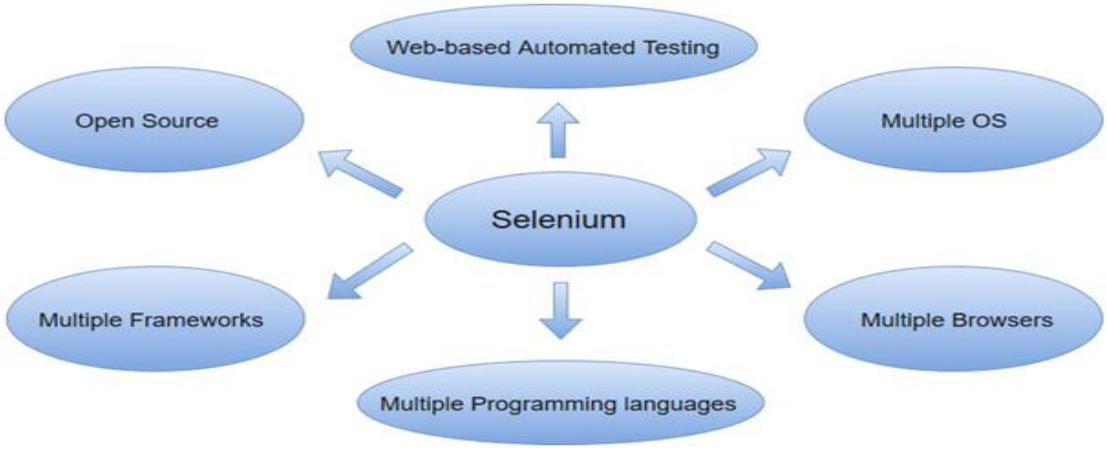
Section 1

1. **Disadvantages of manual testing**
2. **Advantages of automation testing**
3. **Advantages of selenium**
4. **Disadvantages of selenium**
5. **Selenium flavor**
6. **Open browser**
7. **Selenium architecture**
8. **Web driver & its methods**
9. **Basic html coding**
10. **Locators**
11. **Web element and its method**

Selenium

1. Selenium is one of the **most widely used open source** Web UI (User Interface) automation testing suite/tool.
2. It was originally developed by **Jason Huggins in 2004** as an internal tool at Thought Works. Selenium **supports automation** across different browsers, platforms and programming languages.
3. Selenium can be easily **deployed on platforms** such as Windows, Linux, Solaris and Macintosh.
4. Moreover, it **supports OS** (Operating System) for mobile applications like iOS, windows mobile and android.
5. Selenium **supports** a variety of **programming languages** through the use of drivers specific to each language.
6. Languages supported by Selenium **include** C#(C-sharp), Java, Perl, PHP, Python and Ruby. Currently, Selenium Web driver is **most popular with Java** and C#.
7. Selenium **test scripts** can be **coded** in **any** of the supported **programming languages** and can be **run** directly in most **modern web browsers.**
8. **Browsers** supported by Selenium **include** Internet Explorer, Mozilla Firefox, Google Chrome and Safari.

9. Selenium can be used to automate functional tests and can be integrated with automation test tools such as **Maven**, **Jenkins**, **& Docker** to achieve continuous testing.
10. It can also be integrated with tools such as **TestNG**, & **JUnit** for managing test cases and generating reports.



Automation Testing

1. Automation testing uses the **specialized tools** to **automate** the execution of manually designed test cases **without** any human intervention.
2. Automation testing **tools** can **access** the test data, controls the execution of tests and **compares** the actual result **against** the expected result.
3. Consequently, **generating** detailed test reports of the system under test.
4. **Testing** an applications **features** with the **help** of automation tool and **executing** test scripts is called automation testing.

Automation testing covers both functional and performance test on an application.

- **Functional automation** is used for automation of functional test cases. For example, regression tests, which are repetitive in nature, are automated.
- **Performance automation** is used for automation of **non-functional** performance test cases. For example, measuring the response time of the application under considerable (say 100 users) load.

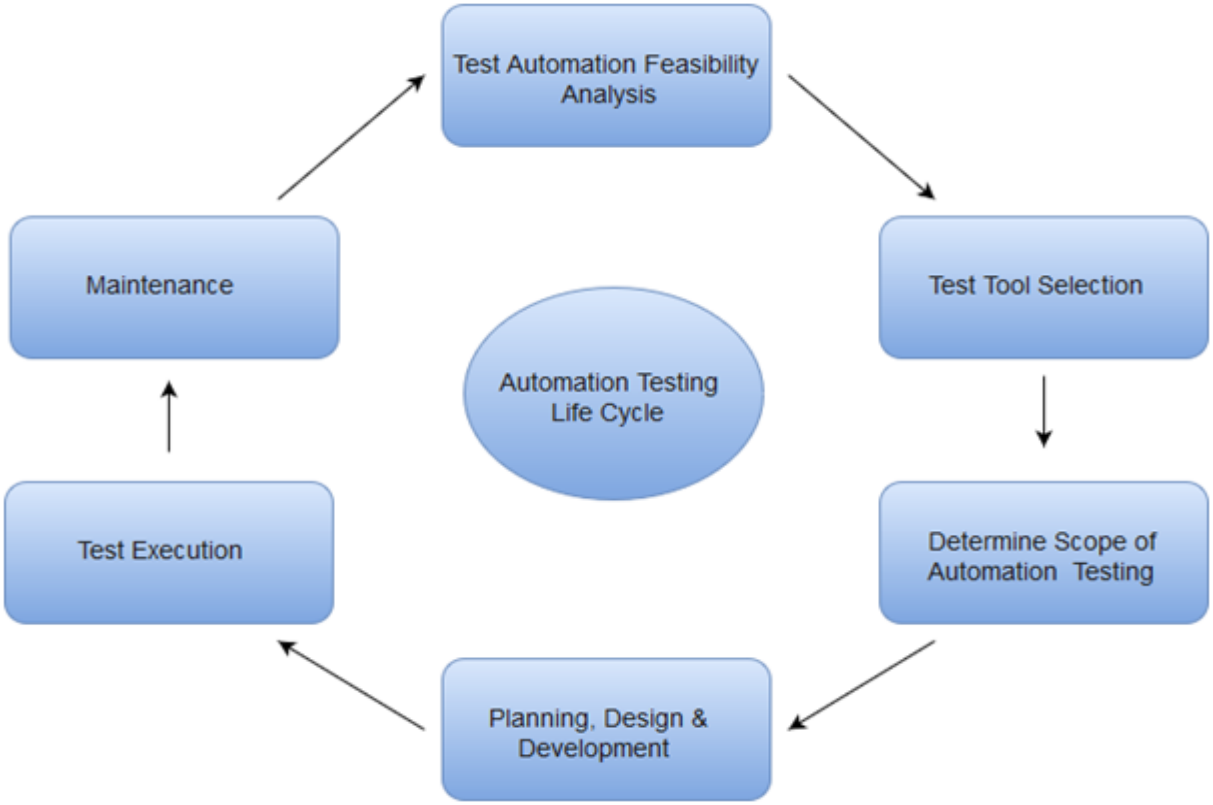
Automation Testing tools which are used for functional automation:

- Quick Test Professional, provided by HP.
- Rational Robot, provided by IBM.
- Coded UI, provided by Microsoft.
- Selenium, open source.
- Auto It, open Source.

Automation Testing tools which are used for non-functional automation:

- Load Runner, provided by HP.
- JMeter, provided by Apache.
- Burp Suite, provided by PortSwigger.
- Acunetix, provided by Acunetix.

Automation Testing Life Cycle



Why Automated Testing

Automation testing has specific advantages for **improving** long-term efficiency of any software. The key benefits of test automation are:

