class are guaranteed to run in the same thread, even if the tests are currently being run with parallel="methods". This attribute can only be used at the class level and it will be ignored if used at the method level. Note: this attribute used to be called <code>sequential</code> (now deprecated).
timeout	The maximum number of milliseconds this test should take.
threadPoolSize	The size of the thread pool for this method. The method will be invoked from multiple threads as specified by invocationCount. Note: this attribute is ignored if invocationCount is not specified

#### Installation:

- Select menu item help in eclipse
- Select install new software
- Click on Add
- Enter name & location (<http://beust.com/eclipse>)-this url id from Testng.org site
- Click on OK
- Select TestNG from work with dropdown
- Select TestNG checkbox (high-level check box)
- Click on Next
- Accept license
- Click on install
- Restart eclipse

#### Step 2: Installing .Jar files for TestNG

- Go to TestNG.org site
- Click on downloads
- Click on "You can download the current release version of TestNG here".
- Add the .jar files to the project

#### Type the example here

```
import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;
import org.testng.annotations.*;
```

```
public class Testng {
```



```

public SeleniumServer objserver;
public Selenium objselenium;
static String username="rajesh";
static String password="suresh123";
@BeforeClass
public void setup()throws Exception
{
    objserver=new SeleniumServer();
    objselenium=new DefaultSelenium("localhost",4444,"*firefox","http://127.0.0.1");
    objserver.start();
    objselenium.start();
    objselenium.windowMaximize();
}
@AfterClass

public void teardown()throws Exception
{
    objselenium.stop();
    objserver.stop();
}
@Test
public void Login()
{
    objselenium.open("/livehrm-2.5.0.2/login.php");
    if(objselenium.getTitle().equals("LiveHRM - New Level of HR Management"))
    {
        System.out.println("Home page Displayed Successfully");
    }
    else
    {
        System.out.println("Do not Displayed");
        return;
    }
    objselenium.type("txtUserName",username);
    objselenium.type("txtPassword", password);
    objselenium.click("Submit");
    objselenium.waitForPageToLoad("30000");
    if(objselenium.getTitle().equals("LiveHRM"))
    {
        System.out.println("Admin page Displayed Successfully");
    }
    else
    {
        System.out.println("Do not Displayed");
    }
    String []a=objselenium.getAllLinks();
    System.out.println("String length:"+a.length);
    objselenium.click("link=Logout");
}
}

```

```

}

```

#### TC-Count of the links in the webpage

```

import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;
import org.testng.annotations.*;

```

```

public class Testng {

```

```

public SeleniumServer objserver;
public Selenium objselenium;
static String username="rajesh";
static String password="suresh123";
@BeforeClass
public void setup()throws Exception
{
    objserver=new SeleniumServer();
    objselenium=new DefaultSelenium("localhost",4444,"*firefox","http://127.0.0.1");
    objserver.start();
    objselenium.start();
    objselenium.windowMaximize();
}
@AfterClass

public void teardown()throws Exception
{
    objselenium.stop();
    objserver.stop();
}
@Test
public void testcase()
{
    objselenium.open("/Livehrm");
    objselenium.type("txtUserName", admin);
    objselenium.type("txtPassword", admin);
    objselenium.click("Submit");
    objselenium.waitForPageToLoad("30000");
    string[]a =objselenium.getAllLinks();
    System.out.println("string length:"+a.length);
}
}

```

}

### TC-Comparing string

```

import org.openqa.selenium.server.*;
import com.thoughtworks.selenium.*;
import org.testng.annotations.*;

```

```

public class Testng {
    public SeleniumServer objserver;
    public Selenium objselenium;
    static String username="admin";
    static String password="admin";
    @BeforeClass
    public void setup()throws Exception
    {
        objserver=new SeleniumServer();
        objselenium=new DefaultSelenium("localhost",4444,"*firefox","http://127.0.0.1");
        objserver.start();
        objselenium.start();
        objselenium.windowMaximize();
    }
    @AfterClass

    public void teardown()throws Exception
    {
        objselenium.stop();
        objserver.stop();
    }
}

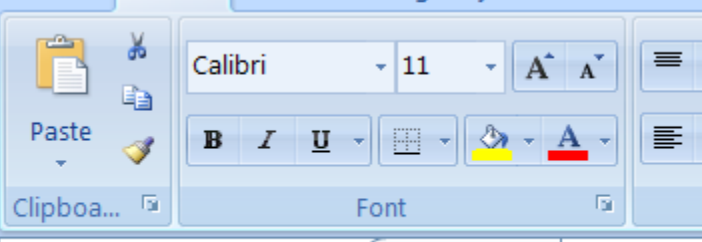
```

```

    }
    @Test
    public void testcase()
    {
        objselenium.open("/Livehrm")
        objselenium.type("txtUserName", rajesh);
        objselenium.type("txtPassword", suresh);
        string uname=objselenium.getText("txtUserName");
        System.out.println(uname);
        if(uname.equals("admin"))
        {
            System.out.println("the user name is admin");
        }
        else
        {
            System.out.println("The user name is not admin");
        }
        int len_uname;
        len_uname=uname.length();
        if(len_uname<9 && len_uname >3)
            System.out.println("The user name is invalid range");
        else
            System.out.println("Then user name is in invalid range");
        System.out.println(len_name)
        objselenium.click("Submit");
        objselenium.waitForPageToLoad("30000");
    }
}

```

### Login Scenario – Providing the login details from Excel sheet Create the Excel Sheet in .xls format



	A	B	C	D
1	<b>Username</b>	<b>Password</b>		
2	suresh	suresh123		
3				
4				
5				
6				

```

//Login using excel sheet with test ng
import java.io.FileInputStream;
import jxl.Sheet;
import jxl.Workbook;
import com.thoughtworks.selenium.*;

```

```

import org.openqa.selenium.server.*;
import org.testng.annotations.*;

public class LoginExcelInput
{
    public SeleniumServer objserver;
    public Selenium objselenium;

    @BeforeClass
    public void setUp() throws Exception
    {
        objserver=new SeleniumServer();
        objselenium=new DefaultSelenium("localhost",4444,"*firefox","http://127.0.0.1");
        objserver.start();
        objselenium.start();
        objselenium.windowMaximize();
    }
    @AfterClass
    public void teardown()throws Exception
    {
        objselenium.stop();
        objserver.stop();
    }
    @Test
    public void Login() throws Exception
    {
        FileInputStream fi=new FileInputStream("E:\\Selenium\\LoginExcel.xls");
        Workbook w=Workbook.getWorkbook(fi);
        Sheet s=w.getSheet(0);
        objselenium.windowMaximize();
        objselenium.open("/orangehrm-2.6/login.php");
        try
        {
            for (int i = 1; i < s.getRows(); i++)
            {
                //Read data from excel sheet
                String s1 = s.getCell(0,i).getContents();
                String s2 = s.getCell(1,i).getContents();
                objselenium.type("txtUserName",s1);
                objselenium.type("txtPassword", s2);
                objselenium.click("Submit");
                objselenium.waitForPageToLoad("30000");
                objselenium.click("link=Logout");
            }
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}

