Date:17/08/2020

**Practical No 1**

**Aim :-** Install Selenium IDE; Write a test suite containing minimum 4 test cases for different formats.

**Installation of Selenium IDE in Chromium Browser**

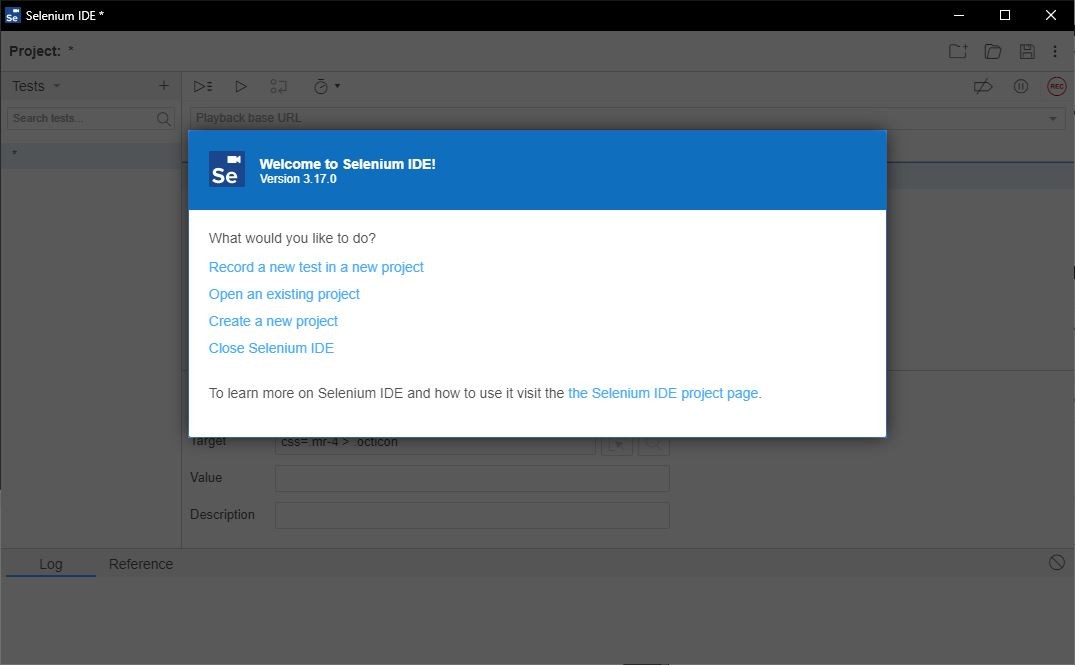
**Step-1 :-** Launch Chromium browser.

**Step-2 :-** Search for Selenium IDE in the ”chrome web store”.

**Step-3 :-** Select ”Add to Chrome” Button which will lead to chromium pop-up asking for permission to add the extension, click ”Add Extension” button.

**Step-4 :-** Chromium will automatically download and install the extension. Which can be accessed from the ”Extensions” Icon Button on the Navigation Bar.

**Step-5 :-** Clicking on the selenium IDE icon, the extension will open a new window with the selenium IDE.



# Creating/Recording and Running a Test Suite

**Step-1 :-** Launch the Chromium browser and open the selenium IDE extension.

**Step-2 :-** When the Welcome screen of selenium IDE is visible, select the option ”create a new project”.

**Step-3 :-** Give an appropriate name to the project and rename the untitled default test case to another appropriate title.

**Step-4 :-** Set the ”Playback base URL” to the URL value of the website/webpage you wish to test.

**Step-5 :-** At this point you have two choices where in you can either manually type the test steps in the IDE or let selenium record the steps for you by clicking the ”Record button ON”.

**Step-6 :-** We will first Record the steps using selenium’s Record functionality. On click- ing the ”Record button ON” or keyboard shortcut ”Ctrl-U”.

**Step-7 :-** A new window will pop up with the base URL provided by us earlier , this window is special because selenium will record our mouse clicks and key inputs. *Be careful not to reveal any sensitive information in the record mode.*

**Step-8 :-** Now perform the testing on various elements of the HTML document either by direct clicks/key inputs or by selecting various methods from the ”right-click selenium drop-down menu” which will provide various options for assert various elements and validate various values.

*→*

*→*

**Step-9 :-** You can try out filling forms, dummy login credential test and a lot more.

**Step-10 :-** After you are done with the test script click on ”Stop Recording” button or keyboard shortcut ”Ctrl-U”.

**Step-11 :-** Save the test script by clicking on ”Save project icon” or keyboard shortcut ”Ctrl-S”.

**Step-12 :-** Next save the script in the desired folder path and selenium will download the ”.side” file to the same .

**Step-13 :-** You can reopen the saved Project by clicking ”Open Project icon” or key- board Shortcut ”Ctrl-O” and selecting the ”.side” file from the desired location.

**Step-14 :-** To run the test click on ”Playback icon” button and selenium will run the test as per the script and log all the events in the window. You can verify which steps succeed and which ones fail with Error messages as well.

**Step-15 :-** The test script can be written manually as well and will produce the same results.

**Selenium IDE Features**

1. Menu Bar Menu bar is positioned at the top most portion of the Selenium IDE

interface. It allows the user to change name/open project/save project and more.

Figure

2: Menu Bar 2. ToolBar The toolbar offers options such as play test/record test

and play all the tests in test suite. More advanced features such as

breakpoints/debugger are also available.

3. Test case panel This panel allows the user to add/delete /modify the selenium

commands. The Command, Target, and Value entry fields display the currently

selected command along with its parameters. These are entry fields where you

can modify the currently selected command. The first parameter specified for a

command in the Reference tab of the bottom pane always goes in the Target

field. If a second parameter is specified by the Reference tab, it always goes in the

Value field. The Comment field allows you to specify in a human readable format

a description of the current command.

4. Navigation Panel Navigation between test cases and test suites is done

through the right hand side of Selenium IDE. Clicking on Tests with the small caret

will open up a menu. When saving the project will be saved as the new .side

format, which will include all test cases and suites combined. It is refered to as a

project.