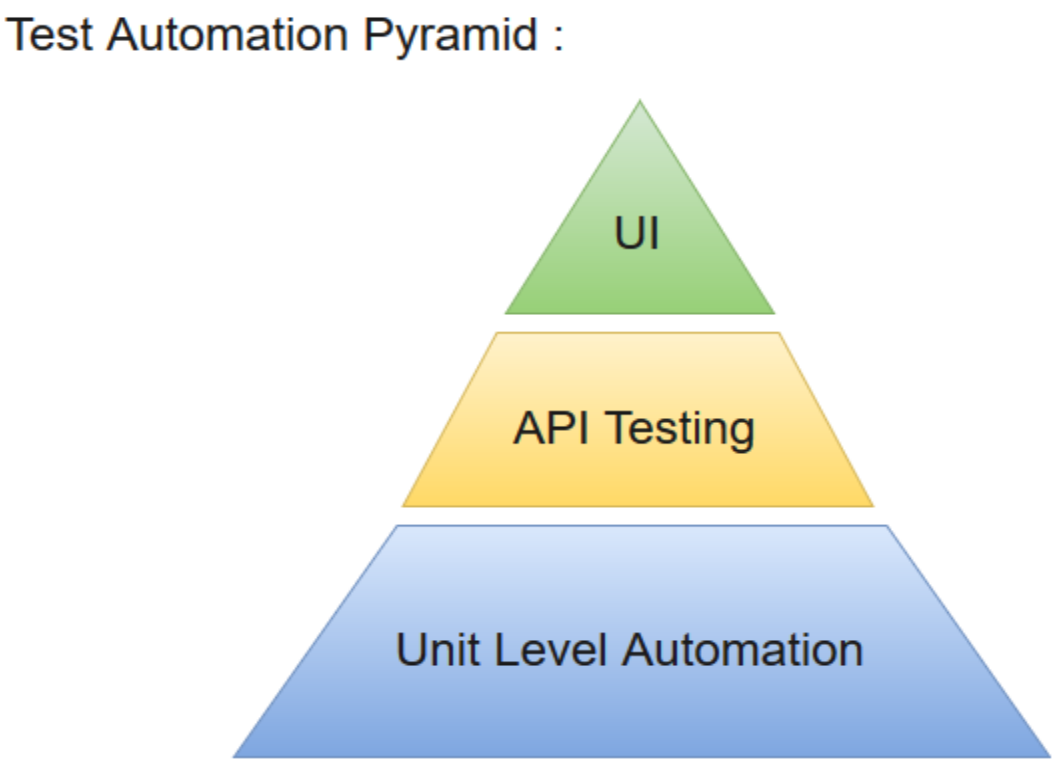
- Automated testing has long been considered **beneficial** for big software organizations. Although, it is often thought to be too **expensive** or **difficult** for smaller companies to implement.
- Automated testing tools can be **programmed** to build and execute test scripts at a **specific time without** involving any human intervention. For instance, automated test can be automatically kicked off overnight, and the testers can analyze the results of the automated the next morning.
- Automated testing tools are **able to** playback pre-recorded and pre-defined actions.
- Automation testing **supports** frequent regression testing.
- It provides **rapid** feedback to developers.
- It provides **unlimited** iterations of test case execution.
- It provides **disciplined** documentation of test cases.
- Automated test **generates** customized defect reports.
- **Less error** prone as compared to manual testing.

Test Automation for Web Applications

If we take a look at the type of software applications prevailing in current market scenario, most of the software applications are written as web-based applications to be run in an internet browser. The testing strategy for web-based applications varies widely among companies and organizations.In an era of highly interactive and responsive software processes where many organizations are using some form of agile methodology, test automation is frequently becoming a requirement for software projects.

The most effective manner to carry out test automation for web application is to adopt a pyramid testing strategy.This pyramid testing strategy includes automation tests at three different levels. Unit testing represents the base and biggest

percentage of this test automation pyramid. Next comes, service layer, or API testing. And finally, GUI tests sit at the top. The pyramid looks something like this:



Selenium Features

- Selenium is an **open source** and **portable** Web testing Framework.
- Selenium **IDE provides** a playback and record feature for authoring tests without the need to learn a test scripting language.
- It can be considered as the leading cloud-based testing platform which helps testers to record their actions and export them as a reusable script with a simple-to-understand and easy-to-use interface.
- Selenium **supports** various operating systems, browsers and programming languages. Following is the list:
    - **Programming Languages:** C#, Java, Python, PHP, Ruby, Perl, and JavaScript
    - **Operating Systems:** Android, iOS, Windows, Linux, Mac, Solaris.
    - **Browsers:** Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.

- It also **supports** parallel test execution which **reduces** time and **increases** the efficiency of tests.
- Selenium can be integrated with frameworks like **Ant** and **Maven** for source code compilation.
- Selenium can also be **integrated** with testing frameworks like **TestNG** for application testing and generating reports.
- Selenium requires fewer resources as compared to other automation test tools.
- WebDriver API has been indulged in selenium whichis one of the most important modifications done to selenium.
- **Selenium web driver does not require server installation, test scripts interact directly with the browser.**
- Selenium commands are categorized in terms of different classes which make it easier to understand and implement.
- Selenium Remote Control (RC) in conjunction with WebDriver API is known as Selenium 2.0. This version was built to support the vibrant web pages and Ajax.

Selenium Limitations

- Selenium **does not support** automation testing for **desktop applications**.
- Selenium requires **high skill sets** in order to automate tests more effectively.
- Since Selenium is **open source software**, you have to rely on community forums to get your technical issues resolved.
- We **can't perform** automation tests on **web services** like SOAP or REST using Selenium.
- We should know at least one of the supported programming languages to **create tests scripts** in Selenium WebDriver.
- It does not have built-in Object Repository like UTF/QTP to maintain objects/elements in centralized location. However, we can overcome this limitation using Page Object Model.
- Selenium does not have any inbuilt reportingcapability; you have to rely on plug-ins like **JUnit** and **TestNG** for test reports.

- It is **not possible to perform testing on images**. We need to integrate Selenium with **Sikuli** for image based testing.
- **Creating test environment** in Selenium takes more time as compared to vendor tools like UFT, RFT, Silk test, etc.
- No one is responsible for new features usage; they may or may not work properly.
- Selenium does not provide any test tool integration for Test Management.

Selenium vs QTP(Quick Test Professional)

Selenium and QTP are the most frequent used automation test tools in the market. Hence, we have compared some of the features of Selenium over QTP.

Features	Selenium	HP QTP
License	Open source tool	Required
Customer support	Dedicated HP support	Selenium community forums
Test Support	Supportsautomation only for web-based applications.	Support tests on both web and desktop based applications.
Resource consumption during test scripts execution	Low resource consumption	High resource consumption
Supported programming languages	Java, C#, Ruby, Python, Perl, PHP and JavaScript	VB Script.
Supported Environments	Android, iOS, Windows, Linux, Mac, Solaris.	Only for Windows
Supported Browsers	Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.	Specific versions of Google Chrome, Mozilla Firefox and Internet Explorer.
Object Repository/Recovery Scenario	Absent	Built-in object repository and recovery scenario.
Browser Controls	None	Controls like favourites bar, backward and forward buttons can be

		accessed within the browser.
Test Report Generation	It relies on external tool for generating test reports.	Built-in test report generation within the tool.
Parameterization	You have to rely on any one of the supported programming language for parameterization.	Built-in tools are available for parameterization.

Selenium Tool Suite/flavor/components

Selenium is not just a single tool but a suite of software, each with a different approach to support automation testing. It comprises of five major components which include:

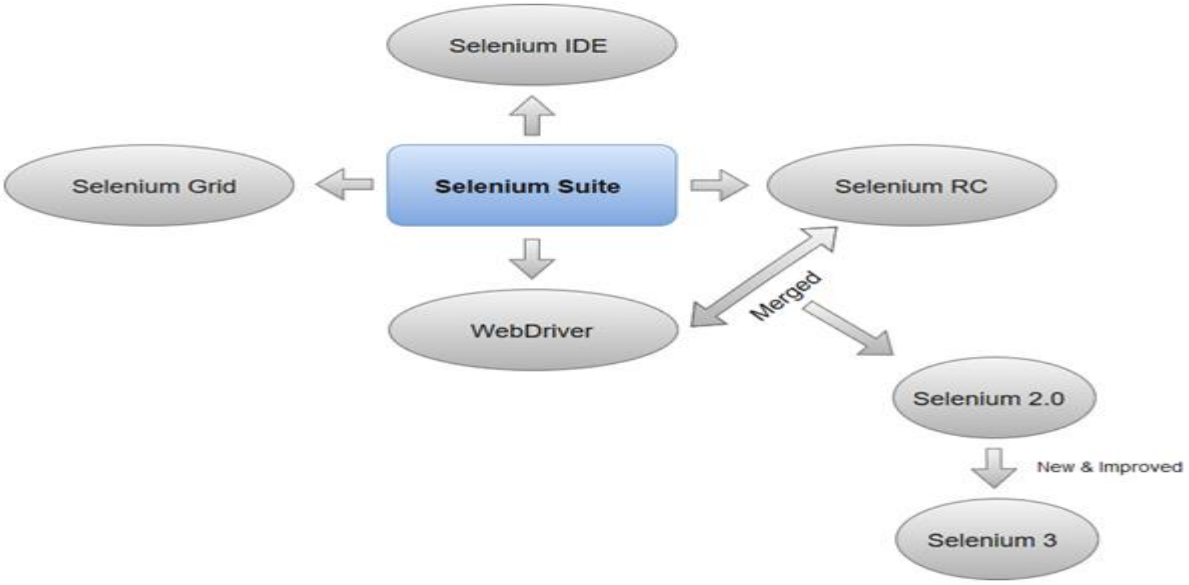
1. Selenium Integrated Development Environment (IDE)
    - We can **run** script in **only** Firefox browser
    - **Record** and **playback** options
    - We **can not do** compatibility testing

2. Selenium Remote Control (Now Deprecated)
    - **Support** compatibility testing (cross browser)
    - We can **run** scripts in java **only**

3. WebDriver
    - **Support** compatibility testing (cross browser)
    - We can **run** scripts in **multiple** languages

4. Selenium Grid
    -

5. Selendroid
    -



Disadvantages of Manual Testing

1. **Compatibility testing** is difficult
2. **Test cycle duration** will be increased
3. More **human efforts** are required
4. **Regression testing** is time consuming

Advantages of automation testing

1. Reusability of **test script**
2. **Compatibility testing** is easy/possible
3. **Project duration** will be reduced
4. Less **human efforts** are required
5. To overcome drawback of **regression testing**
6. **Cost of project** will be reduced
7. It is **reliable** and **efficient**

Some of the automation tools

1. Selenium
2. atp
3. sahi / sahipro
4. selendroid
5. appium

When we should do automation testing

Automation testing tool will be able to perform testing an application but to perform any action as a test environment we need to give commands these commands are called scripting.