```

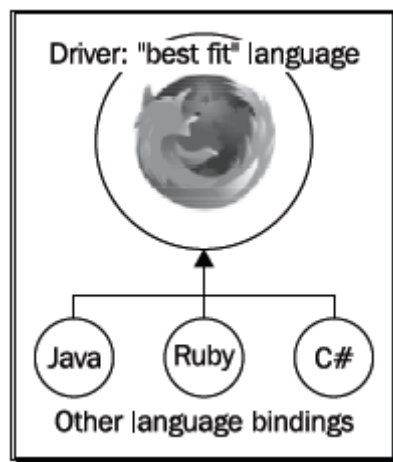
# WebDriver – Selenium 2.0

## Introducing Web Driver

The primary new feature in Selenium 2.0 is the integration of the Web Driver API. Web Driver is designed to providing a simpler, more concise programming interface along with addressing some limitations in the Selenium-RC API.

- The next major version of Selenium is going to change a lot of things on the Selenium internals
- Selenium 2.0 will be in fact be a merge between the Selenium and Web Driver projects
- Currently, Selenium 2.0 is still in development: last version at the time of writing is alpha 6

## Architecture



In Selenium 2 we are now making the Selenium Remote Control Server optional. This is because with Selenium 2 the Selenium Core developers are interacting with the "best fit" language for that browser. An example of this is with Firefox, we interact with the browser using JavaScript in a XPCOM object. This means that we can work on each of the different browsers without the fear of breaking its core elements. It also means that developers do not need to understand a number of different languages when developing bindings.

Once the bindings have been created there is a thin wrapper library created in the popular development languages. In our case we are going to see how our Java tests can interact with Firefox using the Firefox Driver that comes with Selenium 2.

If we want to test browsers that are not available on the machine that we are working on we can use the Remote WebDriver which will launch the browser that we want to test with and work with it as though it were on your machine. This has the obvious benefit of testing web applications on as many browser and operating system combinations as possible.

## Differences & New Features

Selenium 2.0 intends to get rid of Selenium RC

- The driver will directly drive the browser using Web Driver
- Parallelization is easier: the test code takes care of it

Selenium 2.0 comes as a browser extension directly plugging into the browser automation features, allowing to:

- Avoid the JavaScript security model (same origin policy rule) that restricts Selenium 1.x capabilities
- It will be possible to test features on several domains (like Facebook connect for instance)

Selenium 2.0 works at the OS/browser level:

- For instance, command type works at the OS level rather than changing the value of the input elements with JavaScript
- It drives the browser much more effectively: forget the limitations of Selenium 1.x like the file upload or download, pop-ups and dialogs barrier or self-signed certificates problems
- In addition, Web Driver provides a real object-based API giving you the ability to leverage the Page Object pattern very easily
- The merge of the projects combines the strengths of both frameworks: Selenium 2.0 will provide both Selenium 1.x and Web Driver APIs and it will be possible to use these APIs with both Selenium and Web Driver backends

### The Web Driver object

**HtmlUnitDriver**, a pure Java driver based on HtmlUnit browser simulator (all Java-capable OS)

- When running, you won't see any window opening because it doesn't render any page like Firefox would. You can't use it to get size or position of pages' elements
- It is very fast but does not provides JavaScript yet: this feature is still a WIP. If you want to use it anyway, use `driver.setJavascriptEnabled(true)`.

**FirefoxDriver**: supports Firefox (2.x and 3.x, all OS)

**InternetExplorerDriver**: supports InternetExplorer (tested on IE6, 7 and 8, should work on IE5.5 and on Windows XP and Vista)

**ChromeDriver**: drives Google's browser (>= 4.0, all OS)

**Android Driver,**  
**EventFiringWebDriver,**  
**IPhoneDriver,**  
**IPhoneSimulatorDriver,**  
**RemoteWebDriver**

### Locating UI Elements (Web Elements)

Locating elements in Web Driver can be done on the Web Driver instance itself or on a Web Element. Each of the language bindings exposes a "Find Element" and "Find Elements" method. The first returns a Web Element object otherwise it throws an exception. The latter returns a list of Web Elements, it can return an empty list if no DOM elements match the query.

The "Find" methods take a locator or query object called "By". "By" strategies are listed below.

#### By ID

This is the most efficient and preferred way to locate an element. Common pitfalls that UI developers make is having non-unique id's on a page or auto-generating the id, both should be avoided. A class on an html element is more appropriate than an auto-generated id.

**Example of how to find an element that looks like this:**

```
<div id="coolestWidgetEvah">...</div>
```

```
WebElement element = driver.findElement(By.id("coolestWidgetEvah"));
```

“Class” in this case refers to the attribute on the DOM element. Often in practical use there are many DOM elements with the same class name, thus finding multiple elements becomes the more practical option over finding the first element.

Example of how to find an element that looks like this:

```
<div class="cheese"><span>Cheddar</span></div><div class="cheese"><span>Gouda</span></div>
```

```
List<WebElement> cheeses = driver.findElements(By.className("cheese"));
```

#### **By Tag Name**

The DOM Tag Name of the element.

Example of how to find an element that looks like this:

```
<iframe src="..."></iframe>
```

```
WebElement frame = driver.findElement(By.tagName("iframe"));
```

#### **By Name**

Find the input element with matching name attribute.

Example of how to find an element that looks like this:

```
<input name="cheese" type="text"/>
```

```
WebElement cheese = driver.findElement(By.name("cheese"));
```

#### **By Link Text**

Find the link element with matching visible text.

Example of how to find an element that looks like this:

```
<a href="http://www.google.com/search?q=cheese">cheese</a>>
```

```
WebElement cheese = driver.findElement(By.linkText("cheese"));
```

#### **By Partial Link Text**

Find the link element with partial matching visible text.

