

# Software Testing and Quality Assurance

# INDEX

Sr. No.	Aim	Sign
1	Install Selenium IDE and create a test suite containing a minimum of 4 test cases for different web page formats	
2	Conduct a test suite for two different websites using Selenium IDE. Perform various actions like clicking links, filling forms, and verifying content.	
3	Install Selenium Server (Selenium RC) and demonstrate its usage by executing a script in Java to automate browser actions.	
4	Write a program using Selenium WebDriver to automate the login process on a specific web page. Verify successful login with appropriate assertions	
5	Write a program using Selenium WebDriver to update 10 student records in an Excel file. Perform data manipulation and verification.	
6	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.	
7	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.	
8	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	
9	Write a program using Selenium WebDriver to count the number of checkboxes on a web page, including checked and unchecked counts. Perform checkbox identification and counting.	
10	Perform load testing on a web application using JMeter. Generate and analyze load scenarios. Additionally, explore bug tracking using Bugzilla as a tool for logging and tracking software defects	

# Partical 1

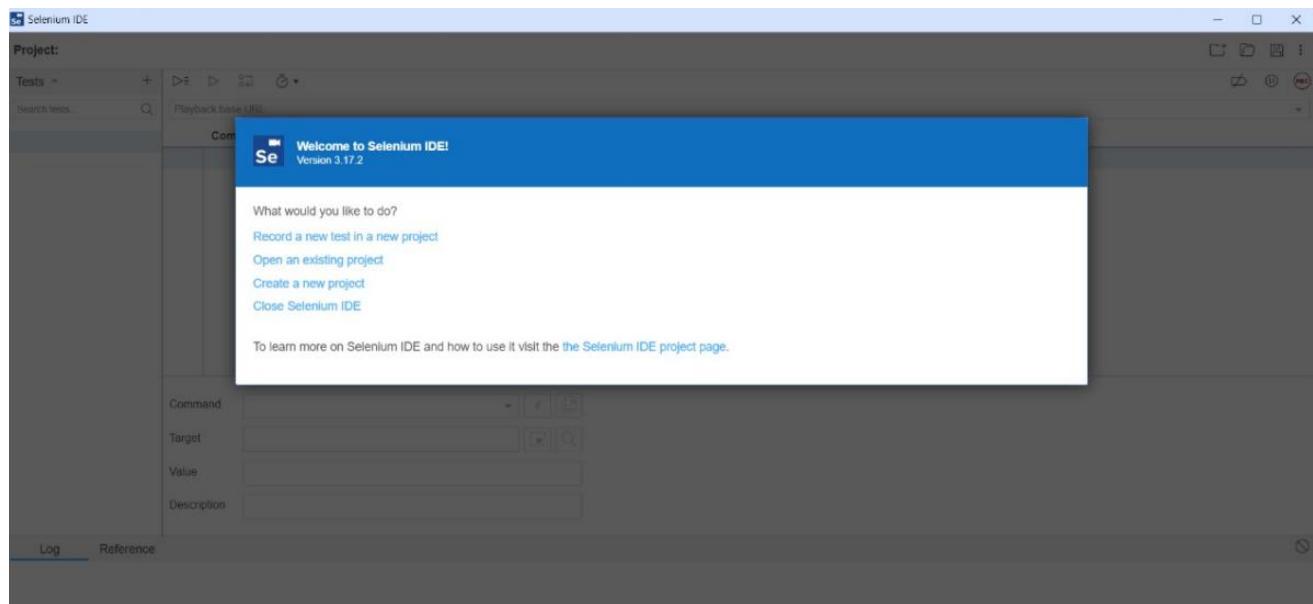
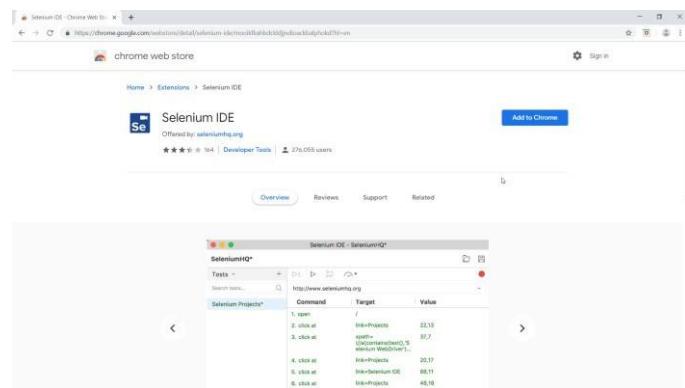
#AIM: Install Selenium IDE. Write a test suite containing minimum 4 test cases for different formats.

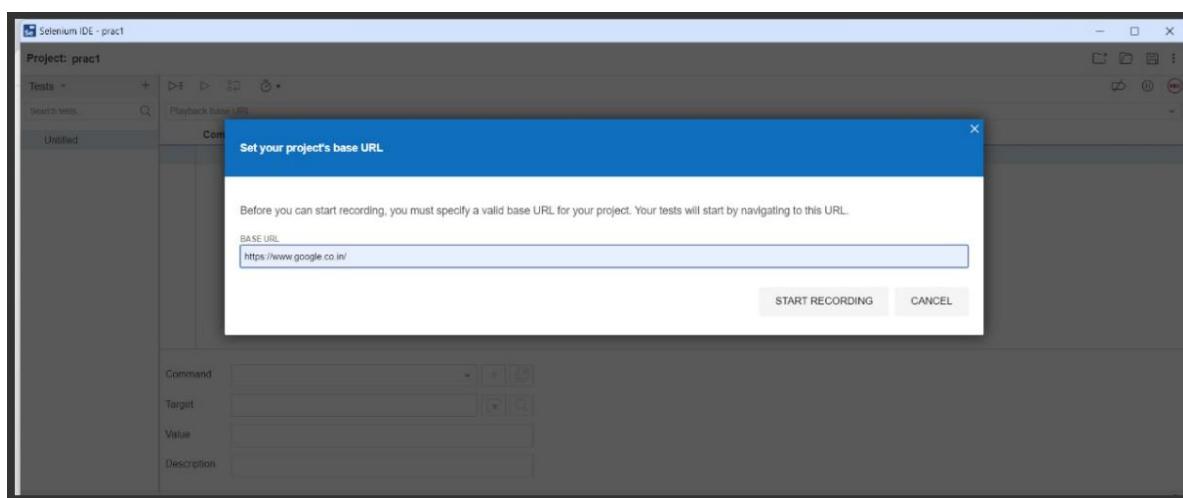
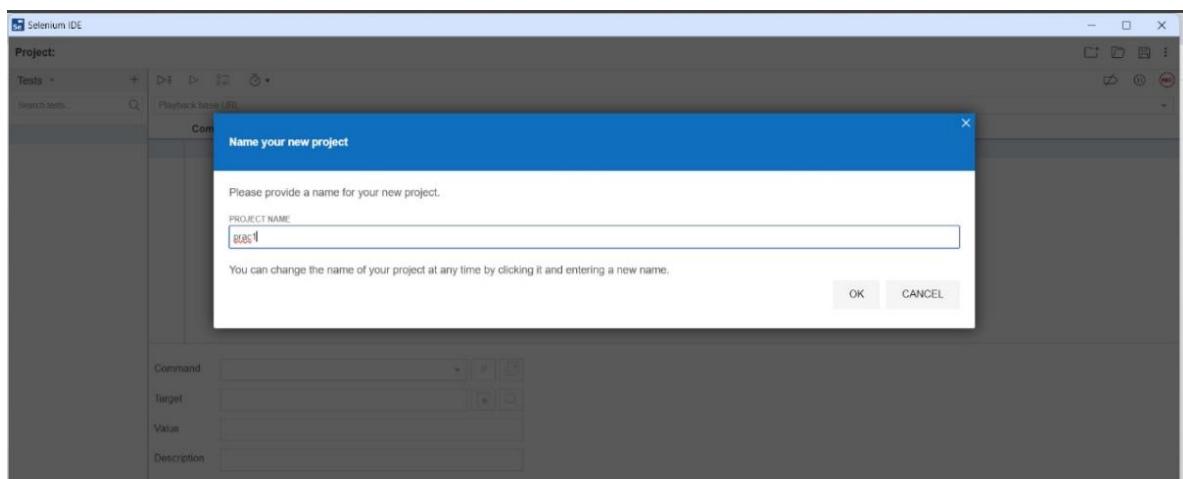
## REQUIREMENTS:

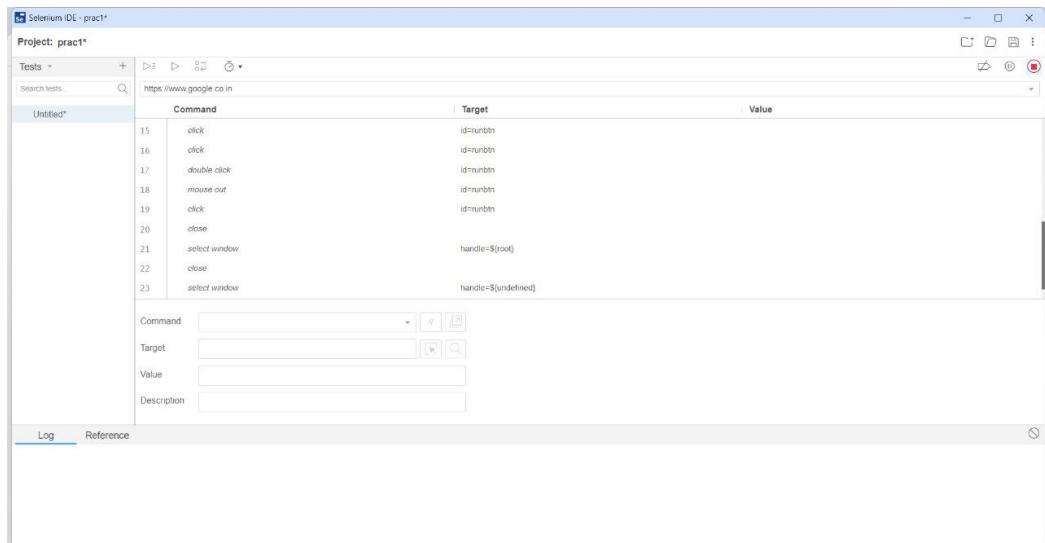
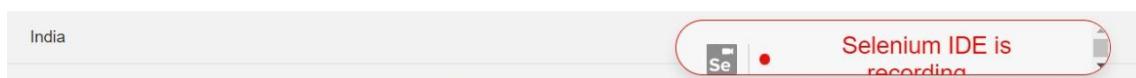
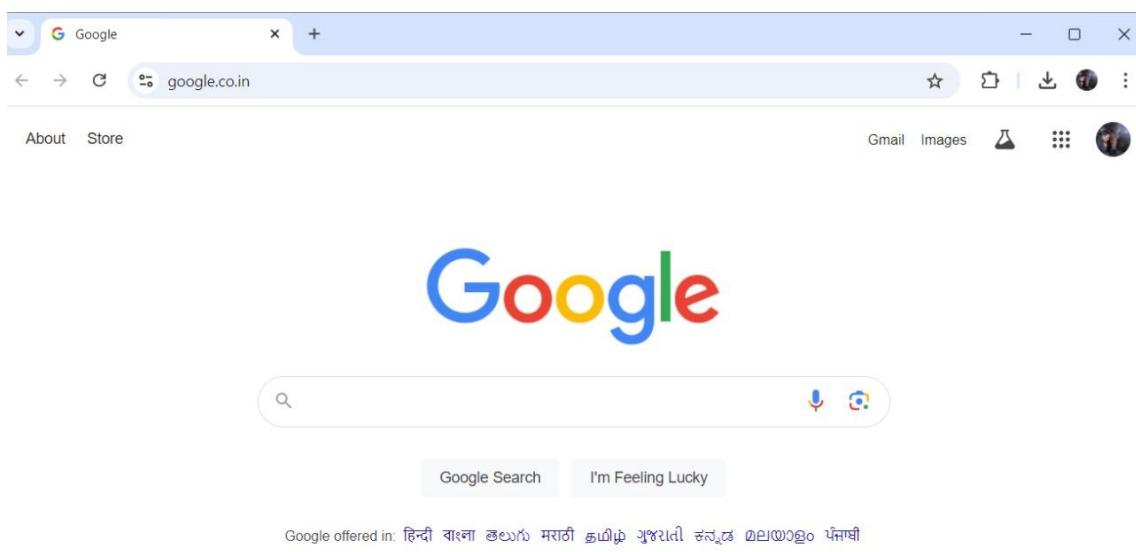
- 1) Selenium IDE extension on a browser.
- 2) NetBeans IDE for creating test case files.

## STEPS:

- 1) Just google “Selenium IDE chrome” and click on the first link. Add the “Selenium IDE” extension to your browser.







## Partical 2

#AIM: Conduct a test case suite for any two websites.

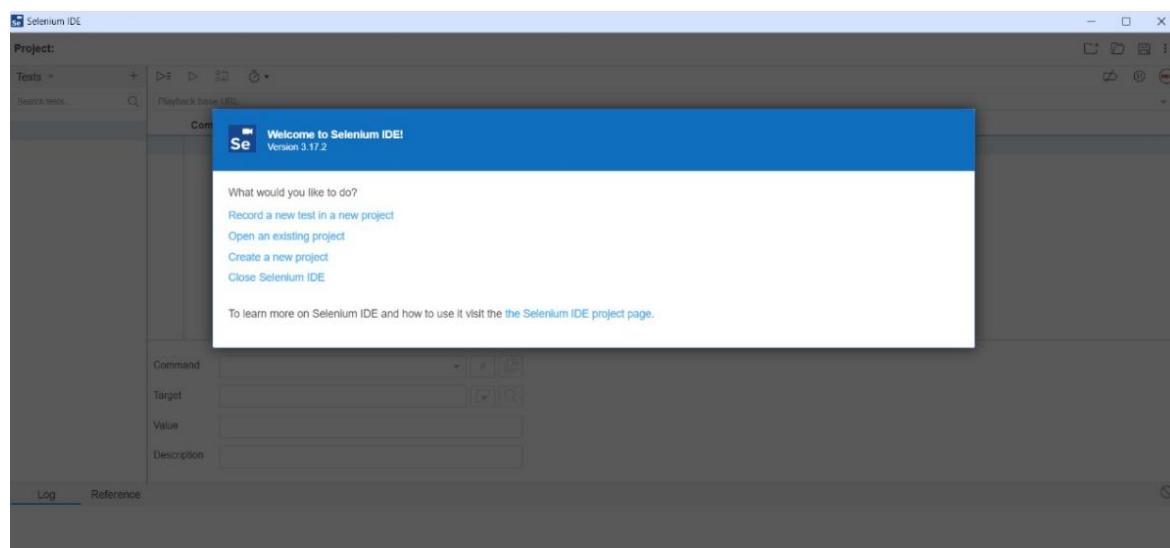
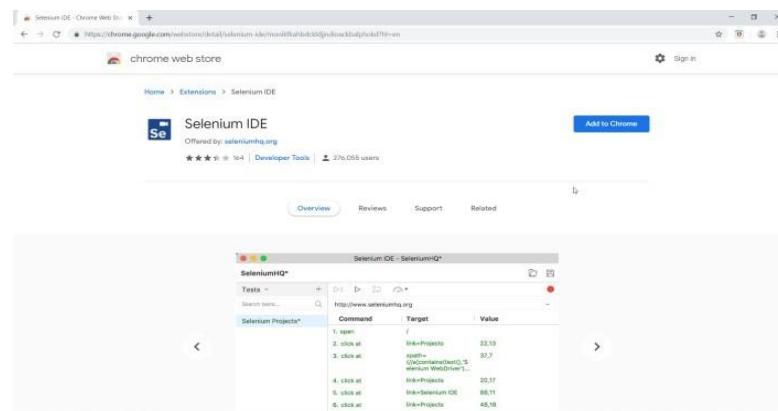
### REQUIREMENTS:

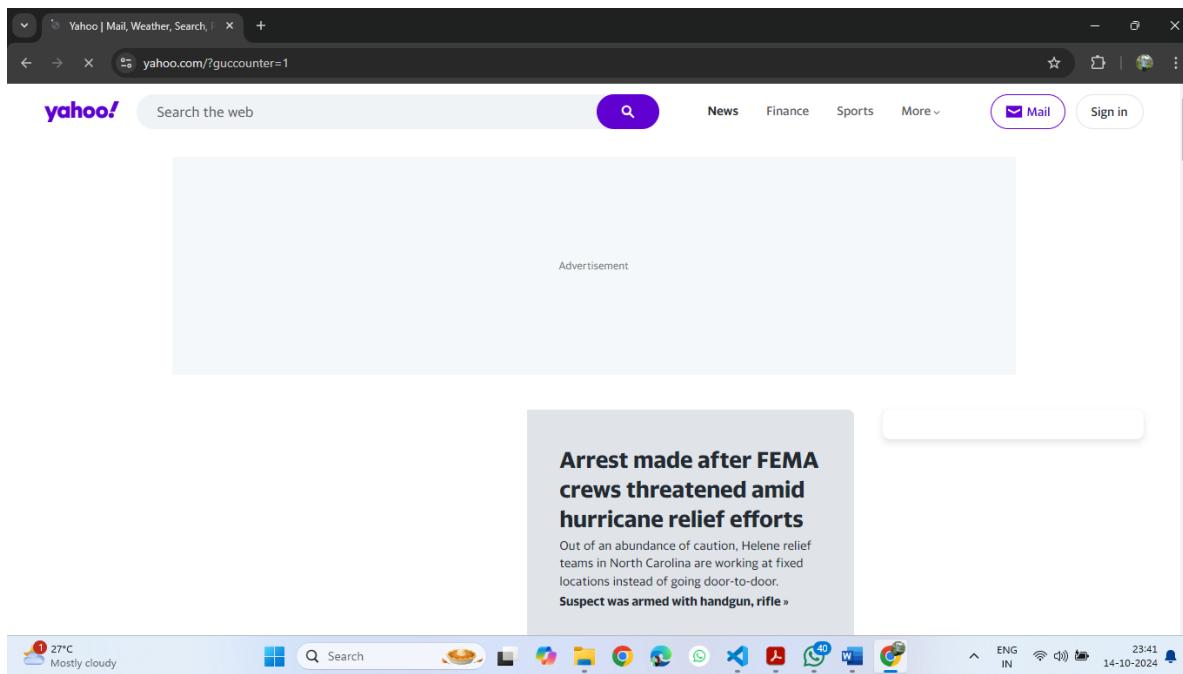
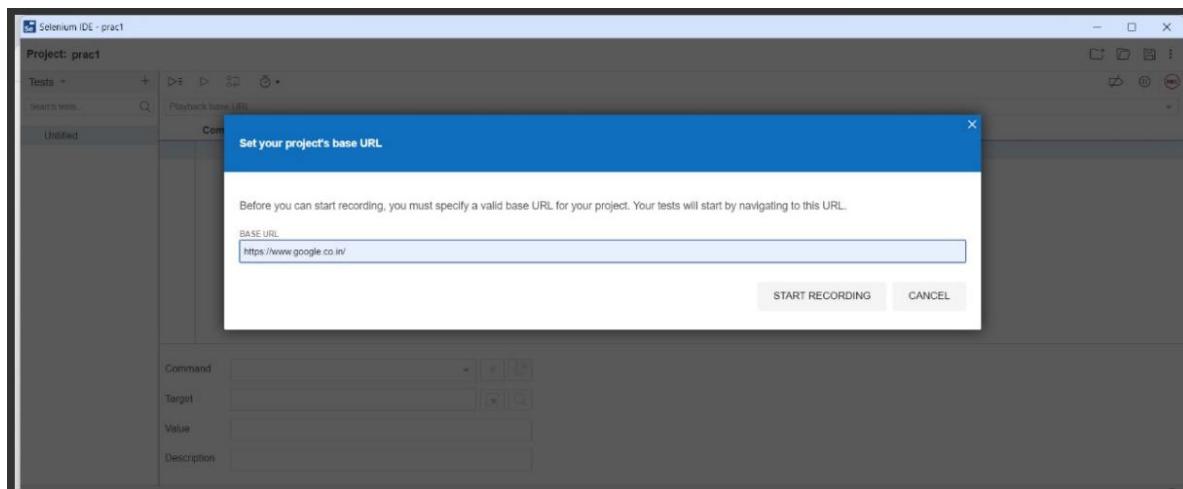
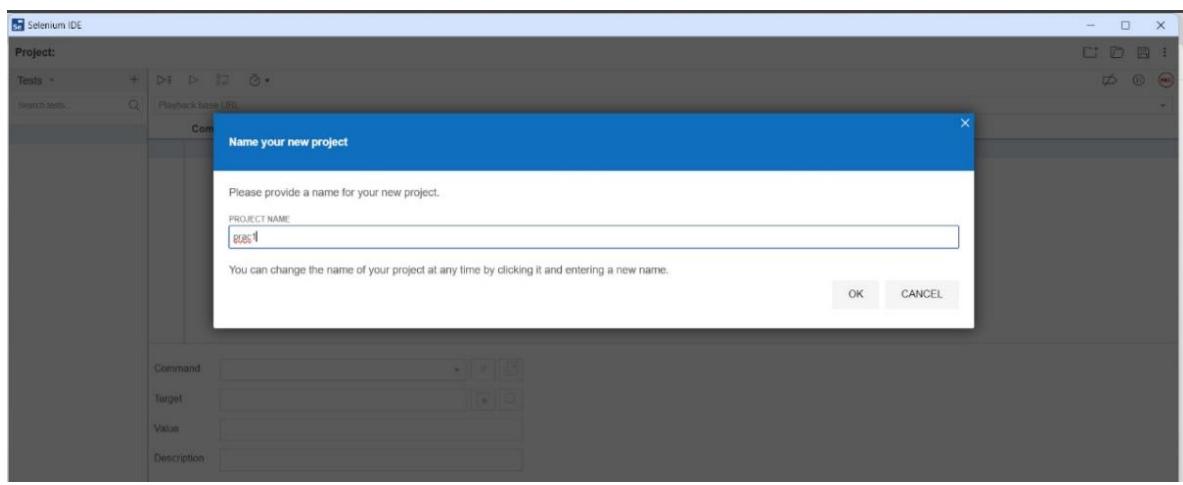
1) Selenium IDE extension on a browser. 2)

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.