Advantages of selenium

1. **Open Source**
2. **Multi Language** Supportable
3. **Cross Browser / Compatibility Testing** Possible
4. **Cross Platform** Is Also Perform

Disadvantages of selenium

1. We Can Automate **Web Based Application**
2. We Can Not Automate **Standalone Application**
3. Can Not Automate **Captcha**
4. Can Not Read **Broadcast**
5. Selenium Will Not Support **File Uploading**
6. **Adhoc Test Cases** Can Not Be Automated

Java concept used in Automation

1. Control Statements 2. Loops 3. Inheritance 4. Polymorphism
5. Interface 6. Casting (Up Casting) 7. Abstraction 8. Encapsulation
6. Arrays 10. String 11. Collection

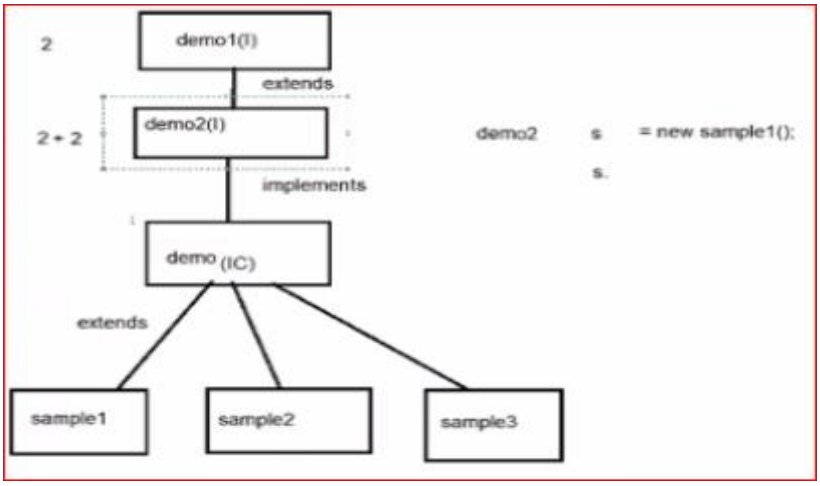
Selenium Setup

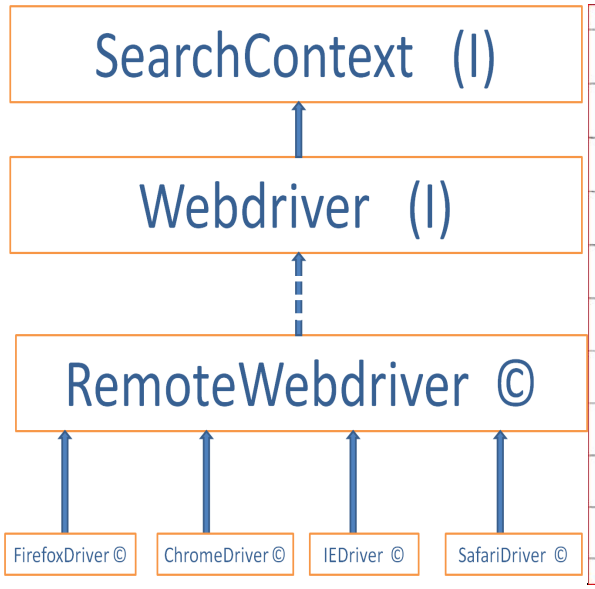
1. Google search
2. Download selenium jar file
3. Select 1st website- download web driver
4. Click on latest stable version **-3.141. 59** and then download fine
5. Create new project in eclipse →create package
6. Right click on project name→build path→configure build path→libraries→Add external JARs→download jar file open→apply→close
7. Create one class→selenium 1st program is browser open→create object of chrome driver class (to check chrome browser is open or not)→

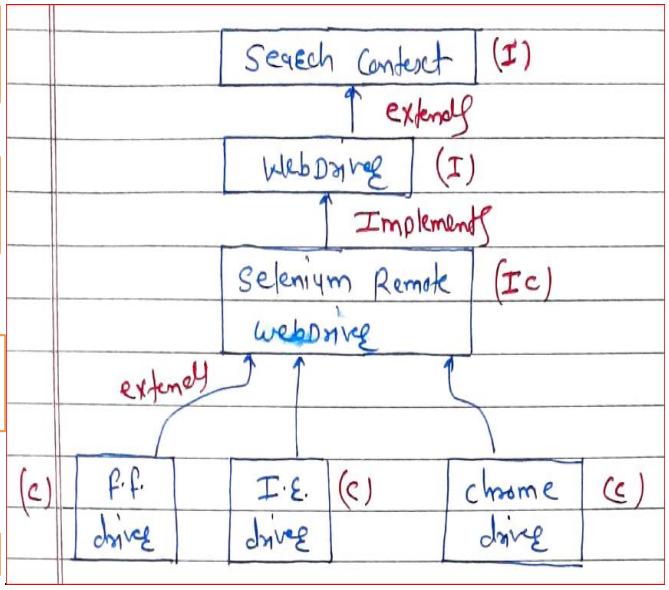
→ WebDriver driver = new Chrome Driver(); →before this we have to mention chrome driver path→Google→download chromedriver.exe file→click on ChromeDriver 88.0.4324.96 →download in zip format→unzip create same name floder→click on new folder→.exe file→click shift and right click→copy path→in class call to System.setProperty("selenium version name we have to use.which browser we have to use.for browser purpose driver",".exe copy path→add one slash");
System.setProperty("webdriver.chrome.driver", "path of .exe file")

WebDriver driver = new ChromeDriver();

Selenium Architecture







1. Search context is a **super most interface** which **contains** abstract methods & **inherited** to web driver.
2. Web driver is an **interface** which **contains** abstract method of **search context** and its **own** abstract methods.
3. All the abstract methods are **overridden** or **implemented** in selenium remote web driver class.
4. Selenium remote web driver-it is **class** which **implements** all the abstract methods of **both interface** (search context and web driver).
5. Selenium remote driver class is **extended** to **browsers** such as Firefox, internet explorer, chrome…..etc.
6. To **run** application in **multiple browsers** (C.T) i.e **writing** test script by using **single browser** but **run** the same script **in multiple browser** we need to **use** runtime polymorphism **by using** up casting in selenium.

```
WebDriver  driver = new Chrome Driver();
```

7. **Create an object** of ChromeDriver class **with** reference of WebDriver interface.

How to open browser in selenium

```
System.setProperty("webdriver.chrome.driver" "path of .exe file")
                    -all mention in small letter
WebDriver  driver = new ChromeDriver();
```

1. **Create an object** of ChromeDriver class and **store** it in 1 ref. variable **with reference** of WebDriver interface.
2. **Before** that we **need** to set path of chromedriver.exe file—System. present in java class

@ Web Driver- it is an **interface** use to **perform** action on browser

Using web driver to perform following actions on browsers-

1. Enter url
2. Maximize
3. Current tab close
4. Multiple tab close at a time

We can not perform action on browser element-
Ex search field, gmail link, images

Why web driver mention in statement object creation of chrome driver?
→To access the methods of web driver so that reason we are use web driver Variable name.(to get web driver method)→because we use selenium web driver or we are the requirements of web driver

Where to use upcasting topic in your selenium?
→At the time of browser opening that time we perform or use upcasting in selenium.

WebDriver driver = new ChromeDriver();
Instead of WebDriver we use **ChromeDriv**e is it possible to open the chrome browser, but in that case we run the script in only one browser.
WebDriver use – so same script run in different browser

Can we write this WebDriver driver = new ChromeDriver(); statement like ChromeDriver driver = new ChromeDriver();
→Yes

Which polymorphism use in selenium?
→Run time polymorphism

How to open a browser in selenium?
→To open a browser in selenium first we need to create an object of chrome driver with reference of web driver. Before writing this statement we need to set the path of chrome.exe file by using statement that is System.setProperty(key,value). In this statements this statement/ method accept two parameter first parameter accept the name of the browser and second parameter accept path of the browser

Methods of Web Driver

1. **get()**
   - This method is use to **open an application** or to **enter url** in a web page or browser.
   - Ex. **WebDriver driver = new ChromeDriver();**

     driver.get(“ url ”);

   - get() method **accept** the string input only
   - To perform action, return type void

Which method is used for url enter in selenium?
→get() method

1. **Webriver exception**- we will get web driver exception when url is not well formatted.
   Ex. driver.get(“ wrong url ”);

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example1
{
public static void main(String[] args)

{
     //open browser
System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
     WebDriver driver = new ChromeDriver();

     //get()-->to enter url in a browser

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

     //driver.get("https://www.amazon.com/");

}
}
```

2. **Close()**
    - This method is use to **close current tab** of the browser
    - Ex. driver.close();
    - To perform action, return type void

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example2
{
     public static void main(String[] args) throws
InterruptedException
     {
          //open browser
          System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

          Thread.sleep(2000);  //wait 3sec

          //get()-->to enter url in a browser
          driver.get("https://www.facebook.com/");

          Thread.sleep(2000);

          //close()-->to close the current tab
          driver.close();
     }
}
```