5. Console Panel The bottom pane, called Console Panel for it’s similarity with

web devtools, is used for different utility functions: Log, Reference, depending on

which tab is selected.

**Test Case 1 :**

**Test URL :** https://login.mailchimp.com/

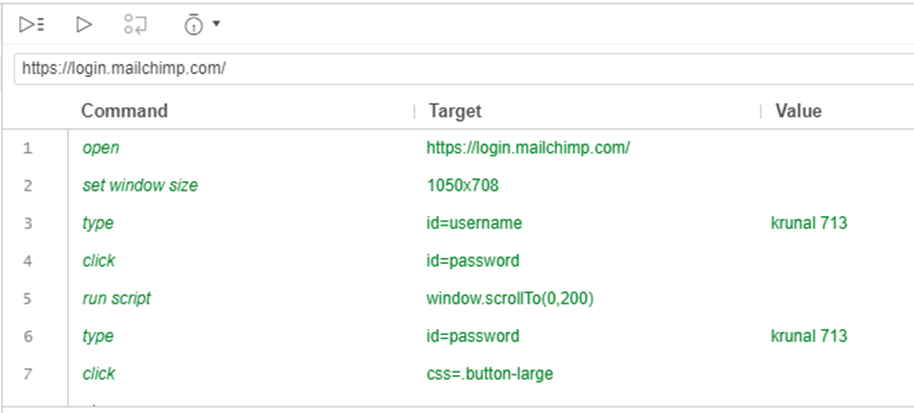
Selenium Commands used :

Command-1 :-set window size

Command-2 :-click

Command-3 :-type

Command-4 :-close



**Test Case 2 :**

**Test URL :** <https://www.shopify.in/tour/ecommerce-website>/

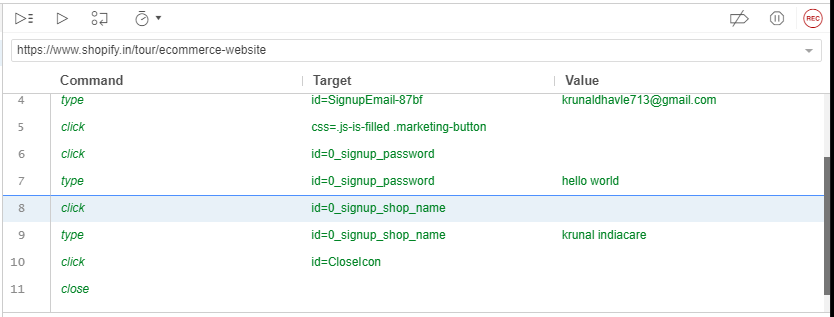
Selenium Commands used :

Command-1 :-open

Command-2 :-type

Command-3 :-click

Command-4 :-close



**Test Case 3 :**

**Test URL :** http://www.google.com/

Selenium Commands used :

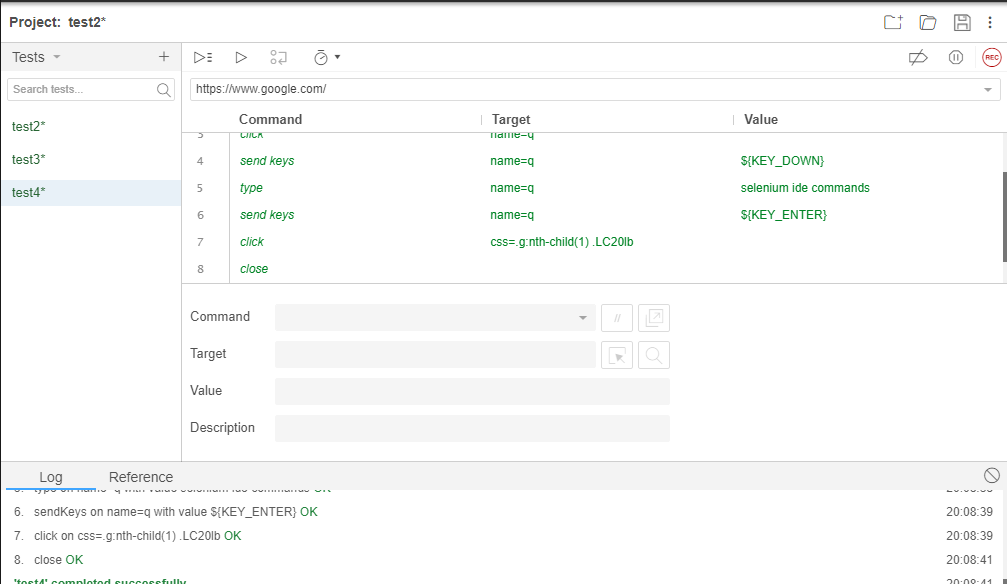
Command-1 :-open

Command-2 :-mouse

Command-3 :-send

Command-4 :-click

Command-5 :-close



**Test Case 4 :**

**Test URL :** https://www.netflix.com/in/Login

Selenium Commands used :