The screenshot shows two windows of the Selenium IDE. The top window is titled 'Selenium IDE - pract1\*' and displays a test script for the URL 'https://www.google.co.in'. The script contains the following commands:

Line	Command	Target	Value
15	click	id=runbtn	
16	click	id=runbtn	
17	double click	id=runbtn	
18	mouse out	id=runbtn	
19	click	id=runbtn	
20	close		
21	select window	handle=\${root}	
22	close		
23	select window	handle=\${undefined}	

The bottom window is also titled 'Selenium IDE - pract1\*' and shows the same test script. A modal dialog box is overlaid on the interface, titled 'Name your new test'. It contains the instruction 'Please provide a name for your new test.' and a text input field labeled 'TEST NAME' with a placeholder 'Untitled'. Below the input field, it says 'You can change it at any time by clicking the : icon next to its name in the tests panel.' There are 'OK' and 'LATER' buttons at the bottom right of the dialog.

## Partical 3

**#AIM: Install Selenium Server(Selenium RC) and demonstrate it using a script in Java.**

**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*”:

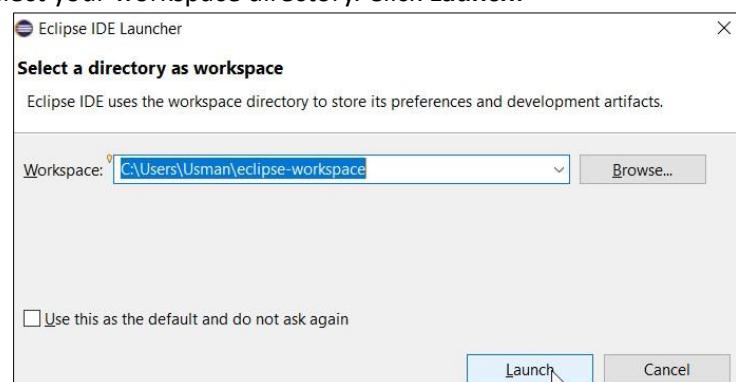
- Visit <https://www.seleniumhq.org/download/>
- Under section “**Selenium Standalone Server**”, click download version “**3.141.59**”
- You’ll get the executable jar file(selenium-server-standalone-3.141.59)

**b)** For “Selenium *Client Driver*”:

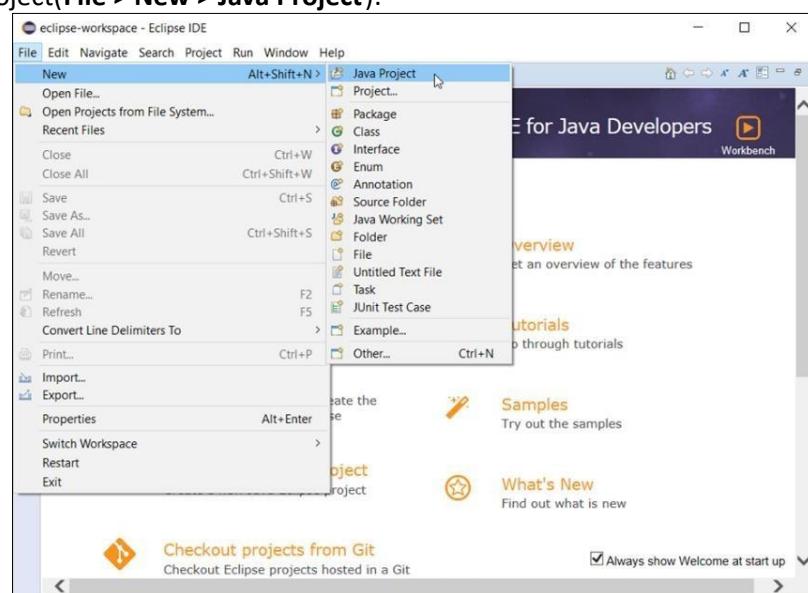
- Visit <https://www.seleniumhq.org/download/>
- Under section “**Selenium Client & WebDriver Language Bindings**”, download the “**3.141.59**” version of Java.
- Extract the file and you’ll see two jar files. From them, we’ll be using this executable jar file(client-combined-3.141.59) **4)** To Download “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:**



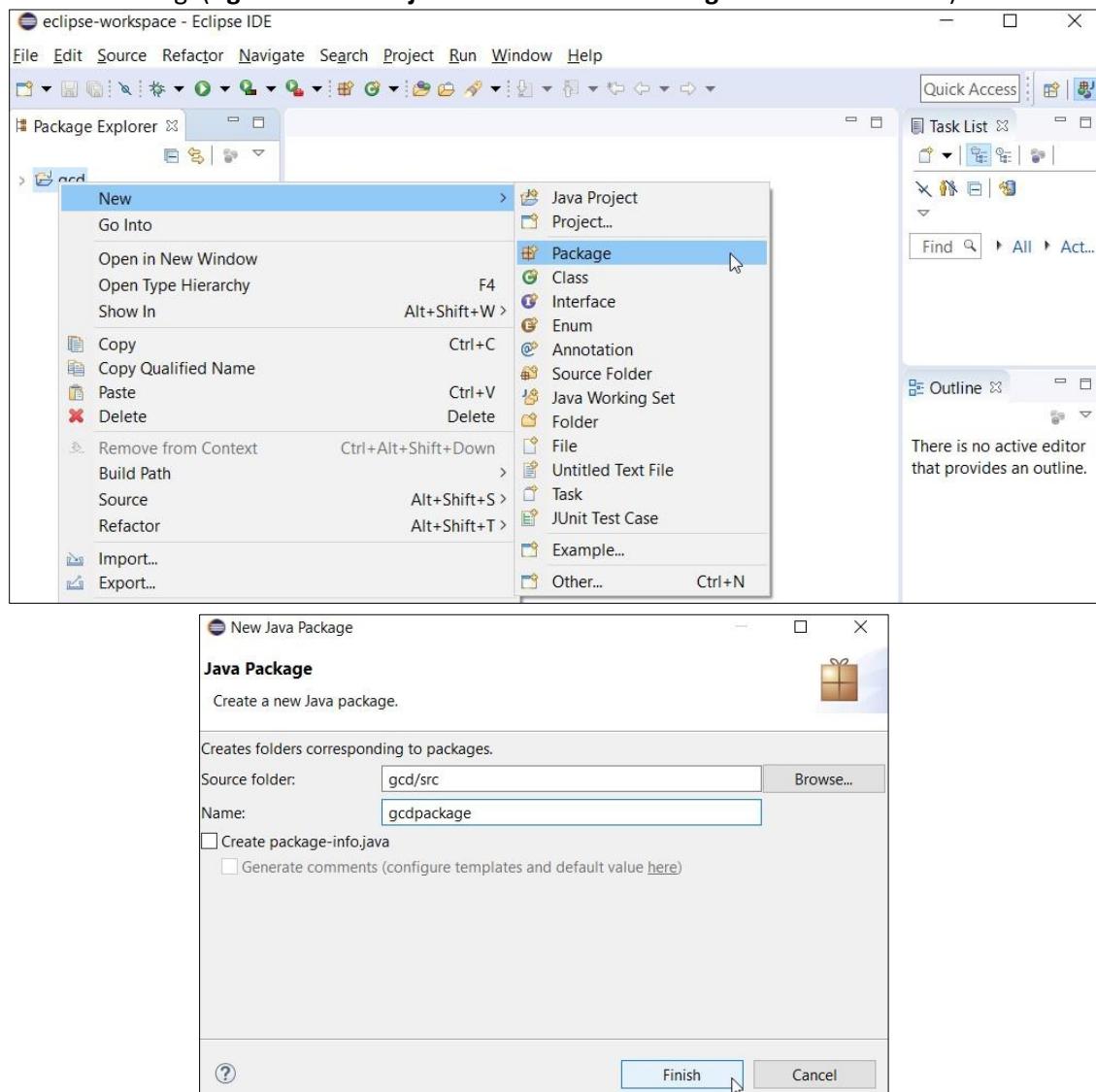
**2) Create a Project(File > New > Java Project):**



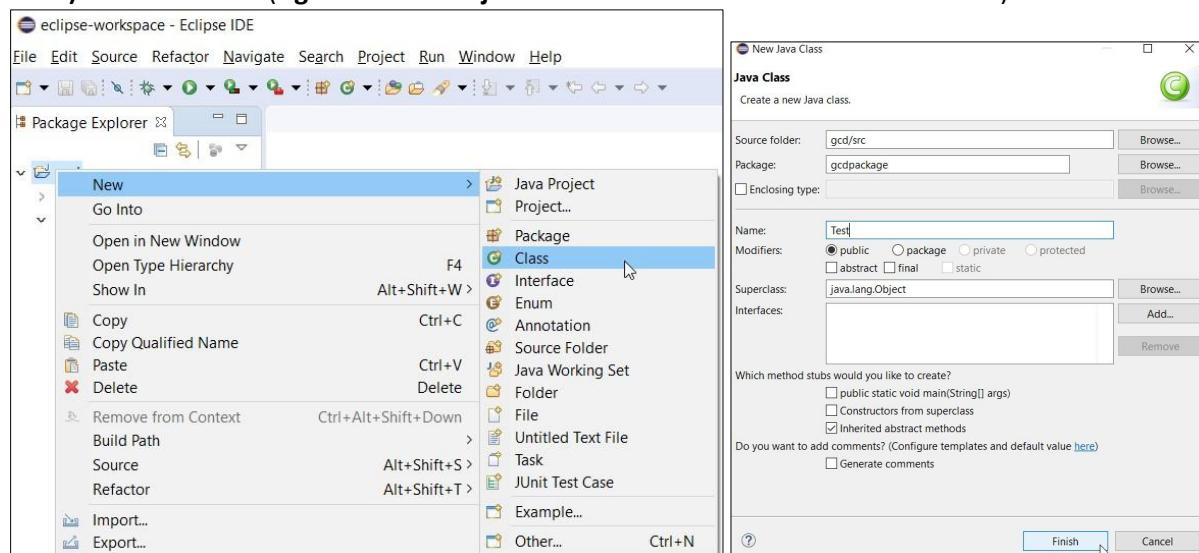
**3) Name the project as "gcd" > click Finish > click Don't Create module:**

4) Close the “Welcome” tab.

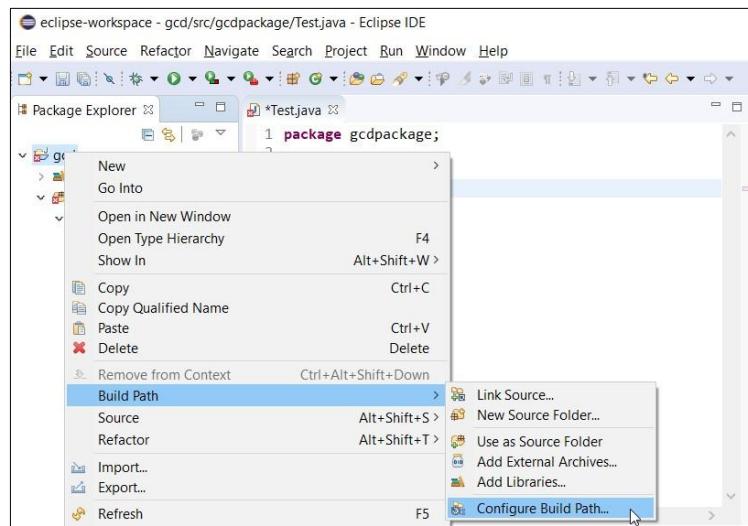
5) Create a Package(right-click on Project Name > New > Package > Name it > Finish):



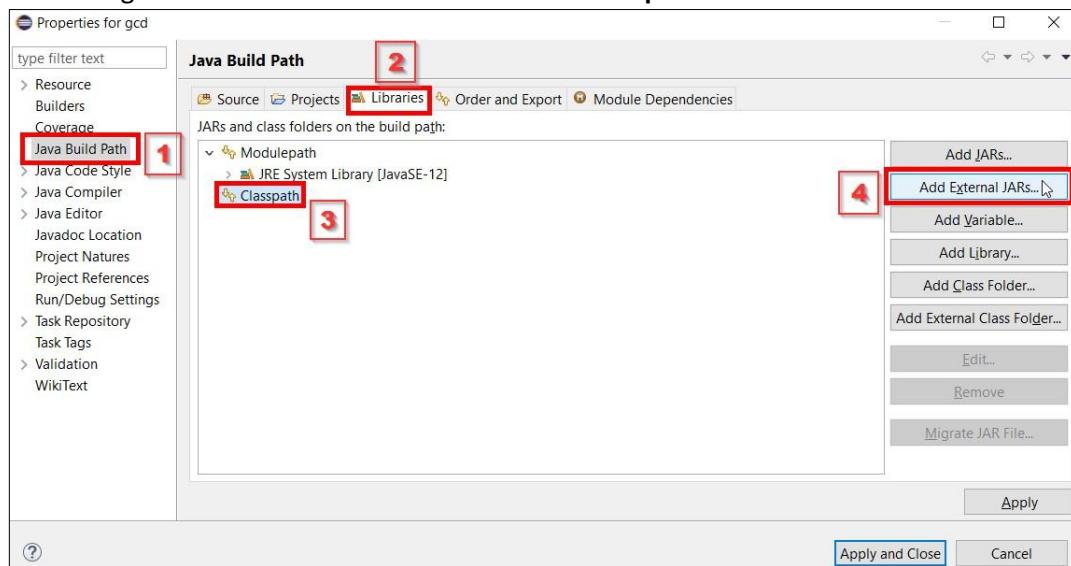
6) Create a Class(right-click on Project Name > New > Class > Name it > Finish):



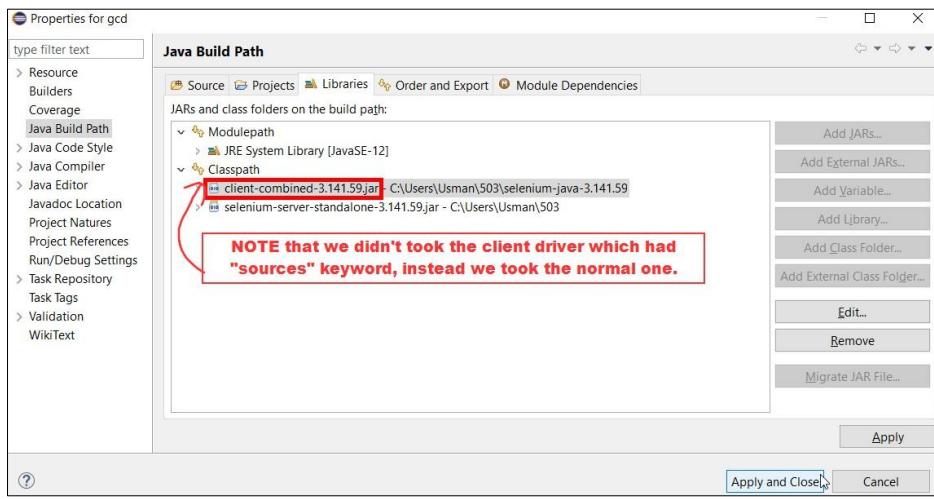
- 7) Adding “**Selenium Server Driver and Client Driver(JAR files)**” in Eclipse IDE: • right-click on Project Name > Build Path > Configure Build Path...



- 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">

```

```

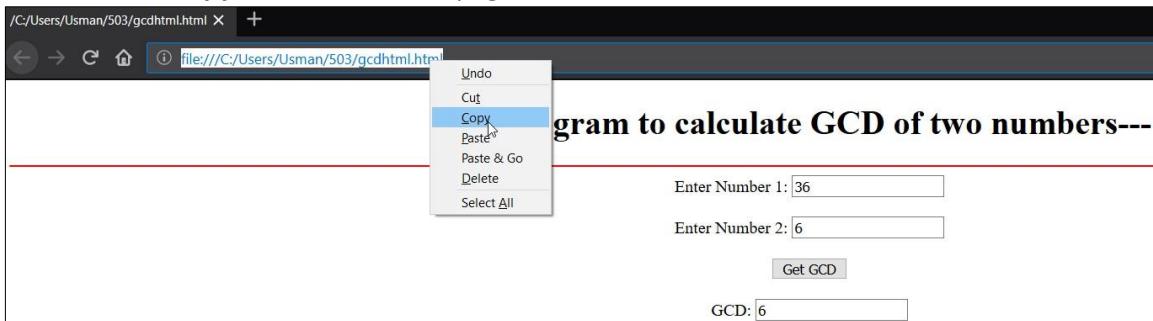
<form name="myform">
    Enter Number 1: <input type="text" name="n1" value=""> <br> <br>
    Enter Number 2: <input type="text" name="n2" value=""> <br> <br>
    <input type="button" name="btn" value="Get GCD" onClick="gcd()"><br><br>
    GCD: <input type="text" name="result" value="">

</form>
</center>
</body>

</html>

```

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



### 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 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();
    }
}