3. **Quite()**
    - This method is the **alternate method** to the close method
    - But the **difference** between the close and quite is that **close()** method **close the current tab** only & **quite()** method will **close all the tabs** of the browser.
    - Ex. driver.quite();
    - To perform action, return type void

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example3
{
     public static void main(String[] args) throws
InterruptedException

     {
          //open browser
      System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
                    WebDriver driver = new ChromeDriver();

                    Thread.sleep(2000);

                    //get()-->to enter url in a browser
                    driver.get("https://www.facebook.com/");

                    Thread.sleep(2000);

          //3.quite()-->to close all the tabs of the browser
                    driver.quit();
     }
}
```

Which method is used to close the browser?
→close() method

Any alternate method for close method?
→quite() method

get(String erg0):void→void meaning to perform action
getTitle():String→String meaning to return output→statement save and store

Note:-Scenario- to enter application url and to verify these application is open or not that time we use getTitle() method

What if you interrupt the browser while running test script?
→We will get "unreachable browser exception"

4. **getTitle()**
    - This method is use to **get title of the web page** as an output
    - Return type of getTitle() is **String→save** and **store** an object

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example4
{
     public static void main(String[] args) throws
InterruptedException
     {
          //open browser
          System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

          Thread.sleep(2000);

          driver.get("https://www.google.com/");

          Thread.sleep(2000);

//getTitle()-->is use to get title of the web page as an output

          String title=driver.getTitle();

          System.out.println(title);

          driver.close();
     }
}
```

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example4_1
{
     public static void main(String[] args) throws
InterruptedException
     {
```

```
            //open browser
            String expResult="Google";

            System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
            WebDriver driver = new ChromeDriver();

            Thread.sleep(2000);

            driver.get("https://www.google.com/");

            Thread.sleep(2000);

//getTitle()-->is use to get title of the web page as an output

            String actResult=driver.getTitle();

            System.out.println(actResult);

            if(actResult.equals(expResult))
            {
               System.out.println("navigated to correct webpage");
            }
            else
            {
               System.out.println("navigated to wrong webpage");
            }
      }
}
```

5. **getCurrentUrl()**
   - This method is use to **get url of the current web page** as an output.
   - Return type of getCurrentUrl() is **String→save** and **store** an object

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example5
{
     public static void main(String[] args) throws
InterruptedException

     {
```

```
            //open browser
            System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
                        WebDriver driver = new ChromeDriver();

                        Thread.sleep(2000);

                        driver.get("https://www.google.com/");

                        Thread.sleep(2000);

                  //5.getCurrentUrl()-->get url of current web page

                  String url = driver.getCurrentUrl();

                        System.out.println(url);
      }
}
```

6. **mazimize()**
   - This method is used to **maximize the browser**

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example6
{
      public static void main(String[] args)
      {
            //open browser
            System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
            WebDriver driver = new ChromeDriver();

            //maximize()-->used to maximize the browser

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

            driver.get("https://www.google.com/");
      }
}
```

Can we minimize the browser using selenium?
→We can't minimize the browser using selenium, but we can change the size and position of the browser

7. **navigate()**
    - This method is to **open an application, move forward, backward and refresh** the browser.
    - Navigate method can be used for **alternate method** for get method
    - Can't pass input directly to the navigate to call the **to** function

Is any alternate method for url enter? Instead of get()?
→navigate() method

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example7
{
     public static void main(String[] args) throws
InterruptedException
     {
          //open browser

     System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

                    Thread.sleep(3000);

                //6.maximize()-->used to maximize the browser

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

                    Thread.sleep(3000);

//7.navigate()-->open an application, move forward, backward and refresh the browser.

     driver.navigate().to("https://www.google.com/");
```

```
                    Thread.sleep(3000);
    driver.navigate().to("https://www.facebook.com/");
                    Thread.sleep(3000);
                    driver.navigate().back();
                    Thread.sleep(3000);
                    driver.navigate().forward();
                    Thread.sleep(3000);
                    driver.navigate().refresh();
    }
}
```

8. **setSize()**
    - this method is use to **change size of the browser** which **accepts** dimensions arguments
    - setSize() method can't accept directly the width and height of the browser. This method accept only dimension arguments.
    - Before using setSize() method we need to **create the object** of the dimension class and then **pass** the width and height of the browser in the **constructor** of the dimension class
    - Dimension class **present** in the selenium
    - Type dime----use control+space
    - To enter width and height value is the browser resolution or picsal value

```
package webdriver_methods;

import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example8
{
    public static void main(String[] args)
    {
        //open browser
```

```
            System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);

            WebDriver driver = new ChromeDriver();

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

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

            //8.setSize()

            Dimension d = new Dimension(200, 500);

            driver.manage().window().setSize(d);
        }
}
```

9. **getSize();**
    - check the **browser size** immediate open of browser
    - to check the size **to call** the printing statement and **inside** the printing statement **to call** the function driver.manage().window().getSize()
    - **return** dimension arguments in terms of width and height

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example9
{
        public static void main(String[] args)
        {
            //open browser
            System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
            WebDriver driver = new ChromeDriver();

            //9.getSize()

        System.out.println(driver.manage().window().getSize());

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

        System.out.println(driver.manage().window().getSize());
```

```
    }
}
```

10.setPosition()

- This method is use to **change position** of the browser which **accept** point argument
- Before using setPosition() method we need to **create the object** of the point class and then **pass** the x and y coordinate of the browser in the **constructor** of the point class
- point class **present** in the selenium
- Type poi----use control+space

```
package webdriver_methods;

import org.openqa.selenium.Dimension;
import org.openqa.selenium.Point;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example10
{
    public static void main(String[] args) throws
InterruptedException
    {
        //open browser
        System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
        WebDriver driver = new ChromeDriver();

        Thread.sleep(3000);

        Dimension d2 = new Dimension(0, 0);

        driver.manage().window().setSize(d2);

        Thread.sleep(3000);

        //10.setPosition()

        Point p = new Point(500, 500);

        driver.manage().window().setPosition(p);
    }
}
```

11.getPosition()

- **check the position** of the browser in terms of x and y coordinates
- to get this coordinate so to **call** the printing statement and **inside** the printing statement to **call** the function

  driver.manage().window().getPosition()
- **return** point arguments in terms of x and y arguments

```
package webdriver_methods;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class example11
{
     public static void main(String[] args) throws
InterruptedException
     {
          //open browser
          System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

          Thread.sleep(3000);

          // getPosition()

     System.out.println(driver.manage().window().getPosition());
     }
}
```

Which type of exception handle or observe in selenium?

1. **Webriver exception**- we will get web driver exception when url is not well formatted.
   Ex. driver.get(“ wrong url ”);

2. **What if you interrupt the browser while running test script?**

→We will get “unreachable browser exception”

Web driver methods

Variable name / object name.methodname();

Variable name / object name = driver

1. .get();

```
driver.get("enter url");
```

2. .close()

```
driver.close();
```

3. .quite()

```
driver.quite();
```

4. .getTitle()

```
String title=driver.getTitle();
Syso(title);
```

5. .getCurrentUrl()

```
String url=driver.getCurrentUrl();
Syso(url);
```

6. .maximize()

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

7. .nevigate()

```
driver.nevigate().to("enter url")
driver,navigate().back();
driver.nevigate().forward();
driver,navigate().refresh();
```

8. .setSize()

```
Dimension d = new Dimension(width value, height value)
driver.manage().window().setSize(d);
```

9. .getSize()

```
Syso(driver.manage().window().getSize());
```

10..setPosition()

```
Point p = new Point(int x, int y)
driver.manage().window().setPosition(p);
```

11..getPosition()

```
Syso(driver.manage().window().getPosition());
```

12.Thread.sleep(int miles)→Java Class used for pause or wait purpose

Html Coding

1. **Hypertext markup language used** for creating a webpage
2. Html coding is **not** case sensitive
3. We can **write** html coding in notepad / notepad++
4. While **saving** html, extension of file should be "filename.html"
5. the meaning of **use** of / before name to close the sentence
6. to open a webpage →<html>→open html & </html>→close html
7. Give title or name of webpage between <title>→open title & </title>→close title. This is dummy webpage
8. To mention some info to the webpage body so to enter the value between <body>→open body & </body>→close body
9. </br>→break to text break and give in the next line

Example of html coding

1. **Write a html code to create dummy webpage**

Webpage

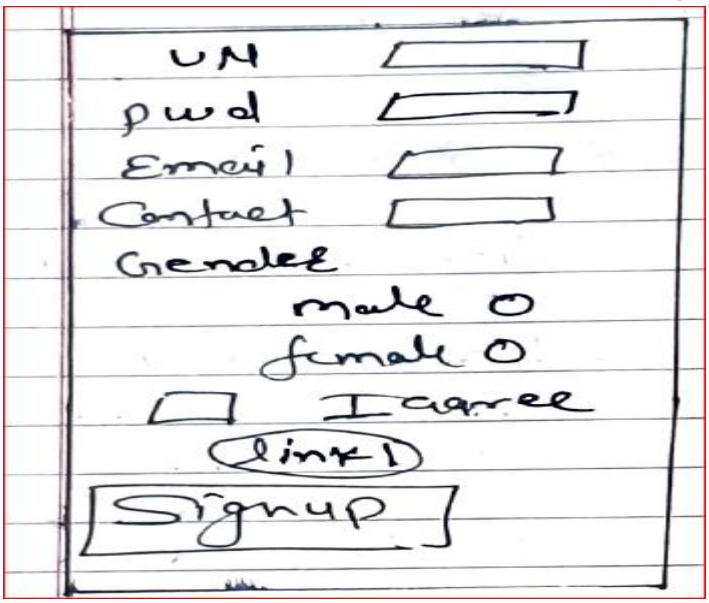
```
<html>
    <title>
        webpage
    </title>
</html>
```

2. **Write a html code to create following webpage**

webpage
Hi Good morning

