Practical-1

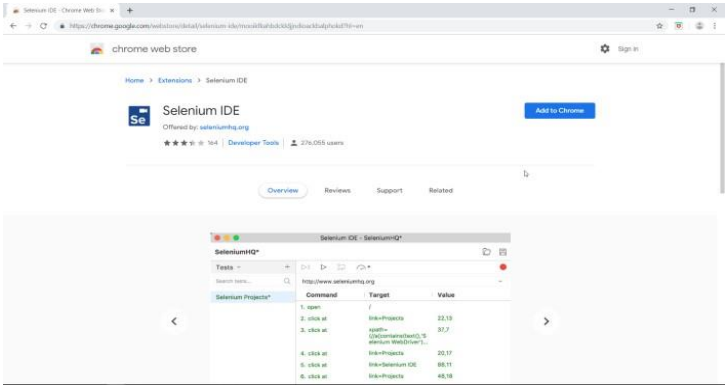
#AIM: Install Selenium IDE and create a test suite containing a minimum of 4 test cases for different web page formats (e.g., HTML, XML, JSON, etc.).

REQUIREMENTS:

1) Selenium IDE extension on a browser.
2) NetBeans IDE for creating test case files.

STEPS:

1) Just **google "Selenium IDE chrome"** and click on the first link. **Add** the **"Selenium IDE" extension** to your browser.



2) For testing, we need a **test case suite**, so we create a **Web Application** in **NetBeans IDE**.

File > New Project > Java Web > Web Application > Add Glassfish Server > Finish.

3) In *index.html*, we **write the following code** for "ARITHMETIC OPERATIONS" test case suite:

---index.html---

```
<html>
  <head>
    <title>prac1</title>
    <script language="javascript">
      function addition()
        {
          var num1=parseInt(document.arithmetic.n1.value);
var num2=parseInt(document.arithmetic.n2.value);
          var result=num1+num2;
          document.arithmetic.res.value=result;
        }

      function subtraction()
        {
          var num1=parseInt(document.arithmetic.n1.value);
var num2=parseInt(document.arithmetic.n2.value);          var
result=num1-num2;
          document.arithmetic.res.value=result;
        }

      function multiplication()
```

```
        {
            var num1=parseInt(document.arithmetic.n1.value);
var num2=parseInt(document.arithmetic.n2.value);
            var result=num1*num2;
            document.arithmetic.res.value=result;
        }

      function division()
        {
            var num1=parseInt(document.arithmetic.n1.value);
var num2=parseInt(document.arithmetic.n2.value);
            var result=num1/num2;
            document.arithmetic.res.value=result;
        }
    </script>
  </head>

  <body>
    <h1 align="center"> Arithmetic Operations </h1>
    <form name="arithmetic">
      <table border="1" align="center">

        <tr>
          <td>Number 1: </td>
          <td><input type="text" name="n1" size="20"></td>
        </tr>

        <tr>
          <td>Number 2: </td>
          <td><input type="text" name="n2" size="20"></td>
        </tr>

        <tr>
          <td colspan=2>
            <input type="button" name="add" value="Add"
onclick="javascript:addition();"> 
            <input type="button" name="sub" value="Subtract"
onclick="javascript:subtraction();"> 
            <input type="button" name="mul" value="Multiply"
onclick="javascript:multiplication();"> 
            <input type="button" name="div" value="Divide"
onclick="javascript:division();"> 
          </td>
        </tr>

        <tr>
            <td colspan="2">Result is: <input type="text" name="res" size="20" value=""></td>
        </tr>
```

```
        </table>
      </form>
    </body>
</html>
```

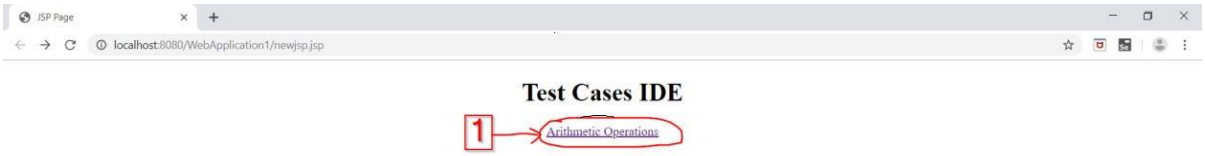
4) Create a **JSP file** *"newjsp.jsp"* with a link to the created HTML file.