```

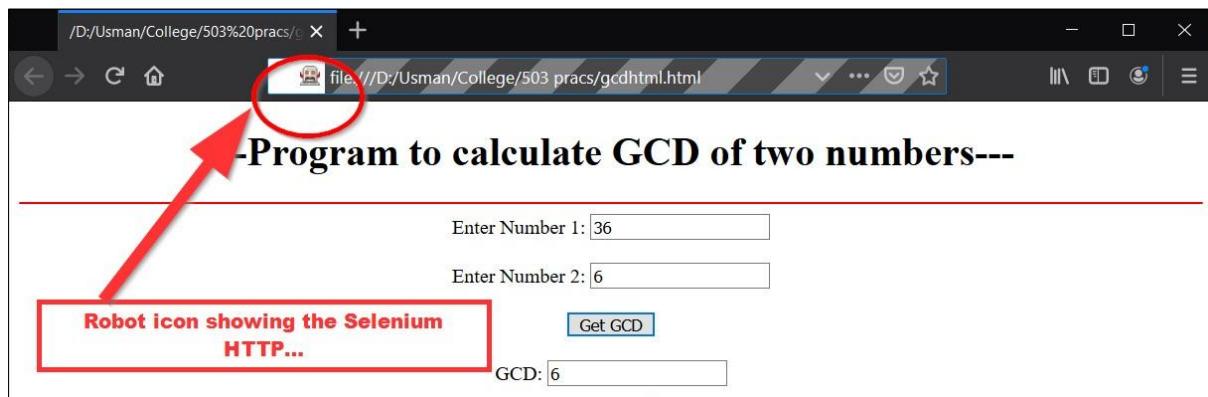
```

        //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:**



**11) 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.

# Partical 4

#AIM: Load Testing using JMeter.

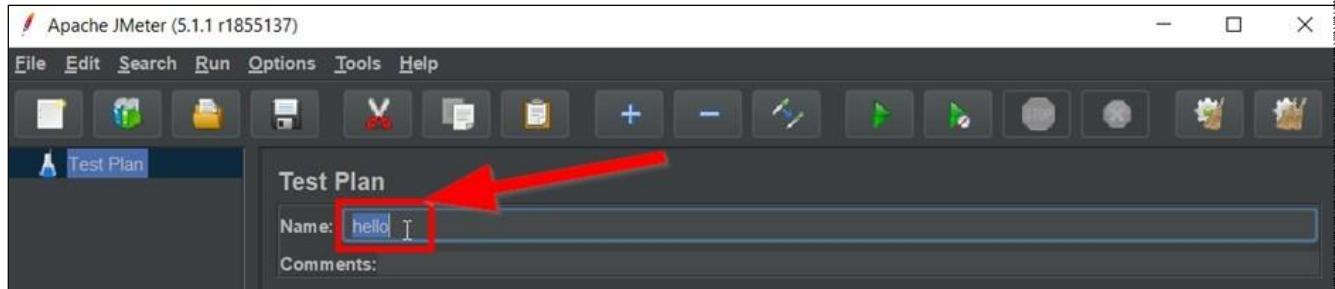
## PRE-REQUISITES:

### 1) To Download “JDK”:

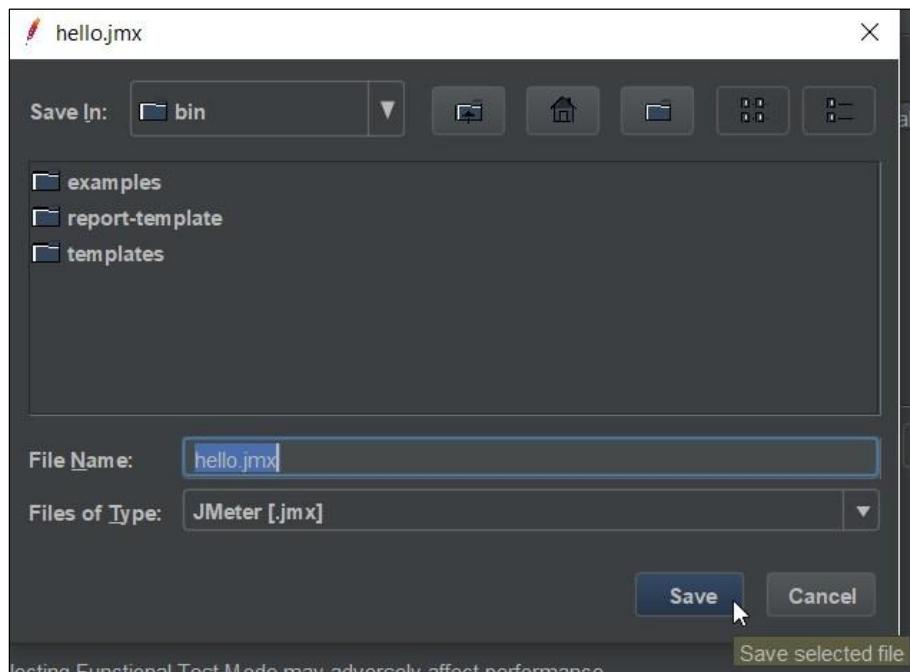
- Visit <https://www.oracle.com/technetwork/java/javase/downloads/jdk12downloads-5295953.html>
- Download this file “jdk-12.0.2\_windows-x64\_bin.exe” and install it.
- 2) To Download “Apache JMeter”:
- Visit [https://jmeter.apache.org/download\\_jmeter.cgi](https://jmeter.apache.org/download_jmeter.cgi)
- Under section “Apache JMeter 5.1.1 (Requires Java 8+)”
- Under sub-section “Binaries”, download “apache-jmeter-5.1.1.zip” file.
- Installation:
  - ❑ Extract the “apache-jmeter-5.1.1.zip” file.
  - ❑ Navigate to: **apache-jmeter-5.1.1 > bin >** the “**ApacheJMeter.jar**” file is your working space.

## STEPS:

### 1) Open “ApacheJMeter.jar” file. Name your **Test Plan** as “hello”:

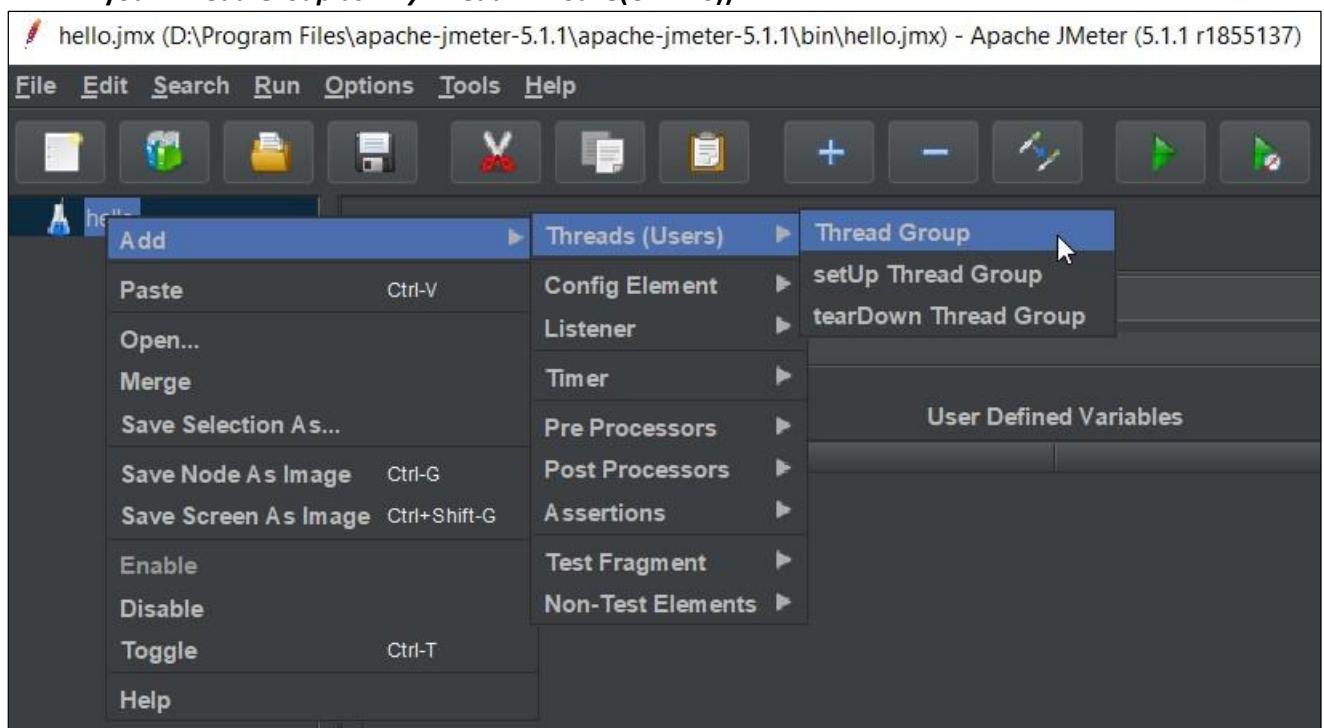


### 2) Save(CTRL+S) your Test Plan. It will be saved as “.jmx”(i.e. Java Management Extensions):



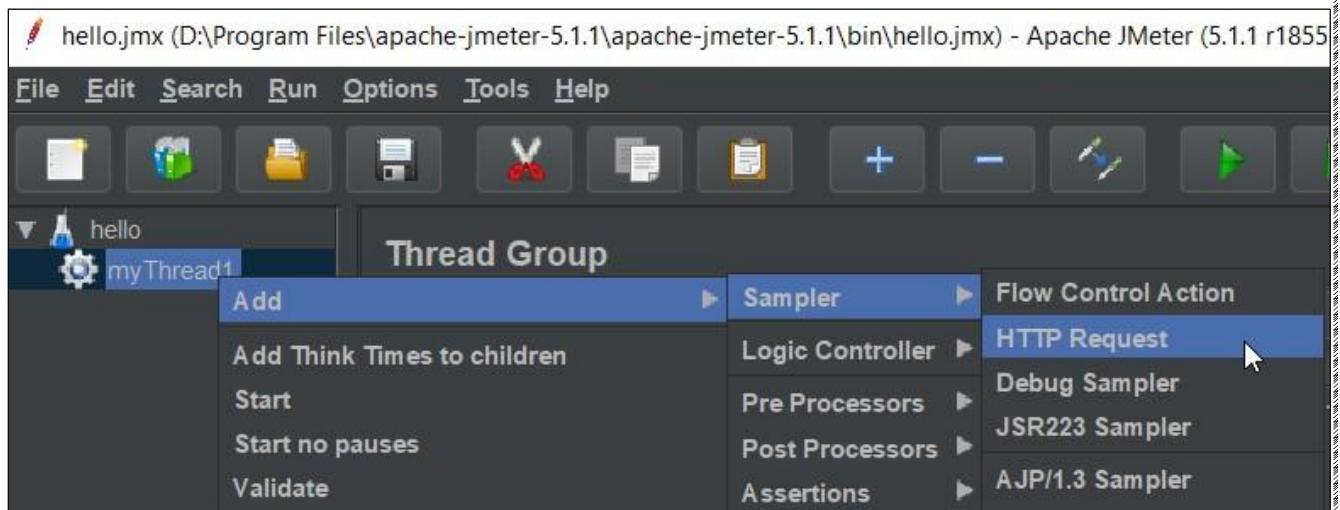
**3) Create a *Thread Group***

(right-click over “hello(*Test Plan*)” > Add > Thread(Users) > Thread Group > Name your *Thread Group* as “*myThread1*” > Save(CTRL+S)):



**4) Under Thread Group(*myThread1*), add a “Sampler”, namely *HTTP Request*:**

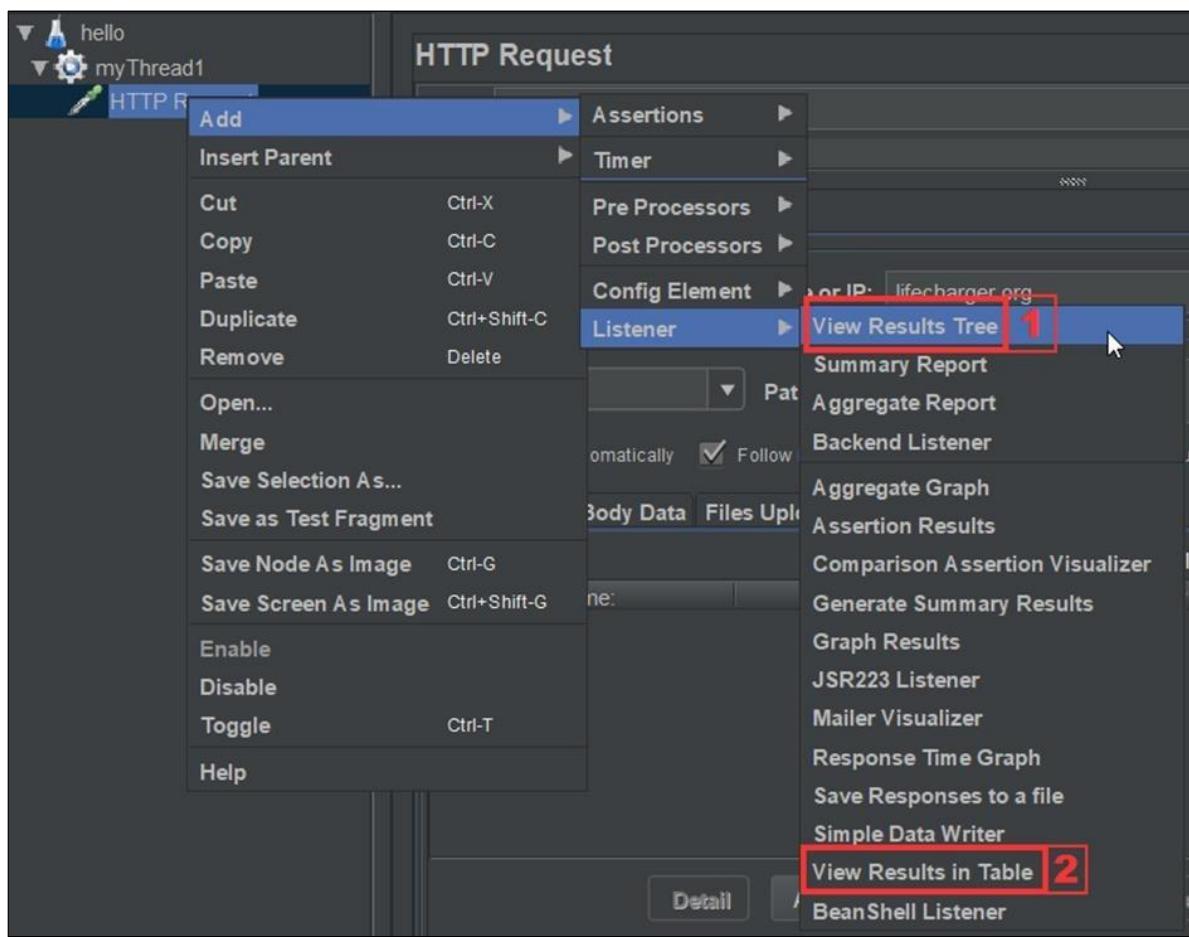
(right-click over “*myThread1(Thread Group)*” > Add > Sampler > HTTP Request):



- 5) Now visit any website page with a next path. Then set **website name** as “**Server Name or IP**” & **next path** as “**Path**” in the Basic panel:

The screenshot shows the Apache JMeter interface and a web browser window side-by-side. In the browser, the URL 'lifecharger.org/when-in-doubt-do-this/' is displayed in the address bar. Red arrows point from the 'Server Name or IP:' field and the 'Path:' field in the JMeter 'HTTP Request' sampler configuration to the corresponding fields in the browser's address bar.

- 6) Under HTTP Request Sampler, add 2 “Listeners”, namely ***View Results Tree*** & ***View Results in Table***:  
**(right-click over HTTP Request(Sampler) > Add > Listener > View Results Tree) & (right-click over HTTP Request(Sampler) > Add > Listener > View Results in Table):**



7) Save(CTRL+S) & Run(CTRL+R) the file and wait for a while. You'll see some:

- **OUTPUTS:**

The screenshot shows the JMeter interface with the 'View Results Tree' listener selected. The 'Text' tab is active, displaying a successful HTTP request sample. A red arrow points from the 'HTTP Request' button in the tabs to the sample details. A cyan box highlights the sample details, and a cyan 'OUTPUT' callout points to it. The sample details include:

- Thread Name: myThread1 1-1
- Sample Start: 2019-08-03 00:08:06 IST
- Load time: 13052
- Connect Time: 4897
- Latency: 8178
- Size in bytes: 42516
- Sent bytes: 275
- Headers size in bytes: 1907
- Body size in bytes: 40609
- Sample Count: 1
- Error Count: 0
- Data type ("text"|"bin"|""): text
- Response code: 200
- Response message: OK

HTTPSAMPLEResult fields:  
ContentType: text/html; charset=UTF-8  
DataEncoding: UTF-8

The screenshot shows the JMeter interface with the 'View Results in Table' listener selected. A cyan 'OUTPUT' callout points to the table. The table has the following columns and data:

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	00:08:06.323	myThread1 1-1	HTTP Request	13052	<span style="color: green;">✓</span>	42516	275	8178	4897

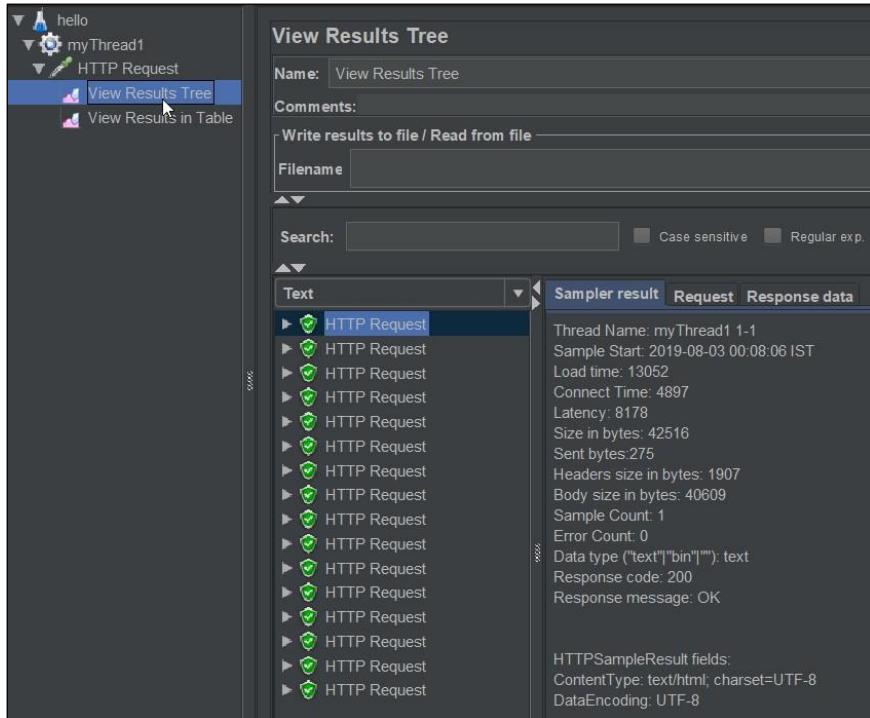
**8) Now change some Thread Properties(users=10; ramp-up period=20; loop count=Forever):**

The screenshot shows the JMeter interface with the 'Thread Group' configuration open. A red box highlights the 'Thread Properties' section, which contains the following settings:

- Number of Threads (users): 10
- Ramp-Up Period (in seconds): 20
- Loop Count:  Forever

**9) Save(CTRL+S) & Run(CTRL+R) the file and wait for a while. You'll see some infinite:**

- **OUTPUTS:**



View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename

Browse... Log/Display Only: Errors Successes Configure

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(m)
16	00:27:10.416	my Thread1 1-1	HTTP Request	16990	✓	42516	275	6990	252
17	00:27:12.420	my Thread1 1-5	HTTP Request	17123	✓	42495	275	7993	269
18	00:27:14.532	my Thread1 1-6	HTTP Request	16339	✓	42516	275	7890	267
19	00:27:15.439	my Thread1 1-2	HTTP Request	15516	✓	42516	275	7993	259
20	00:27:18.433	my Thread1 1-3	HTTP Request	16993	✓	42495	275	7965	258
21	00:27:18.910	my Thread1 1-7	HTTP Request	17281	✓	42516	275	8210	278
22	00:27:22.423	my Thread1 1-8	HTTP Request	15061	✓	42516	275	6071	259
23	00:27:25.417	my Thread1 1-9	HTTP Request	16071	✓	42524	275	6014	259
24	00:27:26.538	my Thread1 1-10	HTTP Request	15378	✓	42517	275	5933	262
25	00:27:25.504	my Thread1 1-4	HTTP Request	16518	✓	42516	275	5990	257
26	00:27:27.407	my Thread1 1-1	HTTP Request	15503	✓	42516	275	7025	260
27	00:27:29.543	my Thread1 1-5	HTTP Request	15941	✓	42517	275	7864	260
28	00:27:30.878	my Thread1 1-6	HTTP Request	17631	✓	42523	275	8538	270
29	00:27:30.953	my Thread1 1-2	HTTP Request	18806	✓	42516	275	8482	257
30	00:27:35.426	my Thread1 1-3	HTTP Request	17313	✓	42523	275	6191	251
31	00:27:36.191	my Thread1 1-7	HTTP Request	18255	✓	42523	275	7242	255
32	00:27:37.493	my Thread1 1-8	HTTP Request	18219	✓	42516	275	6967	259
33	00:27:41.916	my Thread1 1-10	HTTP Request	16316	✓	42516	275	9552	267
34	00:27:42.023	my Thread1 1-4	HTTP Request	16587	✓	42495	275	10409	262
35	00:27:41.489	my Thread1 1-9	HTTP Request	17198	✓	42517	275	8992	263
36	00:27:42.911	my Thread1 1-1	HTTP Request	17076	✓	42516	275	9585	259
37	00:27:48.510	my Thread1 1-6	HTTP Request	12950	✓	42516	275	6909	258
38	00:27:45.485	my Thread1 1-5	HTTP Request	16055	✓	42516	275	8979	264
39	00:27:49.760	my Thread1 1-2	HTTP Request	13732	✓	42516	275	6637	257

## 10) Finish!

### Apache JMeter is:

- designed to load test functional behavior and measure performance.
- used to test performance both on static and dynamic resources, web dynamic applications.

**Some features include:** ability to test many apps/server/protocol types; Test Plan recording, building, debugging; complete report; portable; pure java; multi-threading; offline analysis of results; highly extensible core.

# Partical 4b

#AIM: To study Bugzilla(demo site).

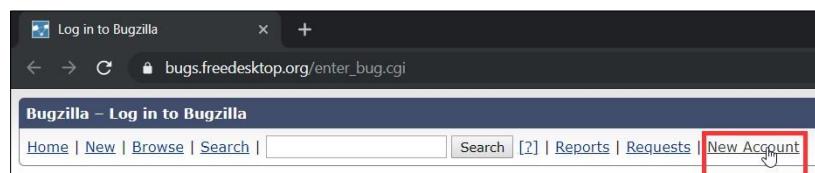
## REQUIREMENTS:

- 1) A stable internet connection.

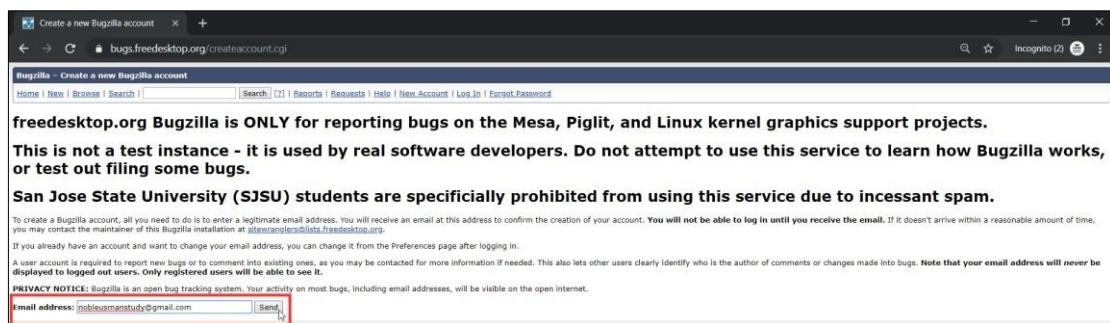
## STEPS:

- 1) Create account and Login:

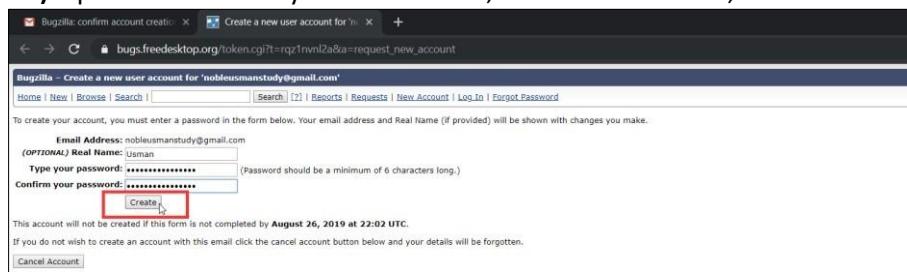
- a) Visit this demo site: [https://bugs.freedesktop.org/enter\\_bug.cgi](https://bugs.freedesktop.org/enter_bug.cgi)
- b) Click on “New Account” in the main menu:



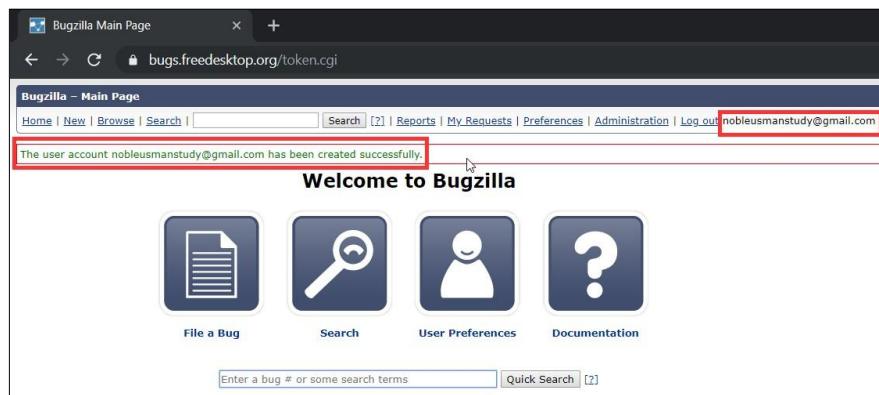
- c) Enter email-id:



- d) Open the link from your email inbox, and then fill details, then click “Create”:



- e) You'll see that you're logged-in:

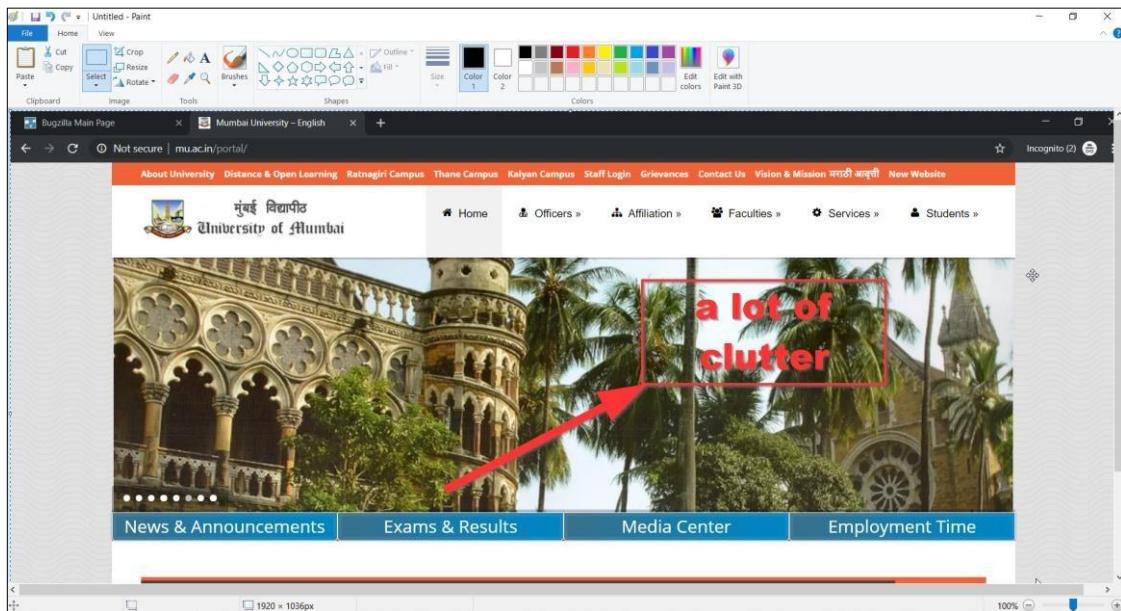


## 2) Creating and submitting a Bug-report:

### a) Detect bug > take its snapshot > save it:

- Visit <http://mu.ac.in/portal/> (or any website you wish to)

Take a snapshot stating the bug:



- Save that snapshot.

### b) Enter a new bug-report:

- Visit the demo site and click on “New” in the main menu:



- Select “DRI(Direct Rendering Infrastructure)” as your product:

The screenshot shows a web browser window with the title 'Enter Bug' and the URL 'bugs.freedesktop.org/enter\_bug.cgi'. The page header includes links for Home, New, Browse, Search, and Reports. A search bar and a 'Search' button are also present. The main content area has a heading 'Bugzilla – Enter Bug' and a note: 'First, you must pick a product on which to enter a bug below:'. Below this, a list of products is shown with the 'DRI' option selected and highlighted with a red box. Other options include Mesa, Spam, and xorg.

**DRI:** Direct Rendering Infrastructure

**Mesa:** Open source 3D drivers and implementation of the OpenGL, OpenGL ES, Vulkan, OpenCL, XvMC, and XvYUV APIs.

**Spam:** Reassign any spam bugs to this product, at which point it will magically disappear.

**xorg:** X.Org X11 Distribution. If you are trying to file a bug and do not see an appropriate component here, please try this one.

- Click on “Show Advanced Fields”:

The screenshot shows a sub-page for the 'DRI' product. The title is 'Enter Bug: DRI' and the URL is 'bugs.freedesktop.org/enter\_bug.cgi?product=DRI'. The page content includes a note about reading bug writing guidelines and looking at frequent errors. A link 'Show Advanced Fields' is highlighted with a red box. Below it, there is a note '(\* = Required Field)' and a field for 'Reporter' with the value 'nobleusmanstudy@gmail.com'.