```
<html>
        <title>
                 webpage
        </title>
        <body>
                Hi
                Good Morning
        </body>
</html>
```

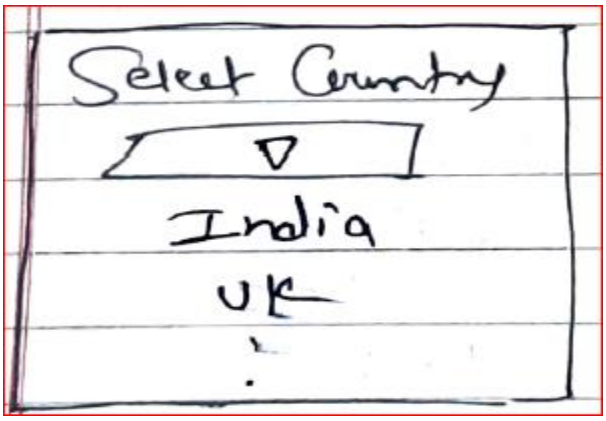
3. **Write a html code to create following login webpage**




```
<html>
        <title>
                login page
        </title>
        <body>
                UN<input type = 'text'> </br>
                PWD<input type = 'password'> </br>
```

```
            Email<input type = 'text'> </br>
            Mob No<input type = 'text'> </br>
            Gender </br>
                Male<input type = 'radio'> </br>
                Feamale<input type = 'radio'> </br>
            <input type = 'checkbox'> I agree </br>
            <a href = "https://www.facebook.com/">link1</a> </br>
            <input type = 'button' value='sign up'
        </body>
</html>
```

4. **Write html code to create listbox**




```
        <html>
            <title>
                Listbox
            </title>
            <body>
                select country
                <select>
                    <option>Ind</option>
                    <option>Aus</option>
                    <option>Sri</option>
                    <option>Pak</option>
                </select>
            </body>
        </html>
```

5. **Write a html code for create web table**

Sr NO.	Book Type	Cost
1	Manual	100
2	SQL	200
3	Java	300

```
<html>
        <title>webtable</title>
        <body>
                <table border = 3>
                        <tr>
                                <th>Sr no</th>
                                <th>Book type</th>
                                <th>Cost</th>
                        </tr>
                        <tr>
                                <td>1</td>
                                <td>Manual</td>
                                <td>100</td>
                        </tr>
                        <tr>
                                <td>2</td>
                                <td>Selenium</td>
                                <td>200</td>
                        </tr>
                        <tr>
                                <td>2</td>
                                <td>Java</td>
                                <td>300</td>
                        </tr>
                </table>
        </body>
</html>
```

Summary

1. To **create a webpage** we need to **use** keyword "**title**"
2. **Every keyword** should be **closed** within **angular brace** using forward slash**(/)**
3. To **create** a **component** or **element** we need to use a keyword "**input**"
4. To **create list** we need to use "**select**" keyword
5. To **create link** we need to use keyword "**a**" with **href** keyword
6. To **create a web table** we need to use keyword "**table**"
7. To create image we need to use "**img**" keyword

1. **Tagname**
   - Any keyword which is **present** immediately **after** angular brace**(<)** less than symbol
   - Eg. html, title, body, tr, table

2. **Attribute**
   - Any keyword which is **present** immediately **after** tagname with equal to symbol **until** greater than symbol
   - Syntax property name = property value
   - **Attribute name = attribute value**
   - Eg. type='text' , id='1234', class='abc', name='xyz'

3. **Text**
   - Any keyword which is **present** in between angular brace**(>)** greater than symbol & angular brace **(<)** less than symbol is known as Text
   - Eg. >sr no<, >link<, >manual<, >java<

Why html coding is required in selenium

- To **identify** an element uniquely and to **perform** action with the help of selenium html coding is necessary

Locators

I. Locators are **used** to identify an element **with** the help of locator types

II. To **identify** an element **present** in webpage we **need** to use 'findElement' method which is **present** in web driver.

Ex. WebDriver driver = new ChromeDriver
Driver.findElement (By arg)

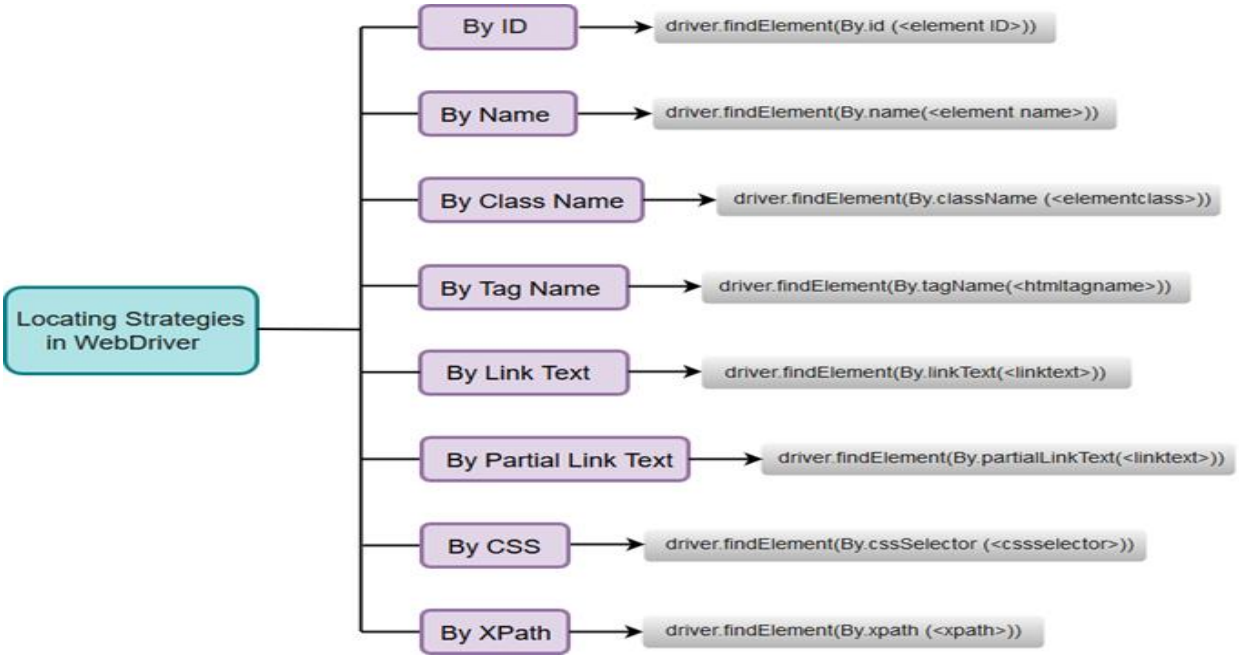
III. findElement **method** will identifies an element **with** the help of By class **which** contains Static methods

IV. All the static methods **present** in By class are **known as** locator types

Locator types

1. Tagname →tagname
2. Id →attribute
3. Name →attribute
4. Class name →attribute
5. Linktext →text
6. Partial linktext →text
7. Css selector →expression
8. Xpath →expression

- All the locator types takes **String arg** as an input & return type is by
  By ele = By.tagname(String arg);
- **Return type** of findElement() is webelement



The following table lists all the Java syntax for locating elements in Selenium WebDriver.

Method	Syntax	Description
By ID	driver.findElement(By.id (<element ID>))	Locates an element using the ID attribute
By name	driver.findElement(By.name (<element name>))	Locates an element using the Name attribute
By class name	driver.findElement(By.className (<element class>))	Locates an element using the Class attribute
By tag name	driver.findElement(By.tagName (<htmltagname>))	Locates an element using the HTML tag
By link text	driver.findElement(By.linkText (<linktext>))	Locates a link using link text
By partial link text	driver.findElement(By.partialLink Text (<linktext>))	Locates a link using the link's partial text
By CSS	driver.findElement(By.cssSelecto r (<css selector>))	Locates an element using the CSS selector
By XPath	driver.findElement(By.xpath (<xpath>))	Locates an element using XPath query

8. Xpath

Types of Xpath

1 Absolute xpath
2 Relative Xpath
3 Xpath by Attribute
4 Xpath by text
5 Xpath by contains
6 Xpath by index

3. Xpath by attribute

Advantage:-

We can **identify** element by **using** attribute **without** finding html tree diagram
Syntax → //tagname[@attribute name = attribute value]

Attribute name = attribute value
Property name = property value
Ex. //input[@id ='abc'

