Selenium:

Selenium is one of the most widely used open source Web UI (User Interface) automation testing suite. Selenium can be easily deployed on platforms such as Windows, Linux, Solaris and Macintosh. Moreover, it supports OS (Operating System) for mobile applications like iOS, windows mobile and android.

Selenium supports a variety of programming languages through the use of drivers specific to each language. Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby. Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari.

**Selenium WebDriver- Installation**

Selenium WebDriver installation process is completed in four basic steps:

1. Download and Install Java 8 or higher version.
2. Download and configure Eclipse or any Java IDE of your choice.
3. Download Selenium WebDriver Java Client
4. Configure Selenium WebDriver

1. Download and Install Java

Require to have Java 8 installed to use Selenium 3.

However, download the latest version of Java Development Kit (JDK) from the link given below.

http://www.oracle.com/technetwork/java/javase/downloads/index.html

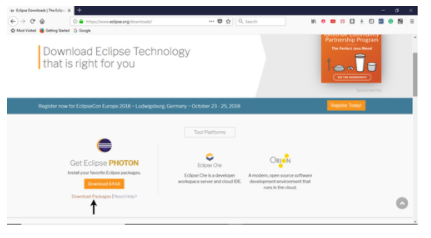
Once you have downloaded and installed the latest version of Java, set path or configure the environment variables in system. Refer the link given below to understand how to set path and configure environment variables in Java.

https://www.javatpoint.com/how-to-set-path-in-java

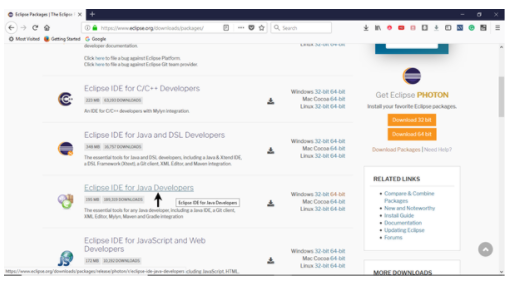
2. Download and Configure Eclipse IDE

Open URL: https://www.eclipse.org/downloads/ .

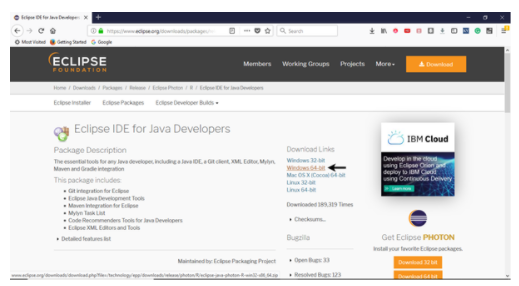
Click on the "Download Packages" link (you can also download the IDE directly from the "downloads page" of Eclipse official website, but recommend to navigate through the download packages section and get "Eclipse IDE for Java Developers").



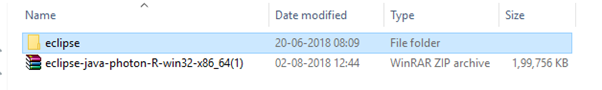
It will redirect to the "Download Packages" section. Scroll down through the webpage and click on "Eclipse IDE for Java Developers".



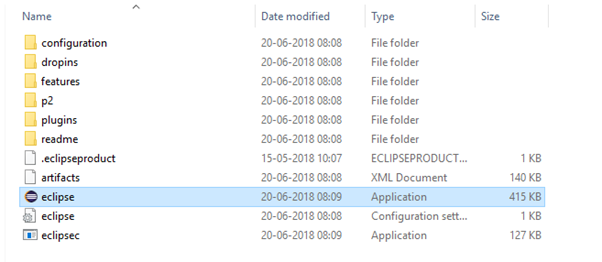
Go to the Download Links section and click on "Windows 64-bit". Select other options to download based on the operating system currently working on.



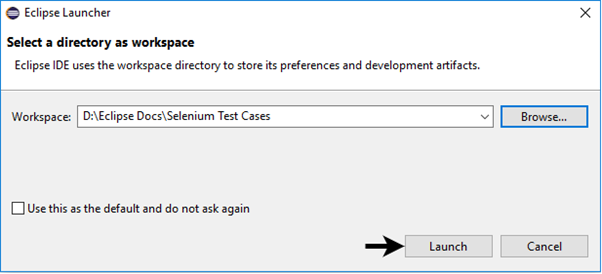
The downloaded file would be in zipped format. Unpack the contents in a convenient directory.