---newjsp.jsp---

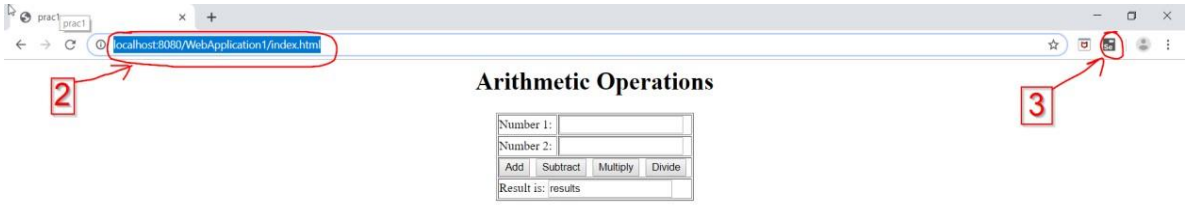
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
  </head>
  <body align="center">
    <h1>Test Cases IDE</h1>
    <a href="index.html">Arithmetic Operations</a>
  </body>
</html>
```

5) Run this JSP file from NetBeans. It will then open up in a browser.

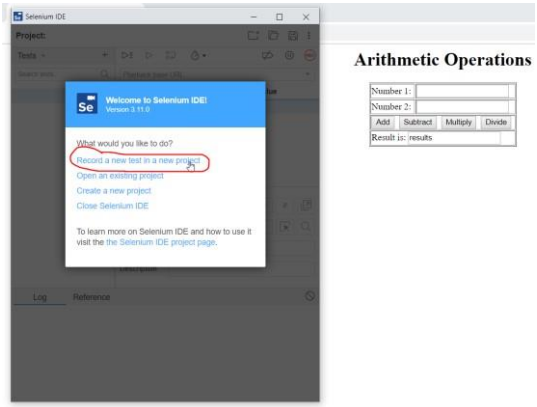
- Then from the browser, **open the link**:



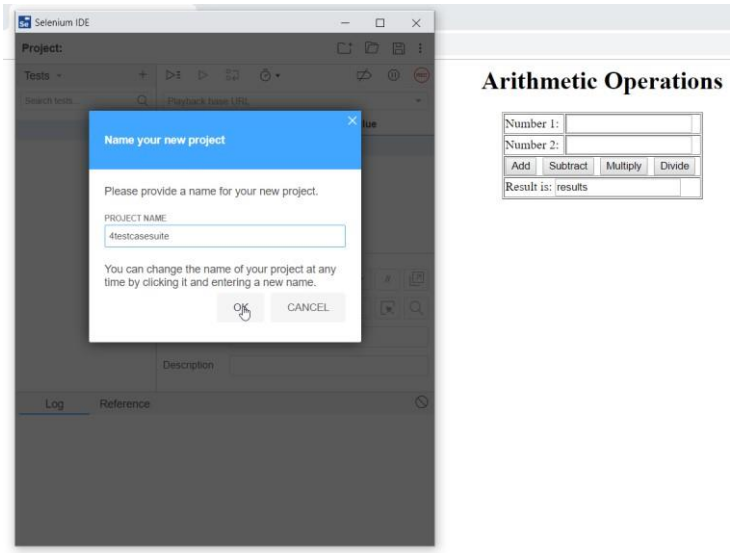
- Now you'll be redirected to ARITHMETIC OPERATIONS page.
  Copy the URL of the page and **open "Selenium IDE" extension**:



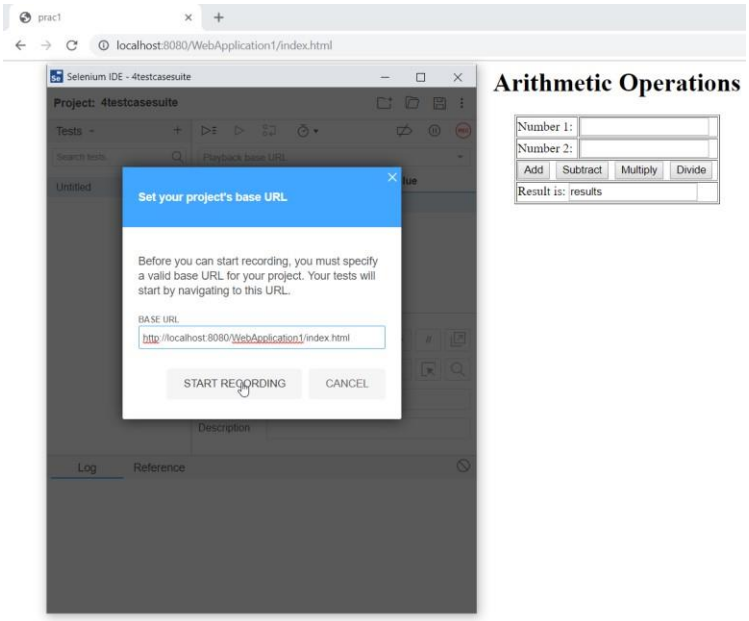
- Now, **Record a new test project in a new project:**



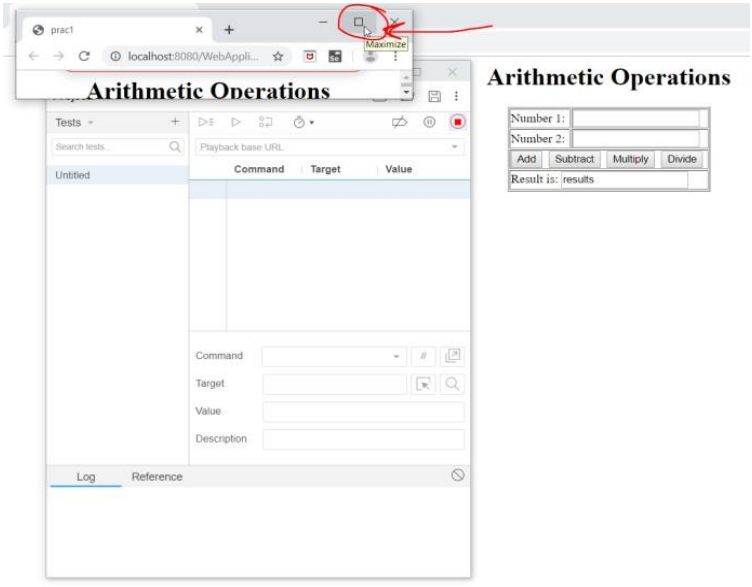
Name your project:



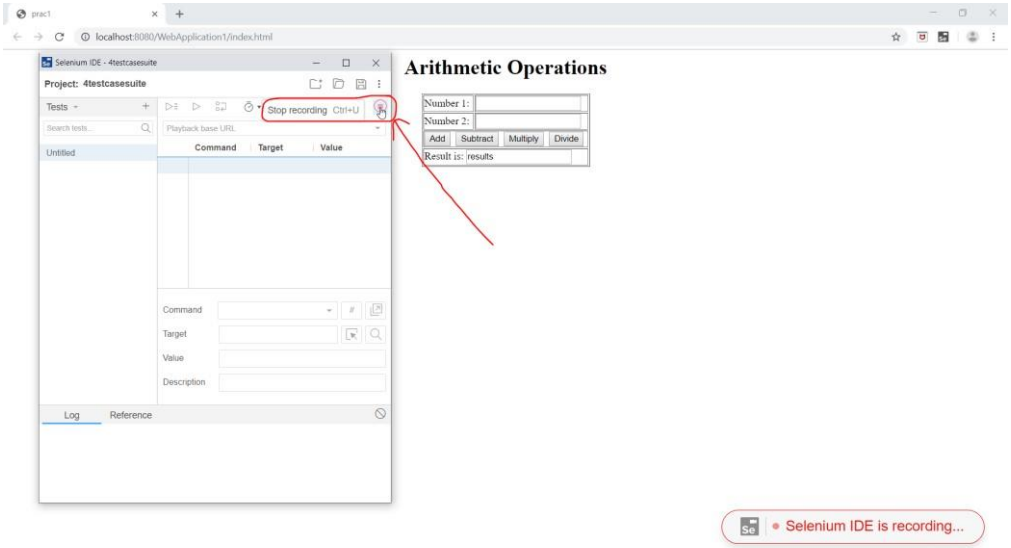
- **Paste that URL** here and click **Start Recording:**



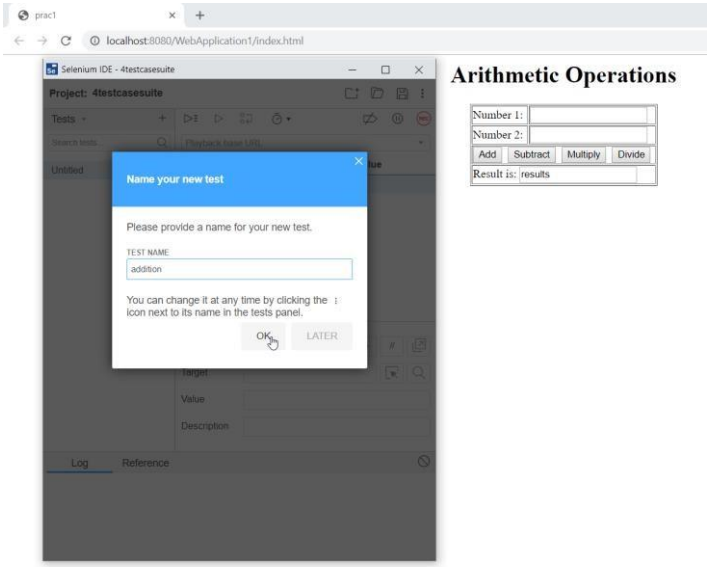
- A **new tab** will be opened, **maximize** that tab(this tab is your working tab for recording):



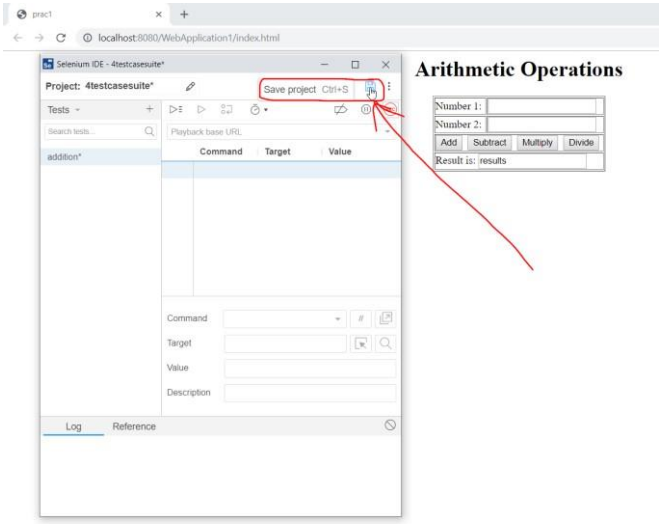
Click on **Stop Recording**:



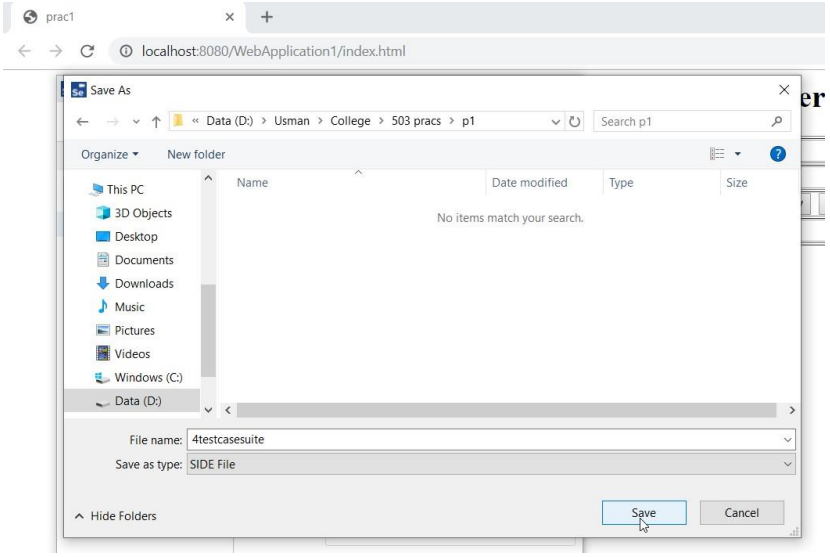
- Just as you stop your recording, you'll be prompted to **Name your new test,** we'll make our new test as "**addition**":



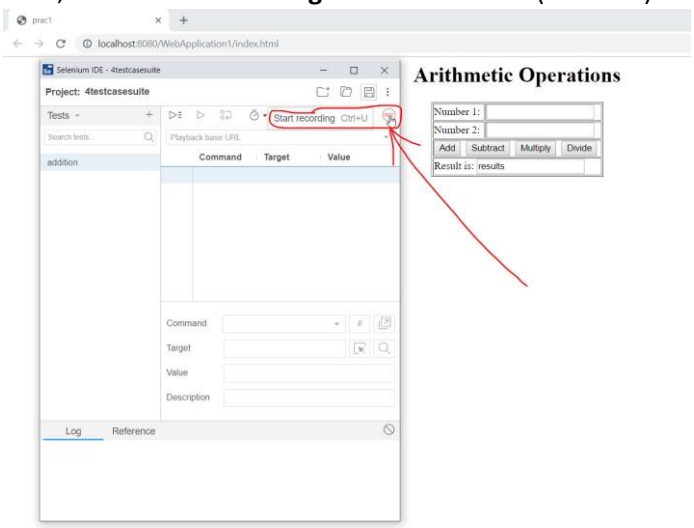
- **Save the project** after each test case:



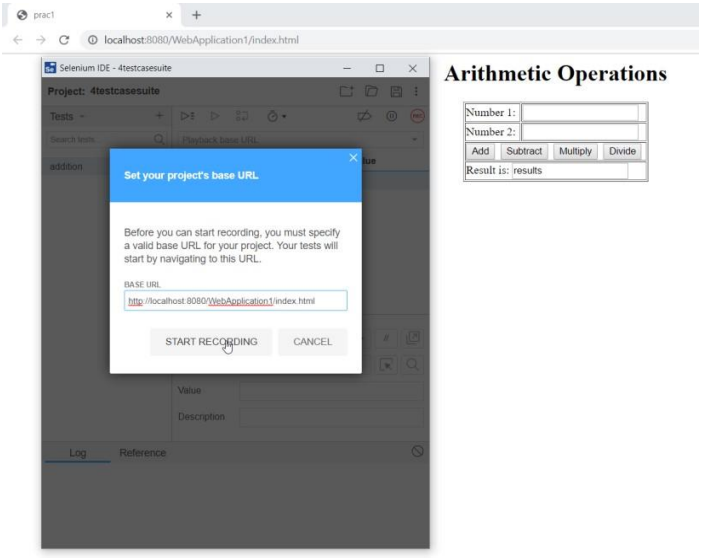
A file with "**.SIDE**" extension will be saved(SIDE abbrev to **S**elenium **IDE**):



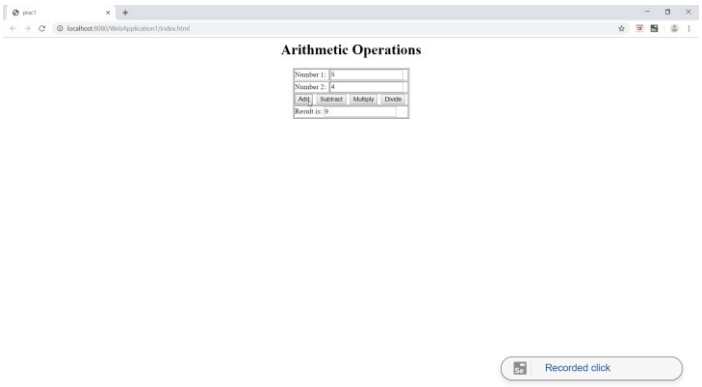
- Now, we'll **Start Recording** our first test case(addition):



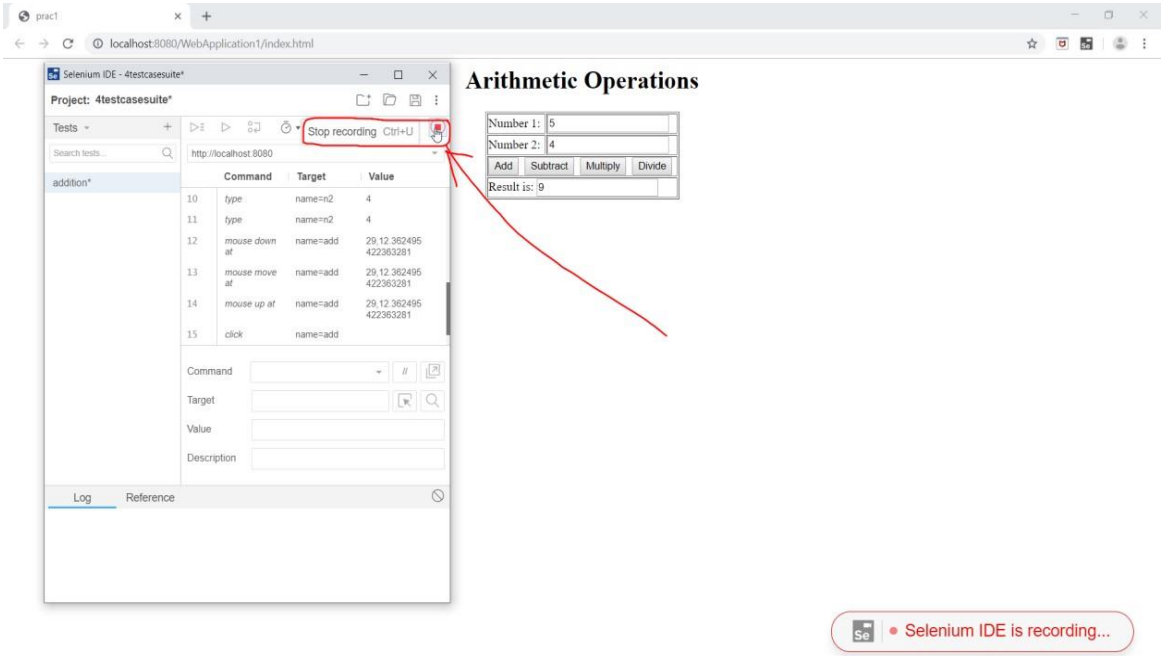
- We'll be prompted to **paste our URL again**, then click **Start Recording:**



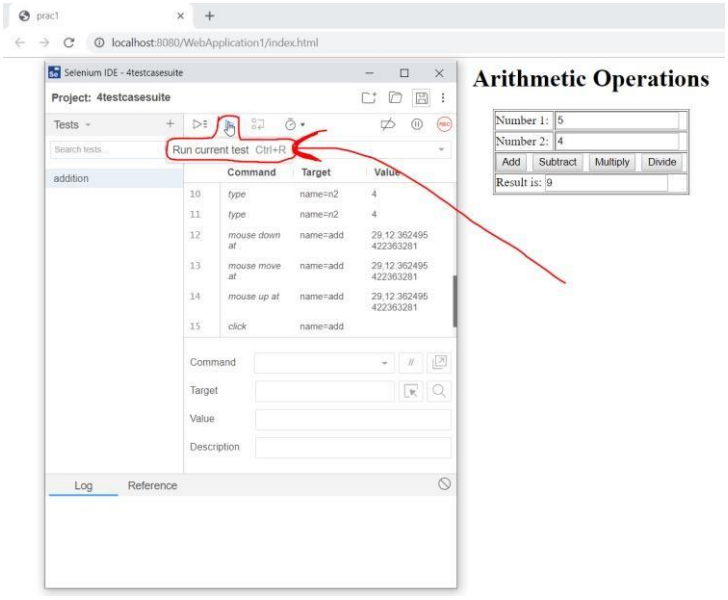
NOTE THAT each clicks are recorded:



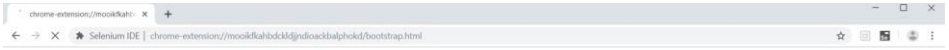
- Now we'll **Stop Recording from the Selenium IDE:**

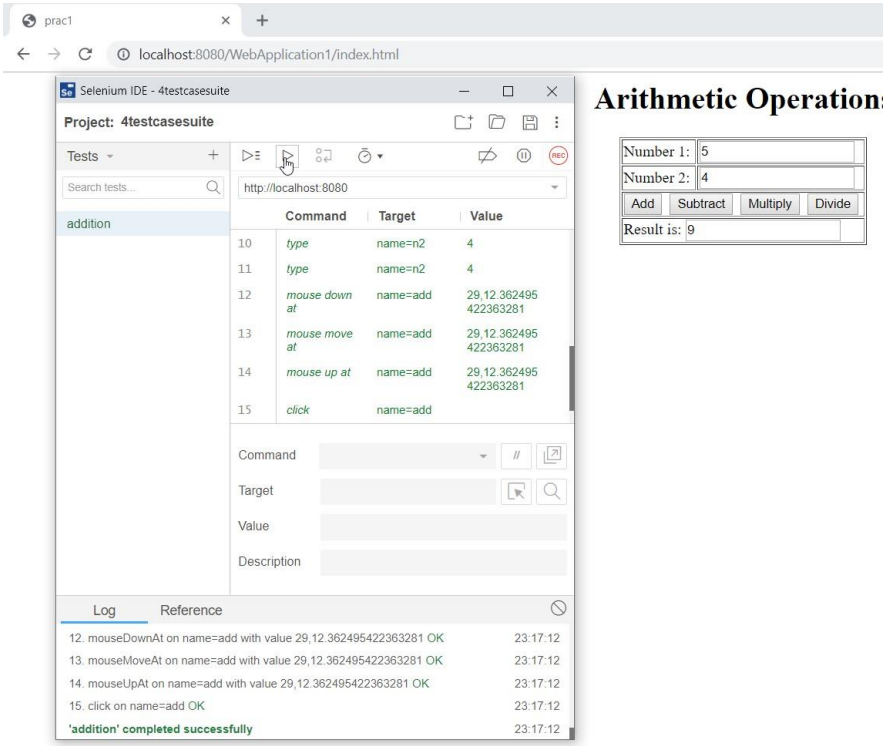


- We'll then **Run current test**(addition):

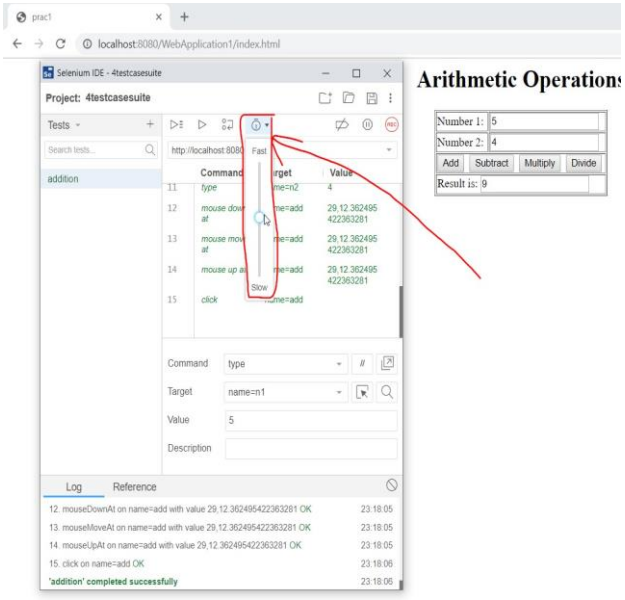


The **test will run automatically** and the **completion will be shown step-by-step** in the **“Log” area** in the Selenium IDE:

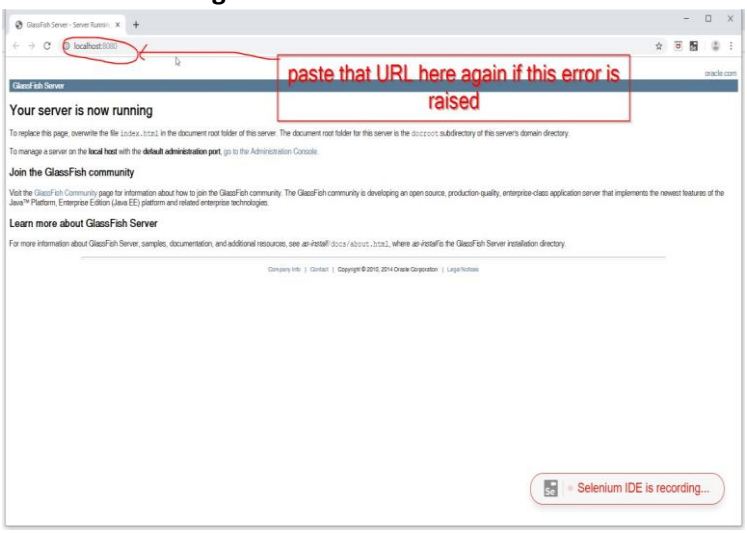




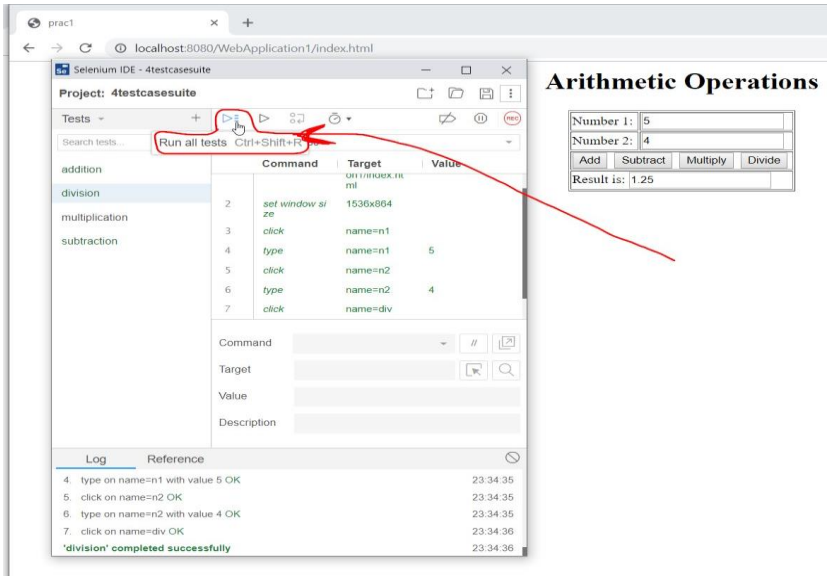
- **NOTE1: You can set the “Test execution speed” too:**



NOTE2: Paste here the URL again if the Glassfish Error occurs between any test case**:**



- Now **create 3 more test cases**(subtraction, multiplication, division). Then "**Run all tests**":



6) Finish!

Practical No: 02

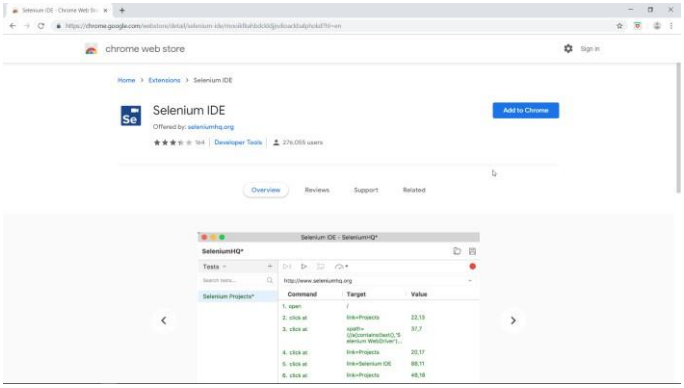
Aim: Conduct a test suite for two different websites using Selenium IDE. Perform various actions like clicking links, filling forms, and verifying content

REQUIREMENTS:

1) *Selenium IDE extension on a browser.* ***2)*** *A stable Internet connection.*

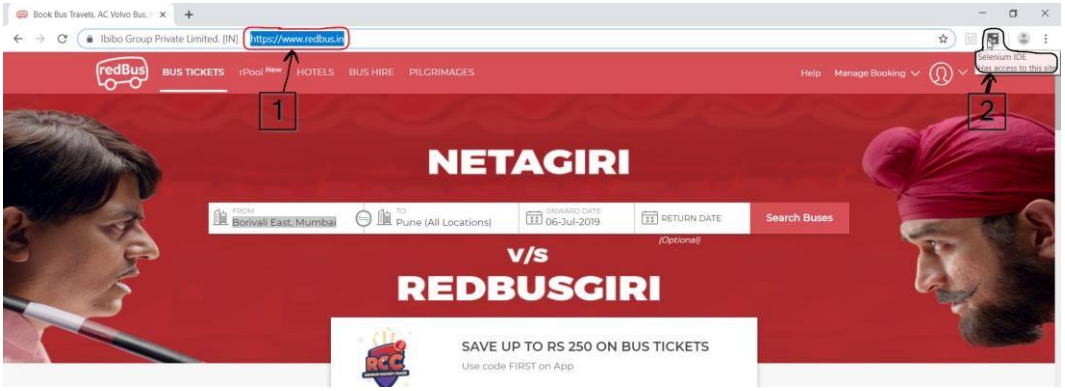
STEPS:

1) *Just* ***google "Selenium IDE chrome"*** *and click on the first link.* ***Add*** *the* ***"Selenium IDE" extension*** *to your browser.*

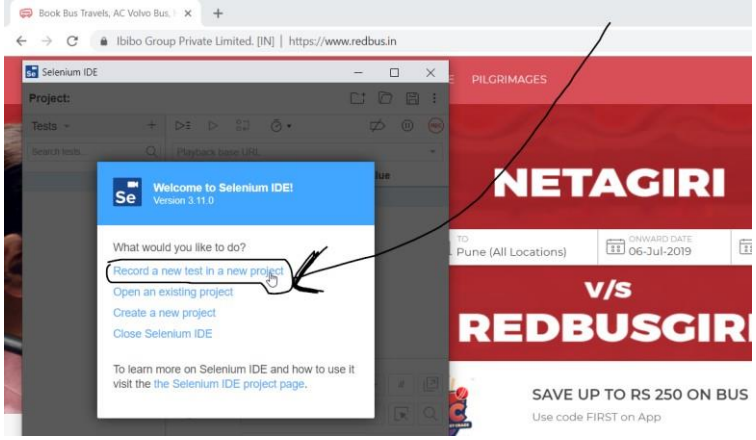


2) *For testing, we'll choose "****redbus.in****" as our first website. We'll be checking some* ***Selenium IDE Commands*** *like* ***assertText, verifyTitle, storeText, echo****. Detailed info for several commands are provided at https://ui.vision/docs/selenium-ide .*

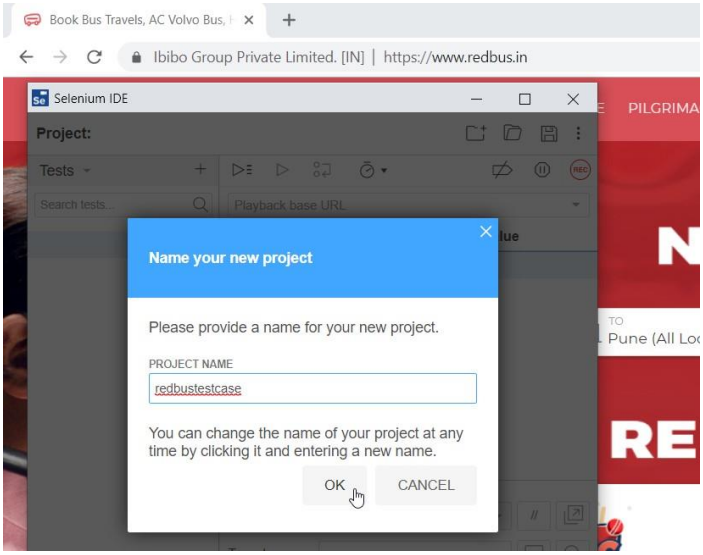
- *Now* ***google "redbus.in".***
- ***Copy the URL*** *of the page and* ***open "Selenium IDE" extension****:*



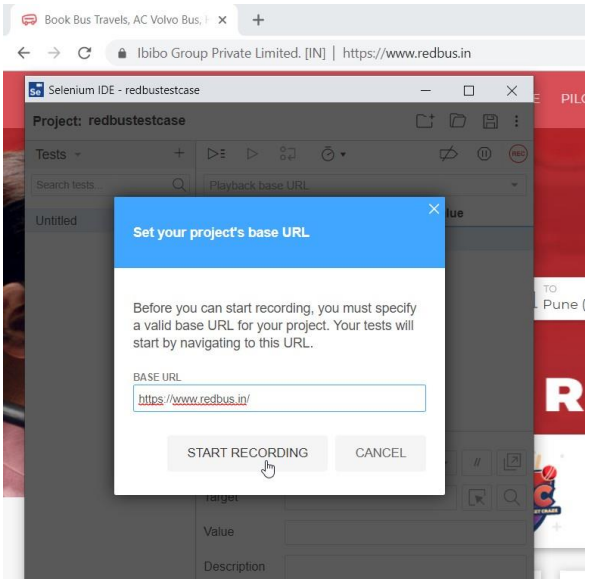
- *Now,* ***Record a new test project in a new project:***



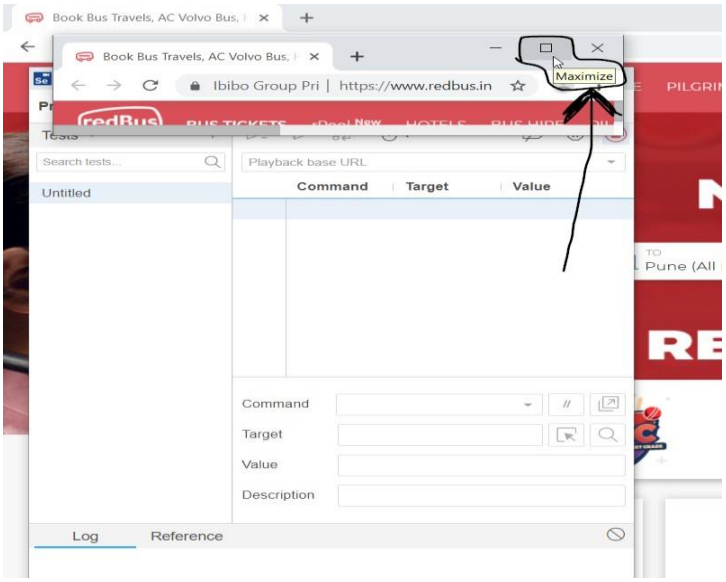
***Name your project**:*



- ***Paste that URL** here and click **Start Recording**:*

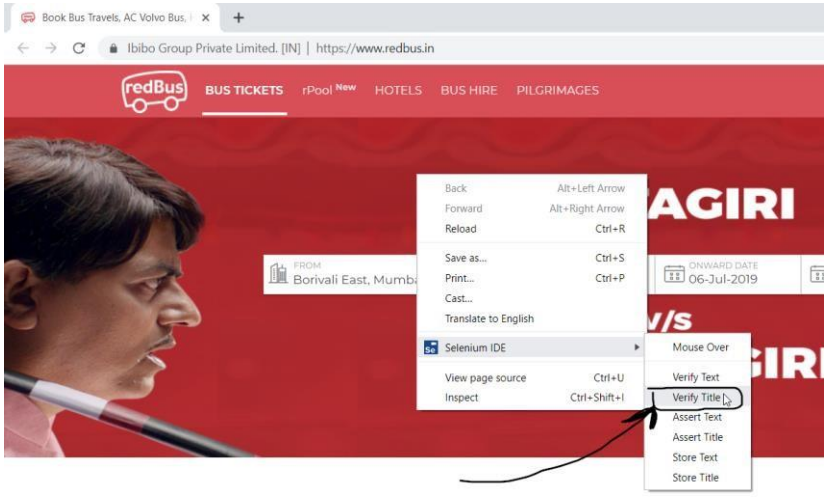


- *A **new tab** will be opened, **maximize** that tab(this tab is your working tab for recording):*

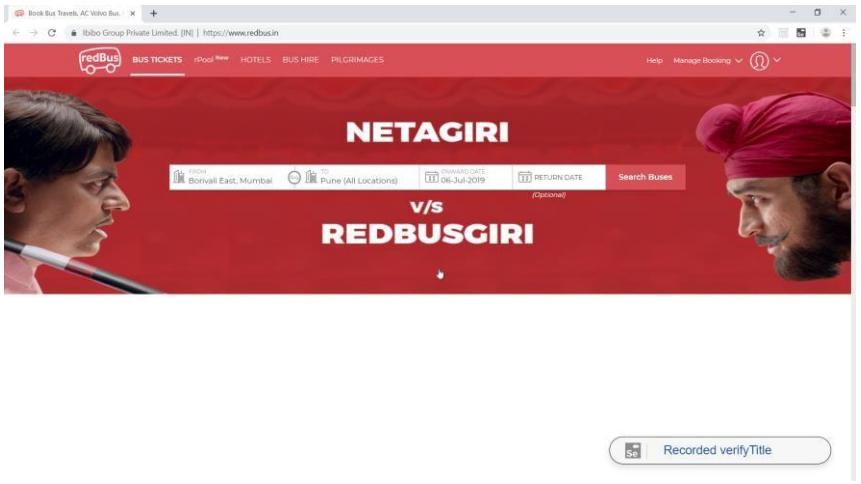


*Now we'll make use of command **verifyTitle**.*

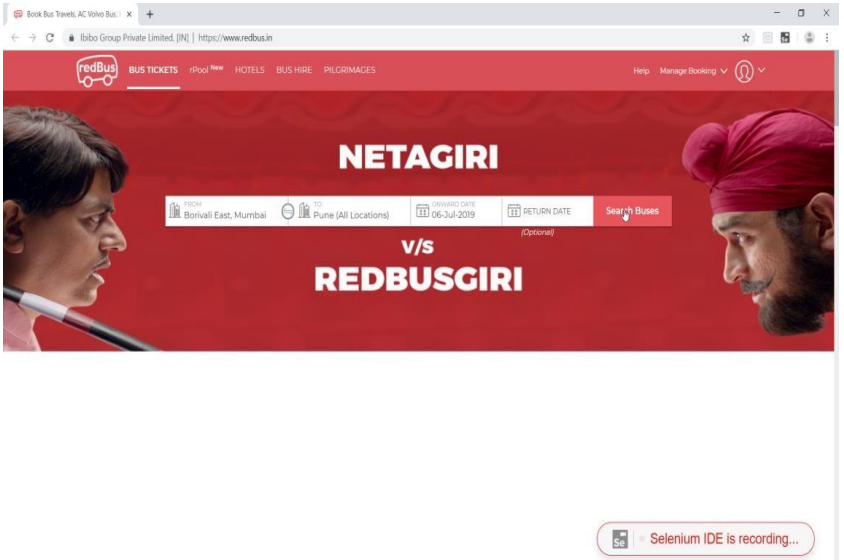
*(This command verifies title of the webpage and keeps on continuing the execution.) So **anywhere on the window**, do **Right Click>** select **Selenium IDE>** click **Verify Title**:*



- ***NOTE THAT** each command is recorded:*



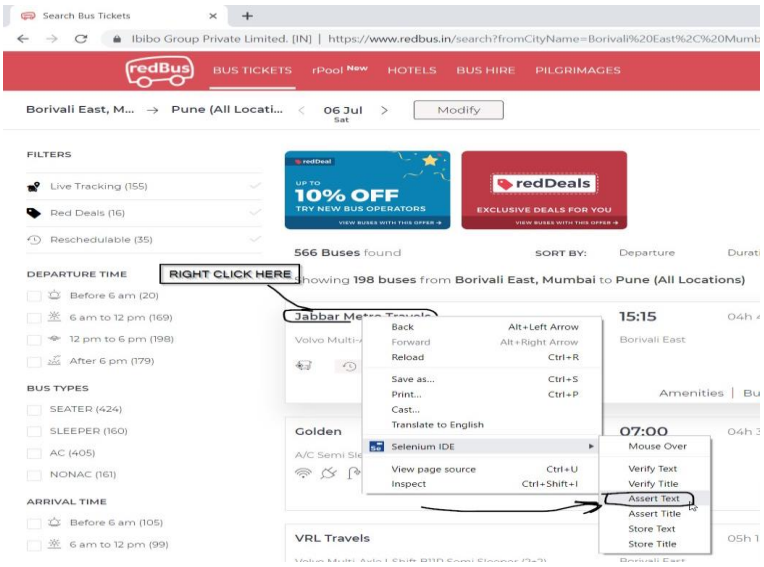
- *Now after filling the FROM, TO, ONWARD DATE, we proceed further by clicking on*
  ***Search Buses:** (make sure not to do unnecessary clicks)*



---

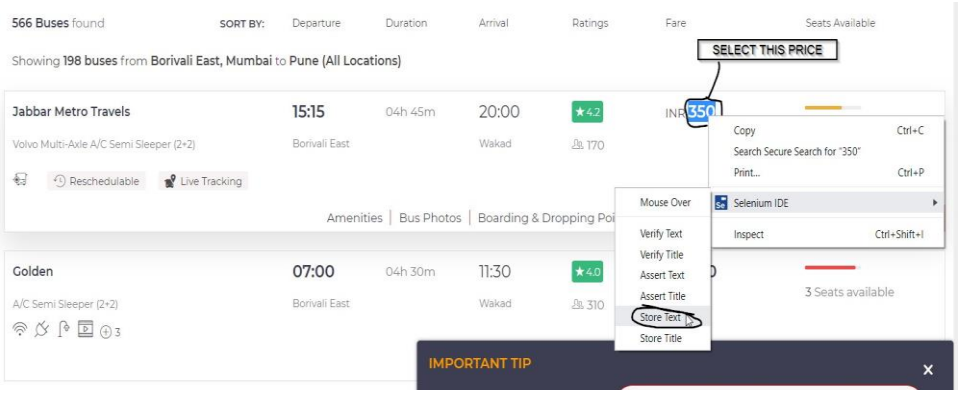
*Now we'll make use of command **assertText**.*
(This command verifies particular text of the webpage and stops the execution of testcase if the match isn't found.)

So, we want to make sure that our automated testcase chooses our specified Bus Provider.
*To ensure that, here we do **Right Click over the name of the bus provider>** then select **Selenium IDE>** and click **Assert Text** :*

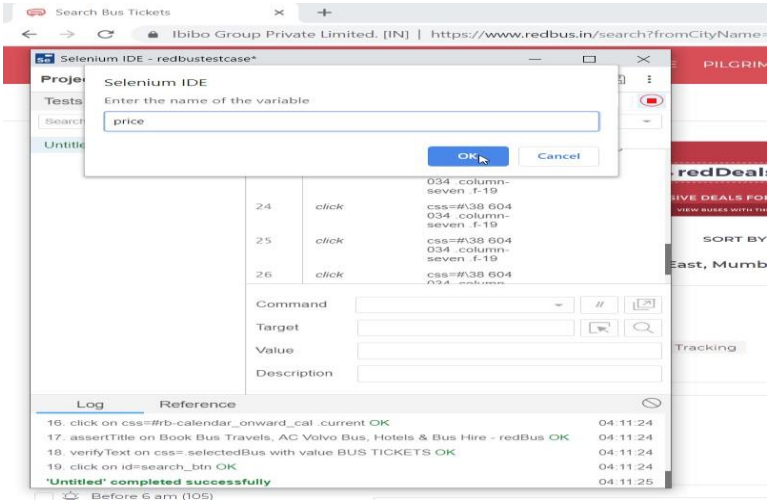


- *Now we'll make use of command **storeText**.*
  (This command stores the text value of page element into variable for future use.) Here we'll store the price of the ticket.