```
<html>
        <body>
                UN<input id = 'abc'   type = 'text'>
                PWD<input id = 'xyz'   type = 'pass'>
        </body>
</html
```

How to identify Xpath by attribute?
→ Syntax → //tagname[@attribute name = attribute value]

Steps to enter UN and PW by using Script in the webpage

1. Open web browser
2. Open Facebook page
3. Right click on the UN and click on inspect
4. Click on element
5. Click on left side arrow
6. Then check UN textbox color blue
7. Control+f
8. Open window
9. Select UN attribute and double click on the attribute value and copy control+c
10. Click on open window and create xpath expression by using formula
11. //tagname[@atribute name = attribute value]
12. After the cratering x path expression if it is match then convert blue color to yellow color or 1 of 1
13. Copy this path and paste to script driver.findelement(by.xpath.(" //tagname[@atribute name = attribute value]")).sendkeys("enter user as per your account UN ex-abc@1234")
14. Same steps follow for PW
15. Same steps follow for login change is replace sendkeys to click()

```java
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class XpathByAttribute1
{
     public static void main(String[] args) throws
InterruptedException
     {
          //to open a chrome browser

     System.setProperty("webdriver.chrome.driver","C:\\Users\\HP
\\Downloads\\chromedriver_win32\\chromedriver.exe");
          WebDriver driver = new ChromeDriver();

          //wait or pause
          Thread.sleep(5000);

          //enter url or open application
          driver.get("https://www.facebook.com/");

          //wait or pause
          Thread.sleep(5000);

          //to enter UN

driver.findElement(By.xpath("//input[@id='email']")).sendKeys("m
angeshchewale5@gmail.com");

          //to enter PWD

driver.findElement(By.xpath("//input[@id='pass']")).sendKeys("94
23350719");

          //to click on login
driver.findElement(By.xpath("//button[@name='login']")).click();

     }
}
```

4. **Xpath by text**

- Sometimes developer may create an element by using tagname and text.
- To identify that text we can't use locator type i.e xpath by attribute.

Syntax= //Tagname[text()='text value']

How to identify Xpath by text?

→ Syntax= //Tagname[text()='text value']

- By using xpath by text we can identify normal text and link text also

```
<html>
	<title>
		webpage
	</title>
	<body>
		<a href = "https://www.facebook.com/">facebook</a> </br>
		<a href = "https://www.google.com/">google</a>
	</body>
</html>
```

Ex. //a[text()='facebook']
//a[text()='google']

```
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class XpathBytext
{
	public static void main(String[] args) throws
InterruptedException
	{
		//open the chrome browser
	System.setProperty("webdriver.chrome.driver","C:\\Users\\HP
\\Downloads\\chromedriver_win32\\chromedriver.exe");

		WebDriver driver = new ChromeDriver();
		//wait or pause
```

```
            Thread.sleep(3000);

            //enter url of webpage
            driver.get("file:///C:/Users/HP/Documents/ex.html");

            Thread.sleep(3000);

//click on the link like facebook right click and inspect
driver.findElement(By.xpath("//a[text()='facebook']")).click();
      }
}

package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public abstract class XpathBytest1
{
      public static void main(String[] args) throws
InterruptedException {

            //To open chrome brower

            System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
            WebDriver driver = new ChromeDriver();

            //To wait or pause
            Thread.sleep(4000);

            //To open or enter url facebook
            driver.get("https://www.facebook.com/");

            //To wait or pause
            Thread.sleep(4000);

//To click on the forgotten password right click and inspects on
the facebook page
driver.findElement(By.xpath("//a[text()='Forgotten
password?']")).click();

      }
}
```

```java
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class XpathBytext2
{
     public static void main(String[] args) throws
InterruptedException
     {
     //To open chrome browser

     System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
     WebDriver driver = new ChromeDriver();

     //after open the browser wait 4sec

     Thread.sleep(4000);

     //to open or enter facbook url

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

     //after open the facebook page wait 4sec

     Thread.sleep(4000);

//to click on create new account on facebook page and right
click on any other element and inspect

driver.findElement(By.xpath("//a[text()='Create New
Account']")).click();
     }
}
```

NOTE-

- If any element **contains** text with spaces while identifying that element using Xpath by text. We **need** to mention space also
  ex. abc
  //span[text()=’ abc ’]
- Space can be **created** by using two ways
  1. Keyboard stroke
  2. Non-breakable space
- If space is **created** by using non-breakable spaces **within** the text. Then we can’t use xpath by text
- So we need to **go** for next type i.e xpath by contains

5. Xpath by contains

- "contains()" is **used** to identify an web element, when we are familiar with some part of the attributes value of an element.
- Xpath by contains **using** two ways
  1. Xpath by attributes
  2. Xpath by text
- **When to use Xpath by contains?**
  1. If the attribute values are long
  2. If the attributes are dynamic id
  3. If space is created by non-breakable space within the text
- Syntax of Xpath by contains by using attributes
  //tagname[contains(@attribute name,’attribute value’)]

- Syntax of Xpath by contains by using text
  //tagname[contains(text(),’text value’)]
- Dynamic id occurs means in the attribute value first some character are constant after some character are changes. So that time we use Xpath by contains with attribute.
- If space is created by non-breakable space within the text. So that time we use Xpath by contains with text.
- At the time of copy all the attribute value instead of that we have to use substring of attribute value in any substring first, middle and last.

```
<html>
  <body>
    UN<input type='text', id='abcdefgh'></br>
    PWD<input type='text', id='xyz123', name='abc123'></br>
    <a href='https://www.facebook.com/'>  facebook  </a>
  </body>
</html>
```

UN

PWD

facebook

```
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

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

     { //Xpath by contains with attributes and text
          //To open browser
     System.setProperty("webdriver.chrome.driver","C:\\Users\\HP
\\Downloads\\chromedriver_win32\\chromedriver.exe");

          WebDriver driver = new ChromeDriver();

          //To pause after opening browser
          Thread.sleep(3000);

          //To open webpage
          driver.get("file:///C:/Users/HP/Documents/ex1.html");

          //To pause after opening webpage
          Thread.sleep(3000);
```

```
          //To enter UN
driver.findElement(By.xpath("//input[contains(@id,\"cdef\")]")).
sendKeys("mangesh123");

          //To pause after entering UN
          Thread.sleep(3000);

          //To enter PWD
driver.findElement(By.xpath("//input[contains(@name,\"c12\")]"))
.sendKeys("abc");

          //To pause after entering PWD
          Thread.sleep(3000);

          //To click on facebook link
driver.findElement(By.xpath("//a[contains(text(),\"boo\")]")).cl
ick();
     }
}
```

6. **Xpath by Index**

- To **create or select** unique xpath for that **reasons** we **use** xapth by index.
- If multiple matching are **present** so that time we **use** xapth by index.
- At the time of multiple matching it's not need to create unique path of first element present in the webpage because selenium automatically every time default action perform on the first element.
- **Syntax** = (xpath expression)[index]

  (//tagname[@attribute name='attribute value'])[index]

  //tagname[@attribute name='attribute value']-xpath by attribute
- 1 of 3 match first then click on yellow path and then click and Control+f window and enter then see element match to path 1 of 1 or to enter index then to see also match 1 of 1.

```
<html>
 <body>
      FN<input type='text'></br>
      LN<input type='text'></br>
      Email<input type='text'></br>
      <a href="https://www.facebook.com/">facebook</a>
 </body>
</html>
```

FN

LN

Email

facebook

```
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class XpathByIndex1
{
     public static void main(String[] args) throws
InterruptedException
     {
          //To open browser
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Dow
nloads\\chromedriver_win32\\chromedriver.exe");
          WebDriver driver = new ChromeDriver();

          //To pause after opening the browser for 3sec
          Thread.sleep(3000);

          //To open/enter url webpage created by using html
          driver.get("file:///C:/Users/HP/Documents/ex2.html");

          //To pause after opening webpage for 3sec
          Thread.sleep(3000);

          //To enter FN
driver.findElement(By.xpath("//input[@type=\"text\"]")).sendKeys
("abc");

          //To pause after enter FN for 3sec
          Thread.sleep(3000);

          //To enter LN
driver.findElement(By.xpath("(//input[@type=\"text\"])[2]")).sen
dKeys("xyz");

          //To pause after enter LN for 3sec
          Thread.sleep(3000);
```

```
          //To enter Email
driver.findElement(By.xpath("(//input[@type=\"text\"])[3]")).sen
dKeys("abc@123");

          //To pause after enter Email for 3sec
          Thread.sleep(3000);

          //To click on facebook link
driver.findElement(By.xpath("//a[text()='facebook']")).click();

     }
}
```

Real time example of xpath by index----open facebook---click on create new account----enter FN---enter surname

```
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class XpathByIndex2
{
     public static void main(String[] args) throws
InterruptedException
     {
          //To open browser
System.setProperty("webdriver.chrome.driver","C:\\Users\\HP\\Dow
nloads\\chromedriver_win32\\chromedriver.exe");
          WebDriver driver = new ChromeDriver();

          //To pause after opening the browser for 3sec
          Thread.sleep(3000);

          //To open or enter url of facebook
          driver.get("https://www.facebook.com/");

          //To pause after opening the facebook for 3sec
          Thread.sleep(3000);

          //To click on create new account
driver.findElement(By.xpath("//a[text()=\"Create New
Account\"]")).click();
```

```
    //To pause after click on the create new account for 3sec
        Thread.sleep(3000);

        //To enter FN
driver.findElement(By.xpath("(//input[@type=\"text\"])[2]")).sen
dKeys("abc");