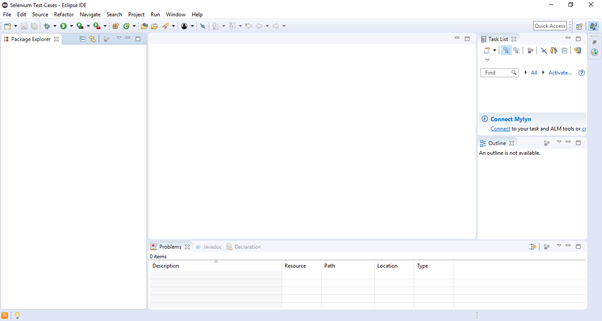
Double click on "eclipse" (.exe file).



To configure the workspace, select a convenient directory to keep all of your Selenium trails and click on Launch button.



It will launch the default interface of Eclipse IDE.

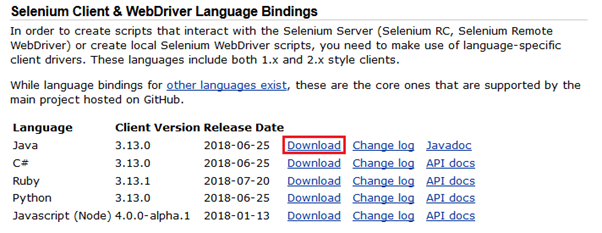


3. Download Selenium WebDriver Java Client

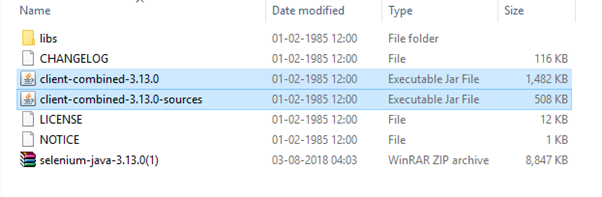
* Open URL: https://docs.seleniumhq.org/download/

It will redirect to the "downloads page" of Selenium official website.

* Scroll down through the web page and locate Selenium Client & WebDriver Language Bindings.
* Click on the "Download" link of Java Client Driver as shown in the image given below.



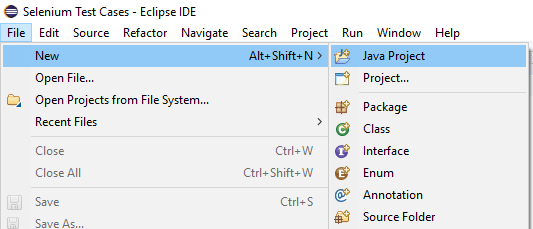
The downloaded file would be in zipped format. Unpack the contents in a convenient directory. It contains the essential jar files required to configure Selenium WebDriver in Eclipse IDE.



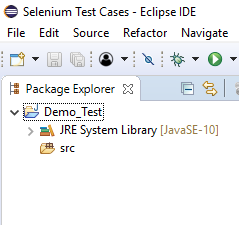
4. Configure Selenium WebDriver

Now Eclipse IDE with Selenium WebDriver. In simple terms, create a new Java Project in Eclipse and load all the essential jar files in order to create Selenium Test Scripts.

* Launch Eclipse IDE
* Create a new Java Project from File > New > Java Project.

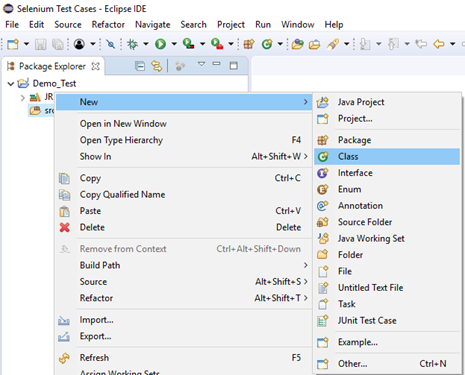


* Give Project name as "Demo\_Test", leave the other fields unaltered and click on "Finish" button.
* It will create a new Java project with the following directories.