- Fill the details, add an attachment, click “Submit Bug”:

**Bugzilla – Enter Bug: DRI**

Home | New | Browse | Search |  Search | [?] | Reports | My Requests | Preferences | Administration | Help | Log out nobleusmanstudy@gmail.com

Before reporting a bug, please read the [bug writing guidelines](#), please look at the list of [most frequently reported bugs](#), and please [search](#) for the bug.

[Hide Advanced Fields](#) (\*) = Required Field

* <b>Product:</b> DRI	<b>Reporter:</b> nobleusmanstudy@gmail.com
* <b>Component:</b> DRM/Radeon	Component Description General DRI issues.
DRM/Tegra	
DRM/Via	
General	
hsakmt	
IGT	
libdrm	

\* **Version:** DRI git  
unspecified  
XOrg git

**Severity:** normal  
**Priority:** high  
**Hardware:** x86-64 (AMD64)  
**OS:** Windows (All)

We've made a guess at your operating system and platform. Please check them and make any corrections if necessary.

**Status:** ASSIGNED  
**Assignee:** nobleusmanstudy@gmail.com

**QA Contact:**   
**CC:**

**Default CC:**

**Alias:** clutter  
**URL:** http://mu.ac.in/portal/  
**\* Summary:** too much clutter

**Possible Duplicates:**

Bug ID	Summary	Status	Action
87805	Wrong formatting of DOCX file: too much breaks no underlining	NEW	Add Me to the CC List
89700	Too much messages "[drm:drm_atomic_set_fb_for_plane] Set [FB:69] for plane state" in dmesg is normal?	CLOSED FIXED	Add Me to the CC List
91454	[KBL/BDW] igt/ gem_render_linear_bits@swap-thrash test is timing out	CLOSED NOTABUG	Add Me to the CC List
92592	Background image is much too dark	RESOLVED MOVED	Add Me to the CC List
93840	[i965] Compiler backend uses too much stack with Alien: Isolation	RESOLVED FIXED	Add Me to the CC List
94482	Incoming/recording sound volume much too soft/low when routed through USB audio	RESOLVED MOVED	Add Me to the CC List
103274	BRW allocates too much heap memory	RESOLVED FIXED	Add Me to the CC List

**Attachment:** Don't add an attachment  
Add an attachment  
File: Enter the path to the file on your computer (or paste text as attachment).  
 Choose File | No file chosen  
(File size limit: 4194304 KB)

\* **Description:** Describe the attachment briefly.  
the clutter

**Content Type:** If the attachment is a patch, check the box below.  
 patch  
Otherwise, choose a method for determining the content type.  
 auto-detect  
 select from list:  PNG image (image/png)  
 enter manually:

**Flags:** Requestee:

**Keywords:** clutter, bad UI  
**Depends on:**   
**Blocks:**   
**See Also:**

Submit Bug |  Remember values as bookmarkable template

- Your bug-report has been created with an ID:

### 3) Check status of your bug-report:

- Search for your bug-id in the main menu search-bar:

### 4) Finish!

## What is Bugzilla?

Bugzilla is an open-source issue/bug tracking system that allows developers to keep track of outstanding problems with their product.

**Alias:** A short, unique name assigned to a bug in order to assist with looking it up and referring to it in other places in Bugzilla.

**Assignee:** The person in charge of resolving the bug.

**Blocks:** This bug must be resolved before the bugs listed in this field can be resolved.

**Bug ID:** The numeric id of a bug, unique within this entire installation of Bugzilla.

**CC:** Users who may not have a direct role to play on this bug, but who are interested in its progress.

**Depends on:** The bugs listed here must be resolved before this bug can be resolved.

**Importance:** The importance of a bug is described as the combination of its Priority and Severity.

**Keywords:** You can add keywords from a defined list to bugs, in order to easily identify and group them.

**Product:** Bugs are categorised into Products and Components.

**QA Contact:** The person responsible for confirming this bug if it is unconfirmed, and for verifying the fix once the bug has been resolved.

**Reporter:** The person who filed this bug.

**Summary:** The bug summary is a short sentence which succinctly describes what the bug is about.

**URL:** Bugs can have URL associated with them—for e.g., a pointer to a web site where the problem is seen.

**Version:** The version field defines the version of the software the bug was found in. #503(prac5)---  
03/08/19---

## Partical 5

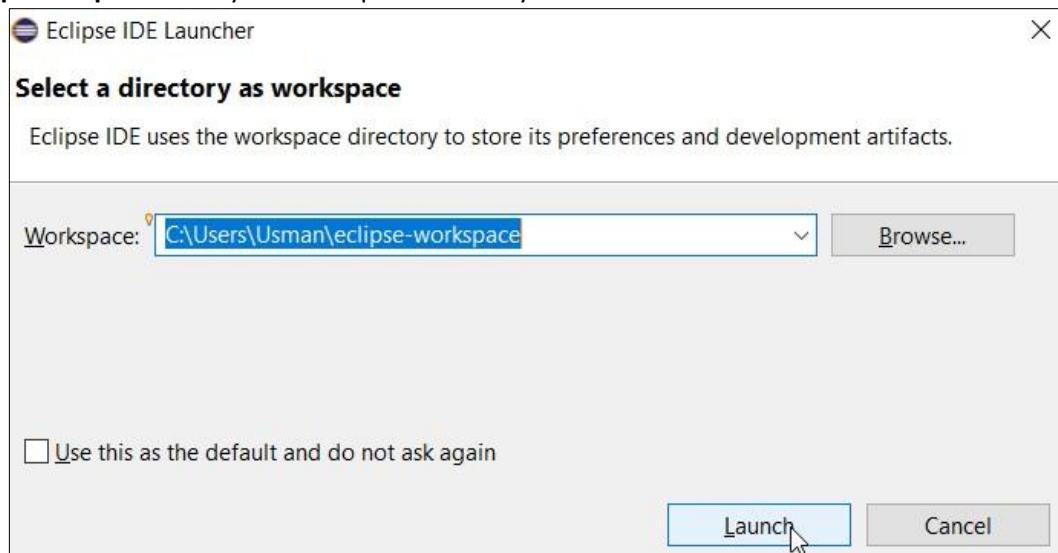
#AIM: Write and test a program to update 10 student records into Excel file(table).

### PRE-REQUISITES:

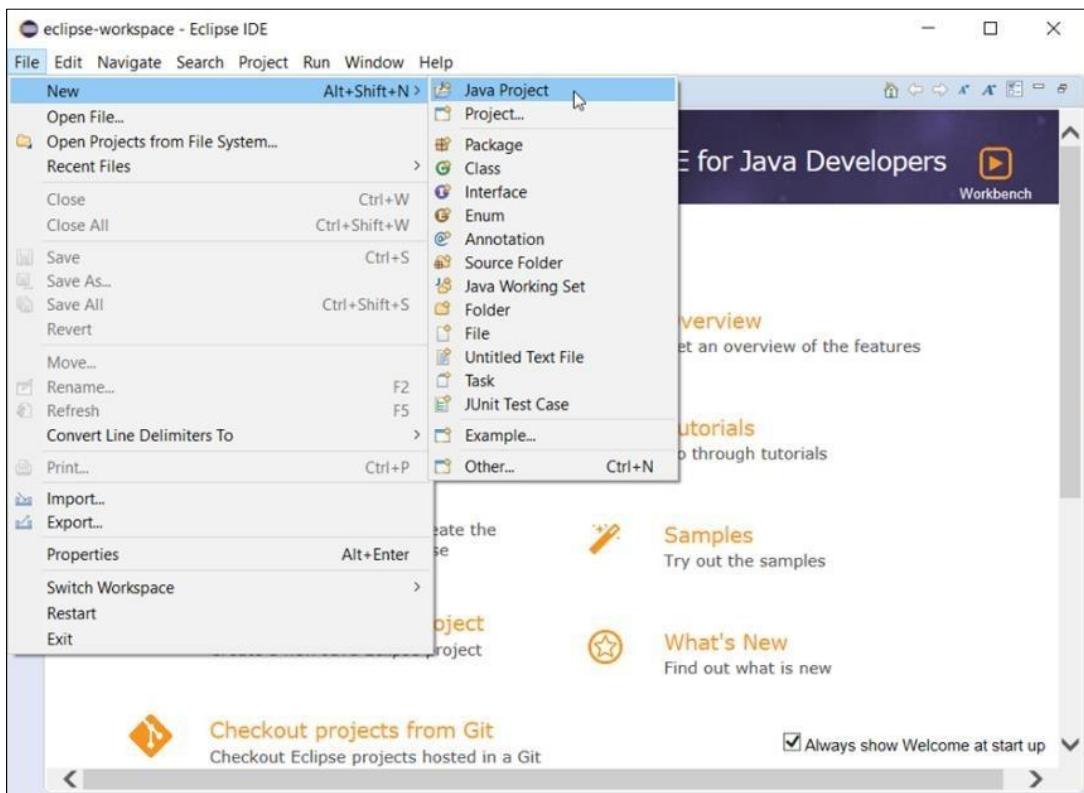
- 1) Check that you have **Eclipse IDE**.
- 2) To Download “**JXL.JAR**”:
  - Visit <http://www.java2s.com/Code/Jar/j/Downloadjxl26jar.htm>
  - Download this file: “**jxl/jxl-2.6.jar.zip( 603 k)**” and **extract it.**(you'll get the .jar file)

### STEPS:

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



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

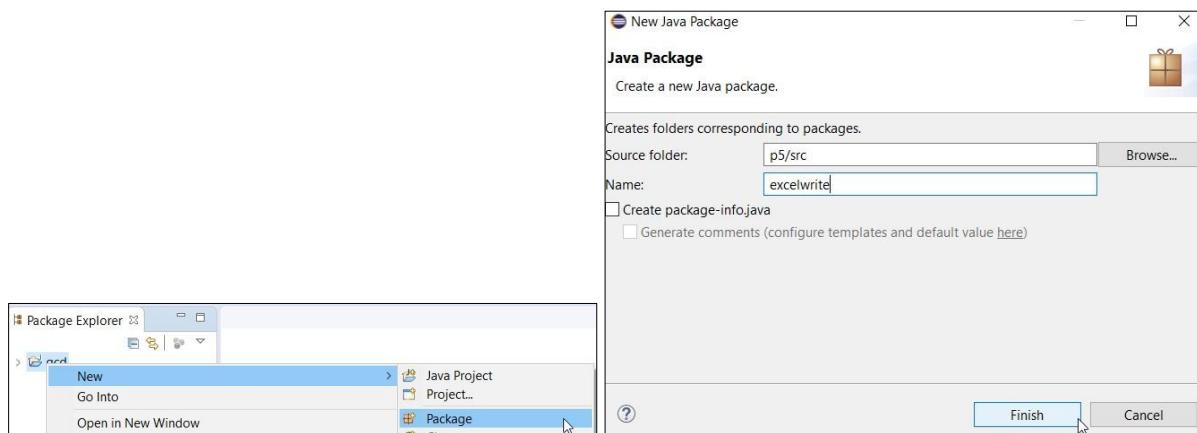


### 3) Name the project as “p5” > click Finish > click Don’t Create module:

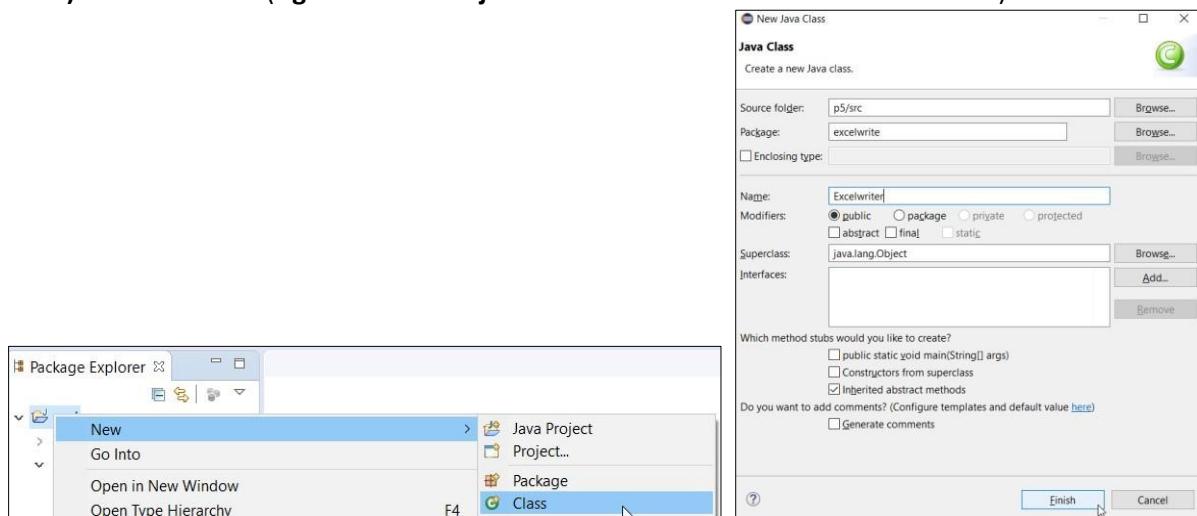
The image contains two side-by-side dialog boxes. The left dialog is titled 'Create a Java Project' and shows fields for 'Project name' (p5), 'Location' (C:\Users\Usman\eclipse-workspace\p5), 'JRE' (JavaSE-12 selected), and 'Project layout' (Create separate folders for sources and class files selected). The right dialog is titled 'Create module-info.java' and shows a 'Module name' field with 'p5' entered, with 'Create' and 'Don't Create' buttons at the bottom.

### 4) Close the “Welcome” tab.

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

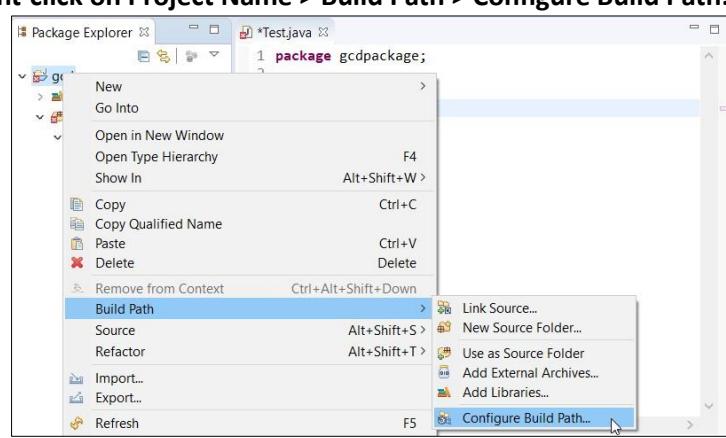


## 6) Create a Class(right-click on Project Name > New > Class > Name it > Finish):

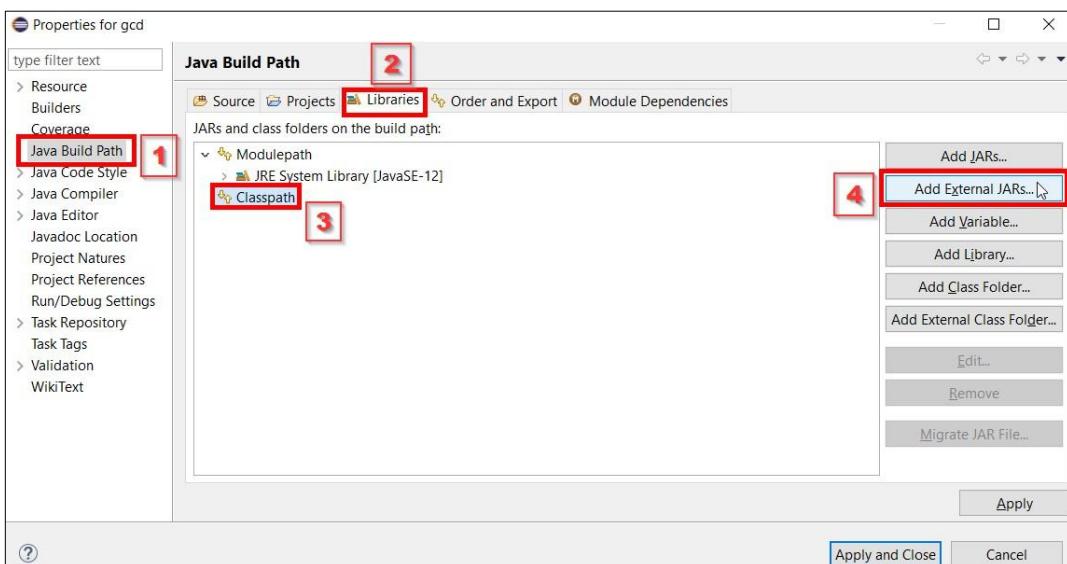


## 7) Adding “JXL(JAR file)” in Eclipse IDE:

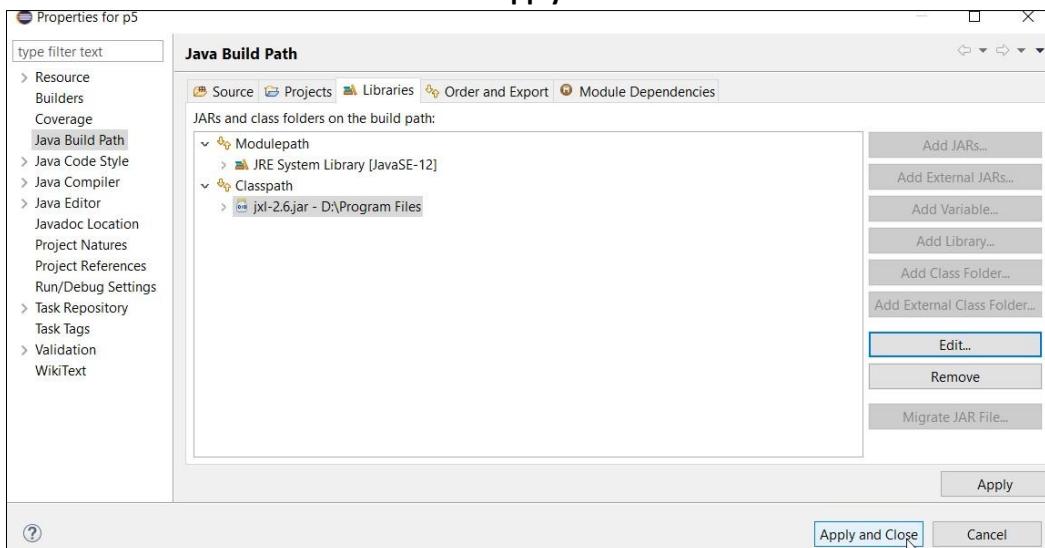
- right-click on Project Name > Build Path > Configure Build Path...



- 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 {
```

```

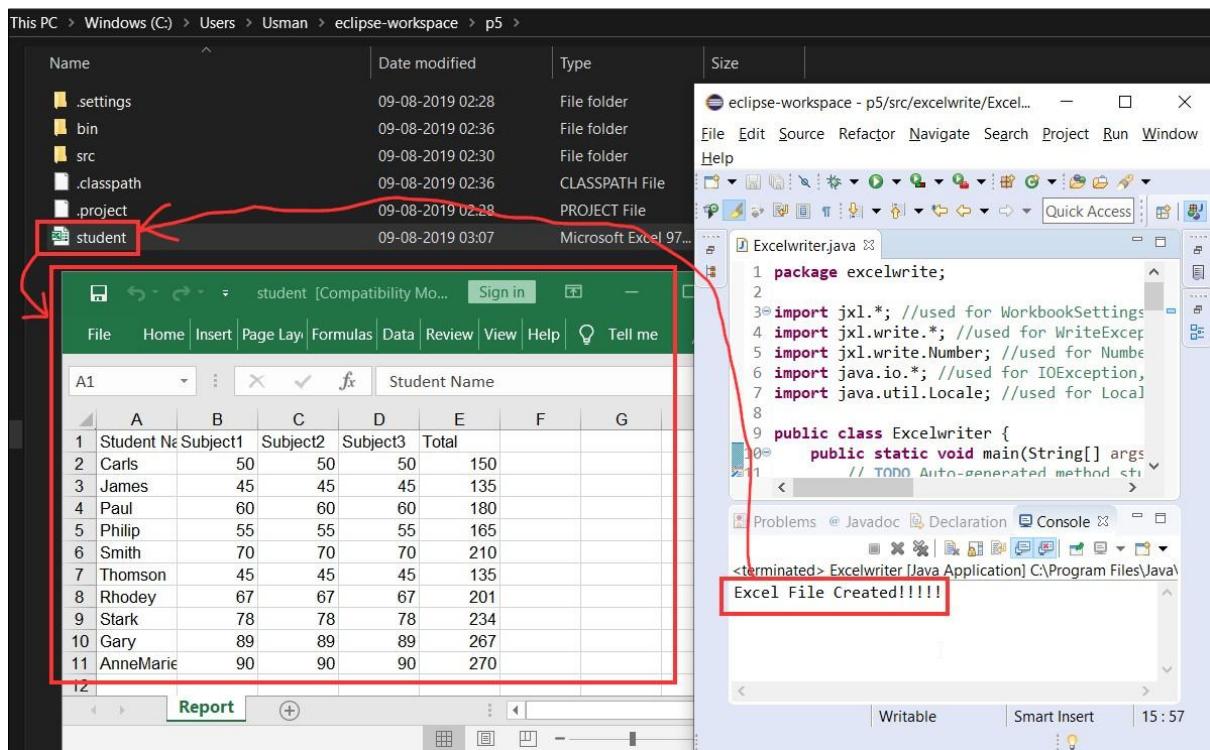
        public static void main(String[] args) throws IOException,WriteException {
            // TODO Auto-generated method stub
            int r=0,c=0;
            String header[]={ "Student
Name", "Subject1", "Subject2", "Subject3", "Total"};
            String
sname[]={ "Carls", "James", "Paul", "Philip", "Smith", "Thomson", "Rhodey", "Stark", "Gary
",
"AnneMarie"};
            int marks[]={ 50,45,60,55,70,45,67,78,89,90,30};

            File 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:**



**10) 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 .

## Partical 6

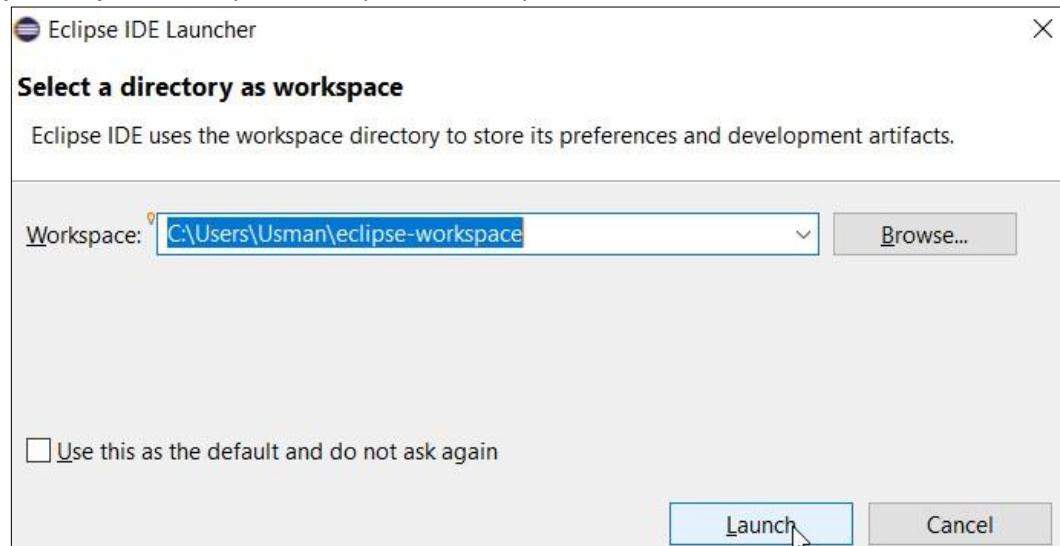
**#AIM:** Write and test a program to select the number of students from Excel file(table) who have scored 60 or more in any one subject(or all subjects).

### PRE-REQUISITES:

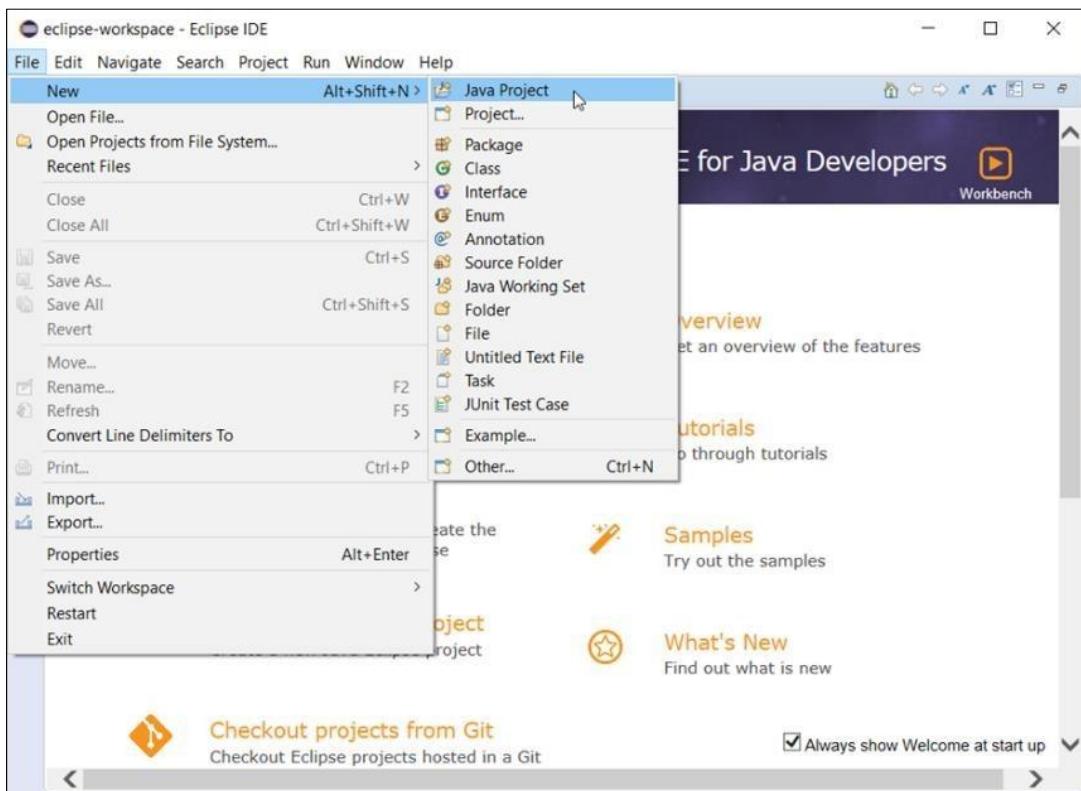
- 1) Check that you have **Eclipse IDE**.
- 2) Check that you have **JXL.JAR**.
- 3) Check that you've the **Excel file**("student.xls") that we'll be working on for reading data.

### STEPS:

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



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

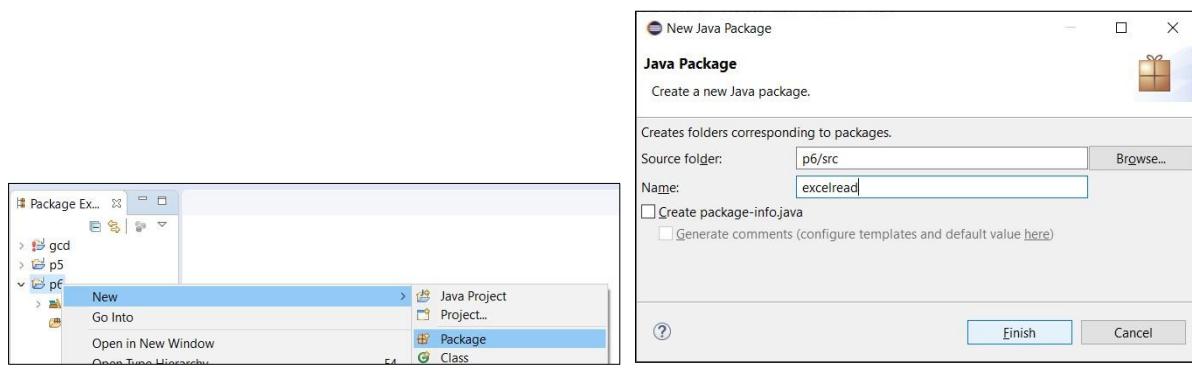


**3) Name the project as “p6” > click Finish > click Don’t Create module:**

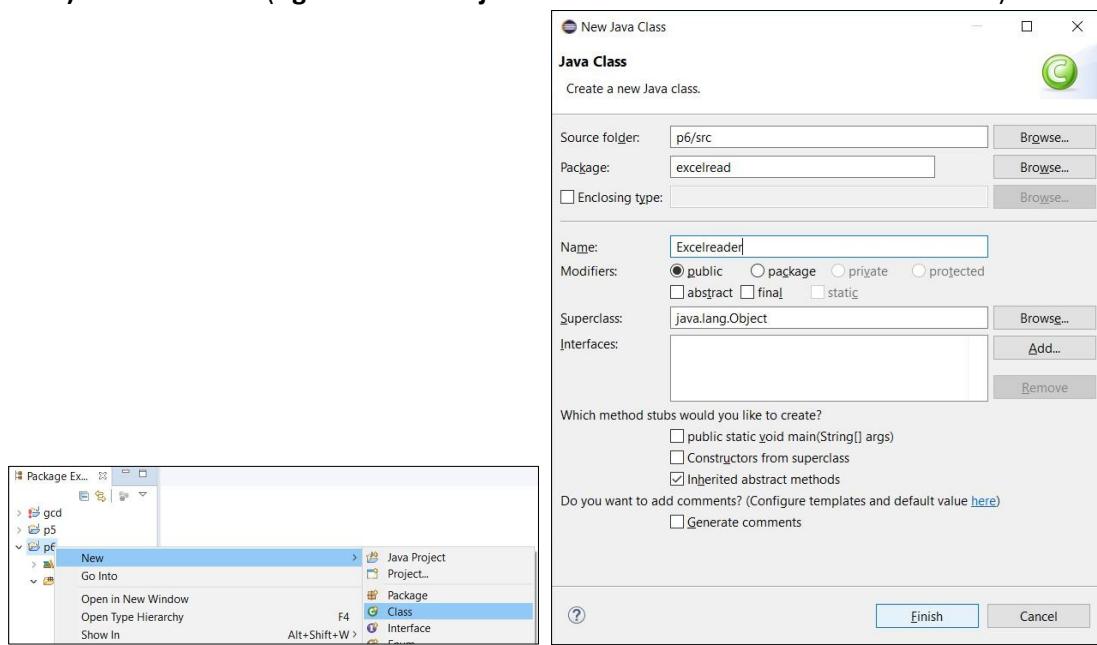
The image shows two overlapping dialog boxes. The left dialog is titled "Create a Java Project" and has fields for "Project name" (set to "p6"), "Location" (set to "C:\Users\Usman\eclipse-workspace\p6"), and "JRE" (set to "JavaSE-12"). The right dialog is titled "Create module-info.java" and has a "Module name" field set to "p6". Both dialogs have "Create" and "Don't Create" buttons at the bottom.

**4) Close the “Welcome” tab.**

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

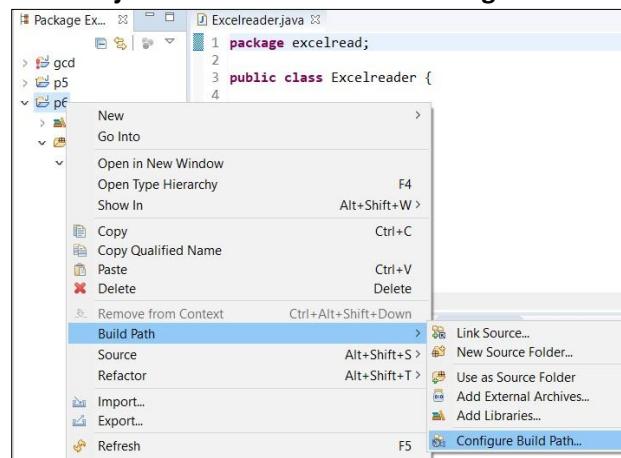


## 6) Create a Class(right-click on Project Name > New > Class > Name it > Finish):

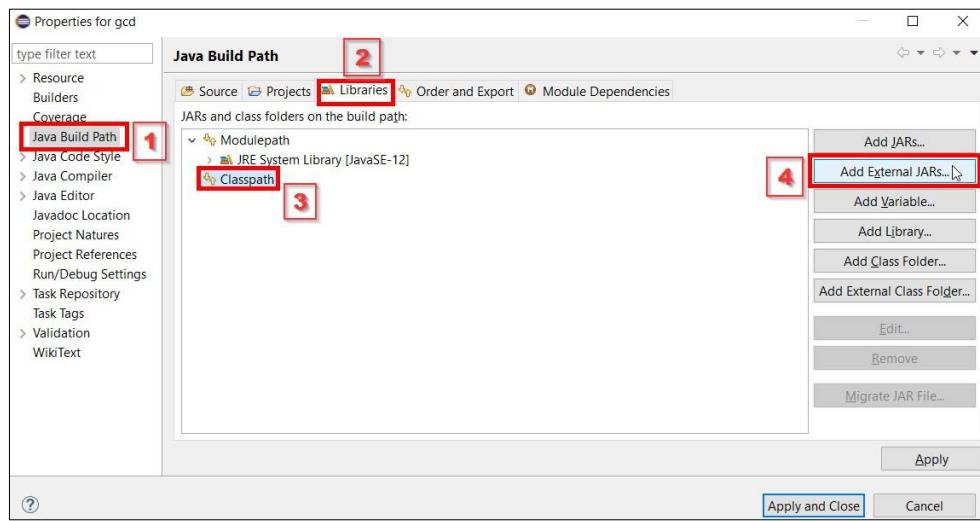


## 7) Adding “JXL(JAR file)” in Eclipse IDE:

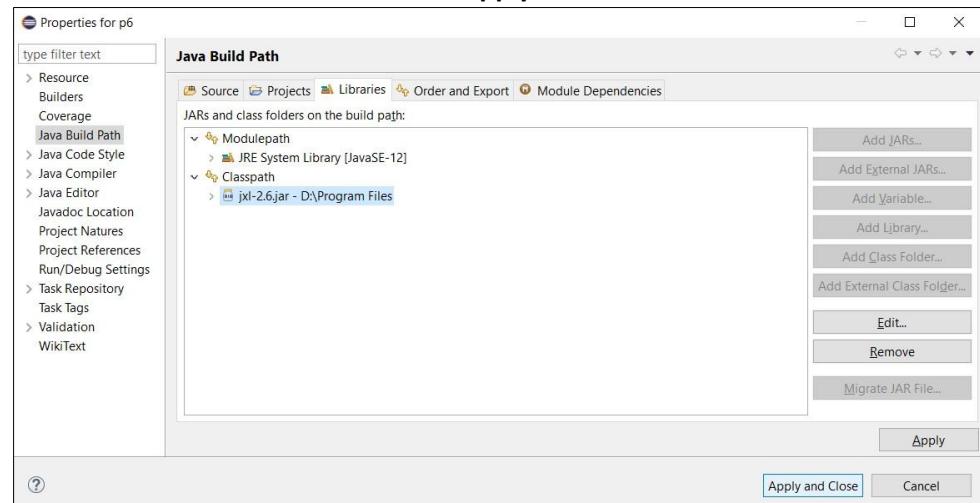
- right-click on Project Name > Build Path > Configure Build Path...



- 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 $\geq 60$  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 sheet = w.getSheet(0);
// Loop over first 10 column and lines
(int j = 0; j < sheet.getRows(); j++) {
            for (int i = 0; i < sheet.getColumns()-1; i++) {
                Cell 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();}
    }

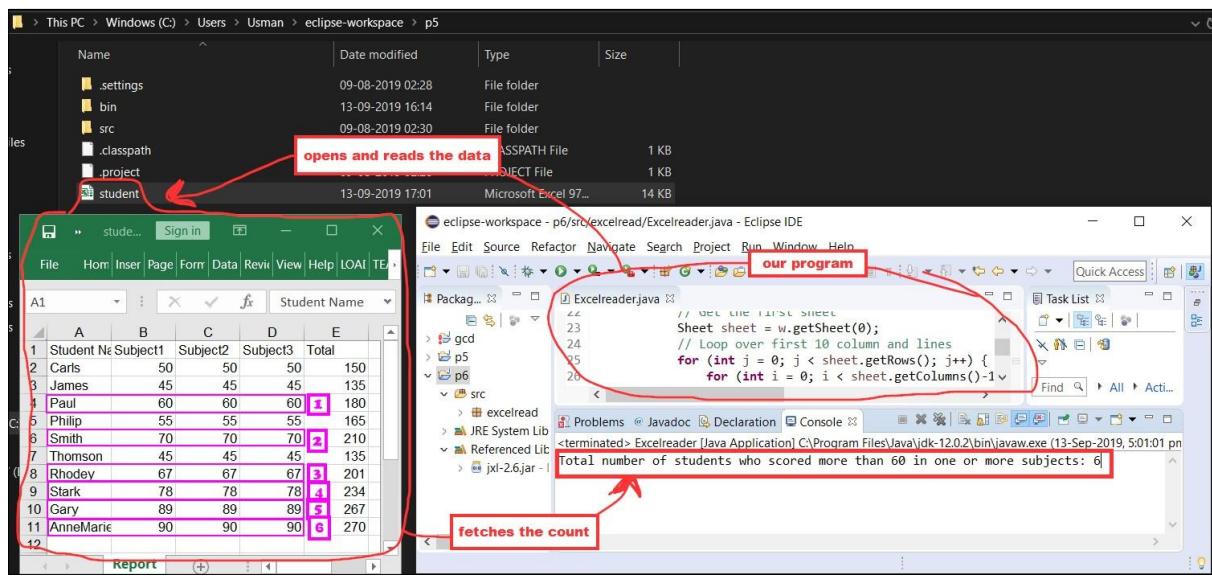
    public static void main(String[] args) throws IOException {
Excelreader test = new Excelreader();

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