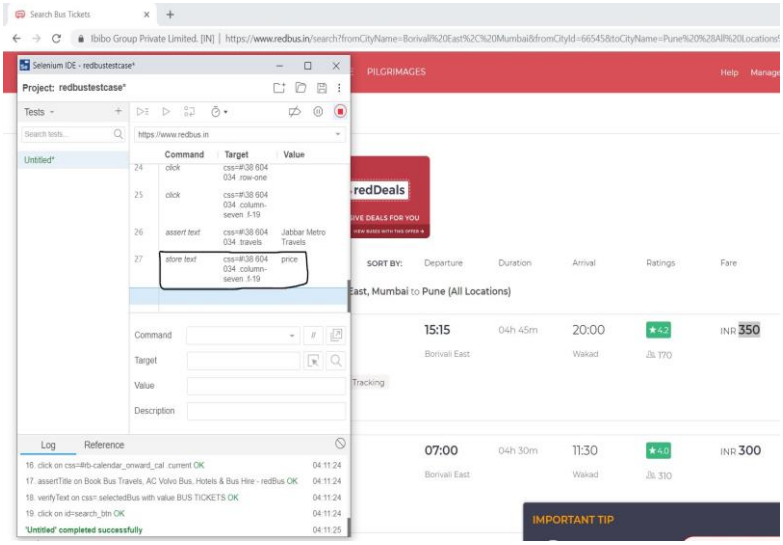
*So, we **select the price> right click** over it> select **Selenium IDE>** click **Store Text :***



*Now Selenium IDE will prompt us to **enter the name of the variable** in which that price will be stored:*



- *After clicking OK, we can see how the target is automatically mapped to the css attribute of the page, and our price is stored:*

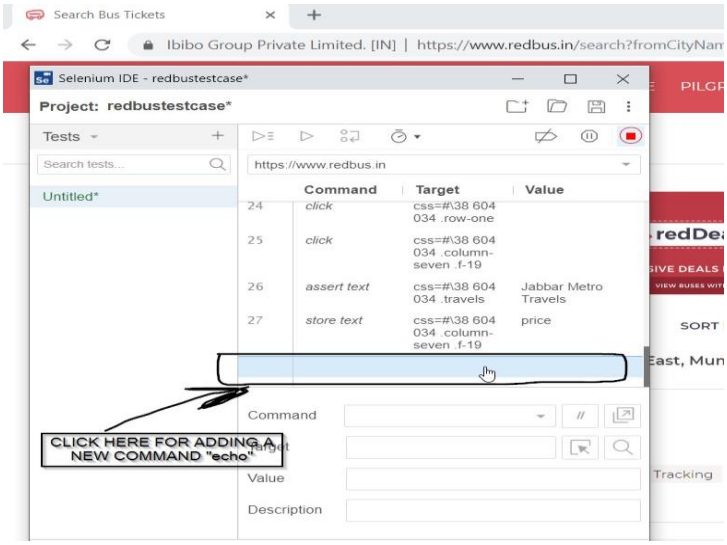


*Now we'll make use of command **echo.***

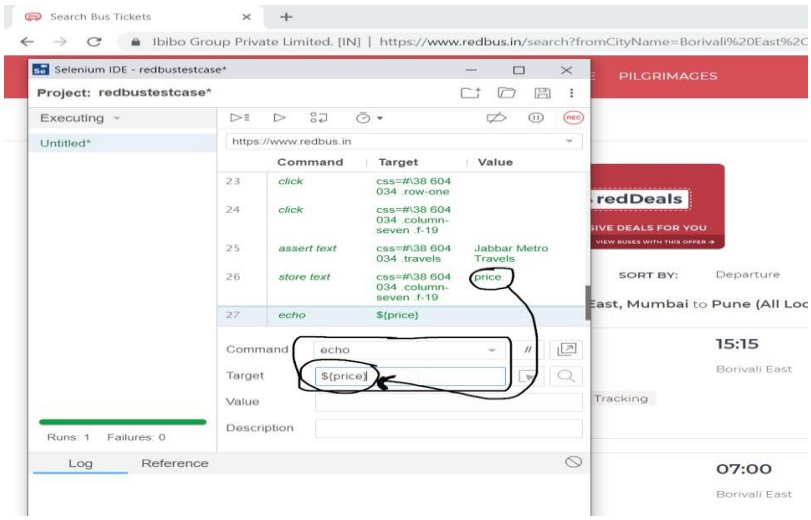
(This command is used to display text(comments, notes) and/or the stored value of variables in the log area of Selenium IDE.)

Here we'll display the price of the ticket from the variable "price" that we stored during the use of storeText command.

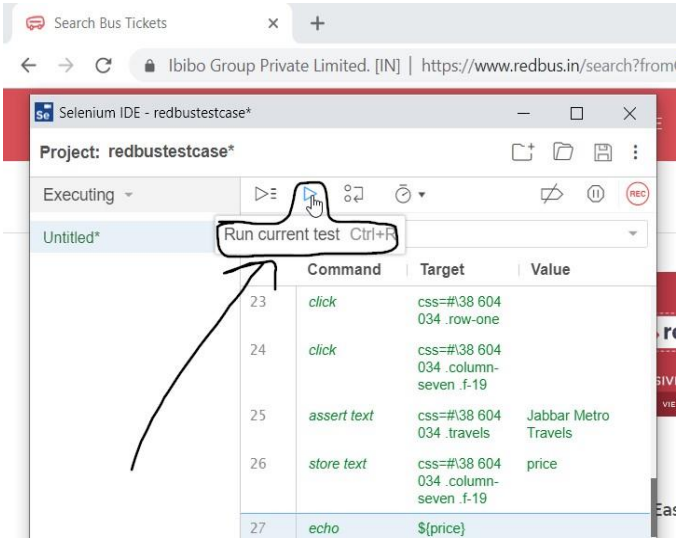
To create a new command, we just click onto the next line and enter our command:



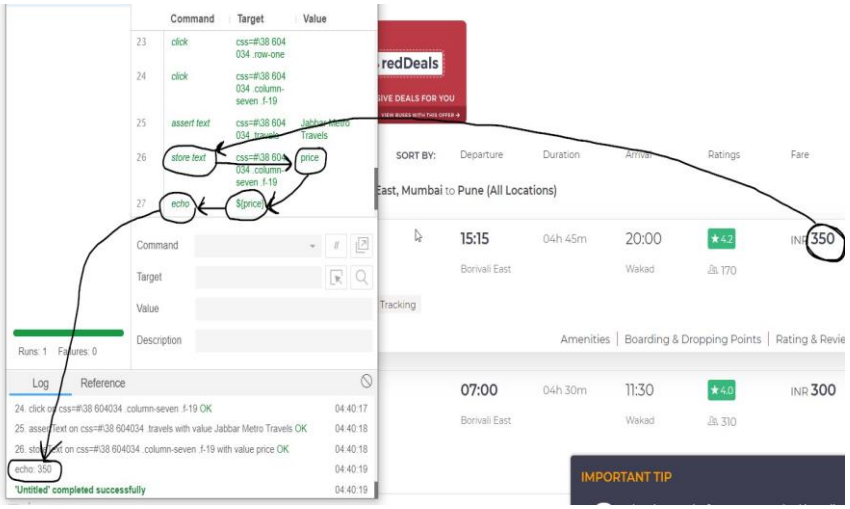
- *We'll enter the Command name **echo**, & Value **${price}** because we are calling the created variable "price" from this command:*



*Now we'll **Run current test**:*



- *After the test gets completed, we can see the **echo: 350** in the log area of Selenium IDE showing the output of the echo command.*
  Here we can see how 350 is stored in variable "price" by storeText command, and that variable is then called by the echo command, which is then returning the value fetched from the page:



3) ***Finish!***

Practical No :03

AIM: Install Selenium Server (Selenium RC) and demonstrate its usage by executing a script in Java or PHP to automate browser actions

PRE-REQUISITES*(*here are w.r.t. Windows 10(64 bit), so choose accordingly w.r.t. your specs)**:***

1) *To Download "**JDK**":*

- ***Visit*** *https://www.oracle.com/technetwork/java/javase/downloads/jdk12downloads-5295953.html*
- ***Download*** *this file "jdk-12.0.2_windows-x64_bin.exe" and* ***install it****.* ***2)*** *To Download "**Eclipse IDE**":*
- ***Visit*** *https://www.eclipse.org/downloads/download.php?file=/oomph/epp/201906/R/eclipse-inst-win64.exe*
- *Click "**Download**".*
- *Installation:*
  - ***Open application****.(click OK if errors occur like "could not find java.dll" & "could not find Java Runtime Environment SE")*
  - *Select "**Eclipse IDE for Java Developers**".*
  - *It will automatically locate the JDK. Choose path, and click "**Install**".*
- *After installation, click "**Launch**" or open Eclipse from START menu in Windows.*

3) *To Download "**Selenium Server Driver and Client Driver(JAR files)**":*

a) *For "Selenium Server Driver":*

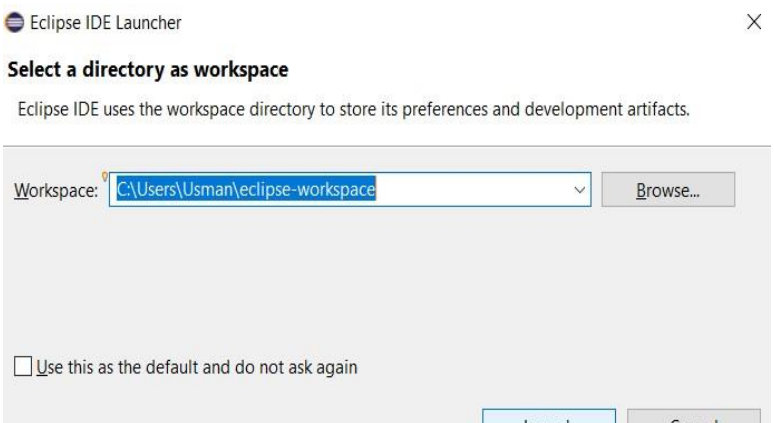
- ***Visit*** *https://www.seleniumhq.org/download/*
- *Under section "**Selenium Standalone Server**", click download version "**3.141.59**"*
- *You'll get the executable jar file(selenium-server-standalone-3.141.59)*

b) *For "Selenium Client Driver":*

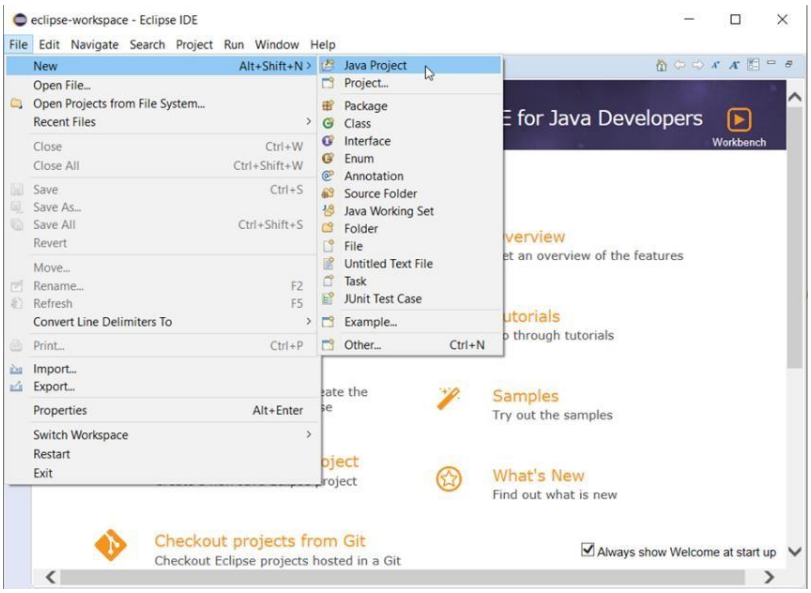
- ***Visit*** *https://www.seleniumhq.org/download/*
- *Under section "**Selenium Client & WebDriver Language Bindings**", download the "**3.141.59**" version of Java.*
- *Extract the file and you'll see two jar files. From them, we'll be using this executable jar file(client-combined-3.141.59)* ***4)*** *To Download "**Gecko Driver**":*
- ***Visit*** *https://github.com/mozilla/geckodriver/releases*
- *Under section "**Assets**", download "geckodriver-v0.24.0-win64.zip" file.*
- *You'll get the application file "geckodriver".*

STEPS:

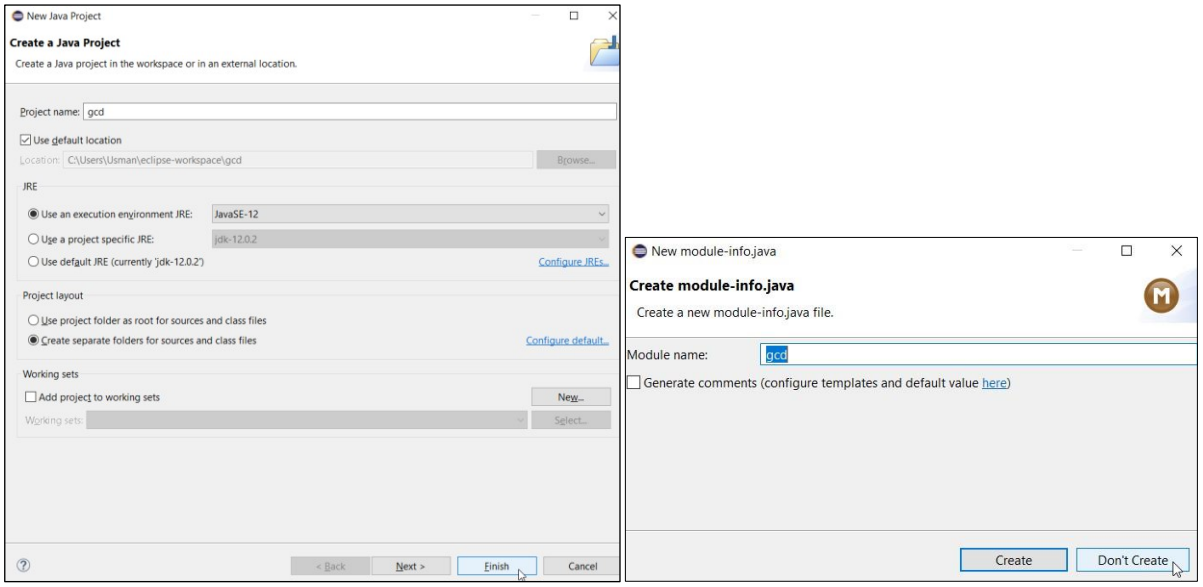
1) ***Open Eclipse****. Select your workspace directory. Click* ***Launch:***



2) *Create a Project File > New > Java Project*

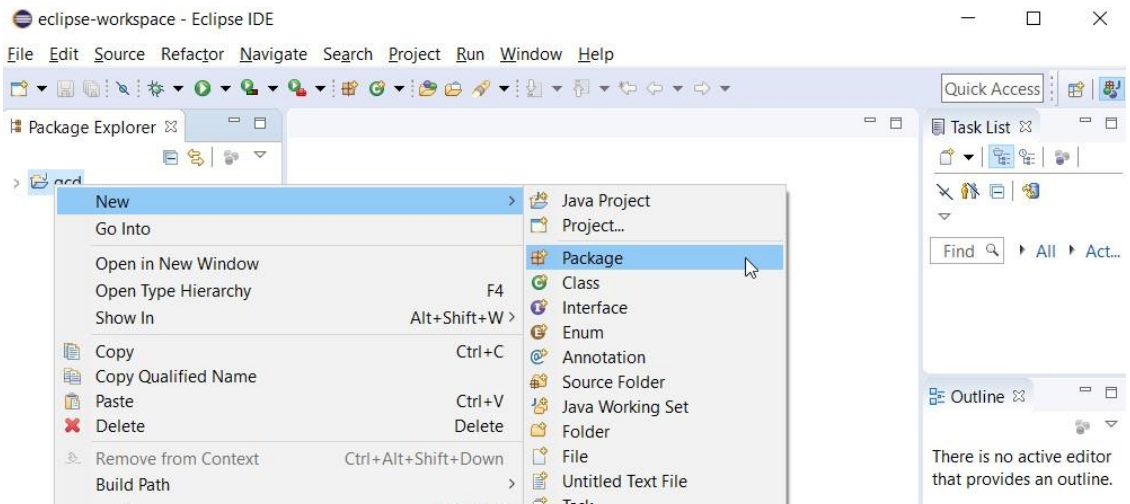


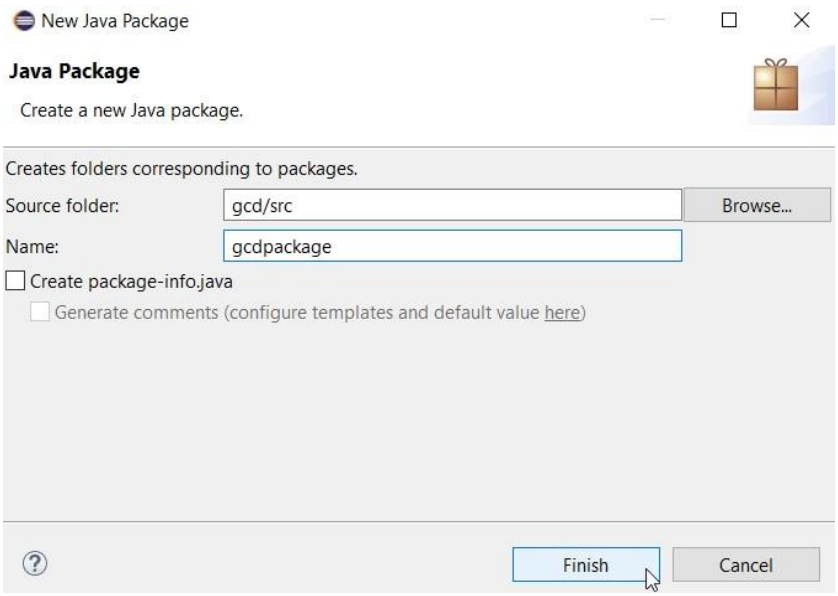
3) ***Name the project*** *as "gcd"* ***>*** *click* ***Finish >*** *click* ***Don't Create*** *module:*



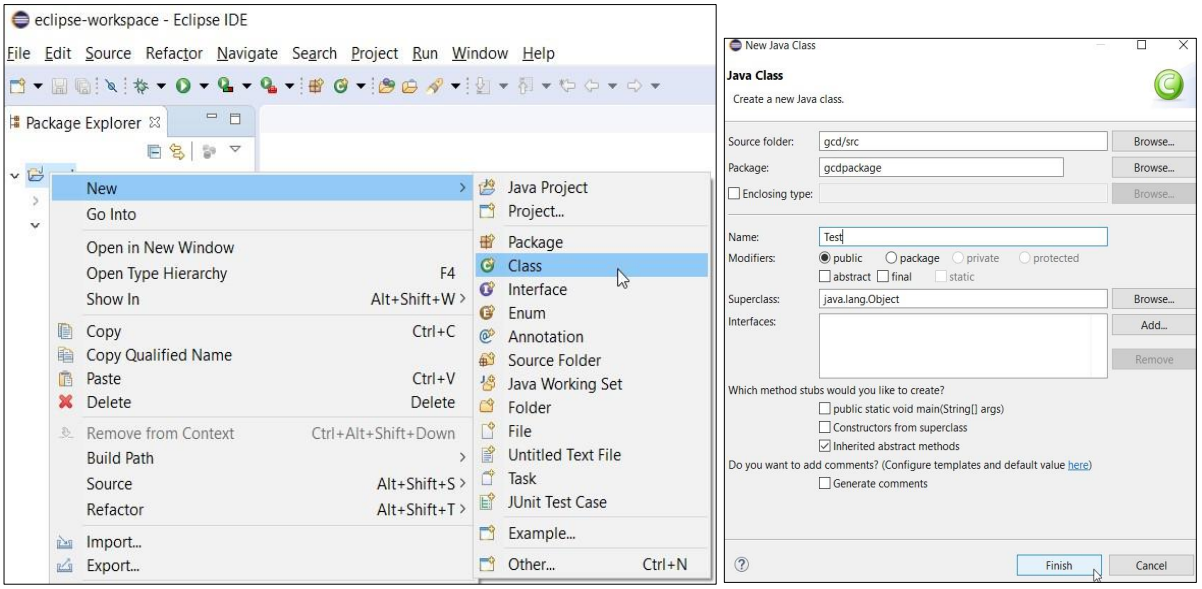
4) ***Close the "Welcome" tab****.*

5) ***Create*** *a Package(****right-click on Project Name > New > Package > Name*** *it* ***> Finish****):*

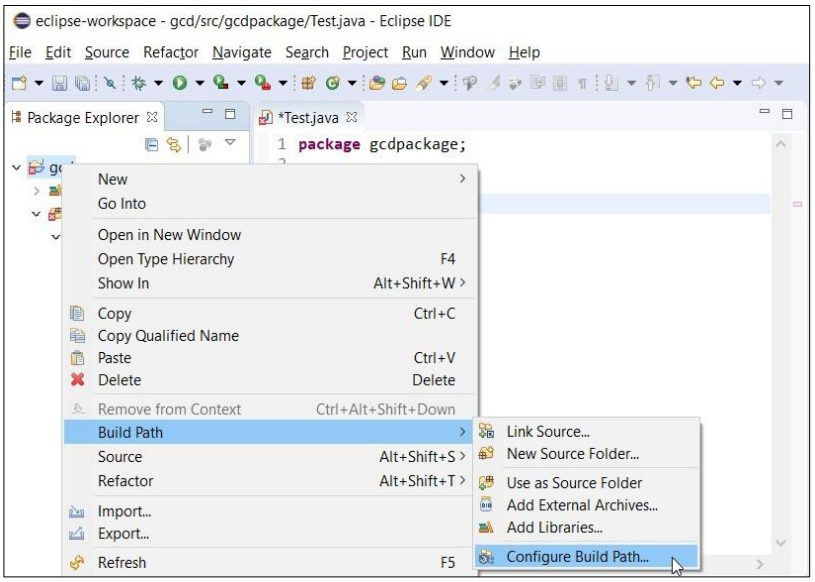




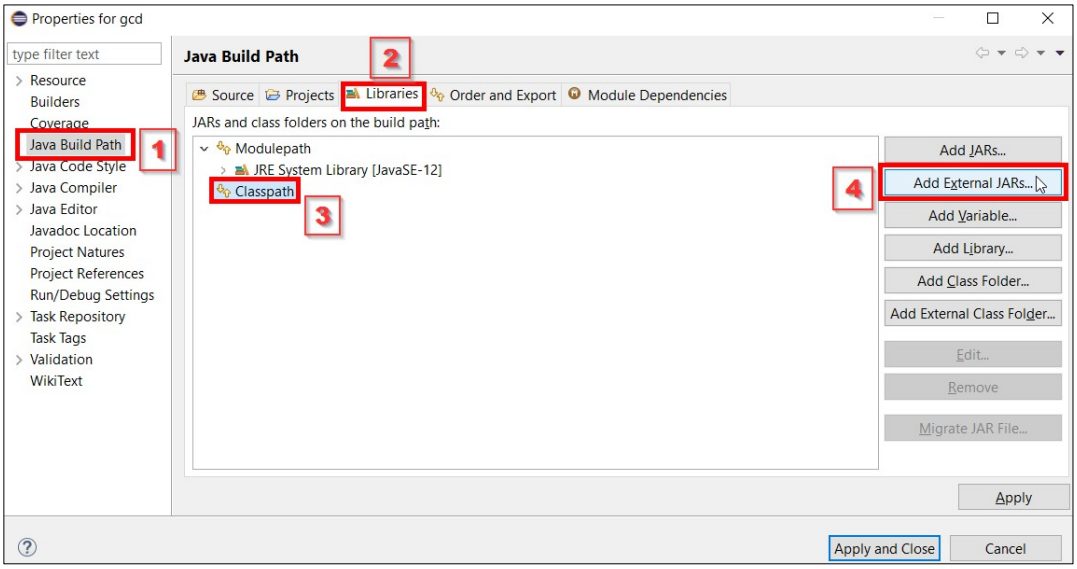
6) ***Create*** *a Class(****right-click on Project Name > New > Class > Name*** *it* ***> Finish****):*



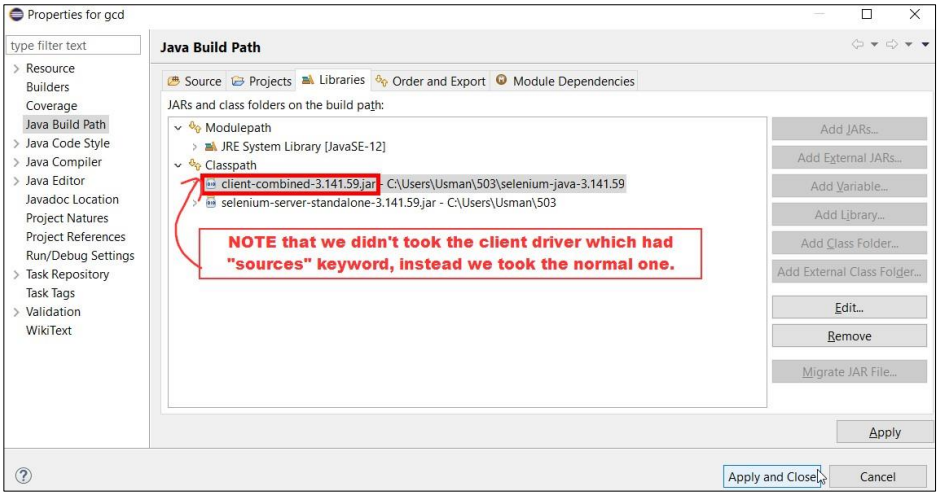
7) *Adding* ***"Selenium Server Driver and Client Driver(JAR files)"*** *in Eclipse IDE:* ***• right-click on Project Name > Build Path > Configure Build Path…***



- *now go under:* ***Java Build Path > Libraries > Classpath >*** *click* ***Add External JARs…***



- *Browse and* ***add JAR files >*** *click* ***Apply and Close :***



8) ***Creating a link for HTML file****(wherein calculation part is present):*

*(NOTE that this file will be run by the '****script*** *in JAVA'(which we'll create later))* •

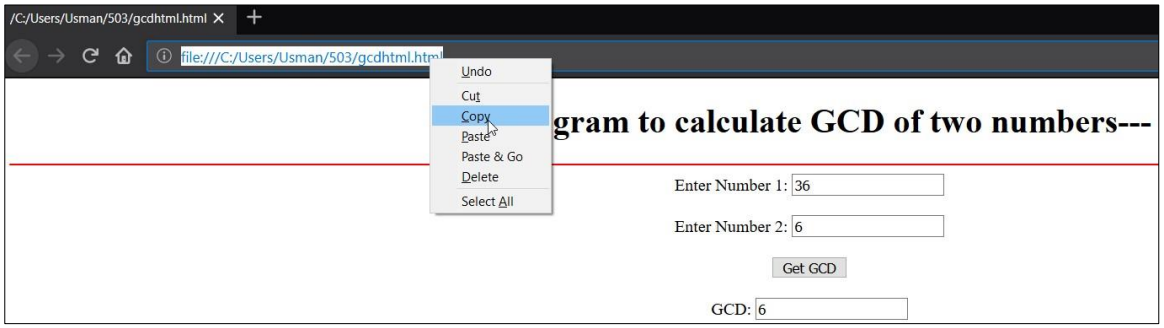
Create a Notepad file *with the following code and save it as "gcdhtml.html":*

---(gcdhtml.html)---