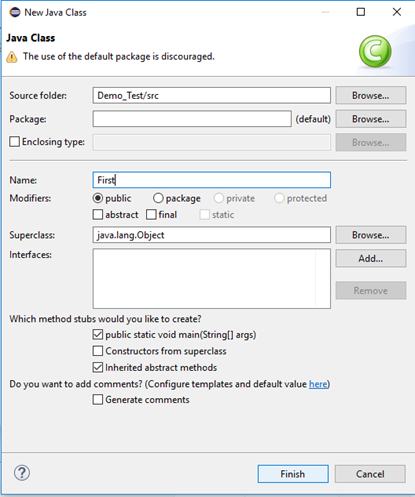


Note: Selenium Test Scripts are always written in ".class" file in Java. Here the project "Demo\_Test" act as a Test Suite that may contain one or more Selenium test cases/test scripts.

Right click on the "src" folder and create a new Class File from New > Class.

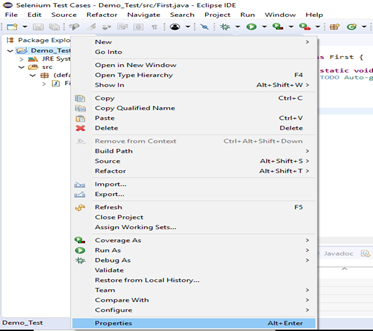


Give your Class name as "First" and click on "Finish" button.



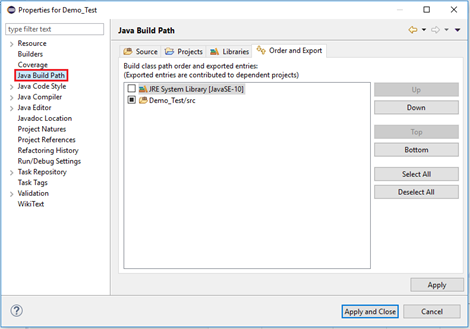
Now, we will add the Selenium jar files in our Test Suite (Demo\_Test).

Right click on "Demo\_Test" folder and select Properties.

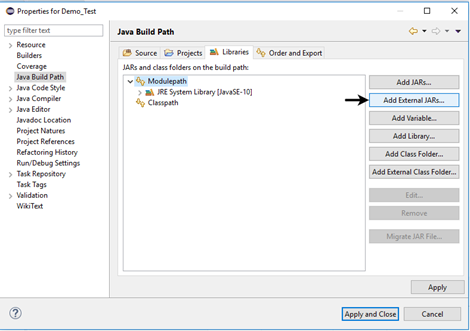


It will launch the Properties window for our "Demo\_Test" Test Suite.

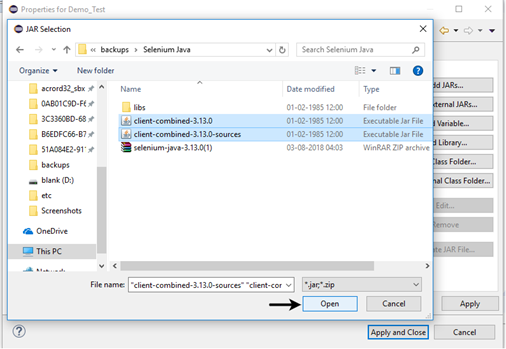
Click on "Java Build Path" option from the left hand side panel.



Switch to Libraries tab and click on "Add External JARs" button.

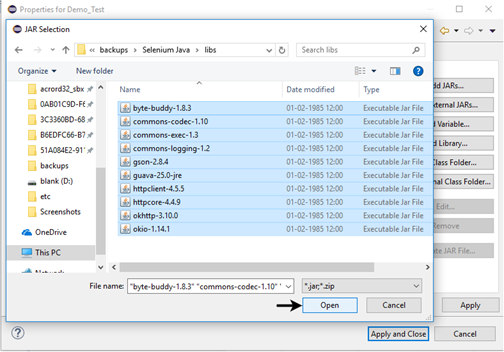


Locate the directory where you have downloaded the Selenium jar files, select the respective jars and click on "Open" button.



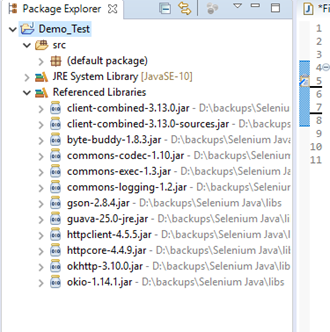
Repeat the same steps for the jars which are present under the "libs" folder.

Open "libs" folder, select all of the respective jar files and click on "Open" button.



Once you get all the Selenium jar files in your Libraries tab, click on Apply and Close button.