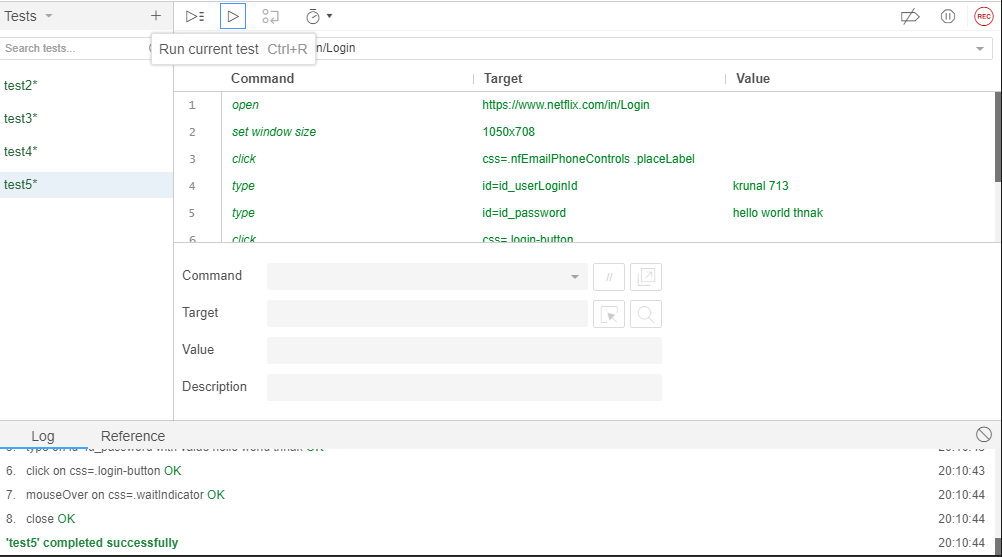
Command-1 :- wait for element

Command-2 : click

Command-3 :- assert checked

Command-4 :- type

Command-5 :- close



Date: 25/08/2020

**Practical No 2**

**AIM:** Install Selenium server and demonstrate it using a script in Java.

**Theory**

**Selenium Java**

Selenium is an umbrella project for a range of tools and libraries that enable and support the automation of web browsers. It provides extensions to emulate user interaction with browsers, a distribution server for scaling browser allocation, and the infrastructure for implementations of the W3C Web-Driver specification that lets you write interchangeable code for all major web browsers.

**ChromeWebDriver**

WebDriver is an open source tool for automated testing of webapps across many browsers.It provides capabilities for navigating to web pages, user input, JavaScript execution,and more. ChromeDriver is a standalone server that implements the W3C WebDriver standard. ChromeDriver is available for Chrome on Android and Chrome on Desktop (Mac, Linux, Windows and ChromeOS).

**Eclipse IDE**

Eclipse IDE is a famous IDE for java development , Eclipse IDE is open source and has been long in development since 2001. It is licensed under ”Eclipse Public License”

Eclipse has a lot of extensions and now has many IDEs for different Programming languages as well.

**Installation of Selenium server in Eclipse IDE**

Step-1 :- Install JDK,Eclipse IDE for ”java developers”.

Step-2 :- Download ”Chromedriver”,”selenium-java-3.13.0.zip” and ”selenium-server- standalone-3.141.59.jar”.

Step-3 :- Launch the Eclipse IDE.

Step-4 :- Create a new java project give a suitable name and save it.

Step-5 :- Eclipse IDE will open the project. Now add a new java class and give it a name.

Step-6 :- To include the libraries and ”jar” files, Unzip the zipped archives,next ”rightclick on the project icon → BuildP ath → Conf igureBuildP ath..”

Step-7 :- Here we will choose ”Add External JARs...” button and navigate to the ”.jar”

file we downloaded/extracted and select them to include in our project. Software Testing & Quality Assurance

Step-8 :- Save the configuration. You have successfully installed selenium for java. Step-9 :- For linking the chrome driver you need to include it in the code by adding a line

System.setProperty("webdriver.chrome.driver","chromedriver path");

**Testing a Website using selenium(java) in eclipsed IDE**

Step-1 :- Import all the important classes we need from the jar packages we included earlier.

import org.openqa.selenium.By;

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

Step-2 :- Next we will set the path property for chromedriver.

Step-3 :- Next Create a WebDriver object for accessing web contents.

Step-4 :- Use ”WebDriver.get()” method to get the base URL .

Step-5 :- we can manipulate the browser DOM elements using ”WebDriver.manage() method”.

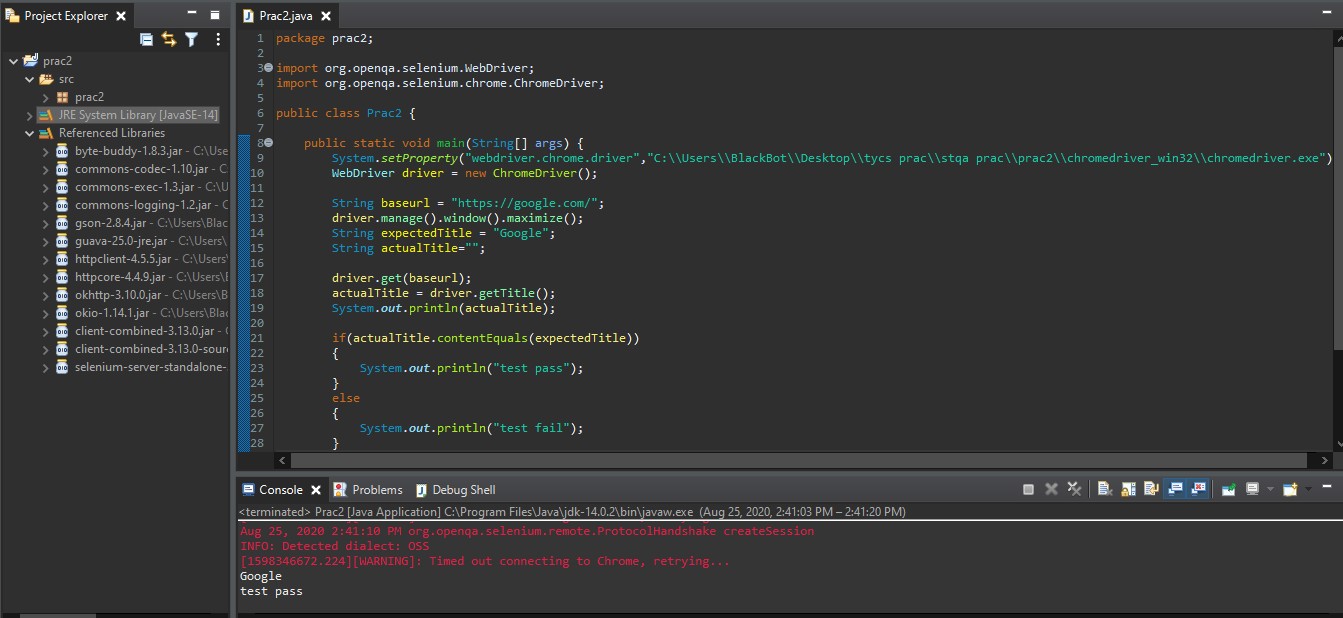
Step-6 :- Next we have a lot of methods to fetch data from the webpage and we can also manipulate them for the ”By” class.