```
<html>
<head>
<script type="text/javascript">
        function gcd()
        {
                var x,y;
                x=parseInt(document.myform.n1.value);
                y=parseInt(document.myform.n2.value);
                while(x!=y)
                {
                        if(x>y){x=x-y;}
                                else{y=y-x;}
                }
```

```
            document.myform.result.value=x;
        }
</script>
</head>
<body>
<center>
        <h1>---Program to calculate GCD of two numbers---</h1>
        <hr color="red">

        <form name="myform">
            Enter Number 1: <input type="text" name="n1" value=""> <br> <br>
            Enter Number 2: <input type="text" name="n2" value=""> <br> <br>
            <input type="button" name="btn" value="Get GCD" onClick="gcd()"><br><br>
            GCD: <input type="text"  name="result" value="">
        </form>
</center>
</body>
</html>
```

- ***Close the file***. *Then* ***right-click > Open with > Firefox Browser***
- ***Copy URL*** *from the webpage:*



Creating the ***script*** *in JAVA*

(NOTE that this ***script*** *will be run by Eclipse IDE)*
(In simple words, it's like we are
-ordering Eclipse to run a script or to do a job
-of opening the HTML file
-and putting the values in the textboxes with the help of Selenium Drivers -and to show the result.
-Hence automating the work in browser)

- *Now we'll put the path of "geckodriver" in a String* ***driverPath***
- *And we'll* ***paste the copied URL in the .get() method*** *of the WebDriver class*

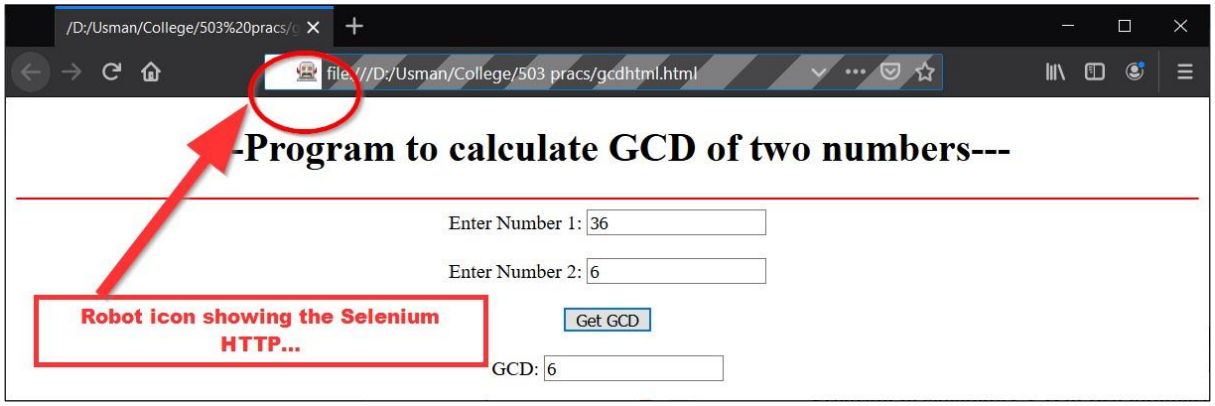
---(Test.java)---

```
package gcdpackage;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxOptions;
import org.openqa.selenium.firefox.FirefoxProfile;
```

```
public class Test {
    static String driverPath = "C:\\Users\\Usman\\503\\geckodriver.exe";
    public static void main(String[] args) {
            System.setProperty("webdriver.gecko.driver",driverPath);
            //DesiredCapabilities capabilities = DesiredCapabilities.firefox();
            //capabilities.setCapability("marionette",true);
            //ProfilesIni allProfiles = new ProfilesIni();
                  FirefoxProfile fp = new FirefoxProfile();
            fp.setPreference(FirefoxProfile.PORT_PREFERENCE,"7055");
            FirefoxOptions options = new FirefoxOptions();
            options.setProfile(fp);
            WebDriver driver=new FirefoxDriver(options);//(capabilities);
            //WebDriver driver=new ChromeDriver();
            driver.get("file:///D:/Usman/College/503%20pracs/gcdhtml.html");
            driver.manage().window().maximize();
                        driver.findElement(By.name("n1")).sendKeys("36");