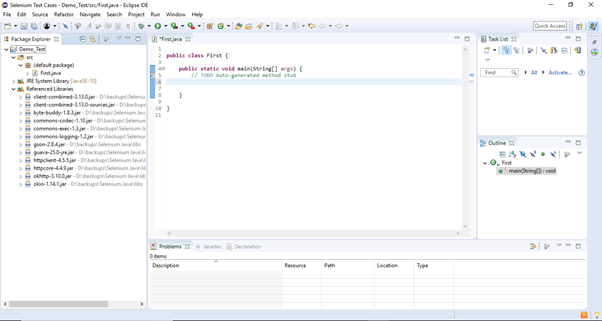
The following image shows the directory structure of our "Demo\_Test" test suite after adding Selenium jars.



Hence, we have successfully configured Selenium WebDriver with Eclipse IDE. Now, we are ready to write our test scripts in Eclipse and run it in WebDriver.

**Lab Question:**

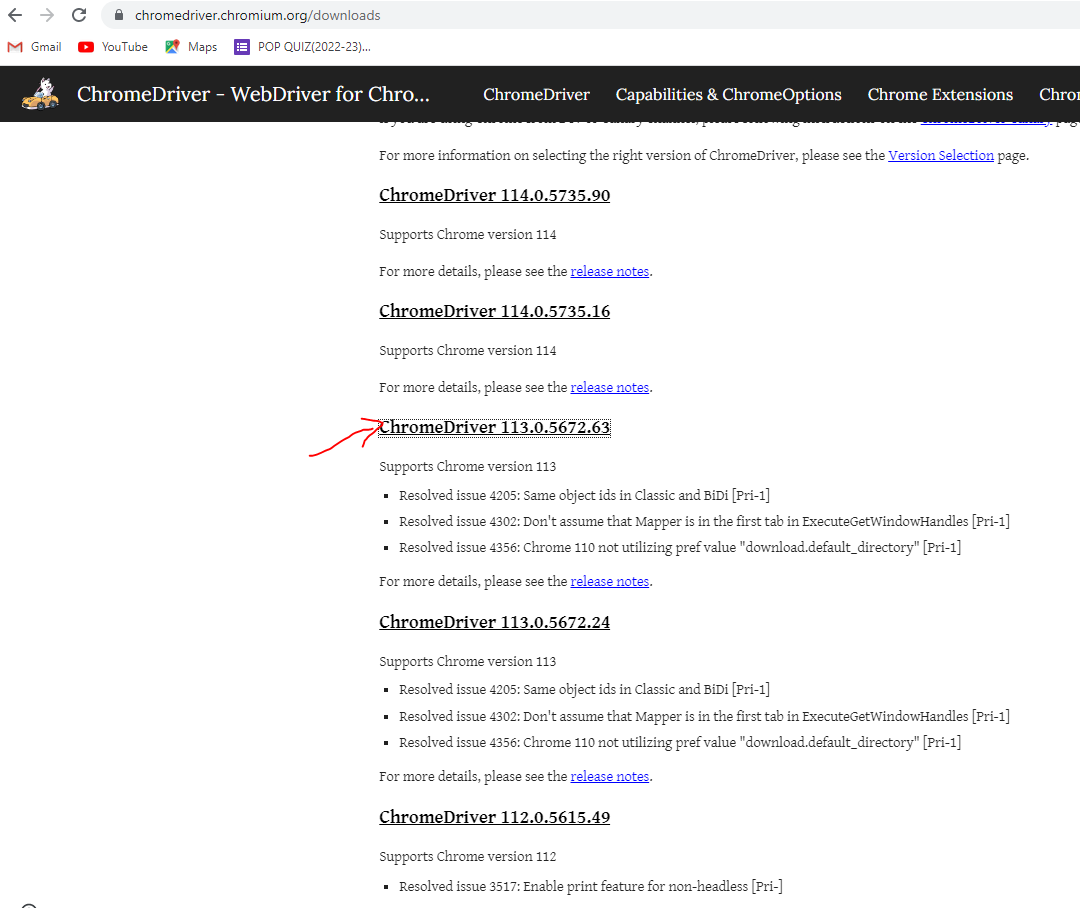
1. Configure Selenium WebDriver with Eclipse IDE and create Selenium Automation Test Script. Under this test, automate the following scenarios:
2. Invoke Google Chrome browser, Open URL: www.google.com, Click on the Google Search text box, Type the value "BMS College of Engineering", Click on the Search button.
3. Invoke Google Chrome browser, Open URL: www.isearch123.com, Click on the Google Search text box, Type the value "BMS College of Engineering", Click on the Search button.
4. Invoke Google Chrome browser, Open Indeed home page, https://www.indeed.co.uk/, Find What field and enter Selenium.
   1. Step1. Launch Eclipse IDE and open project "Demo\_Test" which we have created in the previous section (Configure Selenium WebDriver) of this Tutorial. We will write our first Selenium test script in the "First.class" file under the "Demo\_Test" test suite.