```

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

- **OUTPUT:**



**10) Finish!**

# Partical 7

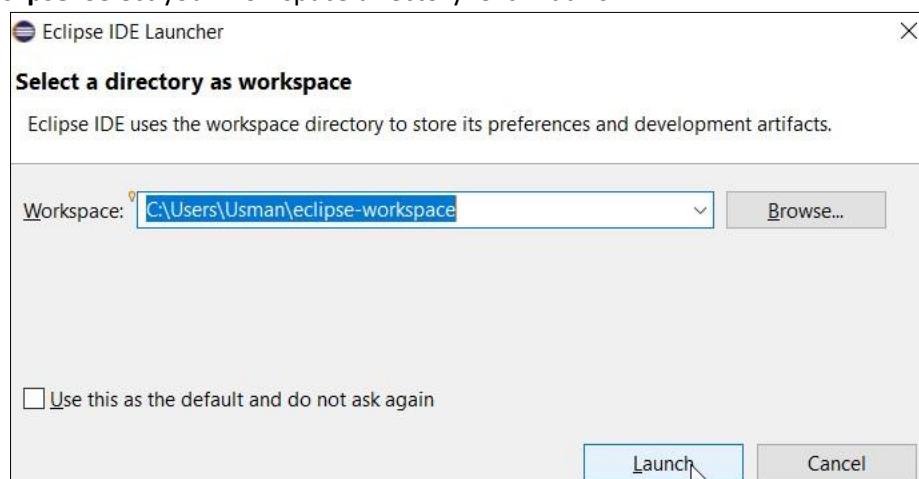
#AIM: Write and test a program for logging in facebook webpage.