driver.findElement(By.name("n2")).sendKeys("6");
            driver.findElement(By.name("btn")).click();
               String
result=driver.findElement(By.name("result")).getAttribute("name=result");
            System.out.println("GCD="+result);
      }
}
```

***10) Run** the file from Eclipse IDE:*

OUTPUT:



Finish!

What is Gecko Driver:

-a WebBrowser engine inbuilt within Mozilla Firefox browser.

-acts as a proxy between Web Driver enabled clients(Eclipse, NetBeans, etc.) and Mozilla Firefox browser or simply acts as a link between Selenium Web Driver tests and Mozilla Firefox browser.

***What is Selenium Remote Control (RC)**:*

-a test tool that allows you to write automated web application UI tests in any programming language against any HTTP website using any mainstream JavaScript-enabled browser.

-Selenium RC comes in two parts.

1) *A server which automatically launches and kills browsers, and acts as a HTTP proxy for web requests from them.*
2) *Client libraries for your favourite computer language.*

Practical No:04

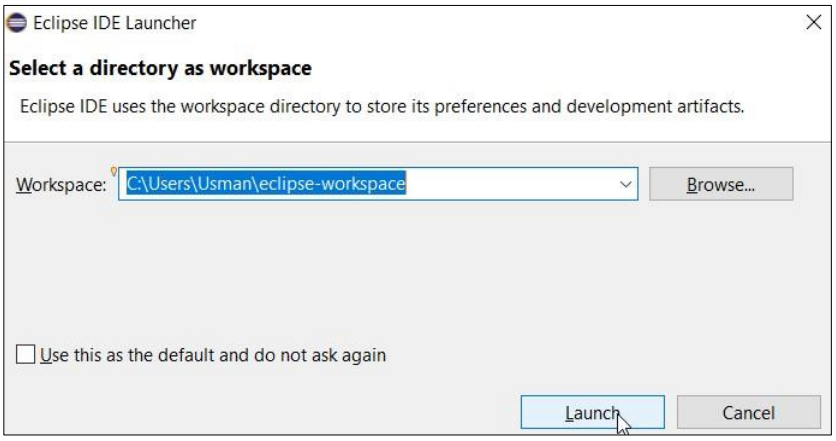
AIM: Write a program using Selenium WebDriver to update 10 student records in an Excel file. Perform data manipulation and verification.

PRE-REQUISITES:

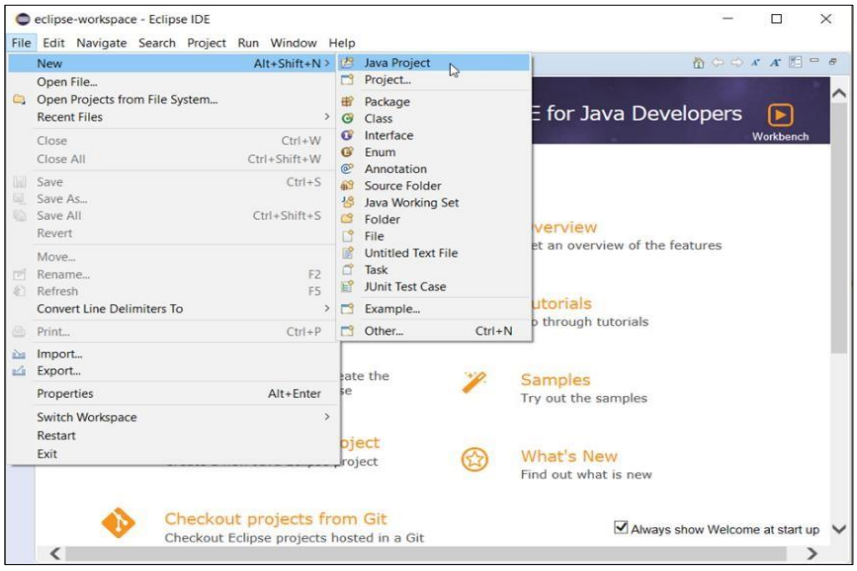
1) *Check that you have **Eclipse IDE**.*
2) *To Download "**JXL.JAR**":*
   - ***Visit** http://www.java2s.com/Code/Jar/j/Downloadjxl26jar.htm*
   - ***Download** this file: "jxl/jxl-2.6.jar.zip(603 k)" and **extract it**.(you'll get the .jar file)*

STEPS:

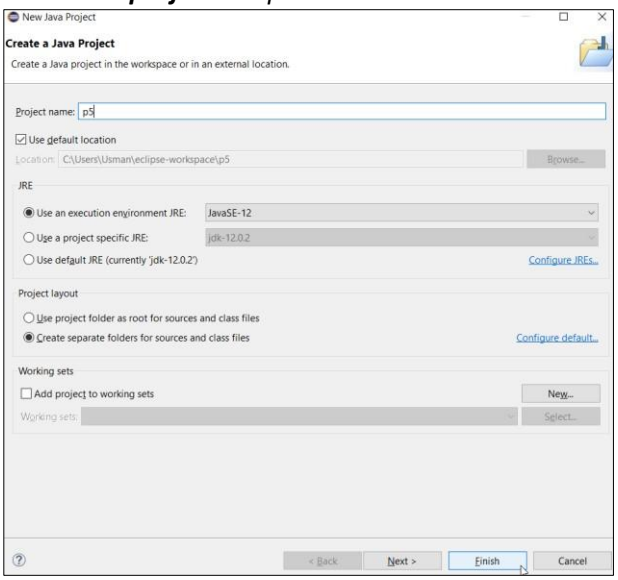
1) ***Open Eclipse**. Select your workspace directory. Click **Launch:***

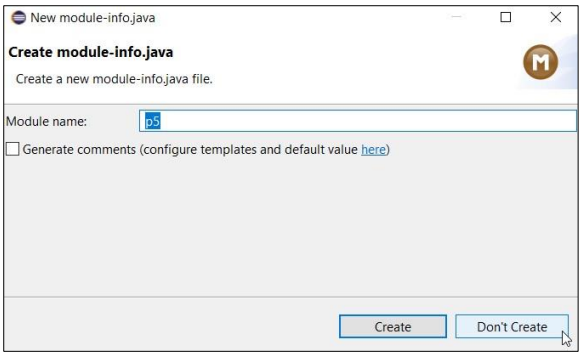


2) ***Create** a Project(**File > New > Java Project**):*



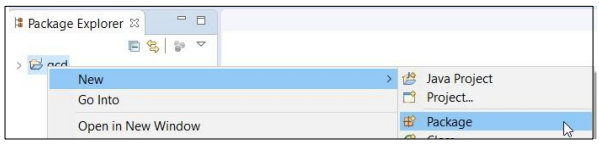
3) ***Name the project*** *as "p5"* ***>*** *click* ***Finish >*** *click* ***Don't Create*** *module:*

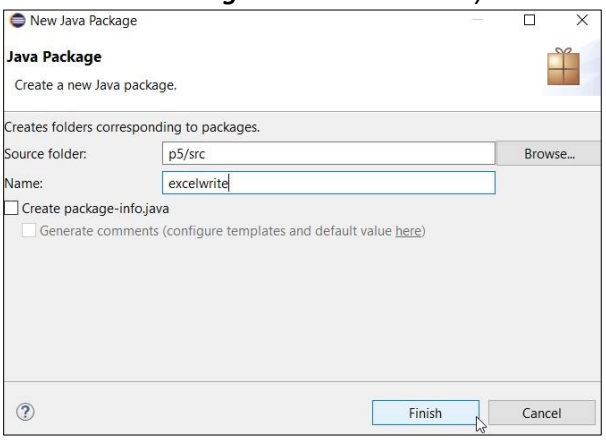




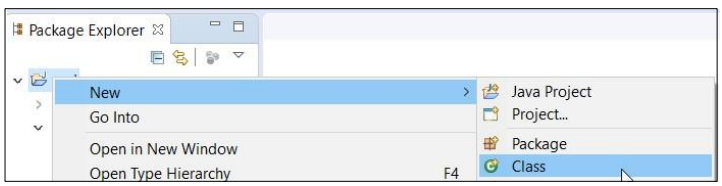
4) ***Close the "Welcome" tab****.*

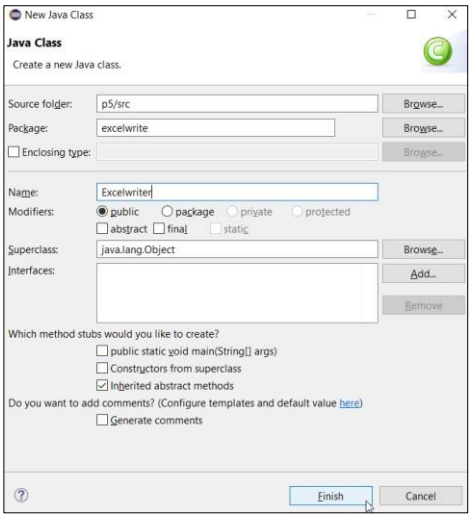
5) ***Create*** *a Package(****right-click on Project Name > New > Package > Name*** *it* ***> Finish****):*





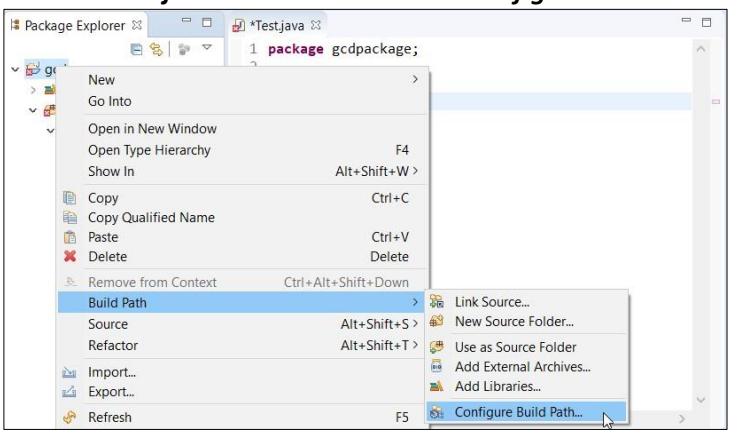
6) ***Create*** *a Class(****right-click on Project Name > New > Class > Name*** *it* ***> Finish****):*



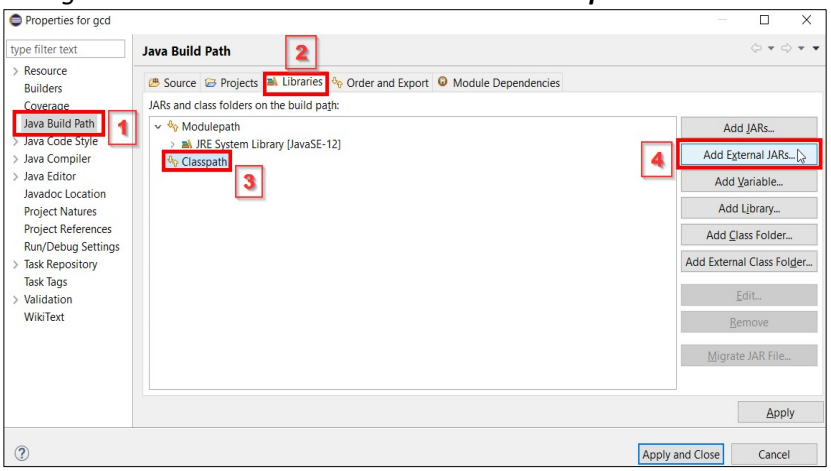


7) *Adding "**JXL(JAR file)**" in Eclipse IDE:*

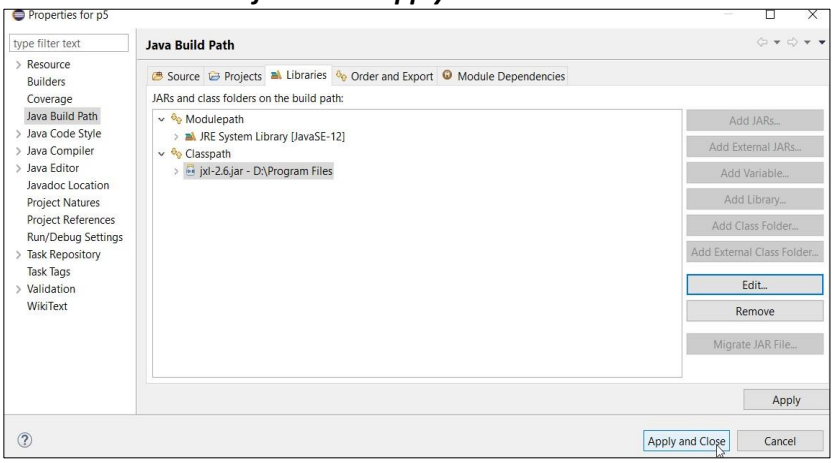
- ***right-click on Project Name > Build Path > Configure Build Path…***



- *now go under: **Java Build Path > Libraries > Classpath >** click **Add External JARs…***



- *Browse and **add JAR file >** click **Apply and Close :***



8) *Creating the **script** in JAVA:*

*(NOTE that this **script** will be run by Eclipse IDE)*
(In simple words, it's like we are
-ordering Eclipse to run a script or to do a job
-of creating and opening .xls file
-and putting the values in the cells with the help of jxl.jar -and to show the result.
-Hence automating the work in a local system(PC)).

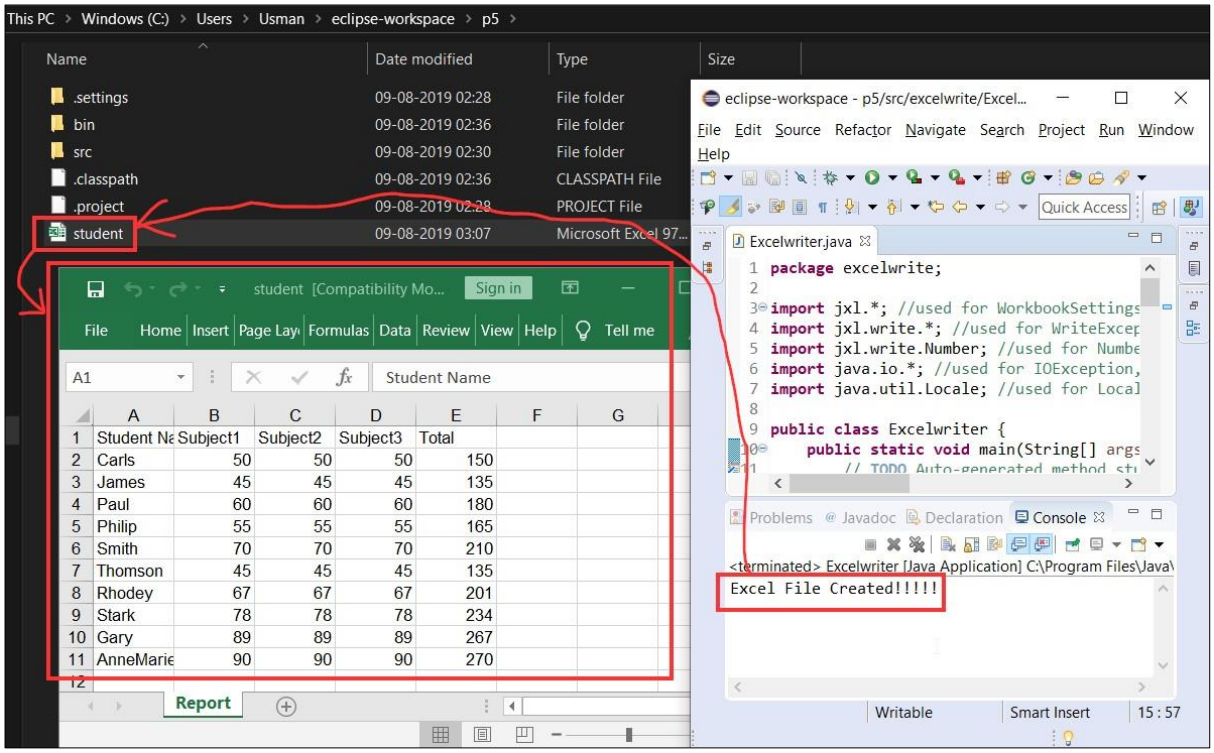
---(Excelwriter.java)---

```
package excelwrite;

import jxl.*; //used for WorkbookSettings,Workbook
import jxl.write.*; //used for WriteException,WritableWorkbook,WritableSheet,Label
import jxl.write.Number; //used for Number
import java.io.*; //used for IOException,File
import java.util.Locale; //used for Locale

public class Excelwriter {
        public static void main(String[] args) throws IOException,WriteException {
            // TODO Auto-generated method stub
                int r=0,c=0;
        String header[]={"Student Name","Subject1","Subject2","Subject3","Total"};
        String
      sname[]={"Carls","James","Paul","Philip","Smith","Thomson","Rhodey","Stark","
      Gary"
,"AnneMarie"};

            int marks[]={50,45,60,55,70,45,67,78,89,90,30};

               File = new File("student.xls");
            WorkbookSettings wbSettings = new WorkbookSettings();
                        wbSettings.setLocale(new Locale("en", "EN"));
            WritableWorkbook workbook = Workbook.createWorkbook(file, wbSettings);
            workbook.createSheet("Report", 0);
            WritableSheet excelSheet = workbook.getSheet(0);
            //creating header row
            for(r=0;r<1;r++) {
                        for(c=0;c<header.length;c++) {
                               Label l=new
Label(c,r,header[c]);
                        excelSheet.addCell(l);
                }
            }
            //filling name in column1
                   for(r=1;r<=sname.length;r++) {
                               for(c=0;c<1;c++) {
                        Label l=new Label(c,r,sname[r-1]);
                        excelSheet.addCell(l);
                }
            }
            //filling name in column2,3,4
                   for(r=1;r<=sname.length;r++) {
                               for(c=1;c<4;c++) {
                        Number num = new Number(c, r, marks[r-1]);
                        excelSheet.addCell(num);
                }
            }
            //filling name in total
                   for(r=1;r<=sname.length;r++) {
                               for(c=4;c<5;c++) {
                        int total=marks[r-1]+marks[r-1]+marks[r-1];
                                    Number num = new Number(c, r, total);
                                                excelSheet.addCell(num);
                }
            }
            workbook.write();
            workbook.close();
```

```
            System.out.println("Excel File Created!!!!!");
        }
}
```

***9) Run** the file from Eclipse IDE:*

OUTPUT:



Finish!

What is JXL.JAR:

-Java libraries are distributed as a ".jar" file

*-jxl.jar is the library of JExcelApi , which is open source java API to read, write, and modify Excel spreadsheets dynamically. It contains all the compiled *.class files, associated metadata and resources that are used by the Java Excel API internally .*

Prasctical No: 05

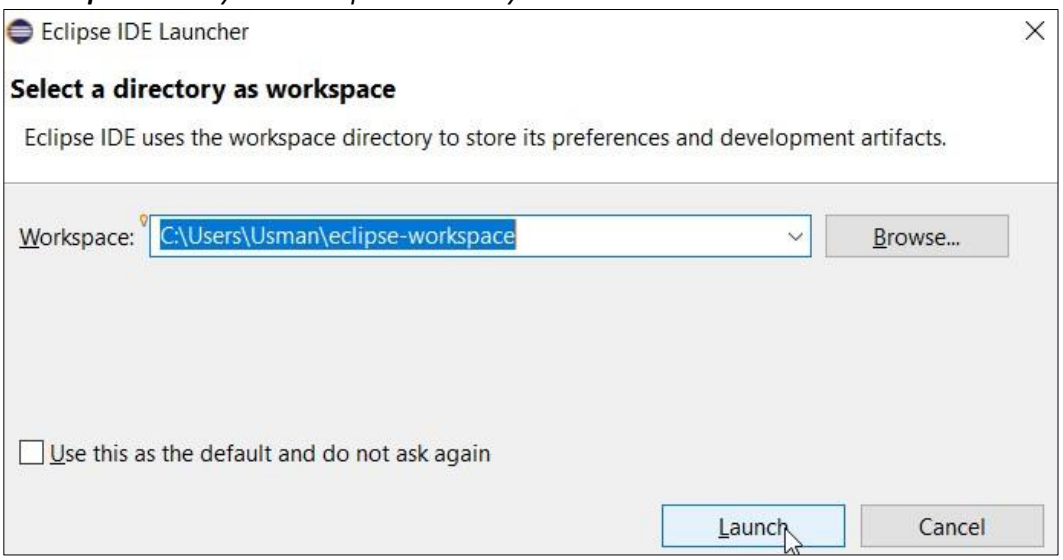
AIM: Write a program using Selenium WebDriver to select the number of students who have scored more than 60 in any one subject (or all subjects). Perform data extraction and analysis

PRE-REQUISITES:

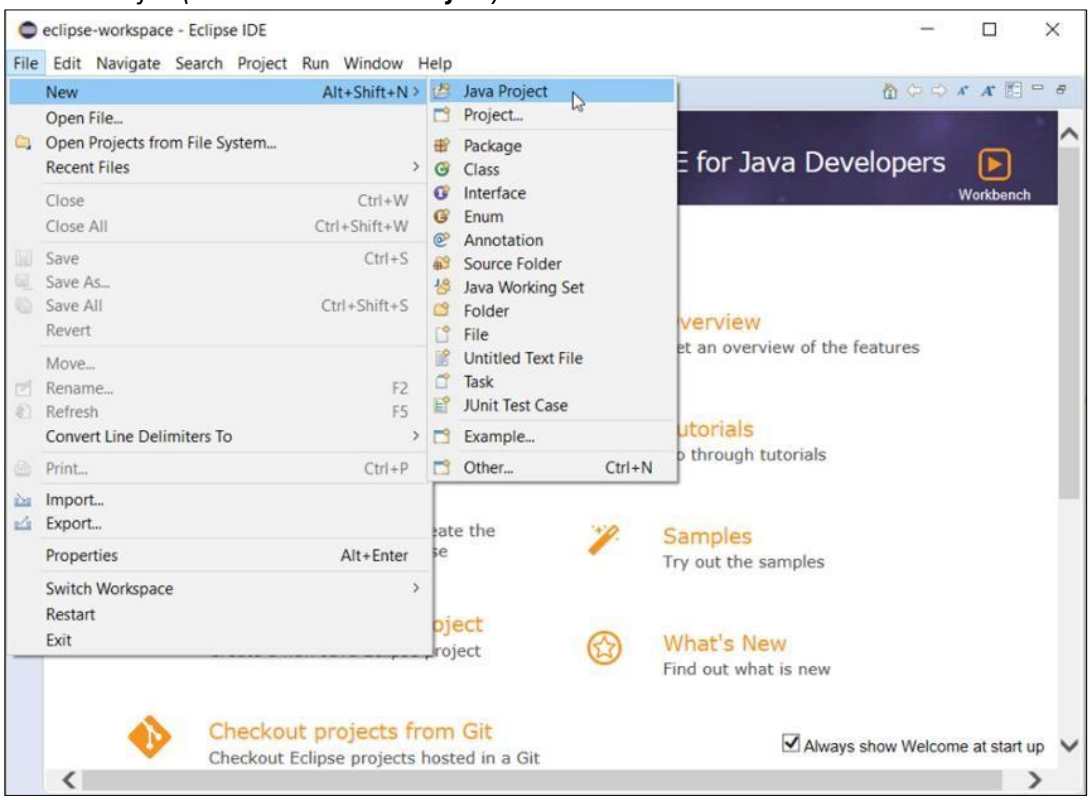
1) *Check that you have **Eclipse IDE**.*
2) *Check that you have **JXL.JAR**.*
3) *Check that you've the **Excel file**("student.xls") that we'll be working on for reading data.*

STEPS:

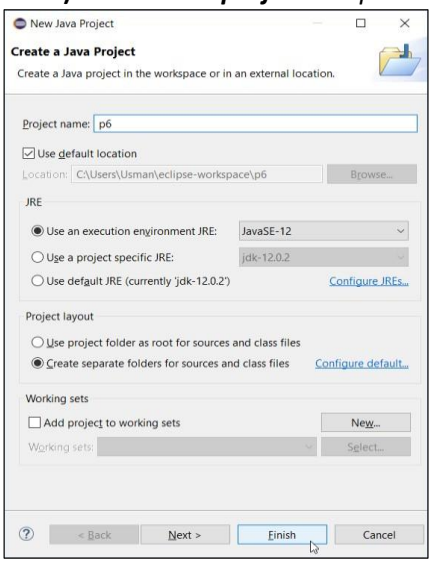
1) ***Open Eclipse**. Select your workspace directory. Click **Launch:***

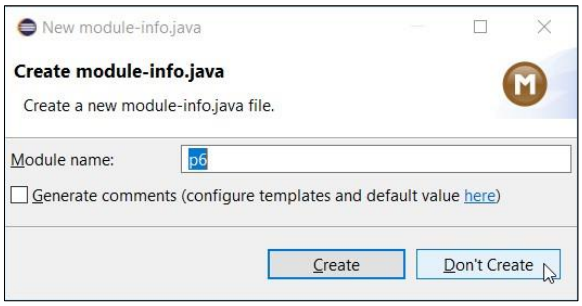


2) ***Create** a Project(**File > New > Java Project**):*



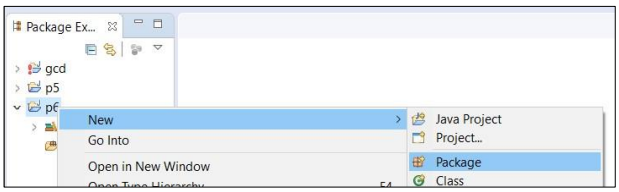
3) ***Name the project*** *as "p6"* ***>*** *click* ***Finish >*** *click* ***Don't Create*** *module:*

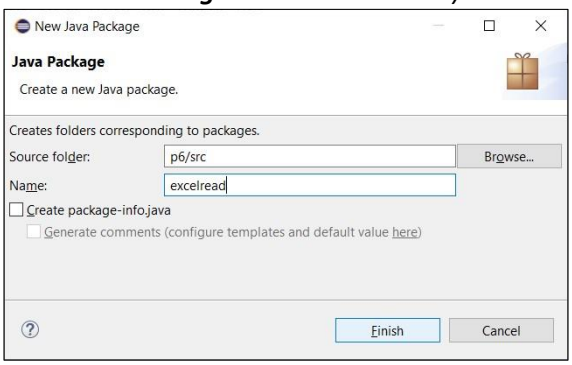




4) ***Close the "Welcome" tab****.*

5) ***Create*** *a Package(****right-click on Project Name > New > Package > Name*** *it* ***> Finish****):*

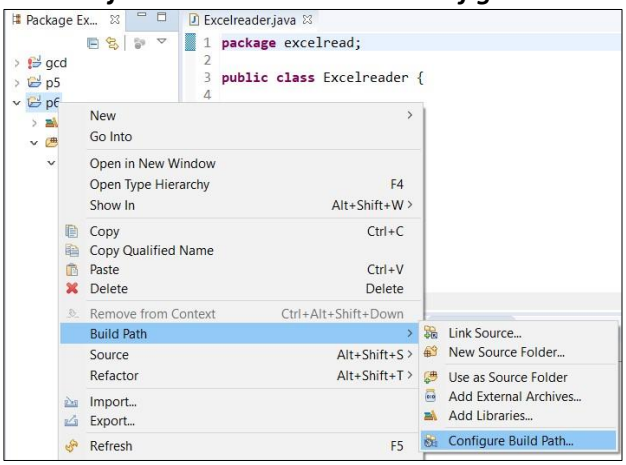




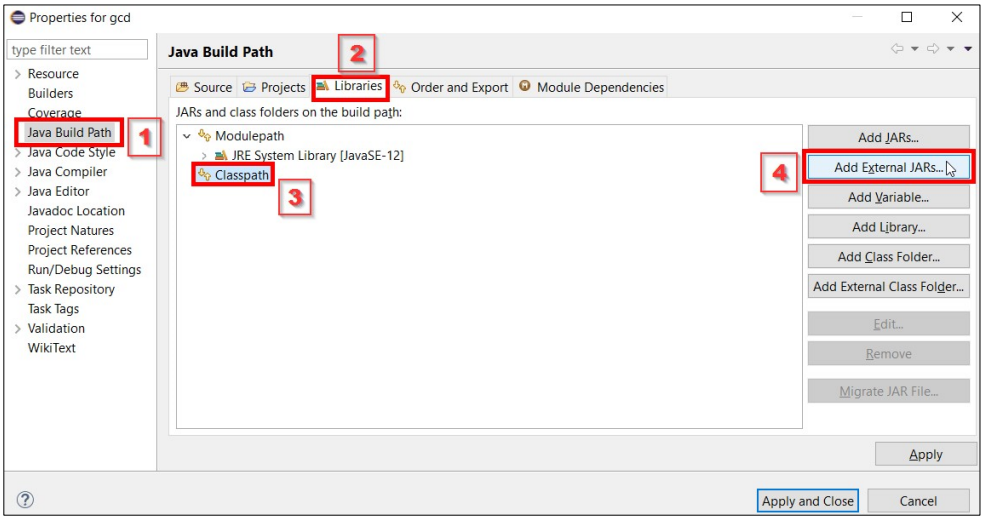
6) ***Create*** *a Class(****right-click on Project Name > New > Class > Name*** *it* ***> Finish****):*

7) *Adding "**JXL(JAR file)**" in Eclipse IDE:*

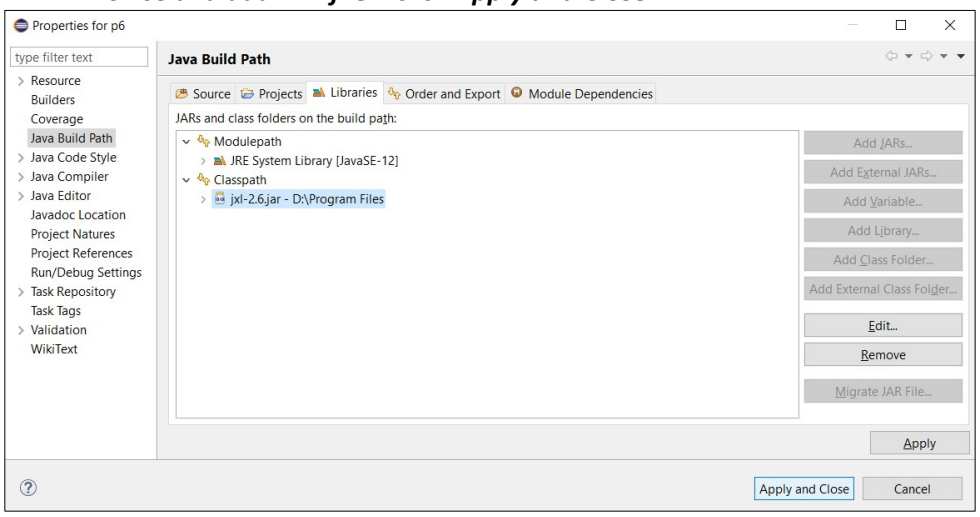
- ***right-click on Project Name > Build Path > Configure Build Path…***



- *now go under: **Java Build Path > Libraries > Classpath >** click **Add External JARs…***



- *Browse and **add JAR file >** click **Apply and Close :***



8) *Creating the **script** in JAVA:*

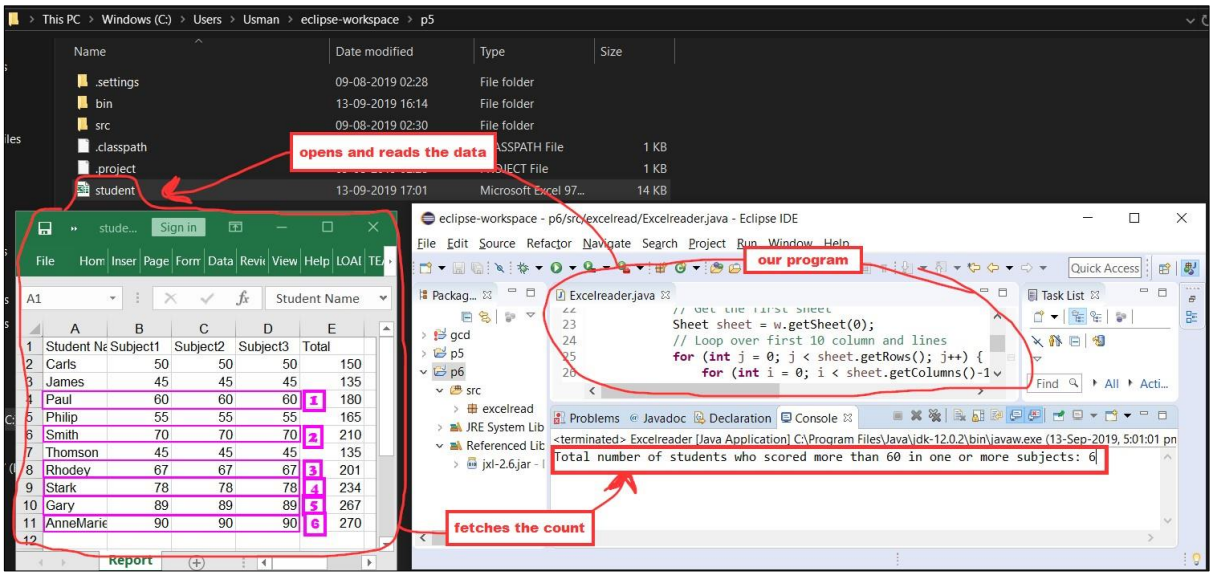
*(NOTE that this **script** will be run by Eclipse IDE)*
(In simple words, it's like we are
-ordering Eclipse to run a script or to do a job
-of opening .xls file
-and fetching the marks count>=60 from the cells with the help of jxl.jar -and to show the result.
-Hence automating the work in a local system(PC)).

---(Excelreader.java)---