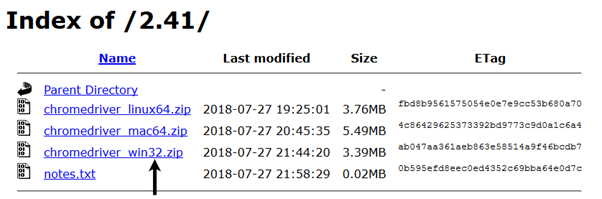
Step2: Find the chrome version in your system ( chrome://settings/help) and

download the chromeDriver with respect to your current chrome version <https://chromedriver.chromium.org/downloads> in your browser.

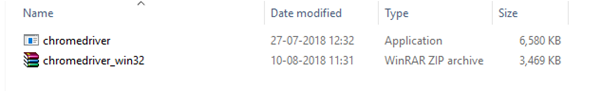
**Step3.** Click on the "ChromeDriver 113.0.5672.63" link (in this Example:my system chrome version is 113.0.5672.63 currently). It will redirect you to the directory of ChromeDriver executables files. Download as per the operating system you are currently on.



For windows, click on the "chromedriver\_win32.zip" download.



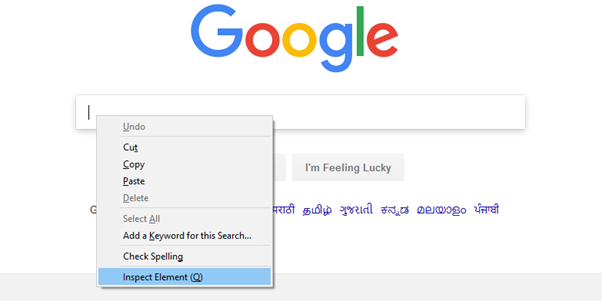
The downloaded file would be in zipped format. Unpack the contents in a convenient directory.



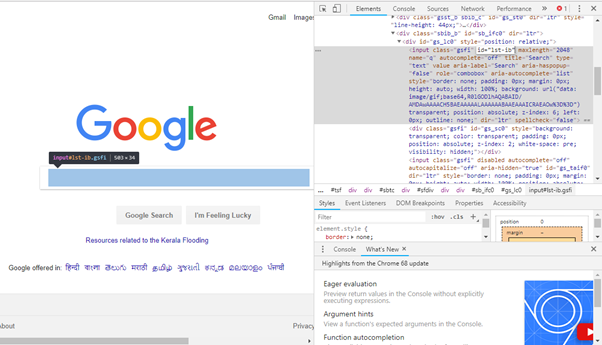
**Step4**. We would need a unique identification for the web elements like Google Search text box and Search button in order to automate them through our test script. These unique identifications are configured along with some Commands/Syntax to form Locators. Locators help us to locate and identify a particular web element in context of a web application.

The method for finding a unique identification element involves inspection of HTML codes.

* Open URL: [https://www.google.com](https://www.google.com/) in your Chrome browser.
* Right click on the Google search text box and select Inspect Element.



* It will launch a window containing all the specific codes involved in the development of the test box.



* Pick the value of id element i.e. "lst-ib".