Step-7 :- Once you are done with the testing close the window using ’WebDriver.close()” method.

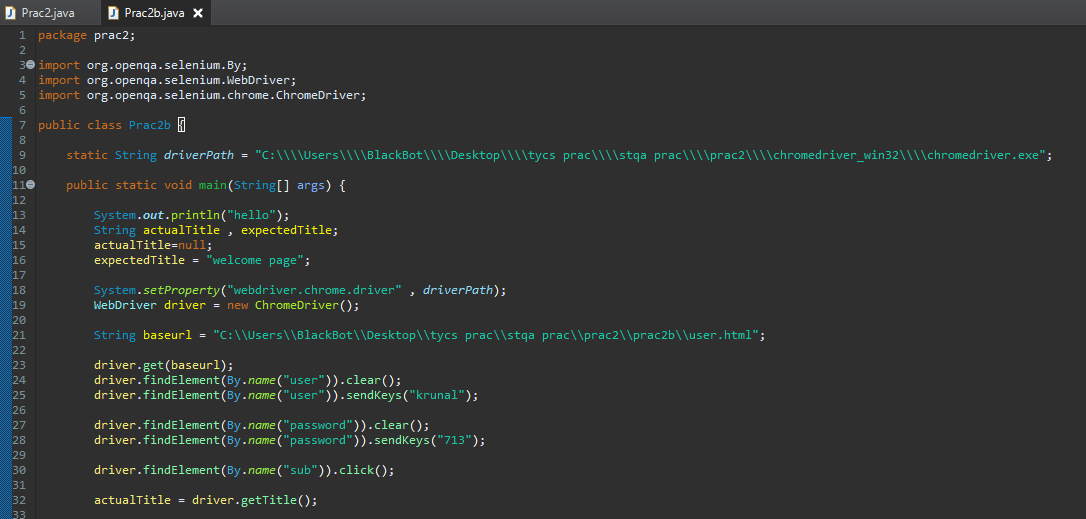
Step-8 :- The base URL can be any site hosted either on localserver or on the world wide web.

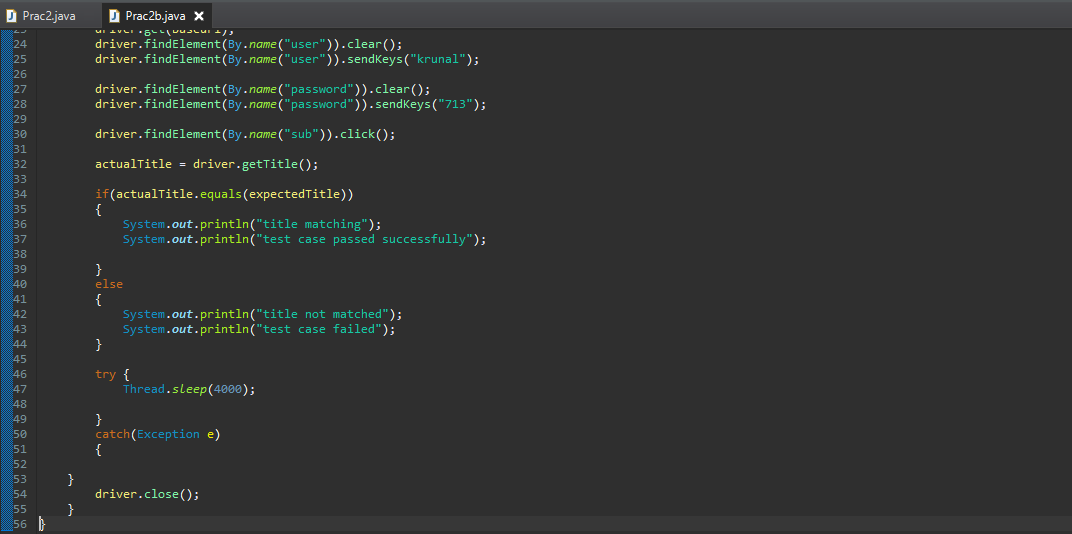
**Code**

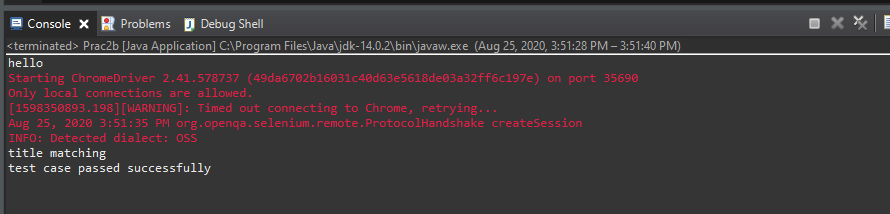
**2A. Test on an External website**



**2B. Test on an Local Webpage**





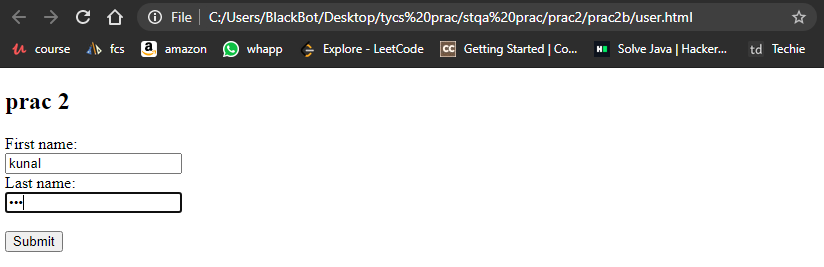


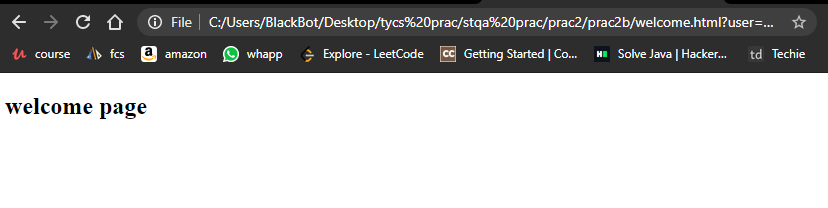
**User.html**

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h2>prac 2 </h2>  <form action="welcome.html">  <label for="fname">First name:</label><br>  <input type="text" id="fname" name="user"><br>  <label for="lname">Last name:</label><br>  <input type="password" id="lname" name="password"><br><br>  <input type="submit" value="Submit" name="sub" >  </form>  </body>  </html> |

**Welcome.html**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> welcome page </title>  </head>  <body>  <h2>welcome page</h2>  </body>  </html> |





Date: 04/09/2020

**Practical no 3**

**AIM:** Write and test a program to login a specific web page using JUnit.

**Theory**

**What is Junit**

The JUnit Platform serves as a foundation for launching testing frameworks on the JVM.