        //To pause after enter FN for 3sec
        Thread.sleep(3000);

        //To enter surname
driver.findElement(By.xpath("(//input[@type=\"text\"])[3]")).sen
dKeys("xyz");

        //To pause after enter surname for 3sec
        Thread.sleep(3000);

        driver.quit();
    }
}
```

1. **Absolute xpath**

- Absolute xpath is **use** to navigate **form** root of the parent to immediate child.
- To **achieve** absolute xpath we need to **use** single forward slash (/).

Disadvantages of Absolute xpath

- Xpath is to **lengthy** and **time** consuming.
- Identify of an element by **developing** html tree diagram is difficult.

2. **Relative xpath**

- Relative xpath is **use** to navigate **form** root of the parent to any child.
- To **achieve** relative xpath we need to **use** double forward slash (//).

Disadvantages of Relative xpath

- Identify of an element by **developing** html tree diagram is difficult.

Other locator types-

1. **tagName**

   Disadvantages-

   If a **webpage** multiple elements are presented with **same tagName** and we **use** tagName locator type to **identify** an element then **selenium** perform action on **first** element present in webpage.

   <html>
   <body>
   UN<input type = 'text'> </br>
   PWD<input type = 'password'>
   </body>
   </html>

   UN

   PWD

```
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class tagName
{
     public static void main(String[] args) throws
InterruptedException
     {
          System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

          Thread.sleep(3000);

     driver.get("file:///C:/Users/HP/Documents/tagname.html");

          Thread.sleep(3000);

     driver.findElement(By.tagName("input")).sendKeys("mac");
```

```
            Thread.sleep(3000);

        driver.findElement(By.tagName("input")).sendKeys("cm123");

        }
}
```

2. **id**

- This locator type is **use** if any element html **contains** id attributes
- When we **can't use id** as a locator type
  - When id attributes is **not present**
  - When id attributes is **duplicate**
- Ex. locator for Contact→(By.id("abc")) in this case selenium perform action on PWD

```
<html>
<body>
        UN<input type = 'text', id='1234'> </br>
        PWD<input type = 'password', id='abc'> </br>
        Contact<input type = 'text', id='abc'>
</body>
</html>
```

UN

PWD

Contact

```
package Locators;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

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

```
            System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
            WebDriver driver = new ChromeDriver();

            Thread.sleep(3000);

            driver.get("file:///C:/Users/HP/Documents/id.html");

            Thread.sleep(3000);

            driver.findElement(By.id("1234")).sendKeys("abc");

            Thread.sleep(3000);

            driver.findElement(By.id("abc")).sendKeys("xyz");

            Thread.sleep(3000);

      driver.findElement(By.id("abc")).sendKeys("9404624368");

      }
}
```

3. **className**

- When **we can use** className as locator type
  - If id is duplicate
  - If id attribute is not present
- When **we can't use** className
  - className is not present
  - className is duplicate

```
<html>
<body>
      UN<input type = 'text', id='1234', class='abc'> </br>
      PWD<input type = 'password', id='abc', class='xyz'> </br>
      Contact<input type = 'text', id='abc', class='abc'> </br>
</body>
</html>
```

UN

PWD

Contact

```
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

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

          System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

          Thread.sleep(3000);

     driver.get("file:///C:/Users/HP/Documents/classname.html");

          Thread.sleep(3000);

     driver.findElement(By.className("abc")).sendKeys("abc");

          Thread.sleep(3000);

     driver.findElement(By.className("xyz")).sendKeys("xyz");

          Thread.sleep(3000);

driver.findElement(By.className("abc")).sendKeys("9404624368");

     }
}
```

4. **name**

- When **we can use** name as locator type
  - If id & className is duplicate
  - If id attribute & className is not present
- When **we can't use** className
  - name is not present
  - name is duplicate

```
<html>
<body>
	UN<input type = 'text', id='1234', class='abc', name='abc1'> </br>
	PWD<input type = 'password', id='abc', class='xyz',name='xyz1'> </br>
	Contact<input type = 'text', id='abc', class='abc', name='abc1'> </br>
</body>
</html>
```

UN

PWD

Contact

```
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class name
{
	public static void main(String[] args) throws InterruptedException
	{
		System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe");
		WebDriver driver = new ChromeDriver();

		Thread.sleep(3000);

		driver.get("file:///C:/Users/HP/Documents/nam.html");
```

```
            driver.findElement(By.name("a1")).sendKeys("cm123");

            driver.findElement(By.name("x1")).sendKeys("123");

            driver.findElement(By.name("a1")).sendKeys("9404624368");
        }
}
```

5. **linkText** **6.partialLinkText**

- If **tagName** is **duplicate** id, className, name **attribute** are **not present** in **html** code of an element then we should **use** linkText/partialLinkText.
- **linkText- used** to identify an element by **taking entire text** as an **input.**
- **patialLinkText- used** to identify an element by **taking few character** of a **text** as an **input.**
- linkText and partialLinkText locators type are **used** to identify elements with the **help** of linkText **present** in links but not for normal text.

```
<html>
<body>
        <a href="https://www.facebook.com/">facebook</a></br>
        <a href="https://www.google.com/">google</a></br>
</body>
</htm>
```

facebook

google

//linkText

```
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

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

```
	{

		System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
		WebDriver driver = new ChromeDriver();

		Thread.sleep(3000);

driver.get("file:///C:/Users/HP/Documents/linktextpartial.html")
;
		Thread.sleep(3000);

		driver.findElement(By.linkText("google")).click();

		Thread.sleep(3000);

		driver.navigate().back();

		Thread.sleep(3000);

		driver.findElement(By.linkText("facebook")).click();
	}
}
```

//partialLinkText

```
package Locators;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

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

		System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
		WebDriver driver = new ChromeDriver();

		Thread.sleep(3000);
```

```
	driver.get("file:///C:/Users/HP/Documents/linktextpartial.h
tml");
		Thread.sleep(3000);

		driver.findElement(By.partialLinkText("ce")).click();

		Thread.sleep(3000);

		driver.navigate().back();

		Thread.sleep(3000);

		driver.findElement(By.partialLinkText("go")).click();
	}
}
```

Web elements and its methods

- **Webelement**- It is an interface **use** to perform **action** on **element** present in a browser

Webelement Methods-

1. **sendKeys():void-WebElement**

- This method is **use** to **enter** value in the **input field**.
- After the entering findElement(By.locator type()) method **.** is mention then after that some methods are present each and every method is an webelement method.
- **1st Approach**- driver.findElement(By.xpath("xpathExpression"))→to find the particular element address identify and sendKeys("enter value")→identify find address to perform action
- In the 1st approach two action perform in same statement.
- **2nd approach**- to identify element and store in variable → WebEelement UN= driver.findElement(By.xpath("xpathExpression"))-return type webEle. to perform action on find element—object name.sendKeys("enter value")
- In the 2nd approach two actions perform in different statement
- **What is return type of findElement?**
  WebElement is a return type of findElement.

```java
package webelement_method;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class sendKeys
{
     public static void main(String[] args) throws
InterruptedException
     {
          //To open browser

          System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

          //To pause of wait after opening browser
          Thread.sleep(3000);

          //To open/enter url facebook
          driver.get("https://www.facebook.com/");

          //To pause of wait after opening facebook
          Thread.sleep(3000);

          //To enter UN or email on facebook
          //1st approach--both action perform on same statement

     //driver.findElement(By.xpath("//input[@id='email']")).send
Keys("abc");

          //2nd approach-action perform on different statement
WebElement UN =
driver.findElement(By.xpath("//input[@id='email']"));
          UN.sendKeys("abc");
     }
}
```

2. **clear():void-WebElement**
   - This method is **use** to **clear** value in the **text field**.
   - **1st approach**-identify and perform action in same statement.
   - In one element only perform **one action** so that time we have to use **1st approach**
   - **2nd approach**-identify and perform action on different statement
   - One element to perform **multiple action** so that time we have to use **2nd approach**.
   - By using Object name/UN we have perform multiple action on one element.

```
package webelement_method;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class clear
{
     public static void main(String[] args) throws
InterruptedException
     {
          //To open browser
          System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

          //To pause of wait after opening browser
          Thread.sleep(3000);

          //To open/enter url facebook
          driver.get("https://www.facebook.com/");

          //To pause of wait after opening facebook
          Thread.sleep(3000);

          //To enter UN or email on facebook
//first approach
     //driver.findElement(By.xpath("//input[@id='email']")).send
Keys("abc");
```

```
          //To pause of wait after entering UN
          //Thread.sleep(3000);

          //To clear()

     //driver.findElement(By.xpath("//input[@id='email']")).clea
r();

//second approach
          WebElement UN =
driver.findElement(By.xpath("//input[@id='email']"));
          UN.sendKeys("abc");
          Thread.sleep(3000);
          UN.clear();
          Thread.sleep(3000);
          UN.sendKeys("xyz");
          Thread.sleep(3000);
          UN.clear();

     }
}
```

3. **click ():void-WebElement**

- Click method is **use** to **click** on button, link also **use** to **select** radio buttons and check box.
- **1st approach**-identify and perform action in same statement.---better to use
- In one link only perform **one action** so that time we have to use 1st approach
- **2nd approach**-identify and perform action on different statement
- One link to perform **multiple action** so that time we have to use 2nd approach.