```
package excelread;
import java.io.File;
import java.io.IOException;
import jxl.Cell;
import jxl.CellType;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;
public class Excelreader {
	private String inputFile;
	public void setInputFile(String inputFile) {this.inputFile = inputFile;}

		public void read() throws IOException {
					File inputWorkbook =
new File(inputFile);
			Workbook w;
				boolean
flag=false;
			int count=0;
			try {
			   w = Workbook.getWorkbook(inputWorkbook);
			  // Get the first sheet
			  Sheet = w.getSheet(0);
			// Loop over first 10 column and lines

			for (int j = 0; j < sheet.getRows(); j++) {
					for (int i = 0; i < sheet.getColumns()-1;
			i++) {                                  Cell =
			sheet.getCell(i, j);                                  if
			(cell.getType() == CellType.NUMBER) {


	if(Integer.parseInt(cell.getContents())>=60){
                                                    flag = true;
                                                   if(flag == true){
                                                     count++;
                                                     flag=false;
                                               }
                                              break;
                                        }
                                   }
                              }
                         }
		System.out.println("Total number of students who scored more than 60
in one or more subjects: " +count);
		}
		catch (BiffException e) {e.printStackTrace();}
	}


	public static void main(String[] args) throws IOException {
                         Excelreader test = new Excelreader();

test.setInputFile("C:\\Users\\Usman\\eclipseworkspace\\p5\\student.xls");
		test.read();
	}
}
```

***9) Run** the file from Eclipse IDE:*

OUTPUT:



Finish!

Practical No: 06

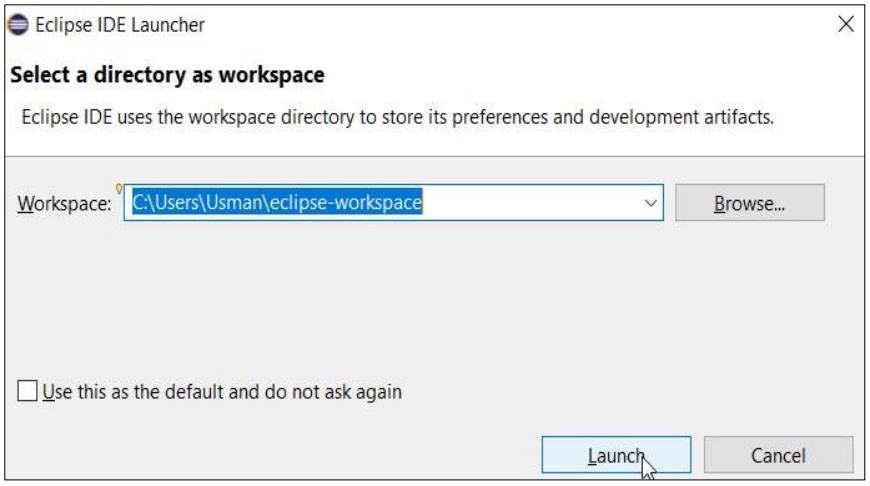
AIM: Write a program using Selenium WebDriver to provide the total number of objects present or available on a web page. Perform object identification and counting

PRE-REQUISITES:

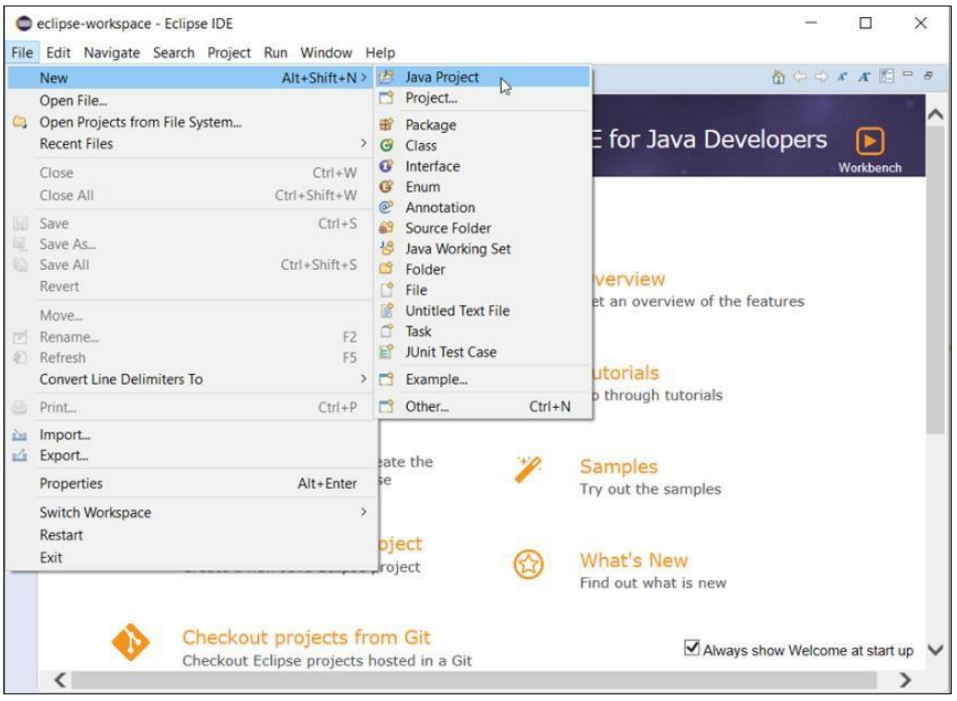
1) *Check that you have **JDK**.*
2) *Check that you have **Eclipse IDE**.*
3) *Check that you have **Selenium Server Driver and Client Driver(JAR files)**.*
4) *Check that you have **Gecko Driver**.*
5) *Check that you have a stable Internet connection.*

STEPS:

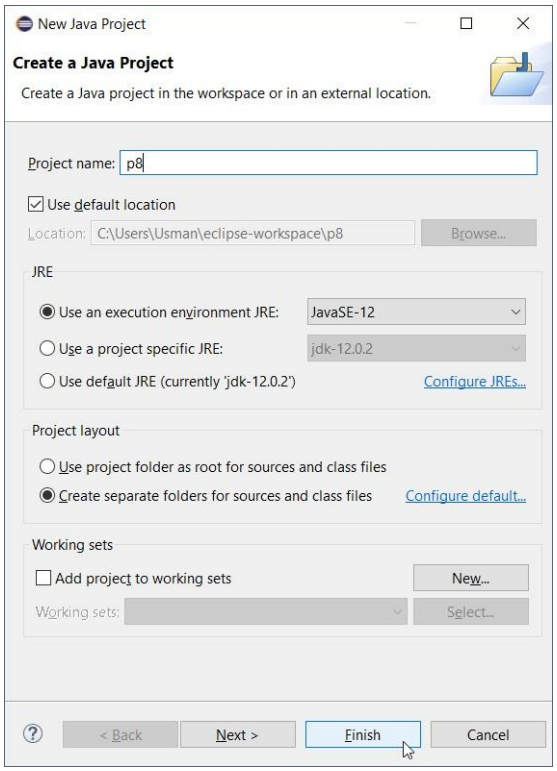
1) Open Eclipse**. Select your workspace directory. Click **Launch:

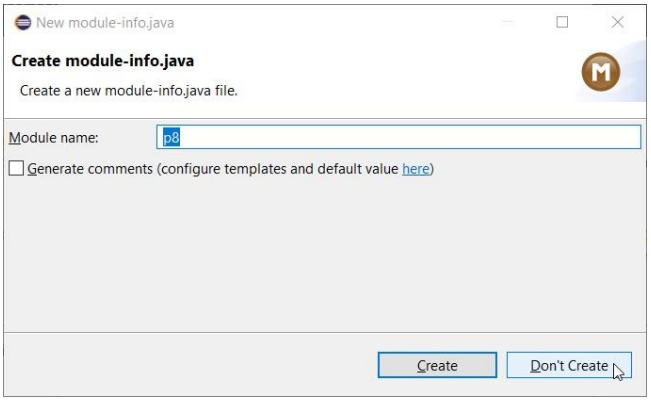


Create** a Project **File > New > Java Project



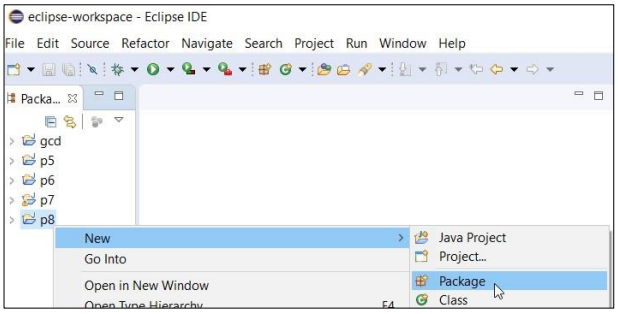
3) ***Name the project*** *as "p8"* ***>*** *click* ***Finish >*** *click* ***Don't Create*** *module:*

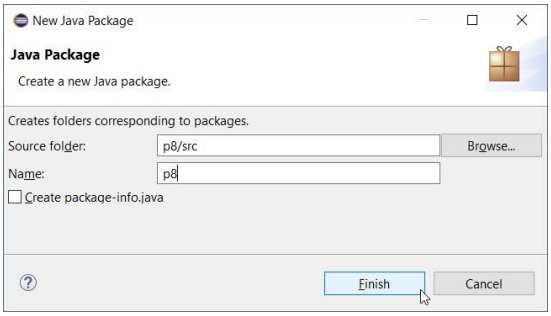




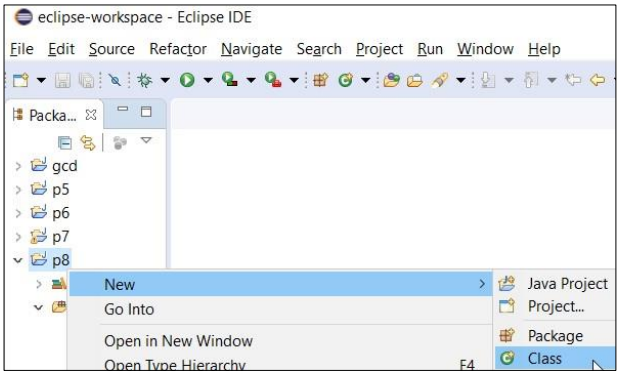
4) ***Close the "Welcome" tab****.*

5) ***Create*** *a Package(****right-click on Project Name > New > Package > Name*** *it* ***> Finish****):*

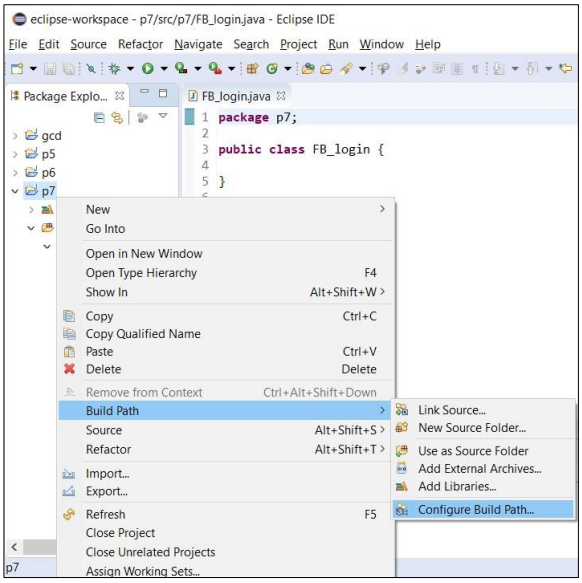




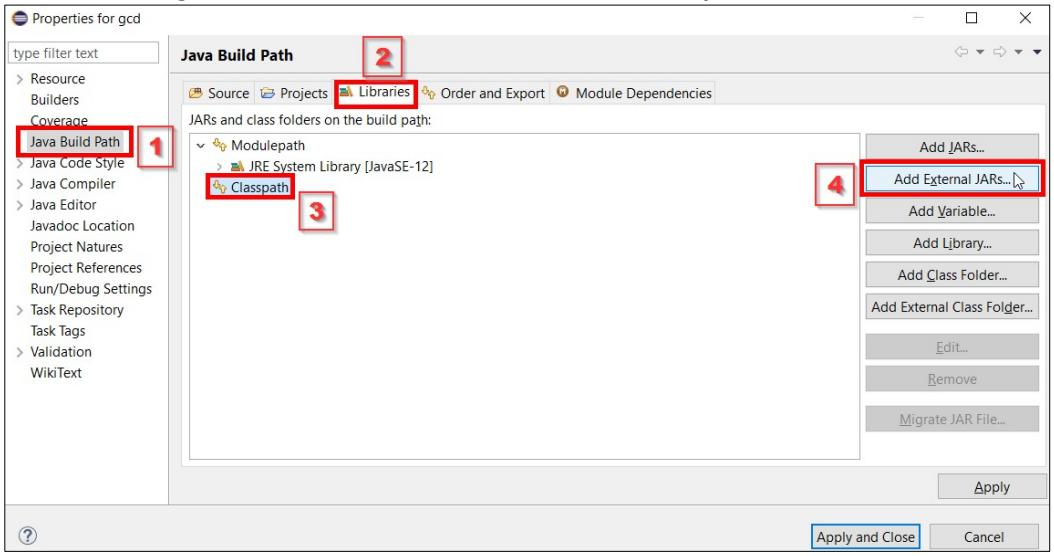
6) ***Create*** *a Class(****right-click on Project Name > New > Class > Name*** *it* ***> Finish****):*



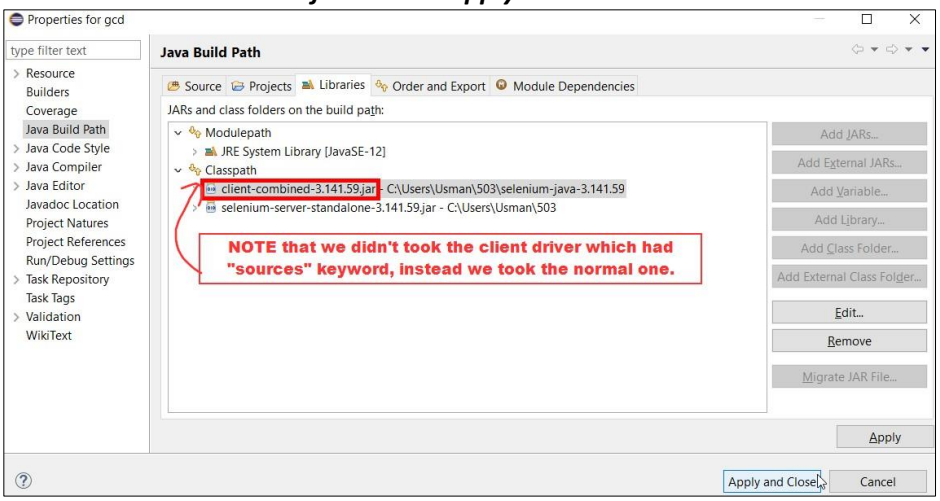
7) *Adding **"Selenium Server Driver and Client Driver(JAR files)"** in Eclipse IDE: • **right-click on Project Name > Build Path > Configure Build Path…***



- *now go under: **Java Build Path > Libraries > Classpath >** click **Add External JARs…***



- *Browse and **add JAR files >** click **Apply and Close :***



8) *Creating the **script** in JAVA:*

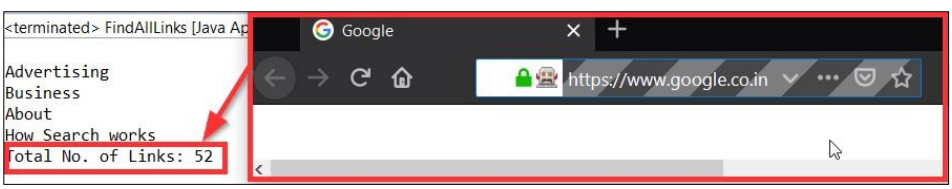
*(NOTE that this **script** will be run by Eclipse IDE)*
(In simple words, it's like we are
-ordering Eclipse to run a script or to do a job
-of opening the browser, visiting the URL
-and finding the <a> tag WebElements with the help of Selenium Drivers -and
to show the result.
-Hence automating the work in browser)

• *Now we'll put the path of "geckodriver" in a String **driverPath***

---(FindAllLinks.java)---