Example of how to find an element that looks like this:

```
<a href="http://www.google.com/search?q=cheese">search for cheese</a>>
```

```
WebElement cheese = driver.findElement(By.partialLinkText("cheese"));
```

#### **By CSS**

Like the name implies it is a locator strategy by css. Native browser support is used by default, so please refer to *w3c css selectors* <<http://www.w3.org/TR/CSS/#selectors>> for a list of generally available css selectors. If a browser does not have native support for css queries, then Sizzle is used. IE 6,7 and FF3.0 currently use Sizzle as the css query engine.

Beware that not all browsers were created equal, some css that might work in one version may not work in another.

Example of to find the cheese below:

```
<div id="food"><span class="dairy">milk</span><span class="dairy aged">cheese</span></div>
```

```
WebElement cheese = driver.findElement(By.cssSelector("#food span.dairy.aged"));
```

#### **By XPATH**

At a high level, WebDriver uses a browser’s native XPath capabilities wherever possible. On those browsers that don’t have native XPath support, we have provided our own implementation. This can lead to some unexpected behaviour unless you are aware of the differences in the various xpath engines.

Driver	Tag and Attribute Name	Attribute Values	Native XPath Support
HtmlUnit Driver	Lower-cased	As they appear in the HTML	Yes
Internet Explorer Driver	Lower-cased	As they appear in the HTML	No
Firefox Driver	Case insensitive	As they appear in the	Yes

Driver	Tag and Attribute Name	Attribute Values	Native XPath Support
		HTML	

This is a little abstract, so for the following piece of HTML:

```
<input type="text" name="example" />
```

```
<INPUT type="text" name="other" />
```

```
List<WebElement> inputs = driver.findElements(By.xpath("//input"));
```

The following number of matches will be found

XPath expression	<a href="#">HtmlUnit Driver</a>	<a href="#">Firefox Driver</a>	<a href="#">Internet Explorer Driver</a>
//input	1 ("example")	2	2
//INPUT	0	2	0

Sometimes HTML elements do not need attributes to be explicitly declared because they will default to known values. For example, the “input” tag does not require the “type” attribute because it defaults to “text”. The rule of thumb when using xpath in WebDriver is that you **should not** expect to be able to match against these implicit attributes.

### Moving Between Windows and Frames

Some web applications have many frames or multiple windows. WebDriver supports moving between named windows using the “switchTo” method:

```
driver.switchTo().window("windowName");
```

### Popup Dialogs

Starting with Selenium 2.0 beta 1, there is built in support for handling popup dialog boxes. After you’ve triggered an action that opens a popup, you can access the alert with the following:

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

### Navigation:

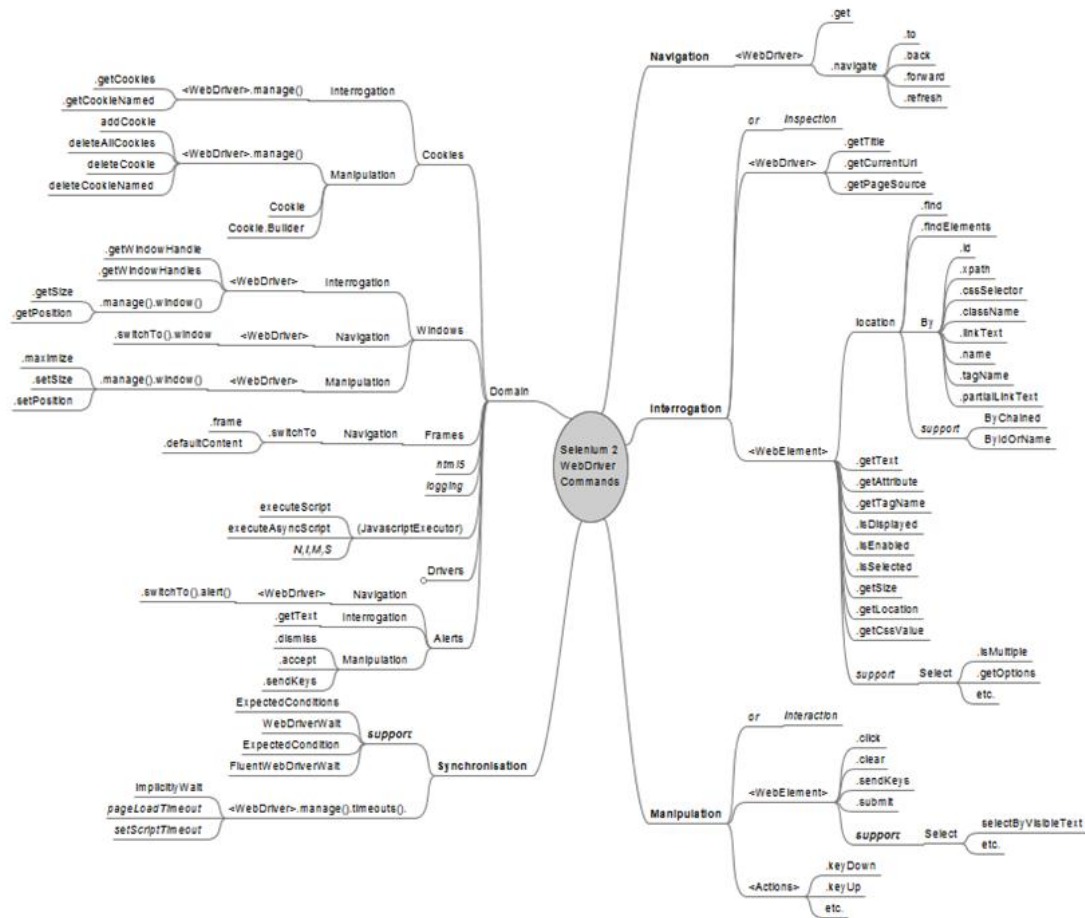
```
driver.navigate().to("http://www.example.com");
```

```
driver.navigate().forward();
```

```
driver.navigate().back();
```



## WebDriver Commands



## Example of a test using HtmlUnitDriver

```

package org.openqa.selenium.example;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.htmlunit.HtmlUnitDriver;

public class Example {
    public static void main(String[] args) {
        WebDriver driver = new HtmlUnitDriver(); // ❶

        // And now use this to visit Google
        driver.get("http://www.google.com"); // ❷

        // Find the text input element by its name
        WebElement element = driver.findElement(By.name("q")); // ❸
  
```

```

        // Enter something to search for
        element.sendKeys("Cheese!"); // ❹

        // Now submit the form. WebDriver will find the form for us from the
        element
        element.submit(); // ❺

        // Check the title of the page
        System.out.println("Page title is: " + driver.getTitle());
    }
}

```

We create a new `WebDriver` object using `HtmlUnit` as a browser.