It also defines the TestEngine API for developing a testing framework that runs on the

platform. Furthermore, the platform provides a Console Launcher to launch the platform

from the command line and a JUnit 4 based Runner for running any TestEngine on the

platform in a JUnit 4 based environment. First-class support for the JUnit Platform also

exists in popular IDEs like eclipse.

**Annotations Used in Junit**

**@Test** Denotes that a method is a test method. Unlike JUnit 4’s @Test annotation, this

annotation does not declare any attributes, since test extensions in JUnit Jupiter

operate based on their own dedicated annotations. Such methods are inherited

unless they are overridden.

**@BeforeAll** Denotes that the annotated method should be executed before all @Test,

@RepeatedTest, @ParameterizedTest, and @TestFactory methods in the current

class; analogous to JUnit 4’s @BeforeClass.

**@AfterAll** Denotes that the annotated method should be executed after all @Test,

@RepeatedTest, @ParameterizedTest, and @TestFactory methods in the current

class; analogous to JUnit 4’s @AfterClass.

**@BeforeEach** Denotes that the annotated method should be executed before each

@Test, @RepeatedTest, @ParameterizedTest, or @TestFactory method in the current class; analogous to JUnit 4’s @Before. Such methods are inherited unless they

are overridden.

**@AfterEach** Denotes that the annotated method should be executed after each @Test,

@RepeatedTest, @ParameterizedTest, or @TestFactory method in the current class;

analogous to JUnit 4’s @After. Such methods are inherited unless they are overridden.

**Steps for Testing a site using Junit**

Step-1 :- Create a new Project and give it a name.

Step-2 :- After the initialization create a new package inside the project and give it a

name.

Step-3 :- To include the libraries and ”jar” files, Unzip the zipped archives,next ”right

click on the project icon → BuildP ath → Conf igureBuildP ath..”Step-4 :- Here we will choose ”Add External JARs...” button and navigate to the ”.jar”

file we downloaded/extracted and select them to include in our project.

Step-5 :- Save the configuration. You have successfully installed selenium for java.

Step-6 :- For linking the chrome driver you need to include it in the code by adding a

Line System.setProperty("webdriver.chrome.driver","chromedriver path");

Step-7 :- Next create a new JUnit Test Case file give it a name.

Step-8 :- Import the following libraries

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.WebDriver;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

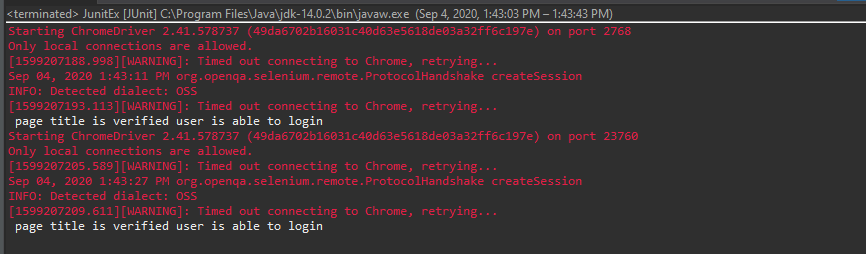
import junit.framework.Assert;

Step-9 :- Add the code.

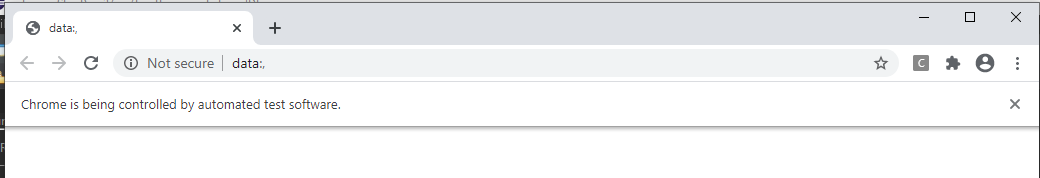
**Code:-**

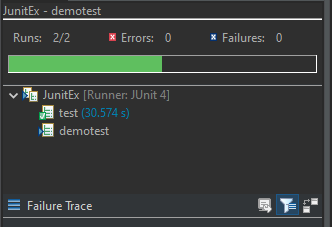
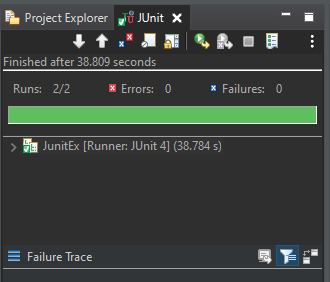
|  |
| --- |
| import static org.junit.Assert.\*;  import org.junit.After;  import org.junit.Assert;  import org.junit.Before;  import org.junit.Test;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  public class JunitEx {  WebDriver driver = null;  @Before  public void setup() {  System.setProperty("webdriver.chrome.driver" , "E:\\tycs\\stqa prac\\prac3\\chromedriver\_win32\\chromedriver.exe" );  driver = new ChromeDriver();  driver.manage().window().maximize();  }    @Test  public void test() throws InterruptedException {    driver.get("http://thedemosite.co.uk/savedata.php");  driver.findElement(By.xpath("//input[@name='username']")).sendKeys("krunal71");  Thread.sleep(1000);  driver.findElement(By.xpath("//input[@name='password']")).sendKeys("kd713");  Thread.sleep(1000);  driver.findElement(By.xpath("//input[@name='FormsButton2']")).click();  Thread.sleep(1000);  Thread.sleep(2000);  Assert.assertTrue("invalid credential" , driver.getTitle().contains("Add a user - FREE PHP code and SQL"));  System.out.println(" page title is verified user is able to login ");  }    @Test  public void demotest() throws InterruptedException {    driver.get("http://demo.guru99.com/test/newtours/");  driver.findElement(By.xpath("//input[@name='userName']")).sendKeys("krunal713");  Thread.sleep(1000);  driver.findElement(By.xpath("//input[@name='password']")).sendKeys("kd713");  Thread.sleep(1000);  driver.findElement(By.xpath("//input[@name='submit']")).click();  Thread.sleep(1000);  Thread.sleep(2000);  Assert.assertTrue("invalid credential" , driver.getTitle().contains("Welcome: Mercury Tours"));  System.out.println(" page title is verified user is able to login ");  }    @After  public void aftertest(){  driver.quit();  }  } |