```
package webelement_method;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

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

```
            //To open browser
            System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
            WebDriver driver = new ChromeDriver();

            //To pause of wait after opening browser
            Thread.sleep(3000);

            //To open/enter url facebook
            driver.get("https://www.facebook.com/");

            //To pause of wait after opening facebook
            Thread.sleep(3000);

            //To click on forgotten password
            //driver.findElement(By.xpath("//a[text()='Forgotten
password?']")).click();

            WebElement forgotten =
driver.findElement(By.xpath("//a[text()='Forgotten
password?']"));
            forgotten.click();
      }
}
```

4. **getText():String-WebElement**
   - This method is **use** to **get** text present in a webpage
   - **Return type** of getText function is **String**
   - **What is return type of getText()?**
     String
   - 1st approach
     - To identify i/p field or element and to perform action on that in one statement means to call findEelment() method after that to call getText function of webelement.
     - This above statement store in one object ex.text and the return type of object is String.
     - Then to call printing statement and inside the printing statement to call store object ex.text
   - 2nd approach
     - 1st to identify i/p filed or element and store in the one object/variable and the return type of object is WebElement.

- Then to call the function of webelement by using objectname.function/method() and these are store in one object and the return type of object is String.
- Then to call printing ststement and inside the printing statement to call the store object.

- To identify element and getText and print in one statement is possible. To call direct printing statement and inside the printing statement to call findEement method and getText function of webelement.

```
package webelement_method;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class getText
{
     public static void main(String[] args) throws
InterruptedException
     {
          //To open browser
          System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

          //To pause of wait after opening browser
          Thread.sleep(3000);

          //To open/enter url facebook
          driver.get("https://www.facebook.com/");

          //To pause of wait after opening facebook
          Thread.sleep(3000);

          //To getText - forgotten password
//String text =
driver.findElement(By.xpath("//a[text()='Forgotten
password?']")).getText();
          //System.out.println(text);

          //second approach
```

```
            WebElement forgotten =
driver.findElement(By.xpath("//a[text()='Forgotten
password?']"));
            String text = forgotten.getText();
            System.out.println(text);
      }
}
```

5. **isEnabled():boolean**
   - This method is **use** to **verify** element is **enabled** or **disabled**.
   - **Return type** of isenabled function **boolean.**
   - If element is **enabled** then **it returns true** otherwise it **returns false**
   - **How to verify button or element is enabled or disabled?**
     By using isEnabled() function
   - **1st approach**
     - To identify element then to verify element to call isEnabled function.
     - This above statement is store in one object or variable ex. object name is result the return type of object is boolean.
     - Then to call printing statement and inside the printing statement call store object.
     - To mention verification by using if else statement means if result is true then print element is enabled and if result is false print element is disabled.
   - **2nd approach**
     - To identify an element and store in one object and return type of object is WebElement.
     - Then call the function of webelement –objectname.function/methd
     - Then these above statement store in one object and return type of object is boolean.
     - Then call to printing statement and inside the printing statement to call store object.
     - To mention verification by using if else statement means if result is true then print element is enabled and if result is false print element is disabled.

```
package webelement_method;
```

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class isEnabled
{
//To verify/identify facebook login button enabled or disabled

     public static void main(String[] args) throws
InterruptedException
     {
          //To open browser
          System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
          WebDriver driver = new ChromeDriver();

          //To pause of wait after opening browser
          Thread.sleep(3000);

          //To open/enter url facebook
          driver.get("https://www.facebook.com/");

          //To pause of wait after opening facebook
          Thread.sleep(3000);

          //approach 1st
          //boolean result =
driver.findElement(By.xpath("//button[text()='Log
In']")).isEnabled();
          //System.out.println(result);

          //to mention verification
          //if(result==true)
          //{
          //   System.out.println("element is enabled");
          //}
          //else
          //{
          //   System.out.println("element is disabled");
          //}

          //approach 2nd
               WebElement loginButton =
driver.findElement(By.xpath("//button[text()='Log In']"));
```

```
            boolean result = loginButton.isEnabled();
            System.out.println(result);

        //to mention verification
            //if(result==true)
            if(loginButton.isEnabled())
            {
                System.out.println("element is enabled");
            }
            else
            {
                System.out.println("element is disabled");
            }
        //close facebook tab
        driver.close();

    }
}
```

6. **isSelected():boolean**
   - This method is **use** to **verify** radio button/checkbox is **selected or not**.
   - **Return type** of isSelected function is **boolean**.
   - If **radio button/checkbox** is **selected** then it **returns true** otherwise it **returns false.**
   - **1st approach**
     - To identify element then to verify element to call isSelected function.
     - This above statement is store in one object or variable ex. object name is result the return type of object is boolean.
     - Then to call printing statement and inside the printing statement call store object.
   - 2nd approach same as isEnabled function

```
package webelement_method;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class isSelected
{
```

```
     //To verify/identify click on create new account button on
facebook inside that verify female radio button is selected or
not

          public static void main(String[] args) throws
InterruptedException
          {
               //To open browser
               System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
               WebDriver driver = new ChromeDriver();

               //To pause of wait after opening browser
               Thread.sleep(3000);

               //To open/enter url facebook
               driver.get("https://www.facebook.com/");

               //To click on create new account
               driver.findElement(By.xpath("//a[text()='Create
New Account']")).click();

               //To pause of wait after opening browser
               Thread.sleep(3000);

               //To identify element and check radio button is
selected or not
               boolean result =
driver.findElement(By.xpath("(//input[@class='_8esa'])[1]")).isS
elected();
               System.out.println(result);
          }
}
```

False

7. **isDisplayed():boolean**
   - This method is **use** to **verify** element is **present or not**.
   - **Return type** of isDisplayed function is **boolean.**
   - If **element** is **present** then it **returns true** otherwise it **returns exception**.

- **1st approach**
  - To identify element then to verify element to call isDisplayed function.
  - This above statement is store in one object or variable ex. object name is result the return type of object is boolean.
  - Then to call printing statement and inside the printing statement call store object.
  - To mention verification by using if else statement means if result is true then print element is displayed and if result is false print element is not displayed.
  - If element are not get then result is NoSuchELementException so that time to handle exception by using try and catch block.
  - In try block to mention exception occur line code and in catch()-bracket mention exception name and call printing statement type any ex, exception handeled.

```
package webelement_method;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class isDisplayed
{

     //To verify/identify facebook logo displayed or not

          public static void main(String[] args) throws
InterruptedException
          {
               //To open browser
               System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
               WebDriver driver = new ChromeDriver();
               //To pause of wait after opening browser
               Thread.sleep(3000);

               //To open/enter url facebook
               driver.get("https://www.facebook.com/");
```

```
                //1st approach
                boolean result =
driver.findElement(By.xpath("//img[@class='fb_logo _8ilh
img']")).isDisplayed();
                System.out.println(result);

                //to mention verification
                if(result==true)
                {
           System.out.println("element displayed on webpage");
                }
                else
                {
     System.out.println("element not displayed on webpage");
                }
           }
}
```

```
true
element displayed on webpage
```

@if element is not getting scenario

```
package webelement_method;

import java.util.NoSuchElementException;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class isDisplayedException
{

     //To verify/identify facebook logo displayed not get

          public static void main(String[] args) throws
InterruptedException
           {
                //To open browser
                System.setProperty("webdriver.chrome.driver",
"C:\\Users\\HP\\Downloads\\chromedriver_win32\\chromedriver.exe"
);
                WebDriver driver = new ChromeDriver();

                //To pause of wait after opening browser
                Thread.sleep(3000);
```

```
                //To open/enter url facebook
                driver.get("https://www.facebook.com/");

                boolean result = false;

                try
                {
                 result =
driver.findElement(By.xpath("//img[@class='  fb_logo _8ilh
img']")).isDisplayed();
                }
                catch(NoSuchElementException e)
                {
                     System.out.println("exception handled");
                }

                System.out.println(result);

                //to mention verification
                if(result==true)
                {
           System.out.println("element displayed on webpage");
                }
                else
                {
     System.out.println("element not displayed on webpage");
                }
           }
}
```

```
false
exception handeled
element not displaed on webpage
```

WebElement:

- it is an interface use to perform action on element present in a browser.
- In the webelement method **is** start functions are return boolean value

Webelement methods:

1. sendKeys(): void:WebElement

- This method is use to **enter value** in the **input field.**

2. clear():void:WebElement

- This method is use to **clear value** in the **text field**.

3. click(): - void:WebElement

- Click method is **use** to **click** on buttons, links also **use** to **select** radio buttons & checkboxes.

4. getText: -String:WebElement

- This method is use to **get text present in a webpage**.
- Return type of getText function is String.

5. isEnabled(): -boolean

- This method is use to **verify** element is **enebled or disabled**.
- Return type of isenabled function boolean
- if element is enabled then it returns true otherwise it returns false

6. isSelected(): -boolean

- This method is use to **verify** radio button/checkbox is **selected or not**.
- Return type of isSelected function is boolean.
- If radio button/checkbox is selected then it returns true otherwise it returns false.

7. isDisplayed(): -boolean

- This method is use to **verify** element is **present or not**.
- Return type of isDisplayed function is boolean.
- If element is present then it returns true otherwise it returns exception.