`get` is the equivalent of `open`

Each element of the page is an instance of `WebElement`

`WebElement` provides the methods to interact with the element. If you call an invalid method on an element (`toggle()` on a Button for instance), an exception will be thrown

The `submit()` method will automatically find the form enclosing the element and submit it. If none is present, it will throw a `NoSuchElementException`

### Interact with the frames and windows of the browser

```

for (String handle : driver.getWindowHandles()) { // ❶
    driver.switchTo().window(handle);
}

...

driver.switchTo().frame("frameName");
driver.switchTo().frame("frameName.0.child"); // ❷

```

- ❶ You can get the list of the windows opened in the browser with `getWindowHandles()`. If you know the handle of the window (`target` attribute of an anchor element) you can just call `driver.switchTo().window("window handle")`
- ❷ You can also access the subframes of a frame using the dot and both name and index

### The driver object also provides navigation methods:

```

driver.navigate().to("http://www.example.com"); //
driver.navigate().forward();
driver.navigate().back();

```

- ❶ `driver.get("url");` is just an alias to `driver.navigate().to("url")`

### TC-Example for Web driver

```

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class webdriver {
    public static void main(String args[]) throws Exception
    {
        WebDriver Browser=new FirefoxDriver();
        Browser.get("http://127.0.0.1LiveHRM - New Level of HR Management");
        if(Browser.getTitle().equals("LiveHRM - New Level of HR Management"))
    }
}

```

```

        {
            System.out.println("Home page Displayed successfully");
        }
        else
        {
            System.out.println("Do not Displayed");
        }

        Browser.findElement(By.name("txtUserName")).sendKeys("rajesh");
        Browser.findElement(By.name("txtPassword")).sendKeys("suresh123");
        Browser.findElement(By.name("Submit")).click();
        Thread.sleep(2000);
        if(Browser.getTitle().equals("LiveHRM"))
        {
            System.out.println("admin page Displayed successfully");
        }
        else
        {
            System.out.println("Do not Displayed");
        }

        Browser.findElement(By.linkText("Logout"));
        Browser.close();
    }
}

```

### //Working with frames - Add new employe

```

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class AddEmp
{
    /**
     * @param args
     * @throws InterruptedException
     */
    public static void main(String[] args) throws InterruptedException {
        WebDriver driver=new FirefoxDriver();
        driver.navigate().to("http://127.0.0.1/orangehrm-2.6/login.php");
        driver.findElement(By.xpath("//input[@type='text']")).sendKeys("suresh");
        driver.findElement(By.xpath("//input[@type='password']")).sendKeys("suresh123");
        driver.findElement(By.xpath("//input[@type='Submit']")).click();
        Thread.sleep(5000L);
        //driver.findElement(By.linkText("Add Employee")).click();
        //Selecting the frame
        driver.switchTo().frame("rightMenu");
        //Clicking on Add Button
        driver.findElement(By.xpath("//*[@id='standardView']/div[3]/div[1]/input[1]")).click();
        driver.manage().timeouts().implicitlyWait(2, TimeUnit.SECONDS);
        driver.findElement(By.xpath("//*[@id='txtEmpLastName']")).sendKeys("aaa");
        Thread.sleep(2000L);
        driver.findElement(By.xpath("//*[@name='txtEmpFirstName']")).sendKeys("bbb");
        //driver.findElement(By.xpath("//*[@id='photofile']")).sendKeys("C:\\img.jpg");
        driver.findElement(By.xpath("//*[@id='btnEdit']")).click();
    }
}

```

```

        Thread.sleep(2000L);
        System.out.println("New Employee Added");
        //driver.navigate().back();
        // driver.navigate().back();
        driver.switchTo().defaultContent();
        driver.findElement(By.xpath("//*[@id='option-menu']/li[3]/a")).click();
        driver.quit();
    }
}

```

### // working with popups

```

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

```

```

public class PopUp {
    public static void main(String args[])throws Exception
    {
        WebDriver d1=new FirefoxDriver();

        d1.get("http://127.0.0.1/orangehrm-2.6/login.php");
        System.out.println(d1.getTitle());
        d1.findElement(By.name("txtUserName")).sendKeys("suresh");
        d1.findElement(By.name("Submit")).click();
        Alert jsalert=d1.switchTo().alert();
        System.out.println(jsalert.getText());
        jsalert.accept();
        d1.findElement(By.name("txtPassword")).sendKeys("suresh123");
        d1.findElement(By.name("Submit")).click();
        Thread.sleep(2000);
        d1.findElement(By.linkText("Logout")).click();
        d1.close();
    }
}

```

### //Entering the text in alert window

```

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

```

```

public class HTML_Alert {

    public static void main(String[] args) {
        WebDriver driver=new FirefoxDriver();
        driver.get("C:\\Documents and
Settings\\Administrator\\Desktop\\Selenium_Livotech_cd\\HtmlProg\\answerOnNextPrompt.html.
html");
        driver.findElement(By.xpath("html/body/a")).click();

        Alert jsalert=driver.switchTo().alert();
        System.out.println(jsalert.getText());
        jsalert.sendKeys("suresh");
    }
}

```

```

        jsalert.accept();
    }
}

```

### //Working with DropDown and page back commands

```

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.By;
public class DropDown {

    /**
     * @param args
     * @throws InterruptedException
     */
    public static void main(String[] args) throws InterruptedException {
        WebDriver driver=new FirefoxDriver();

        driver.navigate().to("file:///E:/Selenium/Selenium_Livetech_Suresh/HtmlProg/goBackAndWait.html");
        driver.findElement(By.xpath("html/body/form/select")).sendKeys("Yahoo");
        driver.findElement(By.xpath("html/body/form/input")).click();
        Thread.sleep(5000L);
        System.out.println(driver.getTitle());
        driver.navigate().back();
        driver.findElement(By.xpath("html/body/form/select")).sendKeys("Google");
        driver.findElement(By.xpath("html/body/form/input")).click();
        Thread.sleep(5000L);
        System.out.println(driver.getTitle());
        driver.navigate().back();
        Thread.sleep(5000L);

        driver.quit();

    }
}