## PRE-REQUISITES:

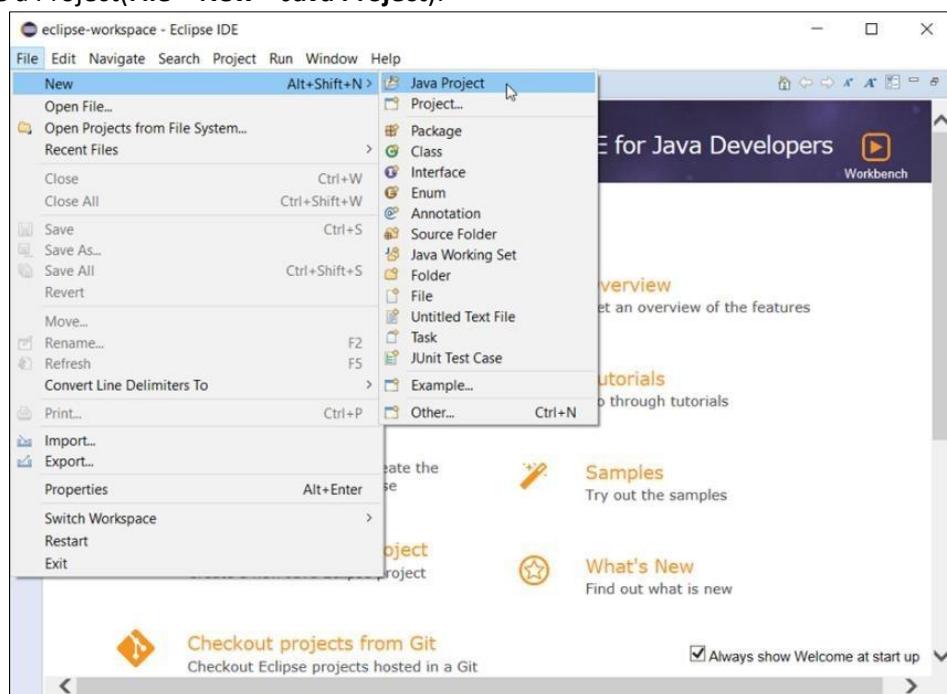
- 1) Check that you have **JDK**.
- 2) Check that you have **Eclipse IDE**.
- 3) Check that you have **Selenium Server Driver and Client Driver(JAR files)**.
- 4) Check that you have **Gecko Driver**.
- 5) Check that you have a stable Internet connection.

## STEPS:

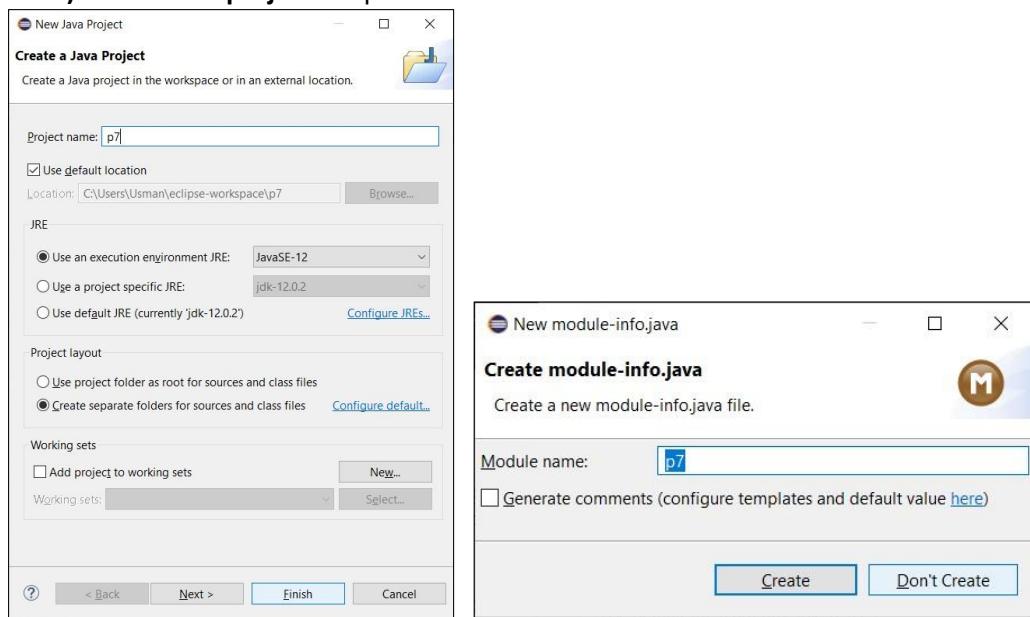
- 1) Open Eclipse. Select your workspace directory. Click **Launch**:



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

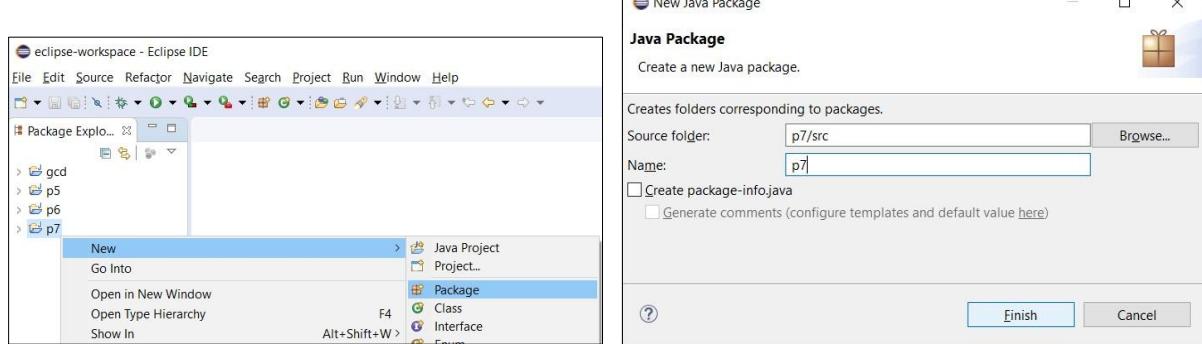


**3) Name the project as “p7” > click Finish > click Don’t Create module:**

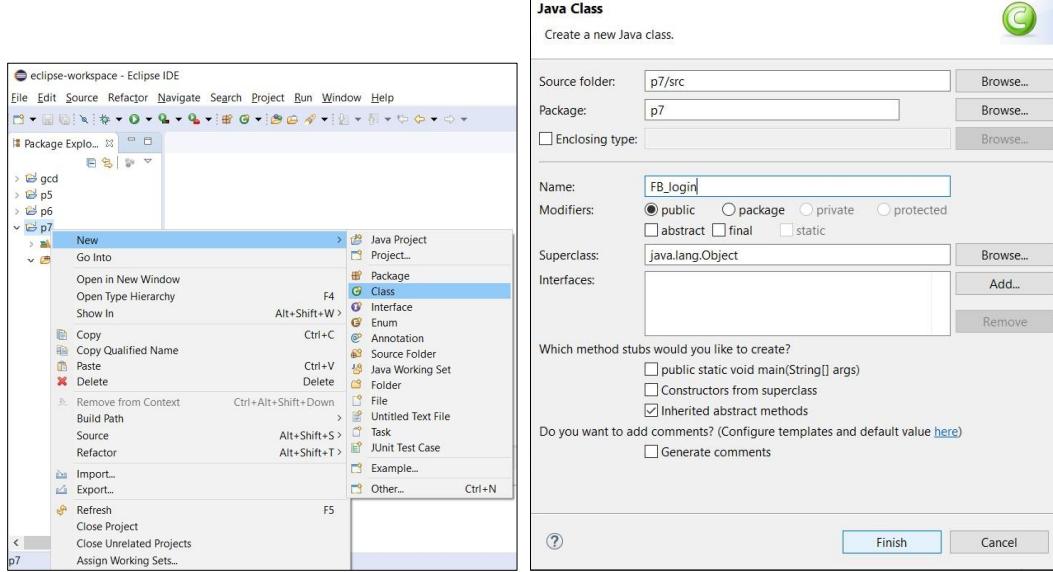


**4) Close the “Welcome” tab.**

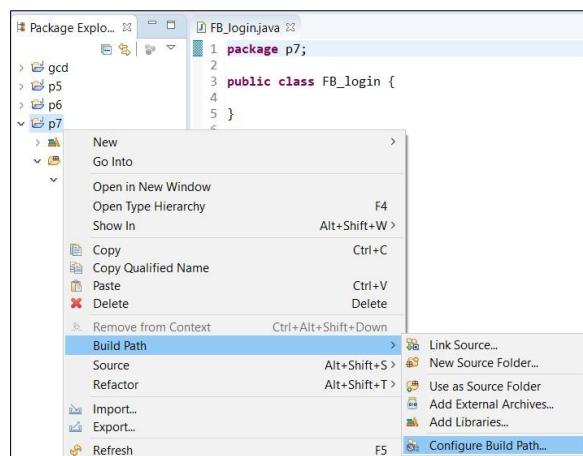
**5) Create a Package(right-click on Project Name > New > Package > Name it > Finish):**



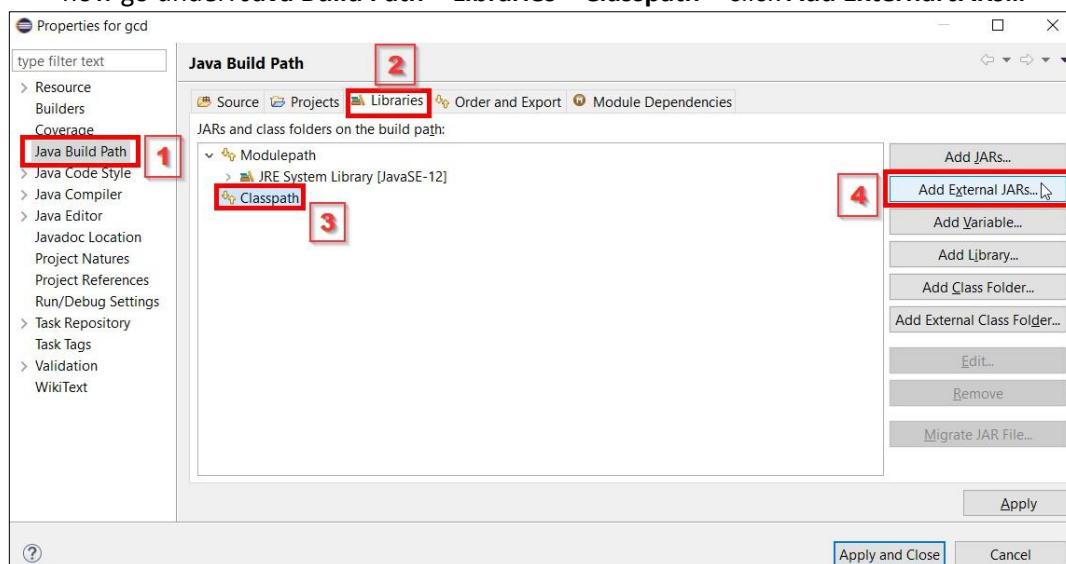
**6) Create a Class(right-click on Project Name > New > Class > Name it > Finish):**



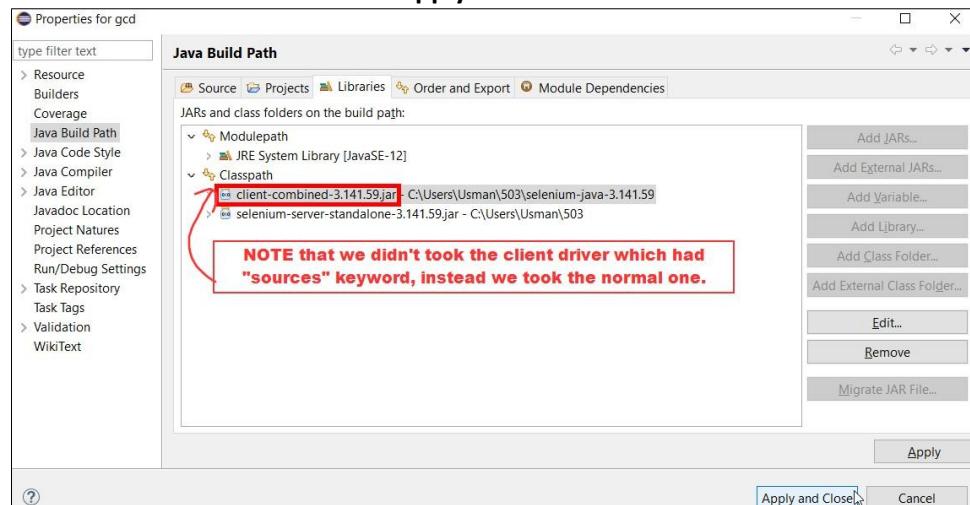
- 7) Adding “**Selenium Server Driver and Client Driver(JAR files)**” in Eclipse IDE: • right-click on Project Name > Build Path > Configure Build Path...



- 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)

The screenshot shows a browser window with the Facebook login page. A red arrow points from the 'Log In' button to the developer tools panel. The developer tools panel shows the HTML structure and the computed styles for the 'Log In' button. A red box highlights the 'input#u\_0\_2' element with the text 'ID for the HTML element("Log In" button), here is "u\_0\_2"'. Another red box highlights the 'remember this ID and put it in the script' text.

- 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();

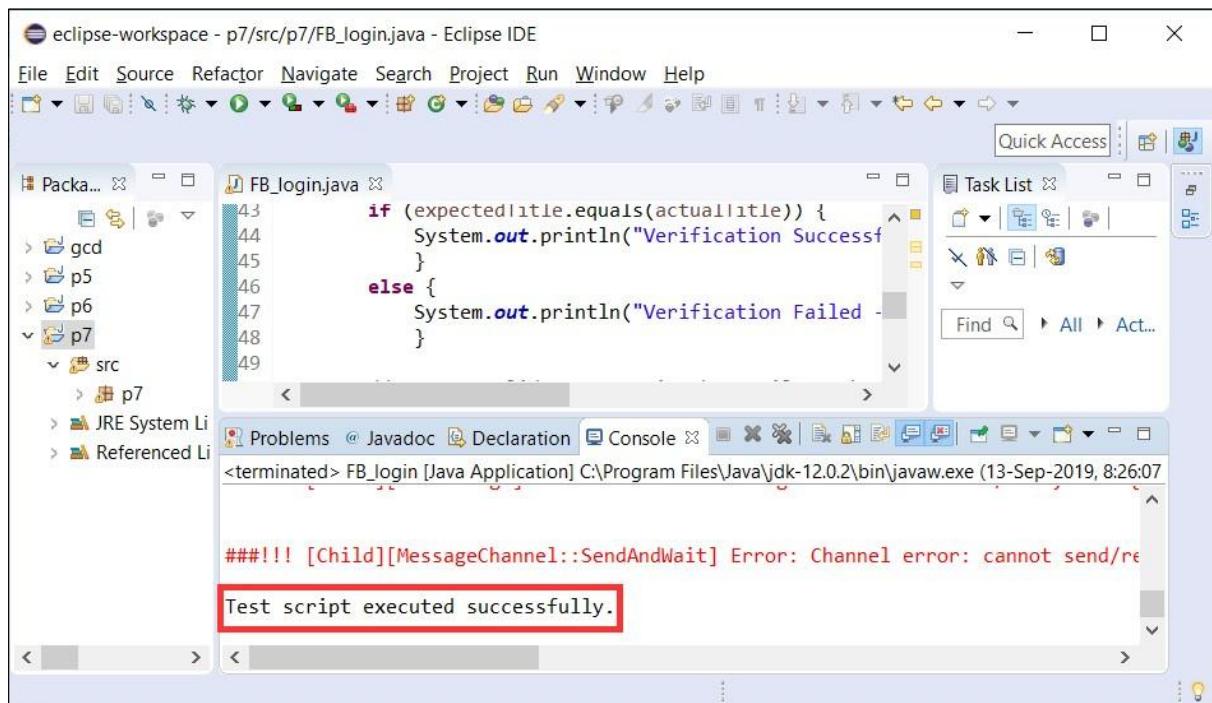
//close the web browser
driver.close();
System.out.println("Test script executed successfully.");