Selenium WebDriver First Test Case

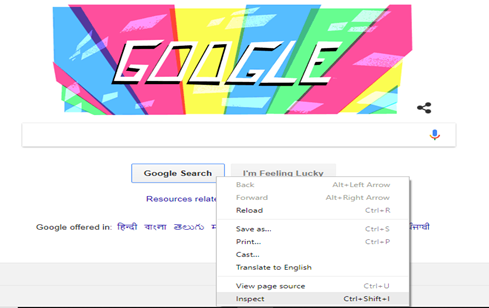
* Given below is the Java syntax for locating elements through "id" in Selenium WebDriver.

driver.findElement(By.id (<element ID>))

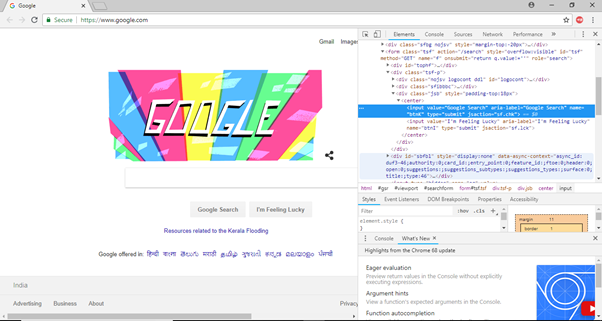
* Here is the complete code for locating Google Search text box in our test script.

driver.findElement(By.id ("lst-ib"))

* Now, right click on the Google Search button and select Inspect Element.



* It will launch a window containing all the specific codes involved in the development of the Google Search button.



* Pick the value of **name** element i.e. "btnK".

Selenium WebDriver First Test Case

* Given below is the Java syntax for locating elements through "name" in Selenium WebDriver.

driver.findElement(By.name (<element name>))

* Here is the complete code for locating Google Search button in our test script.

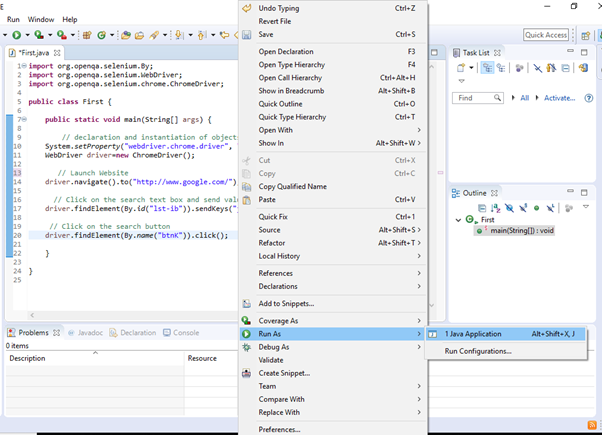
driver.findElement(By.name ("btnK"))

**Step5**. Code

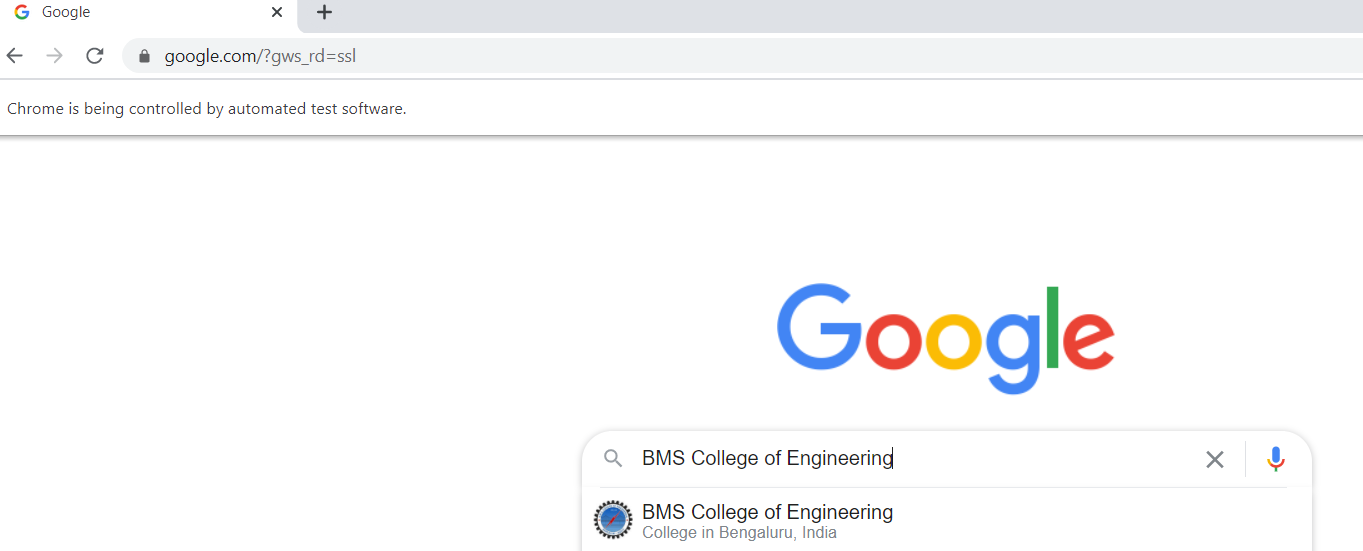
1. **import** org.openqa.selenium.By;
2. **import** org.openqa.selenium.WebDriver;
3. **import** org.openqa.selenium.chrome.ChromeDriver;
4. **public** **class** First {
5. **public** **static** **void** main(String[] args) {
6. // Call the chrome server to invoke chrome
7. System.setProperty("webdriver.chrome.driver","C:\\Selenium\\chromedriver\_win32\\chromedriver.exe");
8. // Create Internet explorer driver to driver the browser
9. WebDriver driver= new ChromeDriver();
10. // Maximize the window server.
11. driver.manage().window().maximize();    // Launch website
12. driver.navigate().to("http://www.google.com/");
13. // Click on the search text box and send value
14. driver.findElement(By.id("lst-ib")).sendKeys("BMS College of Engineering");
15. // Click on the search button
16. driver.findElement(By.name("btnK")).click();
17. }
18. }