```

### Webdriver with Testng

```

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class WebDriverTestng {
    private WebDriver driver;
    @BeforeClass
    public void Startup(){
        driver = new FirefoxDriver();
    }
    @Test (description="OrangeHRM Login")
    public void Login() throws Exception{
        driver.get("http://127.0.0.1/orangehrm-2.6/login.php");
        driver.findElement(By.name("txtUserName")).sendKeys("suresh");
        driver.findElement(By.name("txtPassword")).sendKeys("suresh123");
    }
}

```

```

driver.findElement(By.name("Submit")).click();
Thread.sleep(3000);
if(driver.getTitle().equals("OrangeHRM"))
{
    System.out.println("admin page Displayed successfully");
}
else
{
    System.out.println("Do not Displayed");
}
driver.findElement(By.linkText("Logout"));
}
@AfterClass
public void teardown(){
    driver.quit();
}
}

```

### Webdriver with Excel

```

import java.io.FileInputStream;
import jxl.Sheet;
import jxl.Workbook;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class WebDriverExcel {
    private WebDriver driver;
    @BeforeClass
    public void Startup(){
        driver = new FirefoxDriver();
    }
    @Test (description="OrangeHRM Login using excel sheet")
    public void Login() throws Exception{
        FileInputStream fi=new FileInputStream("E:\\Selenium\\LoginExcel.xls");
        Workbook w=Workbook.getWorkbook(fi);
        Sheet s=w.getSheet(0);
        driver.get("http://127.0.0.1/orangehrm-2.6/login.php");
        try
        {
            for (int i = 1; i < s.getRows(); i++)
            {
                //Read data from excel sheet
                String s1 = s.getCell(0,i).getContents();
                String s2 = s.getCell(1,i).getContents();
                driver.findElement(By.name("txtUserName")).sendKeys(s1);
                driver.findElement(By.name("txtPassword")).sendKeys(s2);
                driver.findElement(By.name("Submit")).click();
                Thread.sleep(3000);
                if(driver.getTitle().equals("OrangeHRM"))
                {
                    System.out.println("admin page Displayed successfully");
                }
                else
            }
        }
    }
}

```

```
        {
            System.out.println("Do not Displayed");
        }
        driver.findElement(By.linkText("Logout"));
    }
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
}
@AfterClass
public void teardown(){
    driver.quit();
}
}
```

# Sikuli Automation Tool

Sikuli automates anything you see on the screen. It uses image recognition to identify and control GUI components. It is useful when there is no easy access to a GUI's internal or source code.

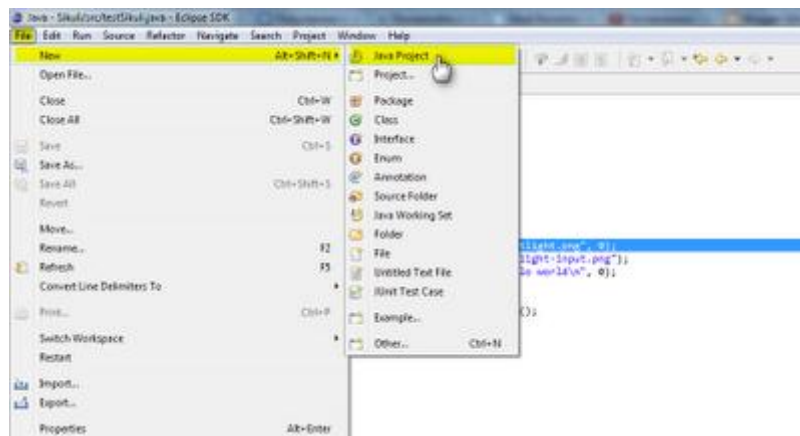
## How to integrate Sikuli script with Selenium WebDriver

Sikuli is a robust and powerful tool to automate and tests user interfaces screenshots. The core of Sikuli Script is written in Java, which means you can use Sikuli Script as a standard JAVA library in your program. This article lets you know how to do that.

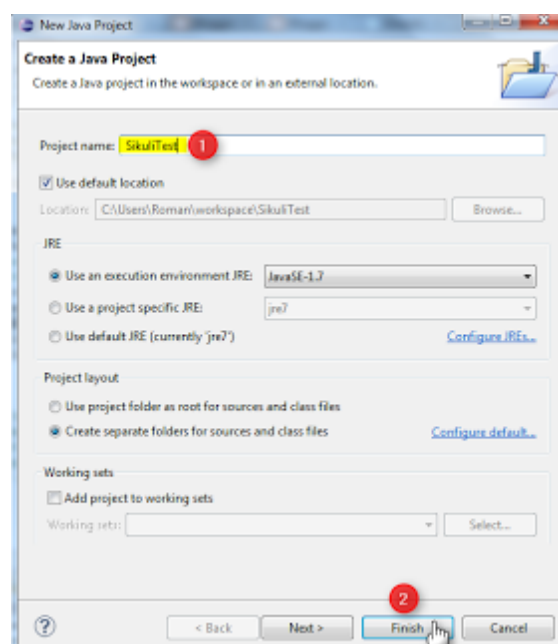
### 1. Download and install Sikuli using the self-extracting installer(<http://www.sikuli.org/download.html>).

Note: Only 32-bit version is provided for using as a standard JAVA library. But SIKULI IDE could run on both 32-bit and 64-bit Windows systems.