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

```

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

- **OUTPUT:**



#### 11) Finish!

## Partical 8

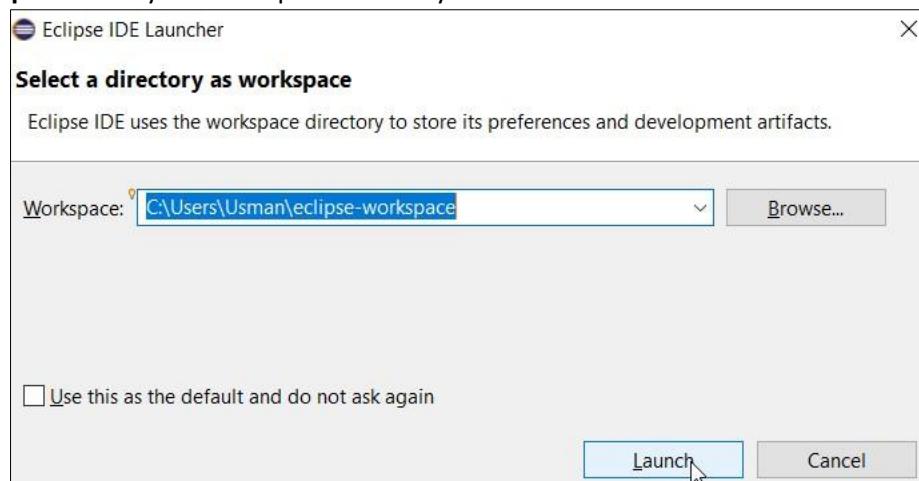
#AIM: Write and test a program to provide total number of objects present/available on webpage.

### PRE-REQUISITES:

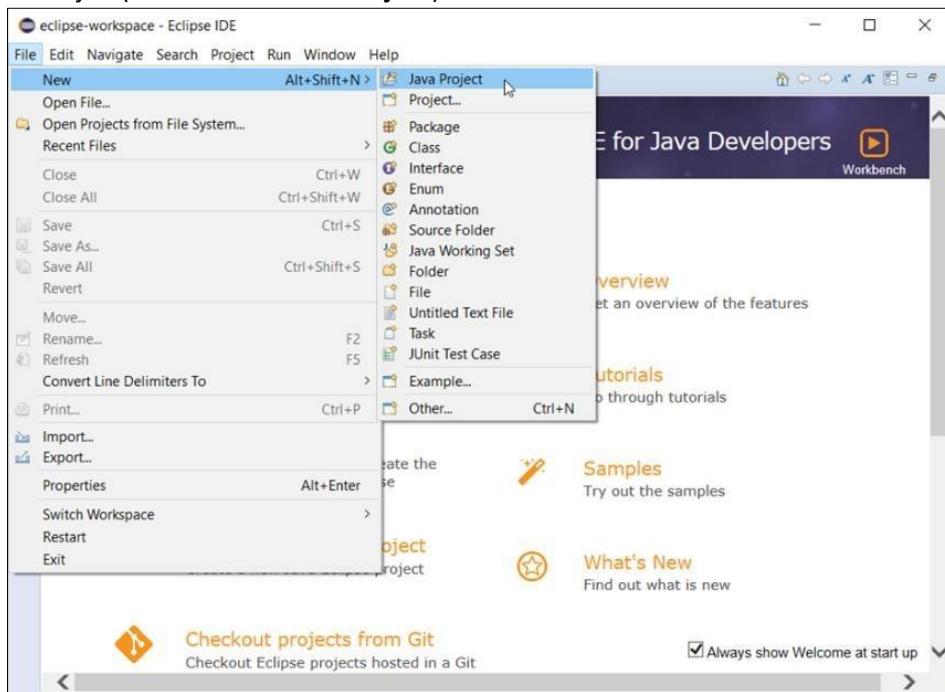
- 1) Check that you have **JDK**.
- 2) Check that you have **Eclipse IDE**.
- 3) Check that you have **Selenium Server Driver and Client Driver(JAR files)**.
- 4) Check that you have **Gecko Driver**.
- 5) Check that you have a stable Internet connection.

### STEPS:

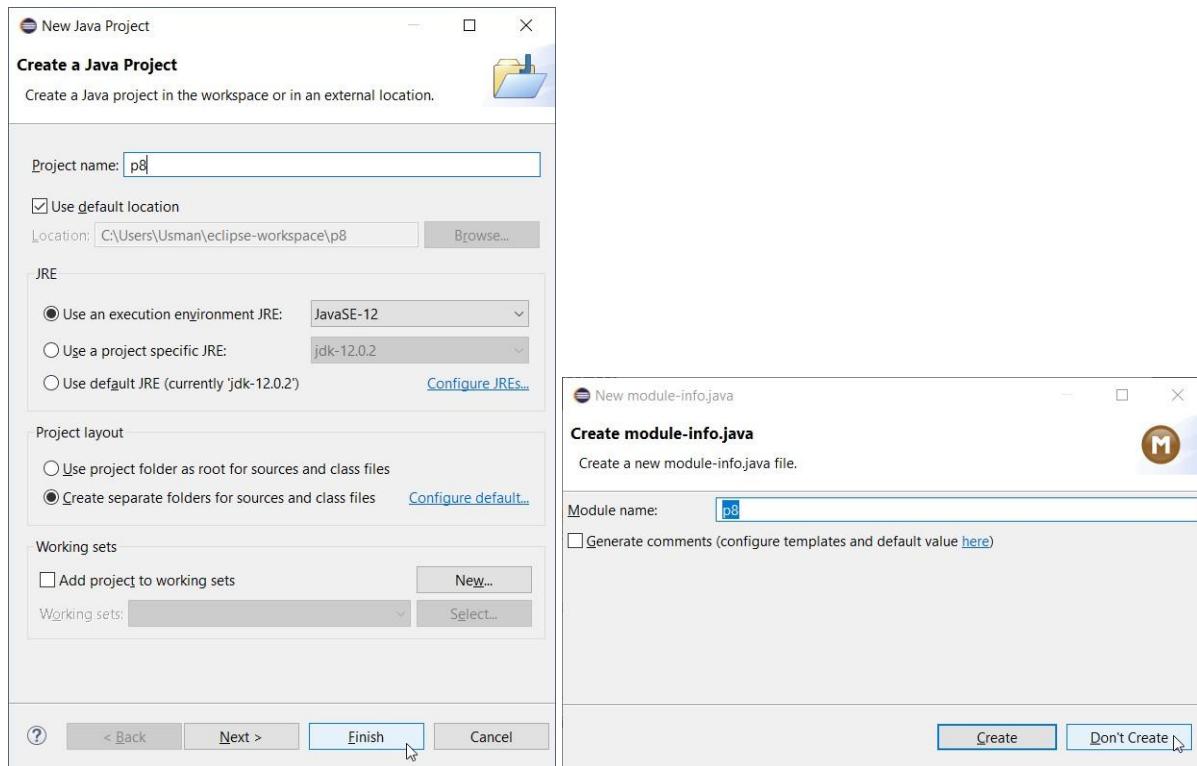
- 1) Open Eclipse. Select your workspace directory. Click **Launch**:



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

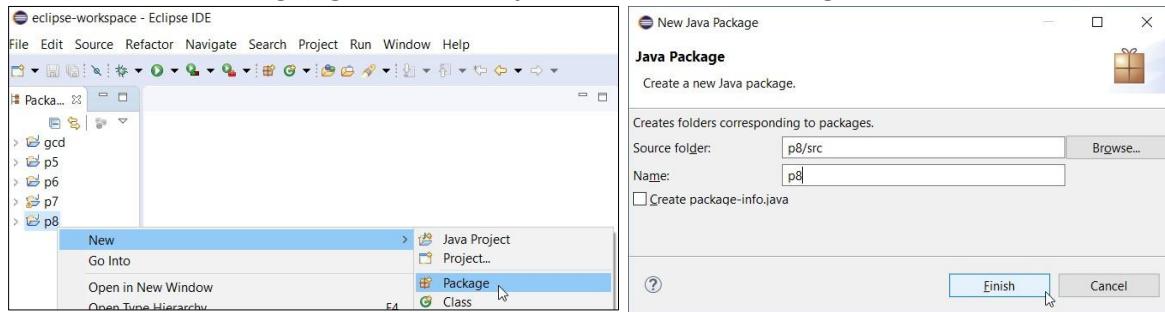


**3) Name the project as “p8” > click Finish > click Don’t Create module:**

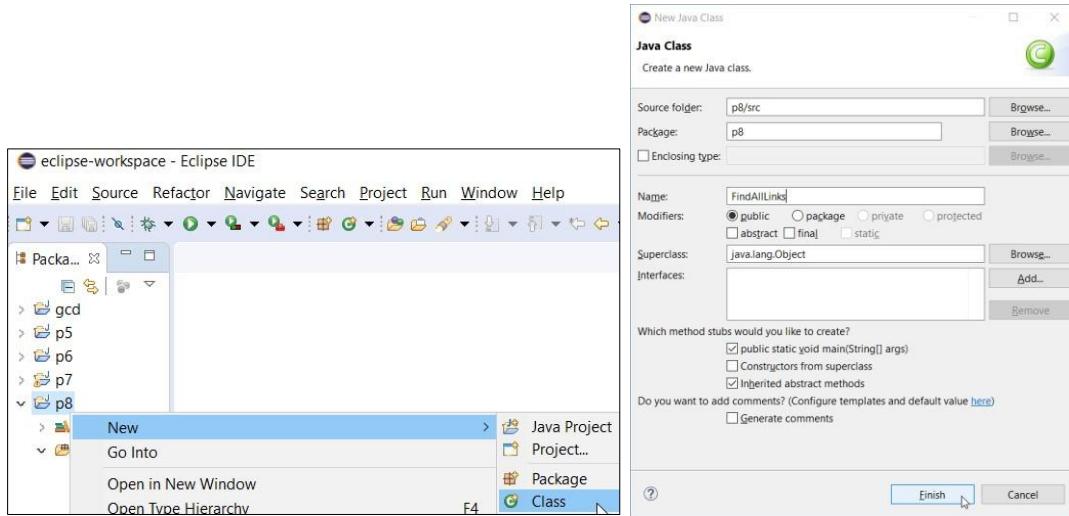


**4) Close the “Welcome” tab.**

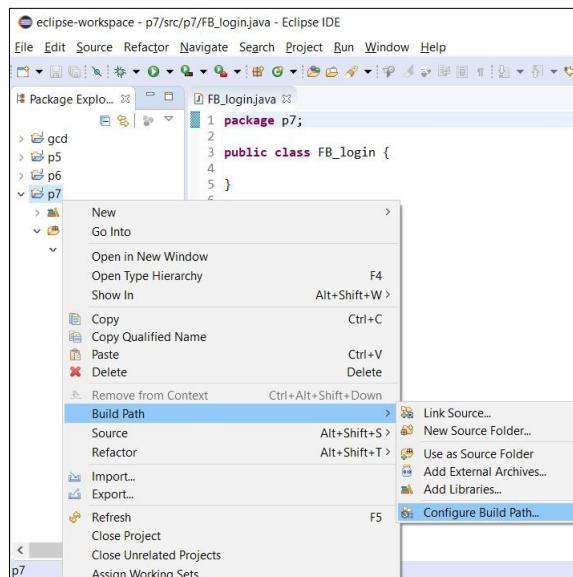
**5) Create a Package(right-click on Project Name > New > Package > Name it > Finish):**



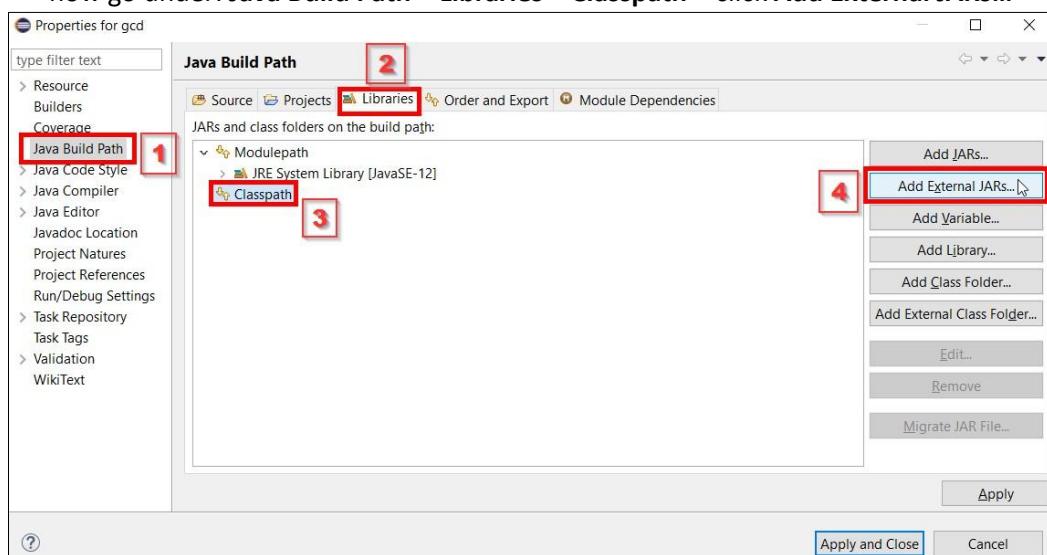
**6) Create a Class(right-click on Project Name > New > Class > Name it > Finish):**



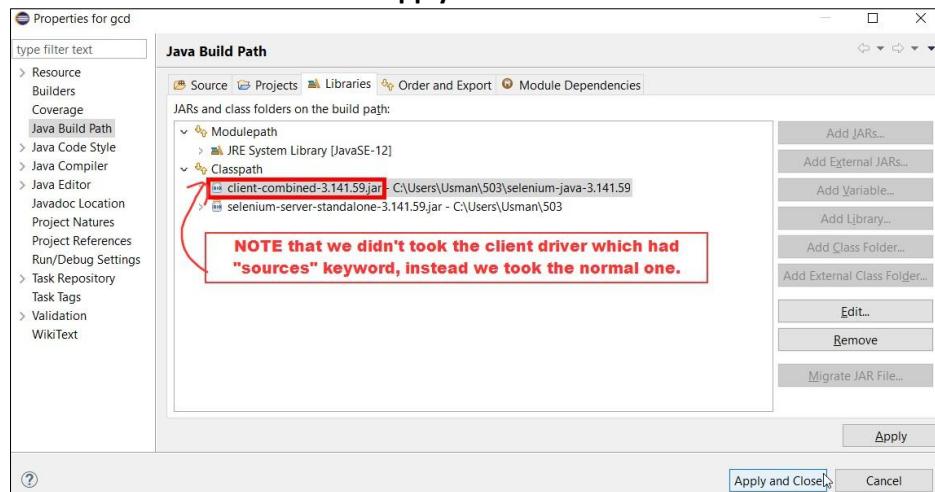
- 7) Adding “**Selenium Server Driver and Client Driver(JAR files)**” in Eclipse IDE: • right-click on Project Name > Build Path > Configure Build Path...



- 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);
    }
}
```

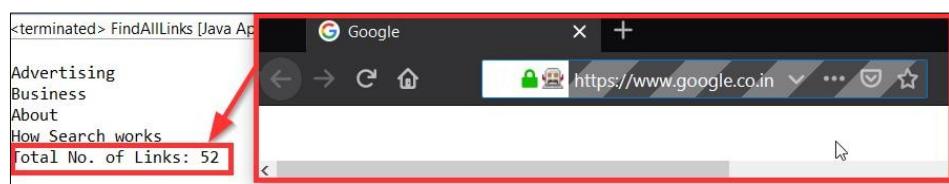
```

        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();
}
}

```

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



# Partical 9

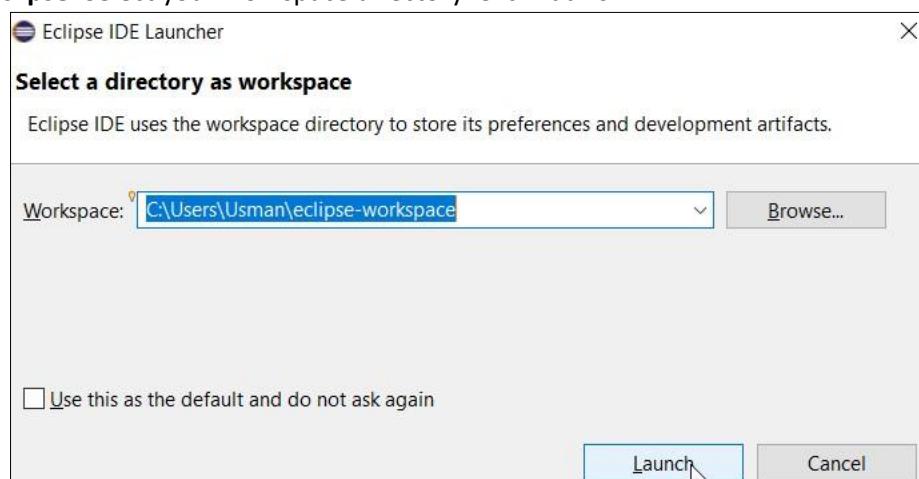
#AIM: Write and test a program to get the number of items in a list/combo box.

## PRE-REQUISITES:

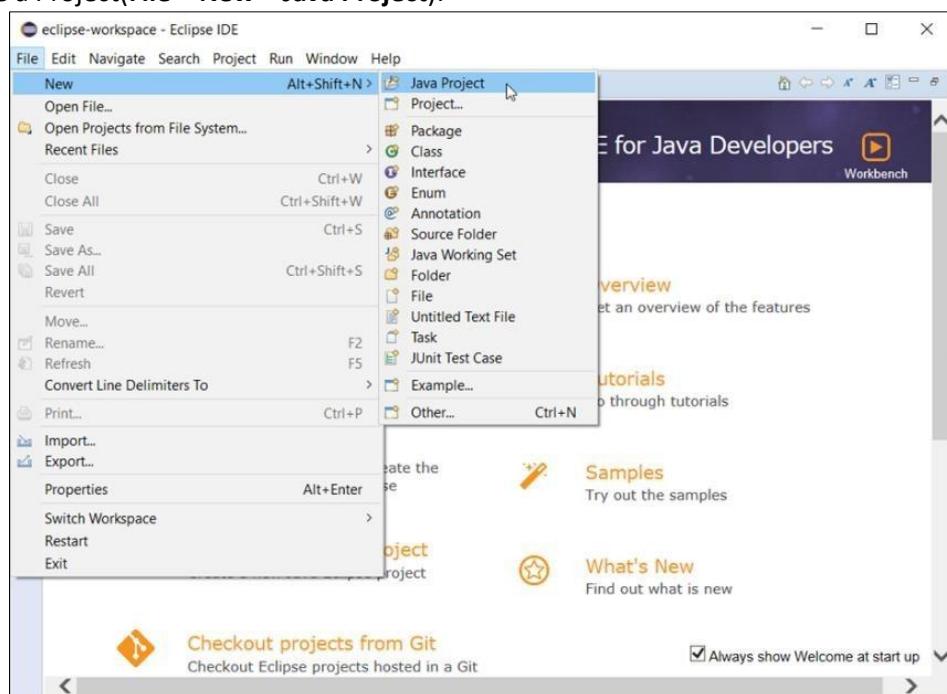
- 1) Check that you have **JDK**.
- 2) Check that you have **Eclipse IDE**.
- 3) Check that you have **Selenium Server Driver and Client Driver(JAR files)**.
- 4) Check that you have **Gecko Driver**.
- 5) Check that you have a stable Internet connection.

## STEPS:

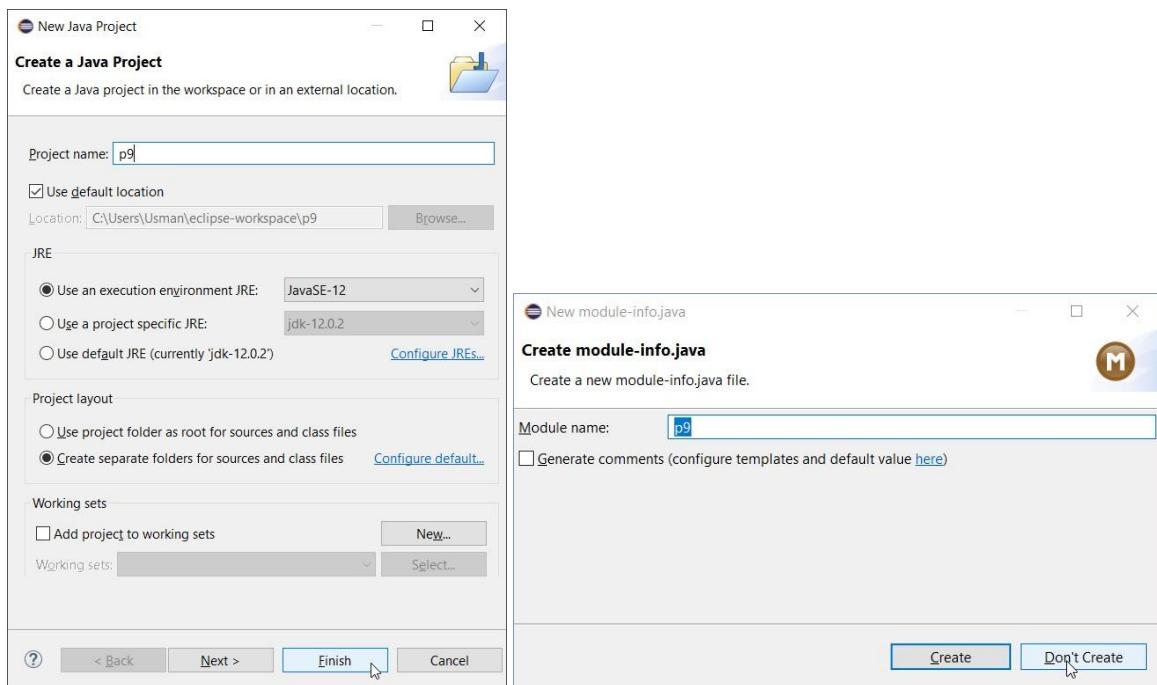
- 1) Open Eclipse. Select your workspace directory. Click **Launch**:



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

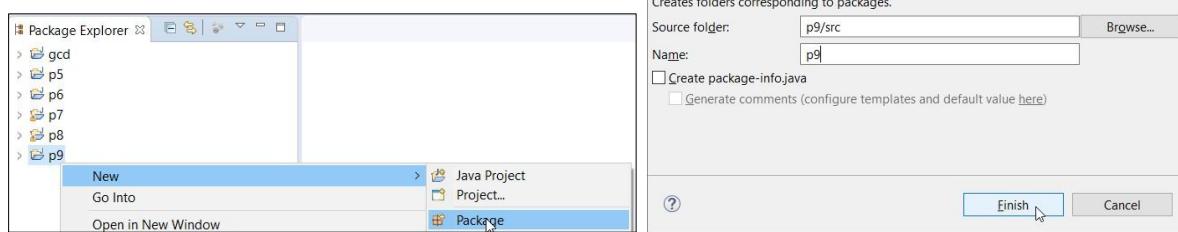


**3) Name the project as “p9” > click Finish > click Don’t Create module:**

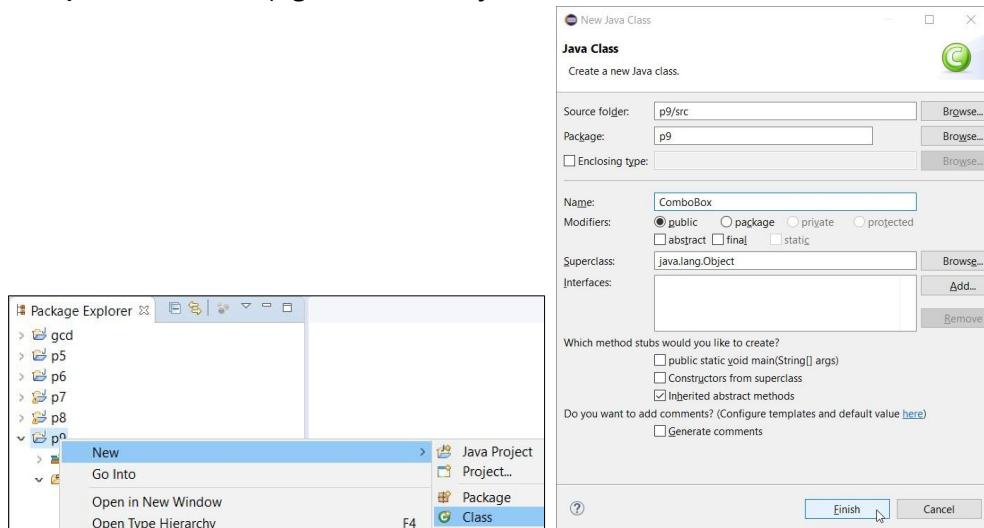


**4) Close the “Welcome” tab.**

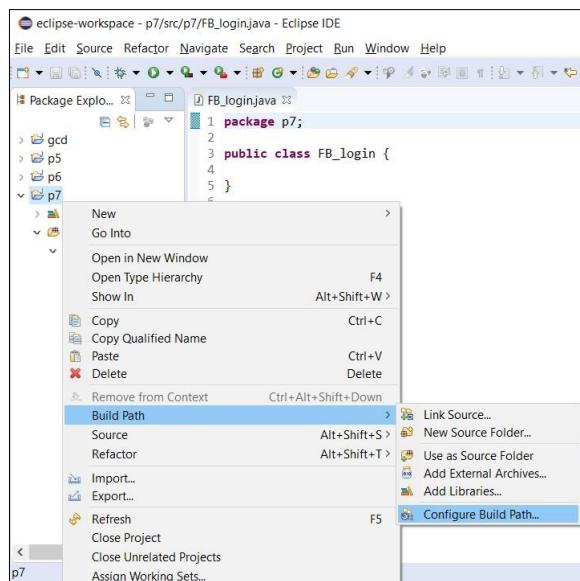
**5) Create a Package(right-click on Project Name > New > Package > Name it > Finish):**



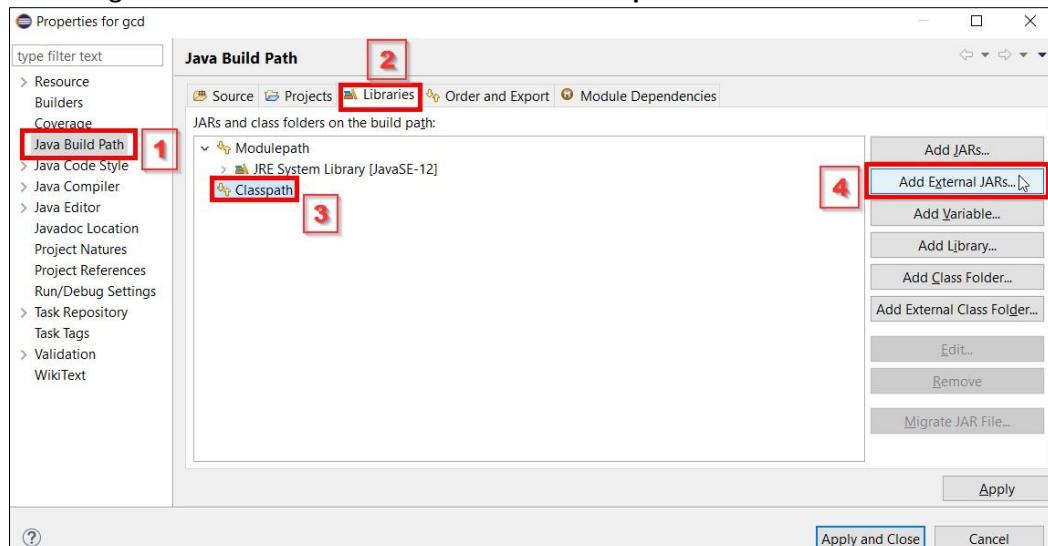
**6) Create a Class(right-click on Project Name > New > Class > Name it > Finish):**



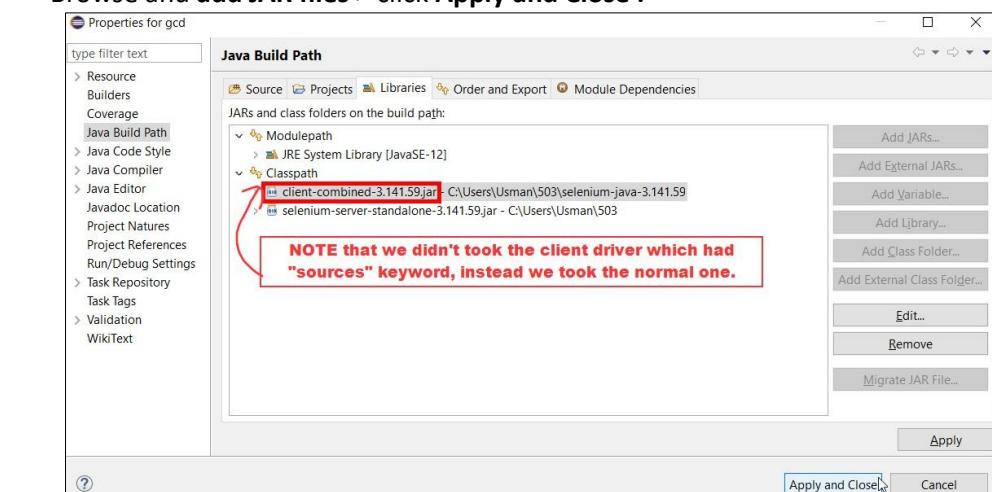
- 7) Adding “**Selenium Server Driver and Client Driver(JAR files)**” in Eclipse IDE: • right-click on Project Name > Build Path > Configure Build Path...