**Note: provide proper path where you have saved your chromeDriver in line number 7 in the above code.**

**Step6**. Right click on the Eclipse code and select **Run As > Java Application**.



**Step7**. The output of above test script would be displayed in Google Chrome browser.



**If ‘id’ element is not displayed, you could give using ‘name’ element.**

// Launch website

driver.navigate().to("http://google.com/");

// Click on the search text box and send value

driver.findElement(By.*name*("q")).sendKeys("BMS College of Engineering");

// Click on the search button

driver.findElement(By.*name*("q")).click();

**(b)**

// Launch website

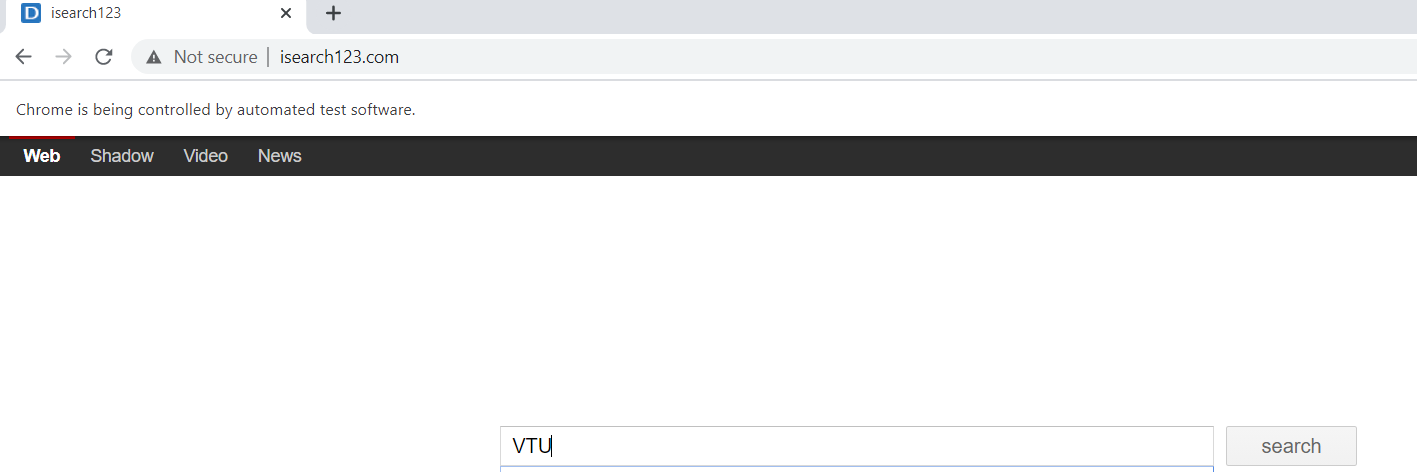
driver.navigate().to("http://isearch123.com/");

// Click on the search text box and send value

driver.findElement(By.*id*("q")).sendKeys("VTU");

// Click on the search button

driver.findElement(By.*name*("q")).click();



**(c)** // Open Indeed home page

driver.get("https://www.indeed.co.uk/");

// Find What field and enter Selenium

driver.findElement(By.*id*("text-input-what")).sendKeys("Selenium");