```
package p8;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.*;
import org.openqa.selenium.firefox.FirefoxOptions;
import org.openqa.selenium.firefox.FirefoxProfile;
import
org.openqa.selenium.firefox.internal.ProfilesIni;

public class FindAllLinks {
	static String driverPath = "C:\\Users\\Usman\\503\\geckodriver.exe";
	public static void main(String[] args) {
		System.setProperty("webdriver.gecko.driver",driverPath);
		 //NOTE THAT: following commented lines are required for old machines
		//DesiredCapabilities capabilities = DesiredCapabilities.firefox();
		//capabilities.setCapability("marionette",true);
		//ProfilesIni allProfiles = new ProfilesIni();
		//FirefoxProfile fp = new FirefoxProfile();
		//fp.setPreference(FirefoxProfile.PORT_PREFERENCE,"7055");
		//FirefoxOptions options = new FirefoxOptions();
		//options.setProfile(fp);
		WebDriver driver = new FirefoxDriver();
			String appUrl
="https://www.google.co.in/";
		driver.get(appUrl);

		java.util.List<WebElement> links =
driver.findElements(By.tagName("a"));

		for (int i = 1; i<links.size(); i=i+1)
		{
			System.out.println(links.get(i).getText());
		}
		System.out.println("Total No. of Links: "+links.size());
		//driver.quit();
	}
}
```

Run the file from Eclipse IDE: OUTPU



Practical No: 07

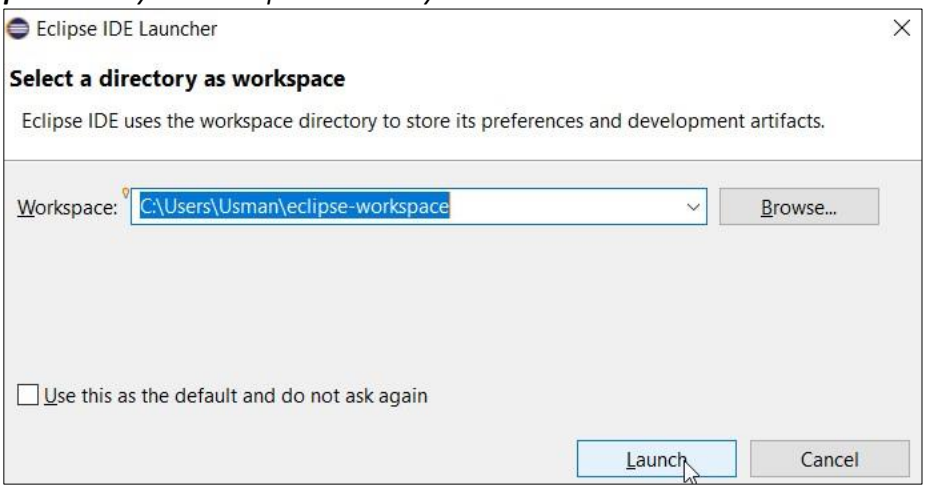
AIM: Write a program using Selenium WebDriver to get the number of items in a list or combo box on a web page. Perform element identification and counting

PRE-REQUISITES:

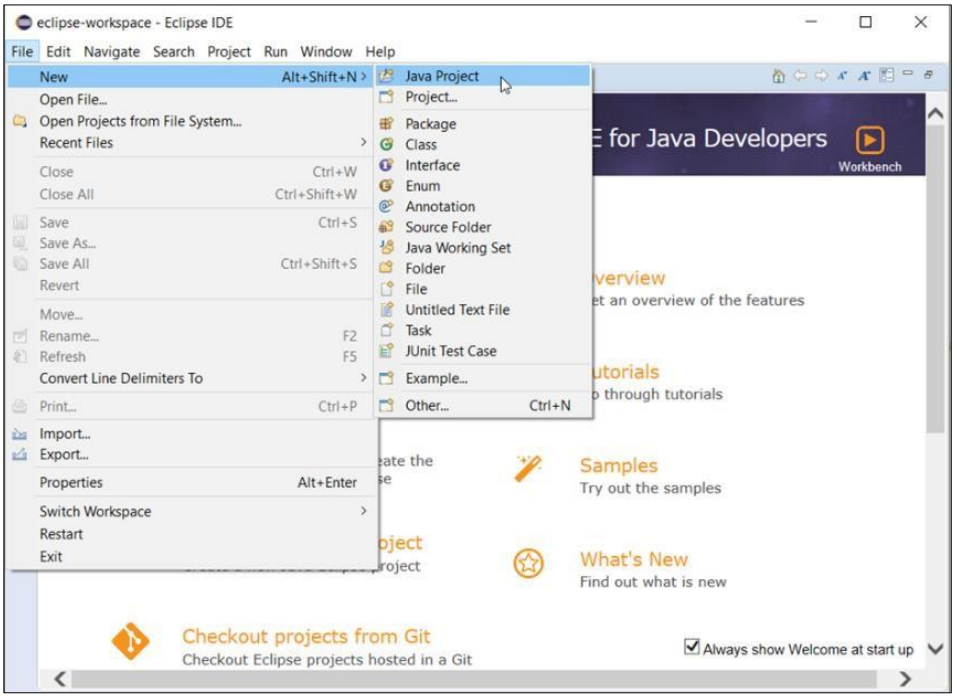
1) *Check that you have **JDK**.*
2) *Check that you have **Eclipse IDE**.*
3) *Check that you have **Selenium Server Driver and Client Driver(JAR files)**.*
4) *Check that you have **Gecko Driver**.*
5) *Check that you have a stable Internet connection.*

STEPS:

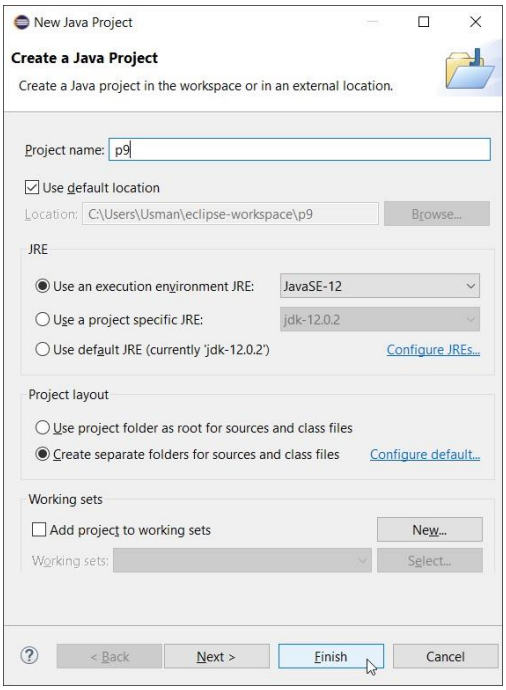
1) Open Eclipse**. Select your workspace directory. Click **Launch:

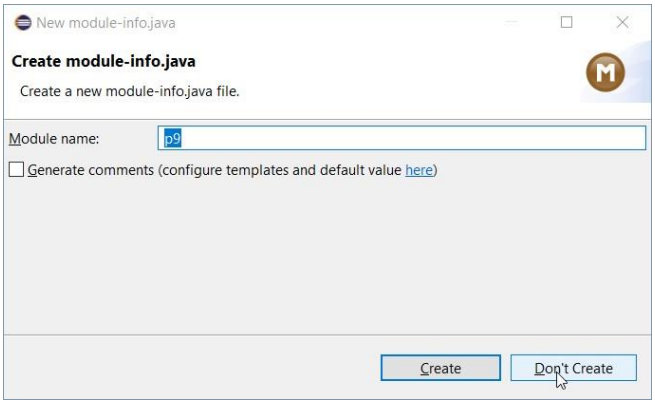


Create** a Project(**File > New > Java Project



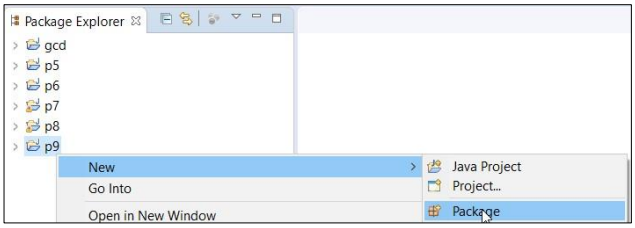
3) ***Name the project*** *as "p9"* ***>*** *click* ***Finish >*** *click* ***Don't Create*** *module:*

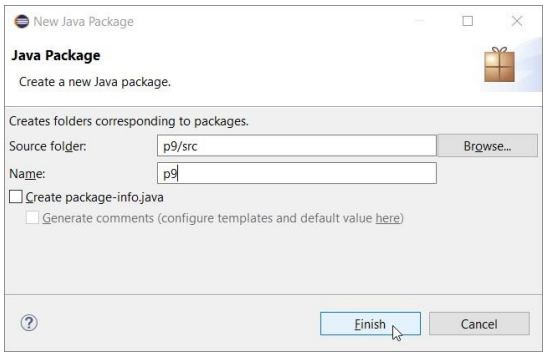




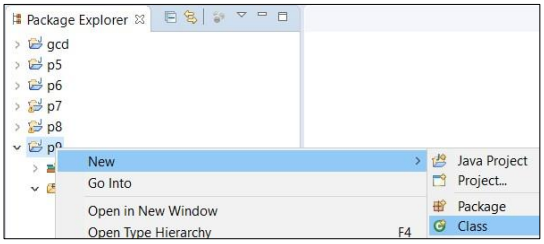
4) ***Close the "Welcome" tab****.*

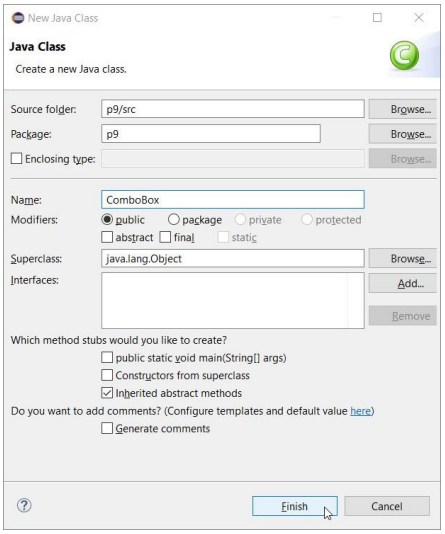
5) ***Create*** *a Package(****right-click on Project Name > New > Package > Name*** *it* ***> Finish****):*





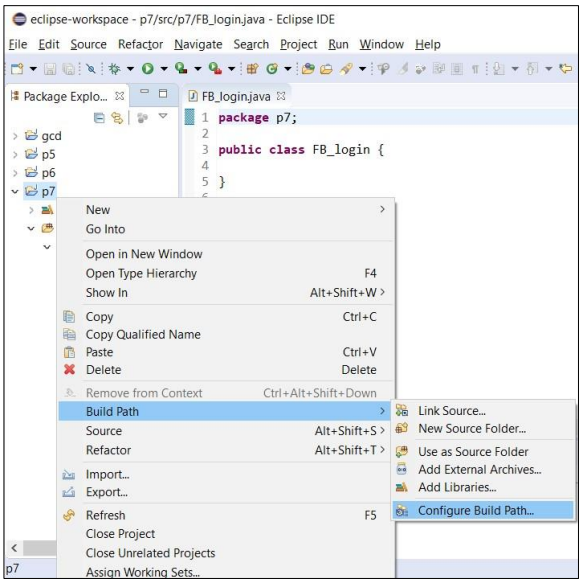
6) ***Create*** *a Class(****right-click on Project Name > New > Class > Name*** *it* ***> Finish****):*



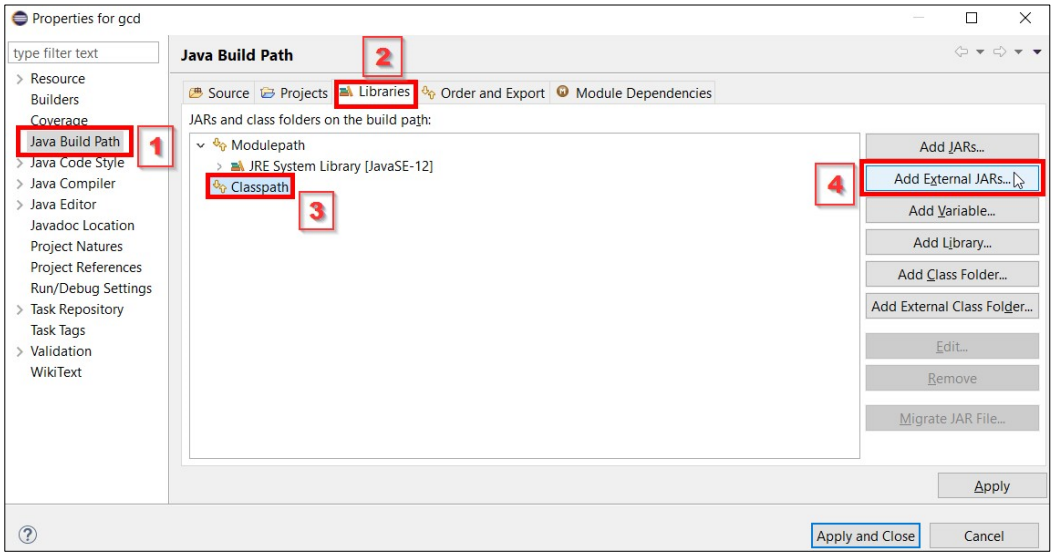


7) *Adding **"Selenium Server Driver and Client Driver(JAR files)"** in Eclipse IDE:*

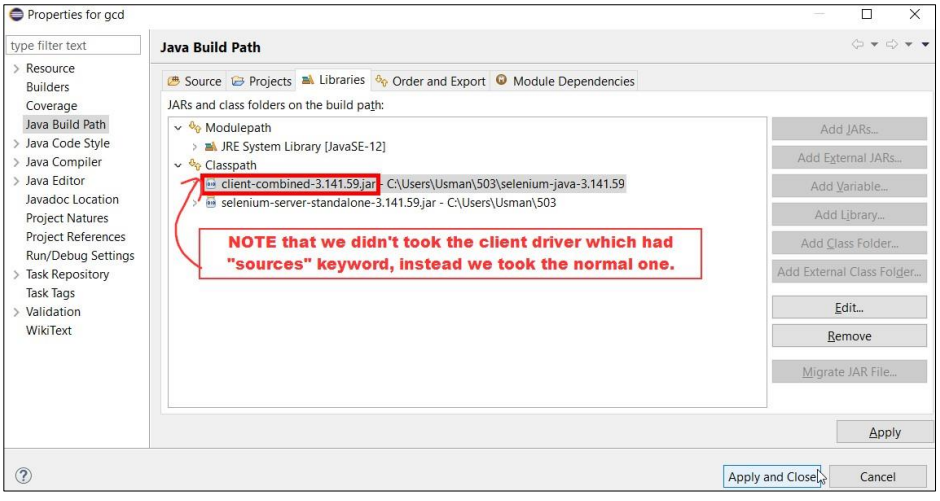
- ***right-click on Project Name > Build Path > Configure Build Path…***



- *now go under: **Java Build Path > Libraries > Classpath >** click **Add External JARs…***



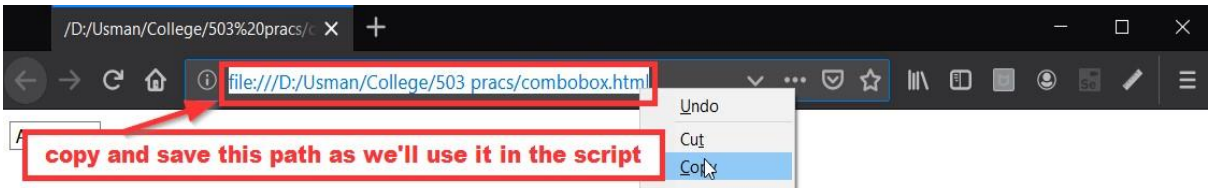
- *Browse and **add JAR files >** click **Apply and Close :***



8) ***Create** HTML file in a notepad **> Save it > Open it in browser > Copy the URL:***
(NOTE THAT: as this is our LOCAL FILE, this file will be opened thru STATIC URL in script)

---(combobox.html)---

```
<select id="continents">
        <option value="Asia">Asia</option>
        <option value="Europe">Europe</option>
        <option value="Africa">Africa</option>
</select>
```



9) *Creating the **script** in JAVA:*
*(NOTE that this **script** will be run by Eclipse IDE)*
(In simple words, it's like we are
-ordering Eclipse to run a script or to do a job
-of opening the browser, visiting the URL
-and finding the WebElement By "ID" with the help of "Select" class and Selenium Drivers -and to show the result.
-Hence automating the work in browser)

- *Now we'll put the path of our LOCAL FILE(combobox.html) in a string*
- *Also put the path of "geckodriver" in a String **driverPath***

---(ComboBox.java)---

```
package p9;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.*;
import org.openqa.selenium.firefox.FirefoxOptions;
import org.openqa.selenium.firefox.FirefoxProfile;
import org.openqa.selenium.firefox.internal.ProfilesIni;
import org.openqa.selenium.support.ui.Select;

public class ComboBox {
	static String driverPath = "C:\\Users\\Usman\\503\\geckodriver.exe";
	public static void main(String[] args) {
	    System.setProperty("webdriver.gecko.driver",driverPath);
	     //NOTE THAT: following commented lines are required for old machines
	     //DesiredCapabilities capabilities = DesiredCapabilities.firefox();
	      //capabilities.setCapability("marionette",true);
	      //ProfilesIni allProfiles = new ProfilesIni();
	      //FirefoxProfile fp = new FirefoxProfile();
	      //fp.setPreference(FirefoxProfile.PORT_PREFERENCE,"7055");
	      //FirefoxOptions options = new FirefoxOptions();
	                  //options.setProfile(fp);
	      WebDriver driver = new FirefoxDriver();
```

```
		String appUrl ="https://www.toolsqa.com/automation-practice-form/";
		//DYNAMIC URL(WEBSITE)
		//String appUrl = "
		file:///D:/Usman/College/503%20pracs/combobox.html"; //STATIC
	URL(LOCAL FILE)
		driver.get(appUrl);

		Select oSelect = new
		Select(driver.findElement(By.id("continents")));
		//Select oSelect = new
		Select(driver.findElement(By.tagName("select")));
		//this works too
		List<WebElement> oSize = oSelect.getOptions();
		int iListSize = oSize.size();