- 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:**

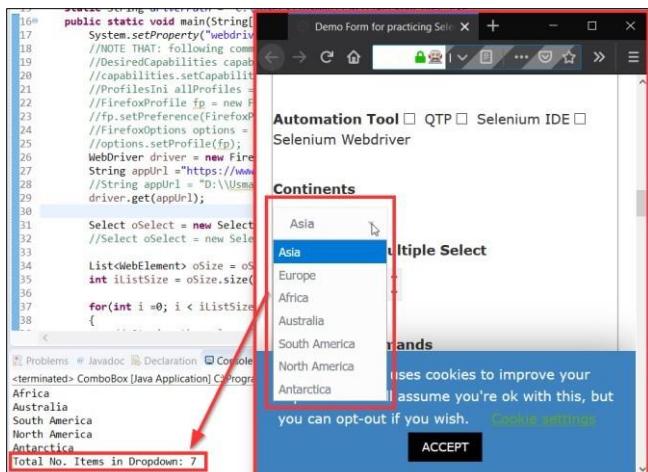
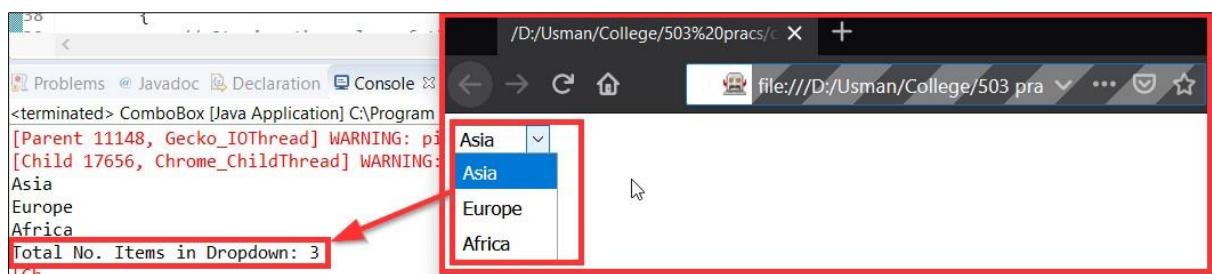


Figure 1: by using the DYNAMIC URL(website)



# Partical 10

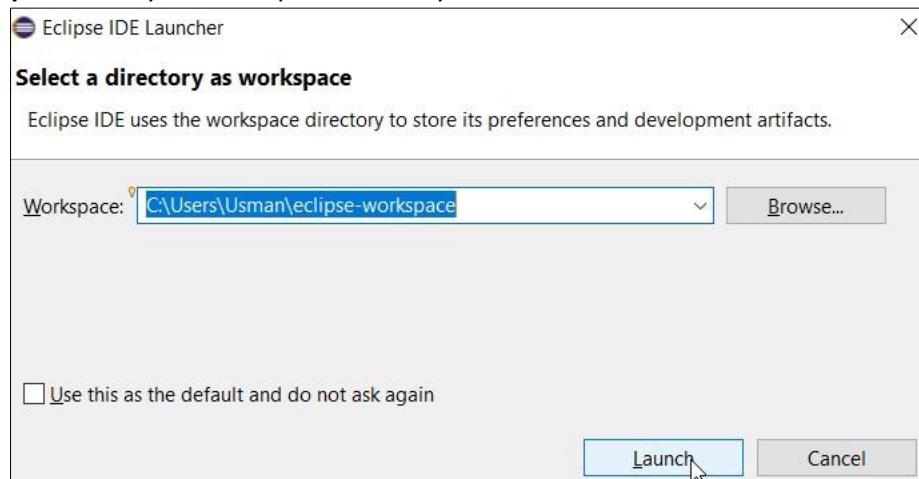
**#AIM:** Write and test a program to count the number of check boxes on the page checked and unchecked count.

## PRE-REQUISITES:

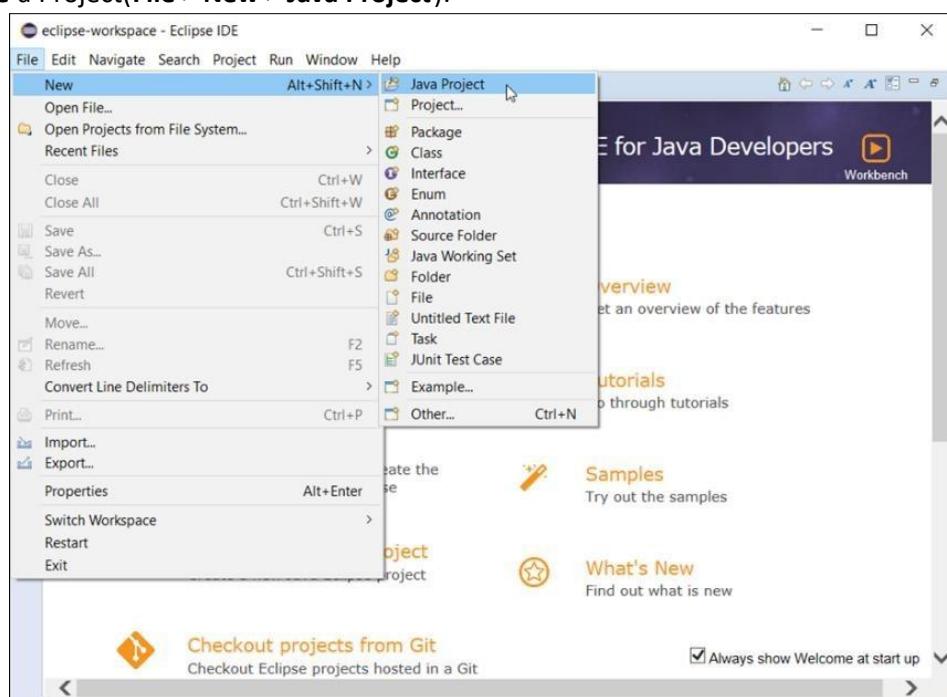
- 1) Check that you have **JDK**.
- 2) Check that you have **Eclipse IDE**.
- 3) Check that you have **Selenium Server Driver and Client Driver(JAR files)**.
- 4) Check that you have **Gecko Driver**.
- 5) Check that you have a stable Internet connection.

## STEPS:

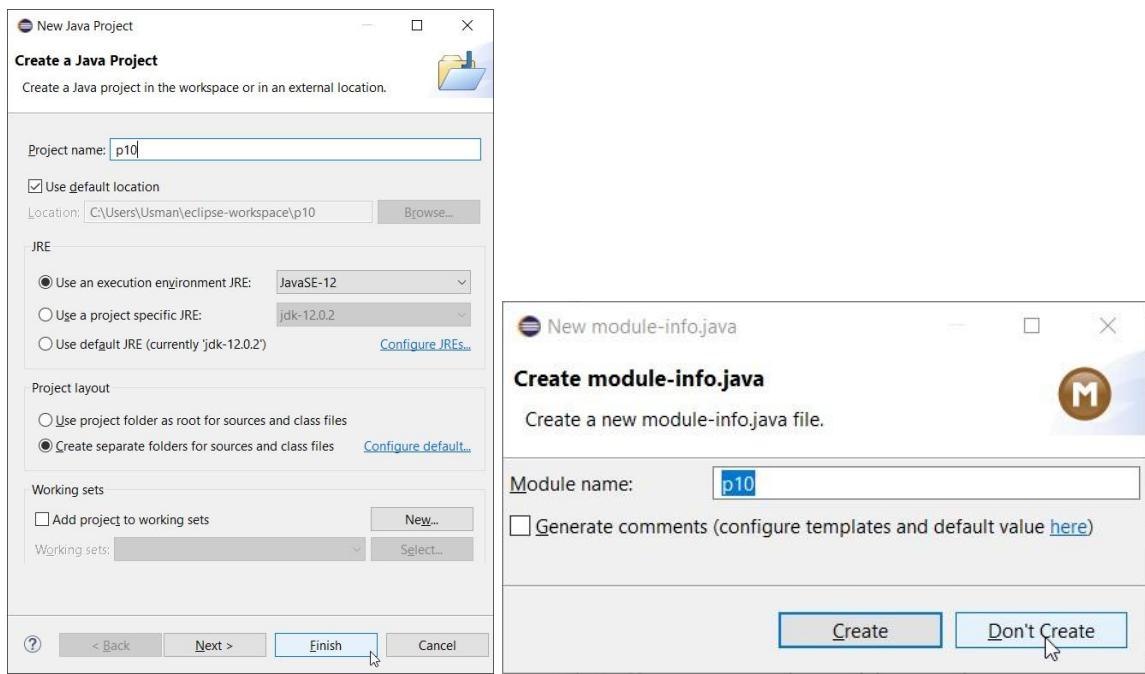
- 1) Open Eclipse. Select your workspace directory. Click **Launch**:



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

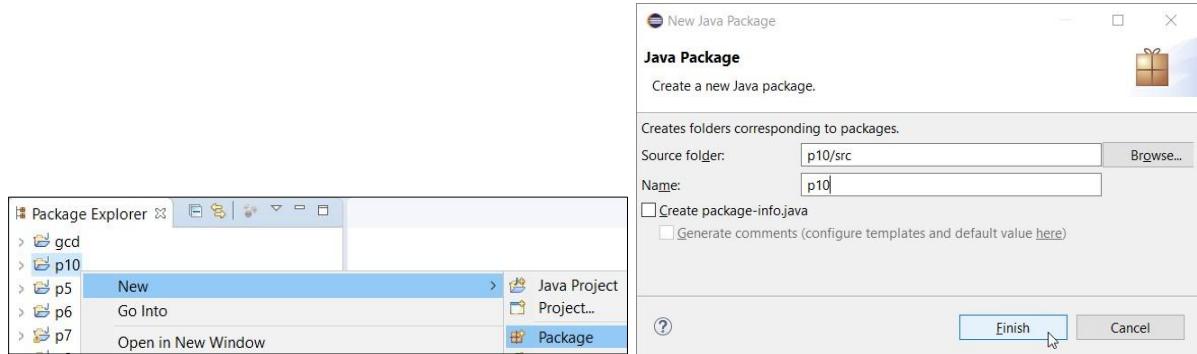


**3) Name the project as “p10” > click **Finish** > click **Don’t Create** module:**

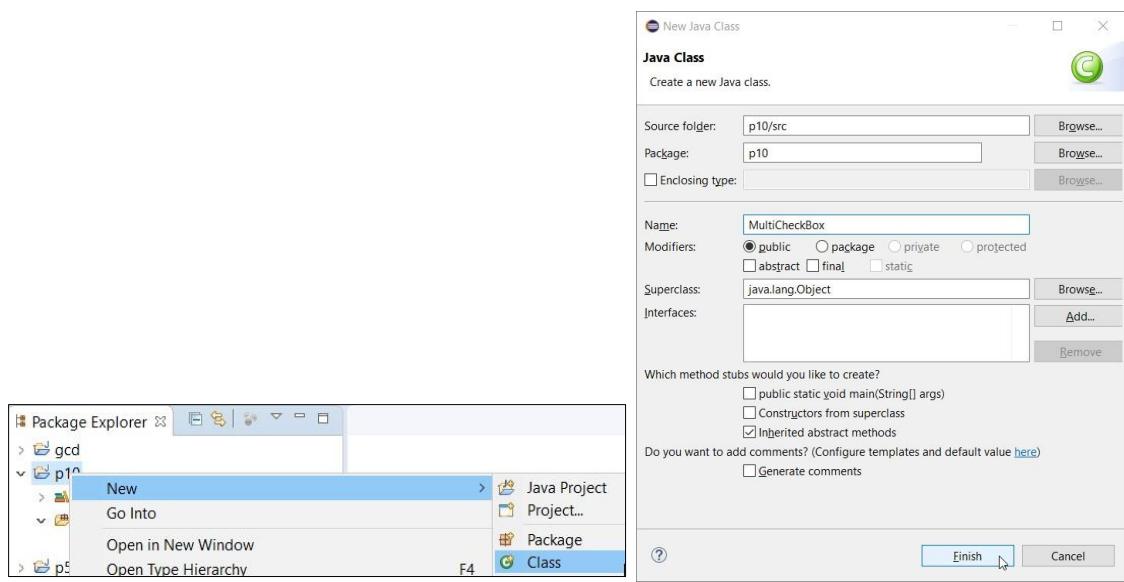


**4) Close the “Welcome” tab.**

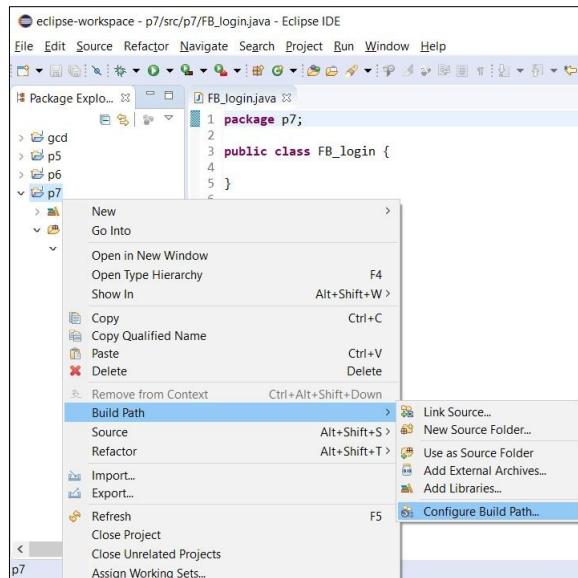
**5) Create a Package(right-click on Project Name > New > Package > Name it > Finish):**



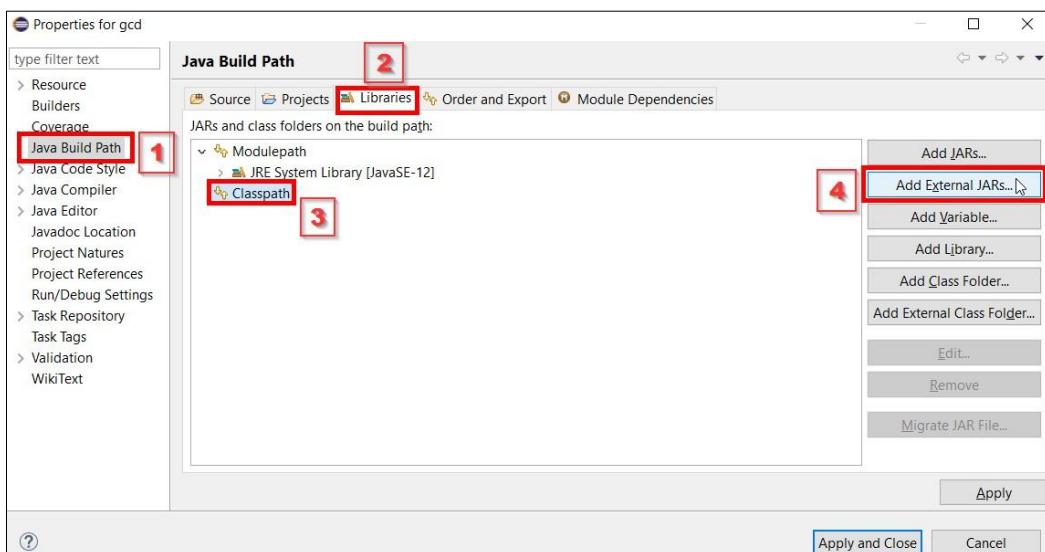
**6) Create a Class(right-click on Project Name > New > Class > Name it > Finish):**



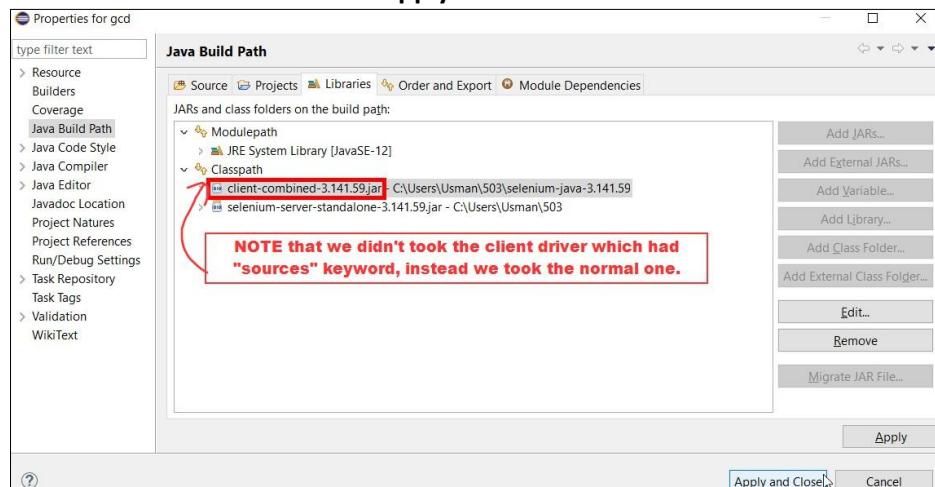
- 7) Adding “Selenium Server Driver and Client Driver(JAR files)” in Eclipse IDE: • right-click on Project Name > Build Path > Configure Build Path...



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



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

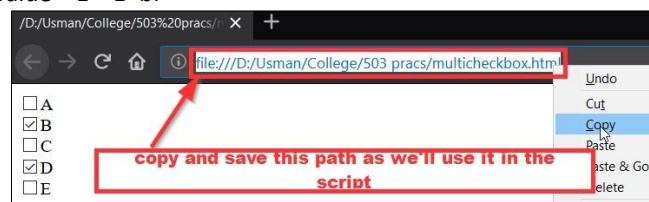


## 8) Create HTML file in a notepad > Save it > Open it in browser > Copy the URL:

(NOTE THAT: as this is our LOCAL FILE, this file will be opened thru STATIC URL in script)

---(multicheckbox.html)---

```
<input type="checkbox" value="A">A<br>
<input type="checkbox" value="B" CHECKED>B<br>
<input type="checkbox" value="C">C<br>
<input type="checkbox" value="D" CHECKED>D<br>
<input type="checkbox" value="E">E<br>
```



## 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 “xpath” with the help of Selenium Drivers -  
 and to show the result.  
 -Hence automating the work in browser)

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

---(MultiCheckBox.java)---

```

package p10;
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;

public class MultiCheckBox {
    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();
        String appUrl = "https://www.toolsqa.com/automation-practice-form/";
        //DYNAMIC URL(WEBBSITE)
        //String appUrl =
        "file:///D:/Usman/College/503%20pracs/multicheckbox.html"; //STATIC URL(LOCAL
        FILE)
        driver.get(appUrl);

        List<WebElement> checkBoxes =
        driver.findElements(By.xpath("//input[@type='checkbox']"));
        int checkedCount=0, uncheckedCount=0;
        for(int i=0; i<checkBoxes.size(); i++){
            System.out.println(i+" checkbox is selected
            "+checkBoxes.get(i).isSelected());
            if(checkBoxes.get(i).isSelected()) {checkedCount++;}
        else {uncheckedCount++;}
        }
        //driver.quit();
        System.out.println("No. of selected checkbox: "+checkedCount);
        System.out.println("No. of unselected checkbox: "+uncheckedCount);
    }
}

```

## 10) Run the file from Eclipse IDE:

- **OUTPUT:**

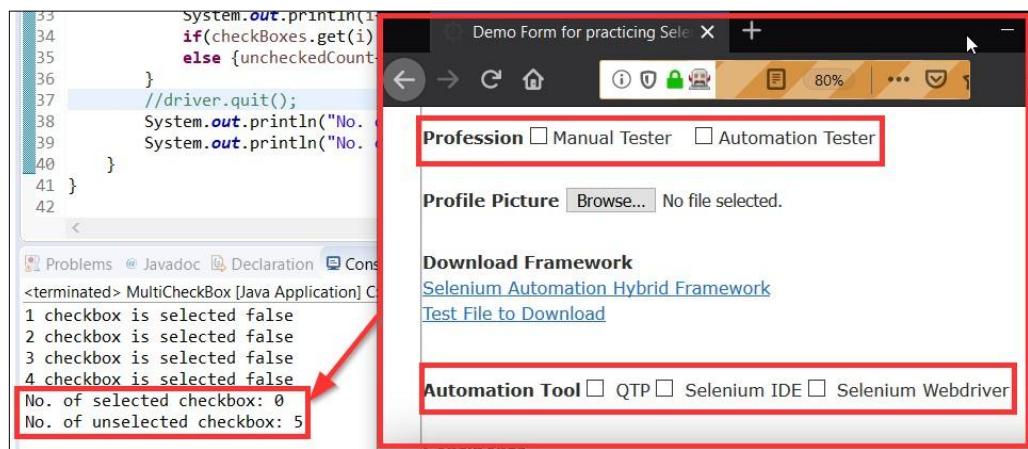


Figure 1: by using the DYNAMIC URL(website)