### 2. Create new Java project (use Eclipse as an example):

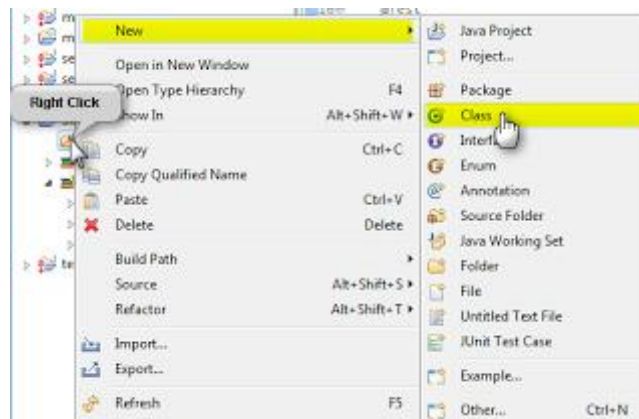


### 3. Fill project name and click Finish:

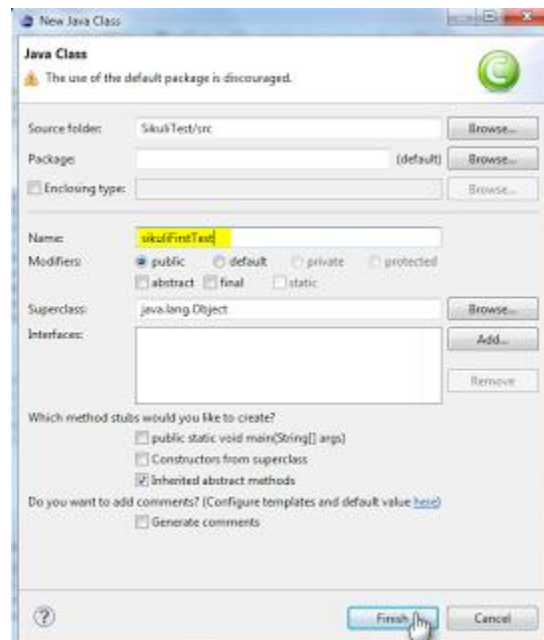




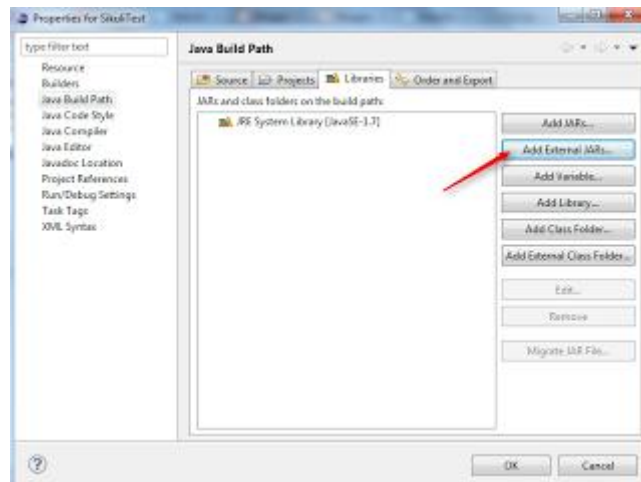
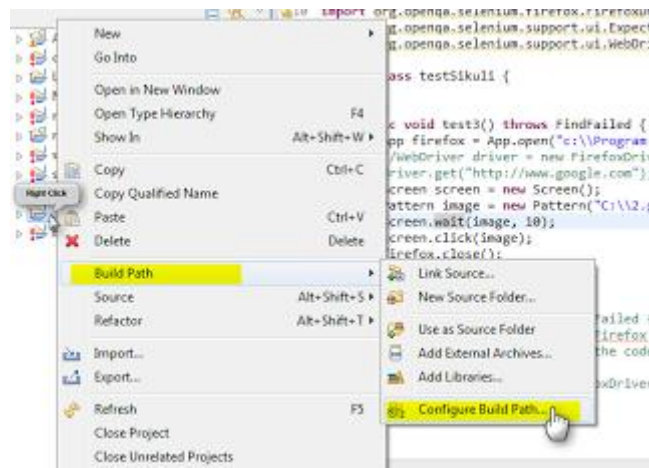
#### 4. Create new class:



#### 5. Fill class name and click Finish:



6. Include sikuli-script.jar, selenium-server-standalone-2.25.0.jar, selenium-java-2.25.0.jar in the CLASSPATH of your Java project.



Get sikuli-script.jar from your Sikuli IDE installation path.

Sikuli Script is packed in a JAR file - sikuli-script.jar. Depending on the operating system you use, you can find the sikuli-script.jar in according places.

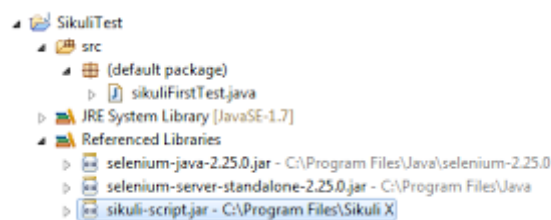
*Windows, Linux:* Sikuli-IDE/sikuli-script.jar

*Mac OS X:* Sikuli-IDE.app/Contents/Resources/Java/sikuli-script.jar

After adding sikuli-script.jar, selenium-server-standalone-2.25.0.jar, selenium-java-2.25.0.jar as a libraries into your project, the project hierarchy should look like this:



After click OK:



7. After configuring in build path, create and initialize an instance of Screen object.

## SIKULI + SELENIUM WEBDRIVER

```
import org.junit.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.sikuli.script.App;
import org.sikuli.script.FindFailed;
import org.sikuli.script.Pattern;
import org.sikuli.script.Screen;

public class sikuliFirstTest {

    @Test
    public void functionName() throws FindFailed {

        // Create a new instance of the Firefox driver
        WebDriver driver = new FirefoxDriver();

        // And now use this to visit Google
        driver.get("http://www.google.com");

        //Create and initialize an instance of Screen object
        Screen screen = new Screen();

        //Add image path
        Pattern image = new Pattern("C:\\\\searchButton.png");

        //Wait 10ms for image
```

```
screen.wait(image, 10);
```

```
//Click on the image  
screen.click(image);  
}  
}
```

### **Here example using SIKULI without Selenium WebDriver:**

```
import org.junit.Test;  
import org.sikuli.script.App;  
import org.sikuli.script.FindFailed;  
import org.sikuli.script.Pattern;  
import org.sikuli.script.Screen;  
  
public class sikuliFirstTest {  
  
    @Test  
    public void functionName() throws FindFailed {  
  
        //Open FireFox application with google home page  
        App firefox = App.open("c:\\Program  
Files\\MozillaFirefox\\firefox.exe");  
  
        //Create and initialize an instance of Screen object  
        Screen screen = new Screen();  
  
        //Add image path  
        Pattern image = new Pattern("C:\\searchButton.png");  
  
        //Wait 10ms for image  
        screen.wait(image, 10);  
  
        //Click on the image  
        screen.click(image);  
  
        //Close firefox  
        firefox.close();  
    }  
}
```

# Automation Framework

## What an Automation Framework is?

A **test automation framework** is a set of **assumptions, concepts** and tools that provide support for automated software testing. The main advantage of such a framework is the low cost for maintenance. If there is change to any test case then only the test case file needs to be updated and the Driver Script and Startup script will remain the same. Ideally, there is no need to update the scripts in case of changes to the application.