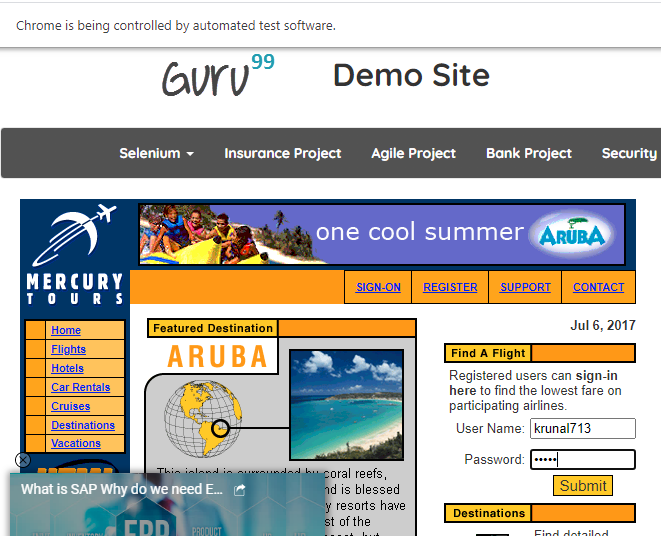
**Output:**









Date:08/09/2020

**Practical no 4**

**AIM:** Write and test a program to update 10 student records into table into Excel file using TestNG.

**Theory**

**What is testNG**

TestNG is a testing framework for the Java programming language created by C´edric

Beust and inspired by JUnit and NUnit. The design goal of TestNG is to cover a wider

range of test categories: unit, functional, end-to-end, integration, etc., with more powerful

and easy-to-use functionalities.

**Annotations in TestNG:-**

**@BeforeSuite:** The annotated method will be run before all tests in this suite have run.

**@AfterSuite**: The annotated method will be run after all tests in this suite have run.

**@BeforeTest**: The annotated method will be run before any test method belonging to

the classes inside the ¡test¿ tag is run.

**@AfterTest**: The annotated method will be run after all the test methods belonging to

the classes inside the ¡test¿ tag have run.

**@Test**: Marks a class or a method as part of the test.

**Jxl Library Functions :**

**Workbook :** Represents a Workbook. Contains the various factory methods and provides a variety of

accessors which provide access to the work sheets.

**WritableWorkbook** : A writable workbook.

**Sheet:**Represents a sheet within a workbook. Provides a handle to the individual cells,

or lines of cells (grouped by Row or Column).

**WriteableSheet;**Interface for a worksheet that may be modified. The most important

modification for a sheet is to have cells added to it

**Label :** A cell containing text which may be created by user applications.

**Steps to install TestNG**

Step-1 :- Create a new Project and give it a name.

Step-2 :- After the initialization create a new package inside the project and give it a

name.

Step-3 :- To include the libraries and ”jar” files, Unzip the zipped archives,next ”right

click on the project icon → BuildP ath → Conf igureBuildP ath..”

Step-4 :- Here we will choose ”Add External JARs...” button and navigate to the ”.jar”

file we downloaded/extracted and select them to include in our project.

Step-5 :- To add TestNG to eclipse navigate to help → InstallNewSof tware...

Step-6 :- Next add the TestNG URL

http://dl.bintray.com/testng-team/testng-eclipse-release/ and

save it as a new source in eclipse.

Step-7 :- Wait for the software repository to fetch all the package data.

Step-8 :- Select TestNG package and all of its sub-packages /dependencies.

Step-9 :- Let the package manager calculate and download the dependencies and requirements.

Step-10 :- Accept the License agreement and finish the installation.

Step-11 :- Next to add a TestNG window to eclipse click on window → showview →

java → T estNG.

Step-12 :- Right click on the project and add a new TestNG class to the project with a

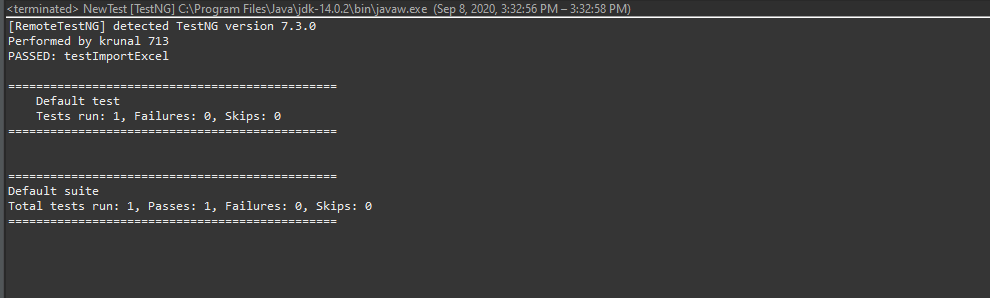
suitable name.

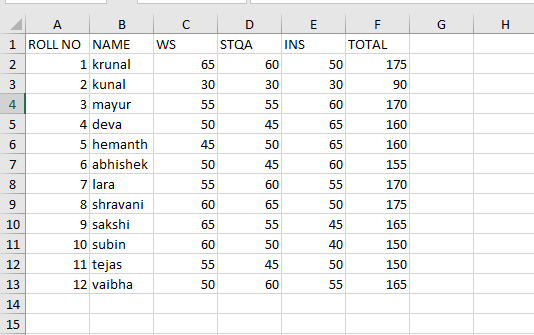
Step-13 :- Add the TestNG Library to your project as well.