		for(int i =0; i < iListSize ; i++)
		{
			// Storing the value of the option
			String sValue = oSelect.getOptions().get(i).getText();
			// Printing the stored value
			System.out.println(sValue);
		}
		System.out.println("Total No. Items in Dropdown: "+iListSize);
		//driver.quit();
	}
}
```

10) Run the file from Eclipse IDE:

- ***OUTPUT:***

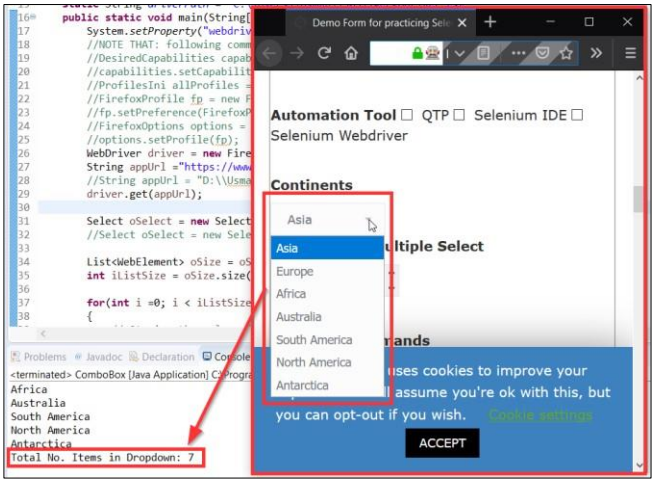


Fig : by using the DYNAMIC URL(website)

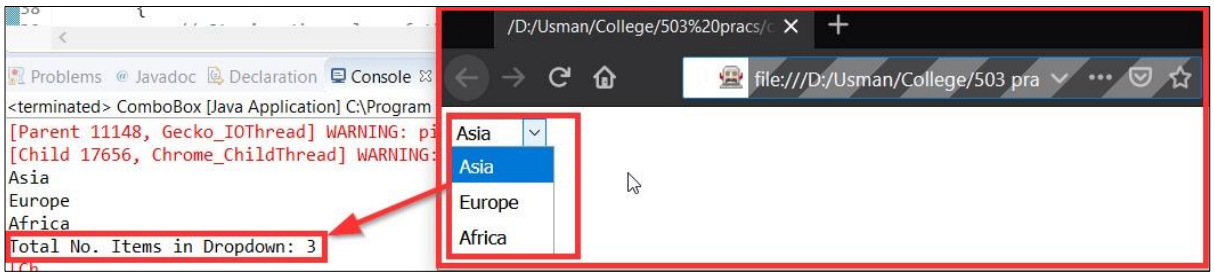


Figure 2: by using the STATIC URL(local file)

Practical No: 08

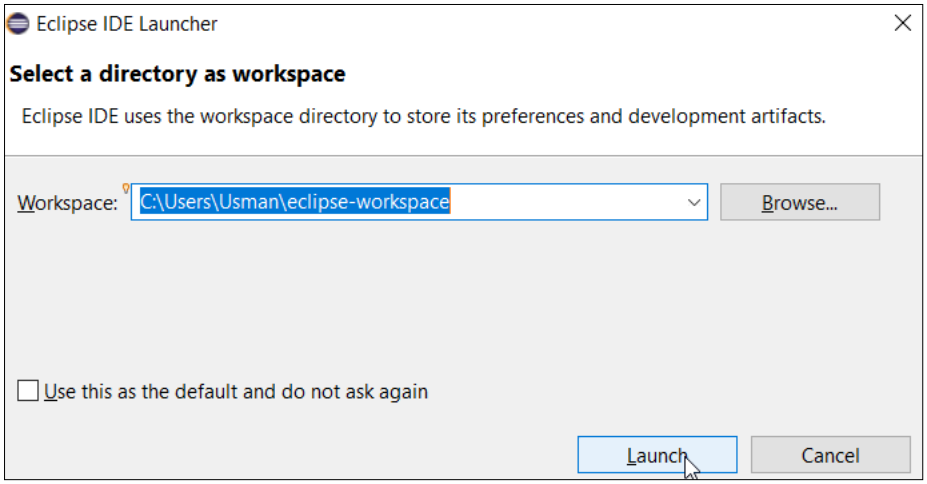
AIM : Write a program using Selenium WebDriver to automate the login process on a specific web page. Verify successful login with appropriate assertions.:

PRE-REQUISITES:

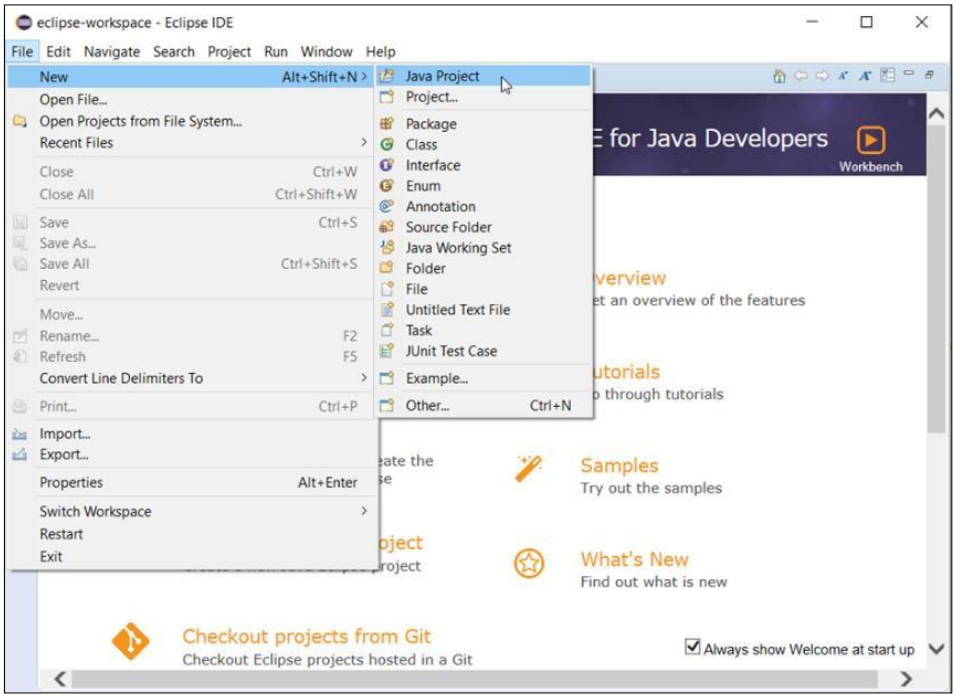
1) *Check that you have **JDK**.*
2) *Check that you have **Eclipse IDE**.*
3) *Check that you have **Selenium** Server Driver **and** Client Driver**(JAR files)**.*
4) *Check that you have **Gecko Driver**.*
5) *Check that you have a stable Internet connection.*

STEPS:

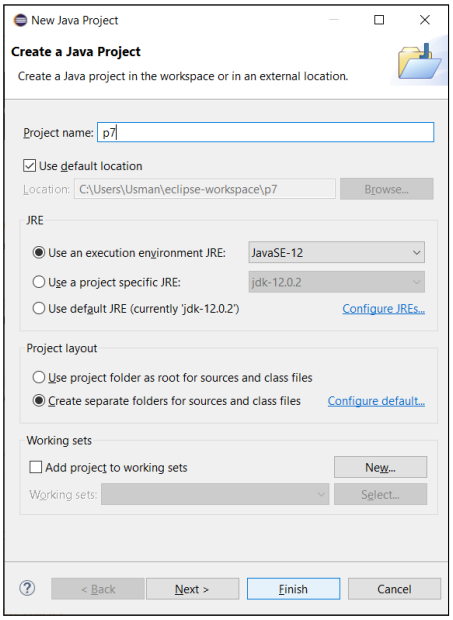
1) ***Open Eclipse**. Select your workspace directory. Click **Launch:***

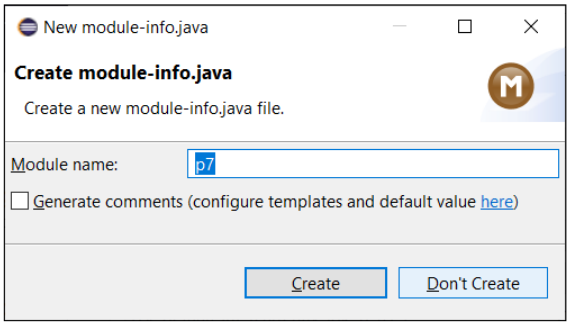


2) ***Create** a Project(**File > New > Java Project**):*



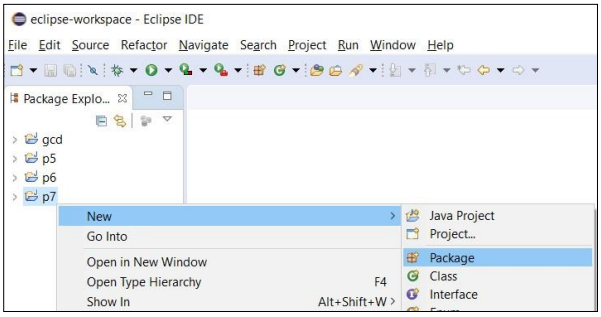
3) ***Name the project*** *as "p7"* ***>*** *click* ***Finish >*** *click* ***Don't Create*** *module:*

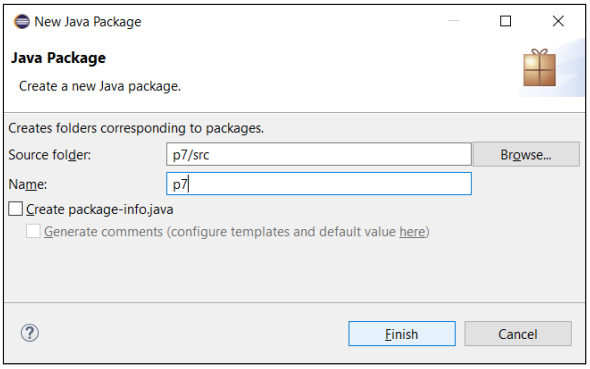




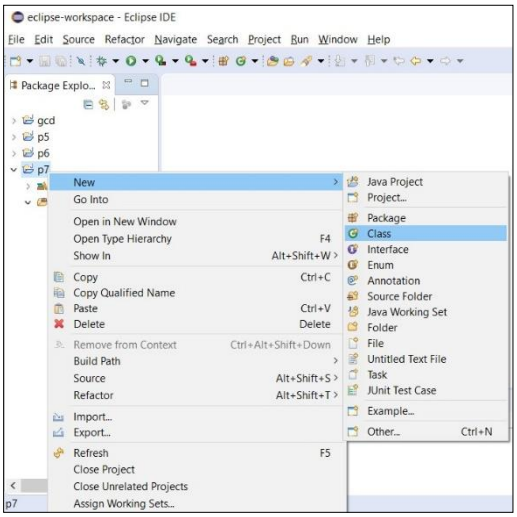
4) *Close the "Welcome" tab.*

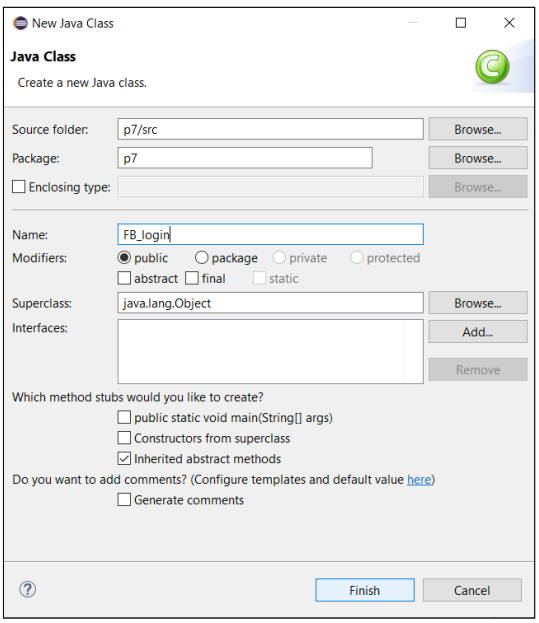
5) ***Create*** *a Package(****right-click on Project Name > New > Package > Name*** *it* ***> Finish****):*





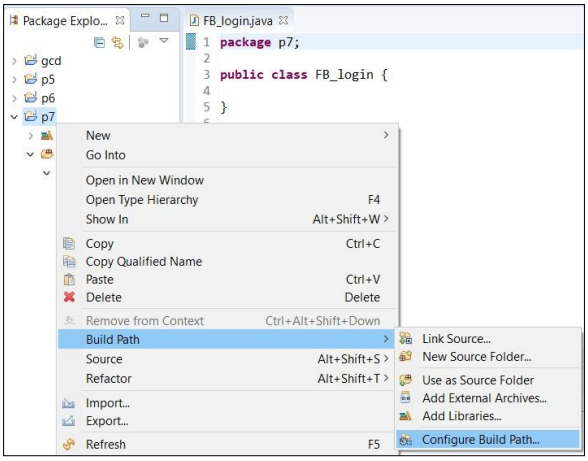
6) *Create* ***a Class****(right-click on Project Name > New > Class > Name* ***it*** *> Finish****):***



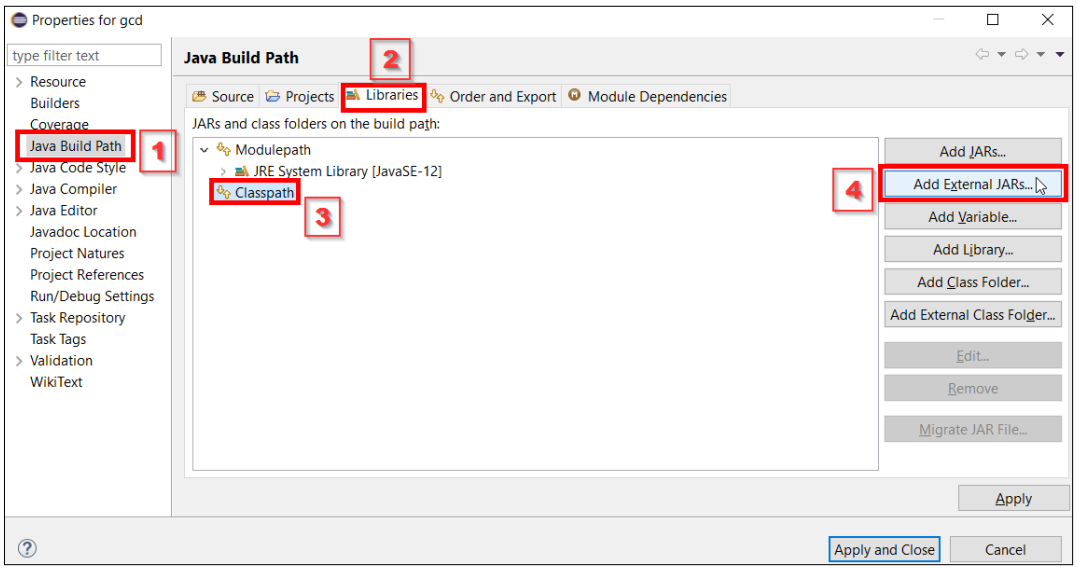


7) *Adding* ***"Selenium* Server Driver *and* Client Driver*(JAR files)"*** *in Eclipse IDE:*

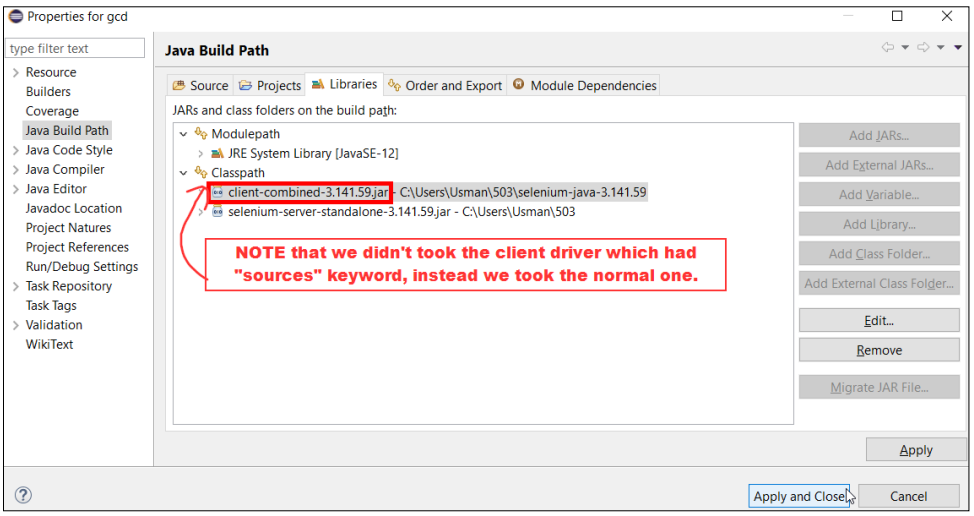
- *right-click on Project Name > Build Path > Configure Build Path…*



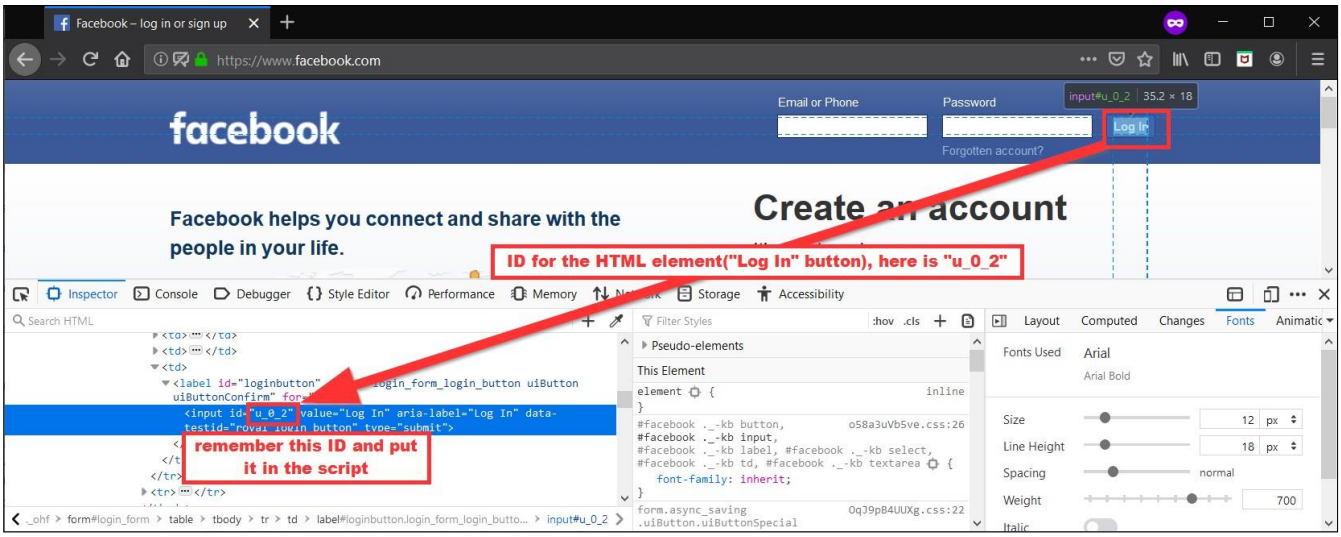
- *now go under:* ***Java Build Path > Libraries > Classpath >*** *click* ***Add External JARs…***



- *Browse and* ***add JAR files >*** *click* ***Apply and Close :***



8) *Fetching ID's of the HTML attributes of the elements from the facebook webpage.*
For instance we looking for the ID behind "Log In" button HTML element
(right-click over element**(button/textbox) **>** click **Inspect Element):
(do the same for EMAIL TEXTBOX & PASSWORD TEXTBOX)



9) *Creating the* **script** in JAVA*:*
*(NOTE that this **script** will be run by Eclipse IDE)*
(In simple words, it's like we are
-ordering Eclipse to run a script or to do a job
-of opening the browser, visiting the URL
-and putting credentials in textboxes and clicking button with the help of Selenium Drivers
-and to show the result.
-Hence automating the work in browser for logging in)

- *Now we'll put the path of "geckodriver" in a String* **driverPath**
- *And in here, we'll **put the ID's in the** .id() **method respectively**.*

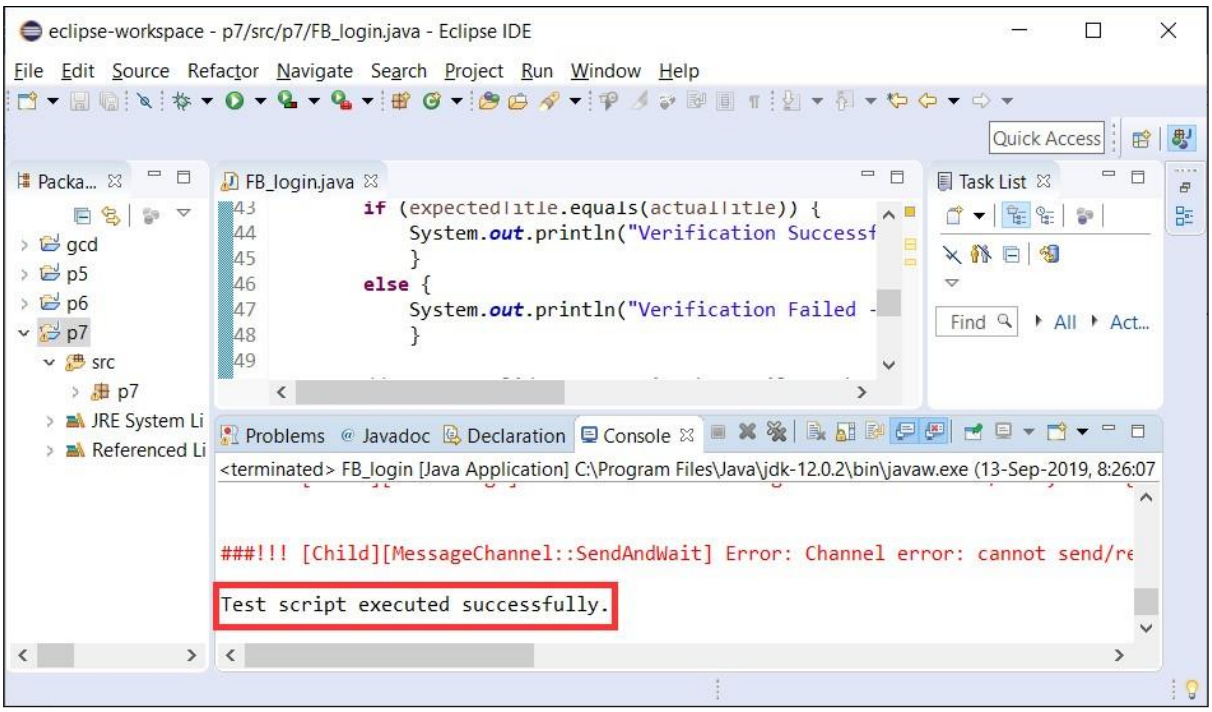
---(FB_login.java)---

```
package p7;

import   org.openqa.selenium.By; import
org.openqa.selenium.Keys; import
org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.*;
import  org.openqa.selenium.firefox.FirefoxOptions;
import  org.openqa.selenium.firefox.FirefoxProfile;
import org.openqa.selenium.firefox.internal.ProfilesIni;

public class FB_login {
        static String driverPath = "C:\\Users\\Usman\\503\\geckodriver.exe";

     public static void main(String[] args) { System.setProperty("webdriver.gecko.driver", driverPath);
                //DesiredCapabilities capabilities = DesiredCapabilities.firefox();
                //capabilities.setCapability("marionette",true);
                ProfilesIni allProfiles = new ProfilesIni();
                FirefoxProfile fp = new FirefoxProfile();
```

```
            fp.setPreference(FirefoxProfile.PORT_PREFERENCE,"7055");
            FirefoxOptions options = new FirefoxOptions();
            options.setProfile(fp);

            //objects and variables instantiation
        WebDriver driver = new FirefoxDriver(options);
        String appUrl = "https://www.facebook.com/";

        //launch the firefox browser and open the application url
        driver.get(appUrl);

        //maximize the browser window
        driver.manage().window().maximize();

        //declare and initialize the variable to store the expected title of the
webpage.
        String expectedTitle = "Facebook – log in or sign up";

        //fetch the title of the web page and save it into a string variable
        String actualTitle = driver.getTitle();

        //compare the expected title of the page with the actual title of the page
and print the result
        if (expectedTitle.equals(actualTitle)) {
            System.out.println("Verification Successful - The correct title is
displayed on the web page.");
        }
          else {
            System.out.println("Verification Failed - An incorrect title is
displayed on the web page.");
        }

        //enter a valid username in the email textbox
        WebElement username = driver.findElement(By.id("email"));
        username.clear();
        username.sendKeys("your email id");

        //enter a valid password in the password textbox
        WebElement password = driver.findElement(By.id("pass"));
        password.clear();
        password.sendKeys("your password");

        password.sendKeys(Keys.ENTER);

        //click on the Sign in button
        WebElement LogInButton = driver.findElement(By.id("u_0_2"));
        LogInButton.click();

        //close the web browser
        driver.close();
        System.out.println("Test script executed successfully.");

        //terminate the program
        System.exit(0);
    }
}
```

10) Run the file from Eclipse IDE:

- ***OUTPUT:***



11) Finish!