## Utility of Test Automation Framework

- Provides an Outline of overall Test Structure
- Ensures Consistency of Testing
- Minimizes the Amount of Code for Development - thereby Less Maintenance
- Maximizes Reusability
- Reduces Exposure of Non-Technical Testers to Code
- Enables Test Automation using Data

## How Many Types of Automation Frameworks are there?

Generally there are 4 Types:

- Data Driven Automation Framework
- Keyword Driven Automation Framework
- Modular Automation Framework
- Hybrid Automation Framework

## Data Driven Automation Framework

In Data Driven Framework, the data is NOT hard-coded in the test scripts. Instead, it is stored in some external files. The test script first connects to the external data source and then extracts the data from that source. Most of the times, excel sheets are used to store the data. Other external data sources that are frequently used are –Text files, XML files, Databases, Combination of more than one external file.

- Scripts are an assembly of function calls
- Data for test cases read in from external source (e.g., Excel spreadsheet, ODBC database)
- Results can be captured externally per script execution (i.e., spreadsheet, database)
- Repeated use of Test Scripts with Different Inputs and Response Data coming out of Predefined Dataset
- Helps in Reducing Coding for Large Test Cases
- Ease of Testing of Time-Consuming & Complex Test Cases

## Advantages of Data Driven Framework

- Since the data is kept separate from the test script, the same script can be run multiple times for different sets of data (which can be kept in multiple rows in the data sheet).
- Changes made in the test data don't affect the test scripts in any way and vice versa.

## Disadvantages of Data Driven Framework

- Additional effort and good technical expertise is required to create functions that connect to the external data sources to retrieve the data.
- Additional amount of time is required in identifying which external data source to use and deciding how the data should be stored or grouped in that external data source.

## Keyword Driven Automation Framework

In Keyword Driven framework, you can create various keywords and associate a different action or function with each of these keywords. Then you create a Function Library that contains the logic to read the keywords and call the associated action.

- As the Name suggests, it enables Keyword Driven Testing or Table Driven Testing
- Data & Keyword Tables being Independent of the Automation Tool
- Enables Documentation of the Functionality of the Application under Test (AUT) in A Tabular Format
- Test cases are expressed as sequence of keyword-prompted actions
- A Driver script executes Action scripts which call functions as prompted by keywords
- No scripting knowledge necessary for developing and maintaining test cases

### Advantages of Keyword Driven Framework

- The keyword and function libraries are completely generic and thus can be reused easily for different applications.
- All the complexity is added in the function libraries. Once the libraries are ready, it becomes very easy to write the actual test script steps in excel sheets.

### Disadvantages of Keyword Driven Framework

- Lot of time and effort needs to be spent initially to create the function libraries. The benefits of the keyword driven framework are realized only after it has been used for longer periods of time.
- Very high programming skills are needed to create the complex keyword library functions.
- It's not easy for new people to understand the framework quickly.

## Modular Automation Framework

Modular Framework is the approach where all the test cases are first analyzed to find out the reusable flows. Then while scripting, all these reusable flows are created as functions and stored in external files and called in the test scripts wherever required.

- Test Script Modularity Framework:  
Enables creation of Small, Independent Scripts representing Modules & Functions of the Application under Test (AUT)
- Test Library Architecture Framework:  
Enables creation of Library Files representing Modules & Functions of the Application under Test (AUT)
- Descriptive programming is used to respond to dynamic applications (e.g., websites)
- Actually, this is a method which can used within other solution types
- Objects defined by parameterized code (i.e., regular expressions, descriptive programming)
- Custom functions used to enhance workflow capabilities

### Advantages of Modular Framework

- Test Scripts can be created in relatively less time as the reusable functions need to be created only once.
- Effort required to create test cases is also lesser due to code reuse.
- If there are any changes in the reusable functions, the changes need to be done in only a single place. Hence script maintenance is easier.

### Disadvantages of Modular Framework

- Since data is still hardcoded in the script, the same test case cannot be run for multiple data values without changing data after each run.
- Additional time is spent in analyzing the test cases to identify with reusable flows.
- Good programming knowledge is required to create and maintain function libraries.

### Hybrid Automation Framework

Hybrid Framework is a framework that is created by combining different features of any of the frameworks mentioned above. Based upon your requirements, you can combine the features of any of the above frameworks to come up with your own version of Hybrid Framework.

- It is the Most Popularly Implemented Framework
- It is a Combination of the Three Types of Frameworks described before
- It has an Ability of Evolving Itself Over a Passage of Time and Over Many Projects

### Advantages of Hybrid Framework

- The main advantage of this approach is that you can use the best features from all the types of frameworks to create your own.

### Disadvantages of Hybrid Framework

- The framework is highly complex and needs very good programming expertise and experience to build a Hybrid Framework from scratch.

### Ten Steps to Implement Automation Framework Approach

#### 1) Identification of the Scope of Testing:

Company Oriented, Product Oriented, Project Oriented

#### 2) Identification of the Needs of Testing:

Identify Types of testing e.g. FT, Web Services etc. and application / modules to be tested

#### 3) Identification of the Requirements of Testing:

Find out the Nature of Requirements, Identification of type of actions for each requirement & identification of High Priority Requirements

#### 4) Evaluation of the Test Automation Tool:

Preparation of Evaluation Checklist, Identification of the Candidate Tools Available, Sample Run, Rate & Select the

#### 5) Identification of the Actions to be automated:

Actions, Validations & Requirements supported by the Tool

#### 6) Design of the Test Automation Framework:

Framework Guidelines, Validations, Actions Involved, Systems Involved, Tool Extensibility Support, Customs Messages & UML Documentation

#### 7) Design of the Input Data Bank:

Identification of Types of Input file, Categorization & Design of File Prototypes

#### 8) Development of the Automation Framework:

Development of Script based upon Framework Design, Driver Scripts, Worker Scripts, Record / Playback, Screen / Window / Transaction, Action / Keyword & Data Driven

#### 9) Population of Input Data Bank:

Different Types of Data Input, Population of Data from Different Data Sources, Manual Input of Data and Parent – Child Data Hierarchy

#### 10) Configuration of the Schedulers:

Identify Scheduler Requirements & Configure the Schedulers

## Benefits of Automation Framework Approach

- Significant Reduction in Testing Cycle Time
- Comprehensive Coverage against Requirements
- Use of a "Common Standard" across the Organization / Product Team / Project Team
- Generation of Reusable Test Scripts ( Utility Functions)
- Systematic Maintenance of Automation Scripts
- Data Pooling