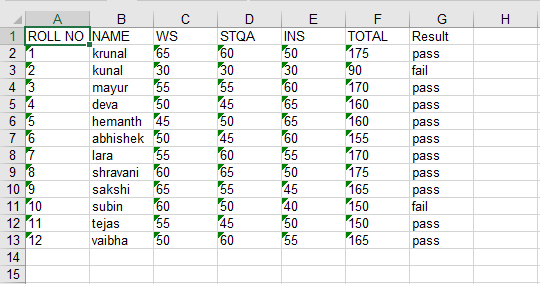
**Code**

|  |
| --- |
| import java.io.FileInputStream;  import java.io.FileOutputStream;  import org.testng.annotations.Test;  import jxl.\*;  import jxl.read.biff.BiffException;  import jxl.write.\*;  import jxl.write.biff.RowsExceededException;  import java.io.\*;  public class NewTest {    @Test  public void testImportExcel() throws IOException, BiffException , RowsExceededException , WriteException  {  FileInputStream fi = new FileInputStream("E:\\tycs\\stqa prac\\prac4\\Student.xls");  Workbook w = Workbook.getWorkbook(fi);  Sheet s = w.getSheet(0);  String a[][] = new String[s.getRows()][s.getColumns()];  FileOutputStream fo = new FileOutputStream("E:\\tycs\\stqa prac\\prac4\\Output.xls");  WritableWorkbook wwb = Workbook.createWorkbook(fo);  WritableSheet ws = wwb.createSheet("result", 0);  System.out.println("Performed by krunal 713");  for (int i = 0; i < s.getRows(); i++)  {  for (int j = 0; j < s.getColumns(); j++)  {  a[i][j] = s.getCell(j , i).getContents();  Label l2 = new Label(j , i ,a[i][j]);  ws.addCell(l2);  Label l1 = new Label(6 , 0 , "Result");  ws.addCell(l1);  }  }  for (int i = 1; i < s.getRows(); i++)  {  for (int j = 2; j < s.getColumns() ; j++)  {  a[i][j] = s.getCell(j , i).getContents();  int x =Integer.parseInt(a[i][j]);  if(x> 40)  {  Label l1 = new Label(6 , i , "pass");  ws.addCell(l1);  }  else  {  Label l1 = new Label(6 , i , "fail");  ws.addCell(l1);  break;  }  }  }  wwb.write();  wwb.close();  }  } |

**Output:**







Date:- 09/10/2020

**Practical no 5**

**AIM:** Write and test a program to select the number of students who have scored more than 60 in any one subject

**Theory**

**What is testNG**

TestNG is a testing framework for the Java programming language created by C´edric

Beust and inspired by JUnit and NUnit. The design goal of TestNG is to cover a wider

range of test categories: unit, functional, end-to-end, integration, etc., with more powerful

and easy-to-use functionalities.

**Annotations in TestNG**

**@BeforeSuite:** The annotated method will be run before all tests in this suite have run.

**@AfterSuite**: The annotated method will be run after all tests in this suite have run.

**@BeforeTest**: The annotated method will be run before any test method belonging to

the classes inside the ¡test¿ tag is run.

**@AfterTest**: The annotated method will be run after all the test methods belonging to

the classes inside the ¡test¿ tag have run.

**@Test**: Marks a class or a method as part of the test.

**Jxl Library Functions :**

**Workbook :** Represents a Workbook. Contains the various factory methods and provides a variety of accessors which provide access to the work sheets.

**WritableWorkbook** : A writable workbook.

**Sheet:** Represents a sheet within a workbook. Provides a handle to the individual cells,

or lines of cells (grouped by Row or Column).

**WriteableSheet :** Interface for a worksheet that may be modified. The most important

modification for a sheet is to have cells added to it

**Label :** A cell containing text which may be created by user applications.

**Steps to install TestNG**

Step-1 :- Create a new Project and give it a name.

Step-2 :- After the initialization create a new package inside the project and give it a

name.

Step-3 :- To include the libraries and ”jar” files, Unzip the zipped archives,next ”right

click on the project icon → BuildP ath → Conf igureBuildP ath..”

Step-4 :- Here we will choose ”Add External JARs...” button and navigate to the ”.jar”

file we downloaded/extracted and select them to include in our project.

Step-5 :- To add TestNG to eclipse navigate to help → InstallNewSof tware...

Step-6 :- Next add the TestNG URL

http://dl.bintray.com/testng-team/testng-eclipse-release/ and

save it as a new source in eclipse.

Step-7 :- Wait for the software repository to fetch all the package data.

Step-8 :- Select TestNG package and all of its sub-packages /dependencies.

Step-9 :- Let the package manager calculate and download the dependencies and requirements.

Step-10 :- Accept the License agreement and finish the installation.

Step-11 :- Next to add a TestNG window to eclipse click on window → showview →

java → T estNG.

Step-12 :- Right click on the project and add a new TestNG class to the project with a

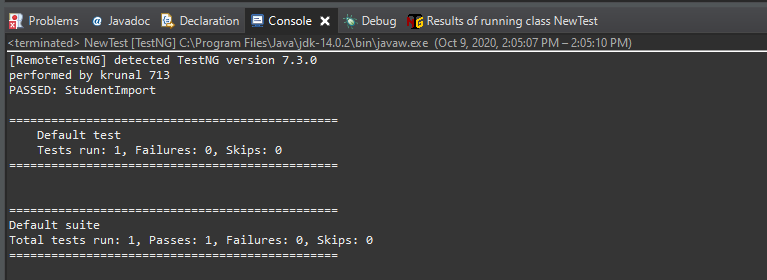
suitable name.

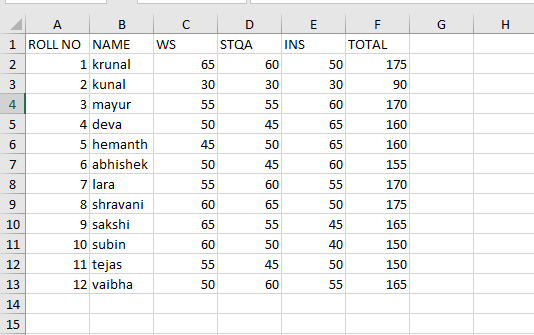
Step-13 :- Add the TestNG Library to your project as well.