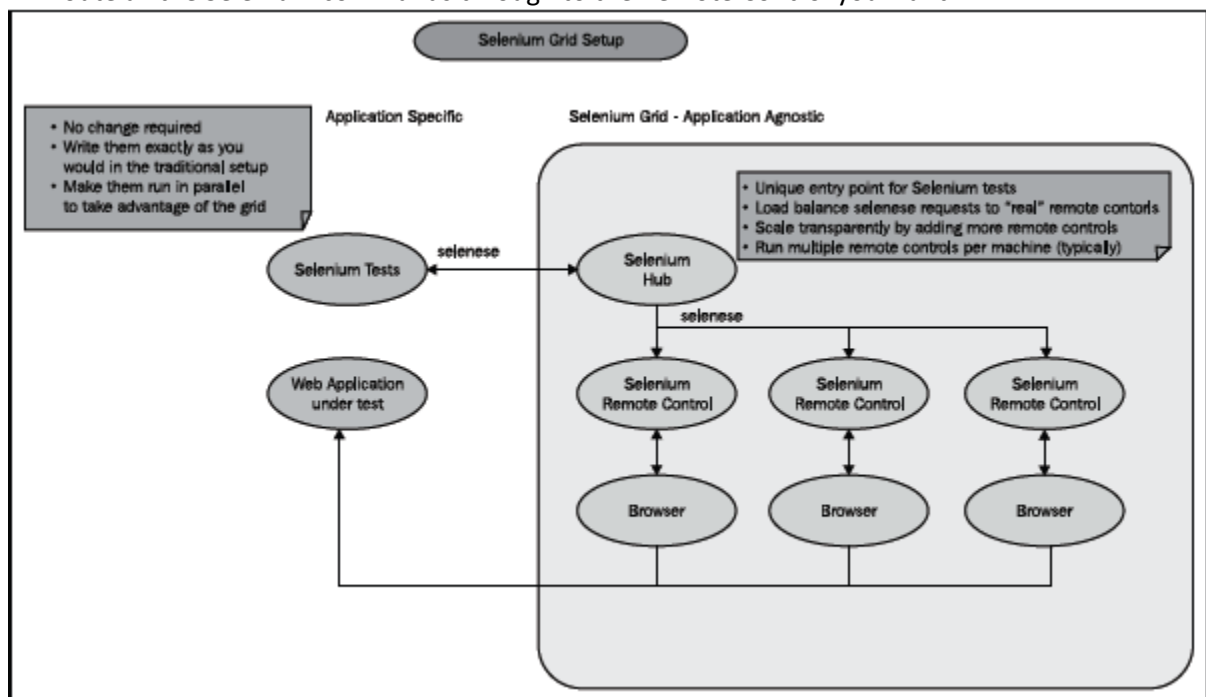
Examples & source code for the frame works will be provided in the class.

## Selenium Grid

### Basic information about selenium grid

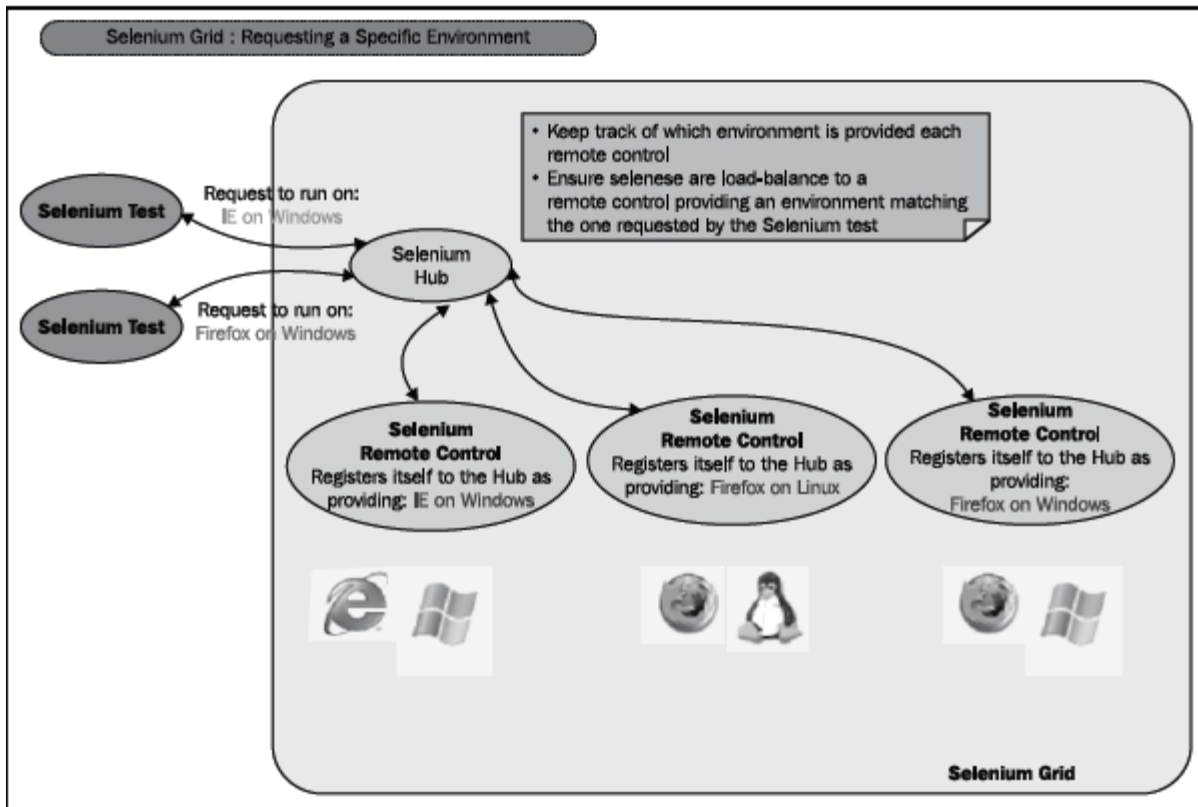
Selenium Grid is a version of Selenium that allows teams to set up a number of Selenium instances and then have one central point to send your Selenium commands to. This differs from what we saw in Selenium Remote Control (RC) where we always had to explicitly say where the Selenium RC is as well as know what browsers that Remote Control can handle.

With Selenium Grid, we just ask for a specific browser, and then the hub that is part of Selenium Grid will route all the Selenium commands through to the Remote Control you want.



Selenium Grid also allows us to, with the help of the configuration file, assign friendly names to the Selenium RC instances so that when the tests want to run against Firefox on Linux, the hub will find a free instance and then route all the Selenium Commands from your test through to the instance that is registered with that environment. We can see an example of this in the next diagram.





\*\*\*\*\*Good Luck\*\*\*\*\*