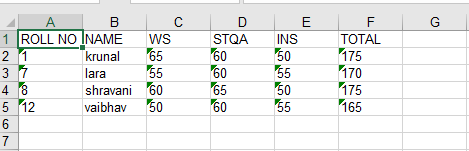
**Code** :-

|  |
| --- |
| import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  import org.testng.annotations.Test;  import jxl.Sheet;  import jxl.Workbook;  import jxl.read.biff.BiffException;  import jxl.write.Label;  import jxl.write.WritableSheet;  import jxl.write.WritableWorkbook;  import jxl.write.WriteException;  import jxl.write.biff.RowsExceededException;  public class NewTest {  @Test  public void StudentImport() throws IOException , BiffException , RowsExceededException , WriteException  {  FileInputStream fi = new FileInputStream("E:\\tycs\\stqa prac\\prac5\\Student.xls");  Workbook w = Workbook.getWorkbook(fi);  Sheet s = w.getSheet(0);  String a[][] = new String[s.getRows()][s.getColumns()];  FileOutputStream fo = new FileOutputStream("E:\\tycs\\stqa prac\\prac5\\studentdata.xls ");  WritableWorkbook wwb = Workbook.createWorkbook(fo);  WritableSheet ws = wwb.createSheet("result", 0);  int c=0;  System.out.println("performed by krunal 713");  for (int i = 0; i < s.getRows(); i++)  {  for (int j = 0; j < s.getColumns();  {  if(i >= 1)  {  String b= new String();  b = s.getCell(3,i).getContents();  int x= Integer.parseInt(b);  if( x < 60)  {  c++;  break;  }  }  a[i][j] = s.getCell(j, i).getContents();  Label l2 = new Label(j, i-c, a[i][j]);  ws.addCell(l2);  }  }  wwb.write();  wwb.close();  }  } |

**OUTPUT:**







**Part B:-**

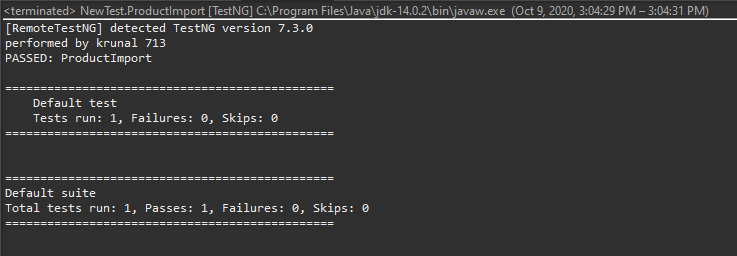
**Aim:-** Write and test a program to read the data from product excel sheet and select the products which

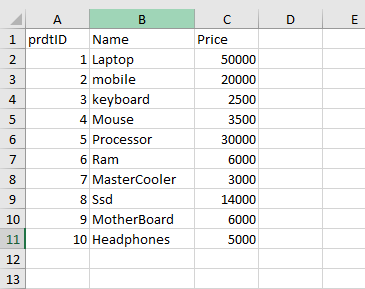
price is more than 10000

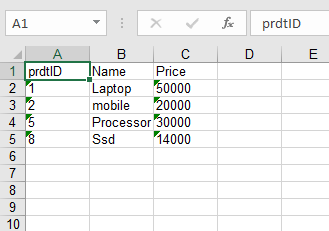
**Code:-**

|  |
| --- |
| import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  import org.testng.annotations.Test;  import jxl.Sheet;  import jxl.Workbook;  import jxl.read.biff.BiffException;  import jxl.write.Label;  import jxl.write.WritableSheet;  import jxl.write.WritableWorkbook;  import jxl.write.WriteException;  import jxl.write.biff.RowsExceededException;  public class NewTest {  @Test  public void ProductImport() throws IOException , BiffException , RowsExceededException , WriteException  {  FileInputStream fi = new FileInputStream("E:\\tycs\\stqa prac\\prac5\\product.xls");  Workbook w = Workbook.getWorkbook(fi);  Sheet s = w.getSheet(0);  String a[][] = new String[s.getRows()][s.getColumns()];  FileOutputStream fo = new FileOutputStream("E:\\tycs\\stqa prac\\prac5\\productdata.xls ");  WritableWorkbook wwb = Workbook.createWorkbook(fo);  WritableSheet ws = wwb.createSheet("result", 0);  int c=0;  System.out.println("performed by krunal 713");  for (int i = 0; i < s.getRows(); i++)  {  for (int j = 0; j < s.getColumns();  {  if(i >= 1)  {  String b= new String();  b = s.getCell(3,i).getContents();  int x= Integer.parseInt(b);  if( x < 60)  {  c++;  break;  }  }  a[i][j] = s.getCell(j, i).getContents();  Label l2 = new Label(j, i-c, a[i][j]);  ws.addCell(l2);  }  }  wwb.write();  wwb.close();  }  } |

**Output:**







Date:17/10/2020

**Practical no 6**

**AIM:** Write and test a program to provide total number of objects present / available on the page

**Theory**

**WebElement** : Represents an HTML element. Generally, all interesting operations to

do with interacting with a page will be performed through this interface.

WebElements objects are Selenium(Java) equivalent of HTML elements in the UI such

as Form, Button, Select, A, P, H0-6, etc..

**java.util.List<WebElement>** : This data structure allows one two list down WebElements in Java which could be sub-elements or nested elements under the selector. The object has methods such as object.size() which returns the size of the list,

object.get(index) which returns a element from the object at a specified index, object.get(index).getText() returns the text of the element.

**WebDriver.findElement():** Find the first WebElement using the given method.

This method is affected by the ’implicit wait’ times in force at the time of execution. The

findElement(..) invocation will return a matching row, or try again repeatedly until the

configured timeout is reached. findElement should not be used to look for non-present

elements, use findElements(By) and assert zero length response instead.

**By.tagName()** : By is a mechanism used to locate elements within a document from

selenium code. This allows us to access elements by multiple ways such as tag names, ids,

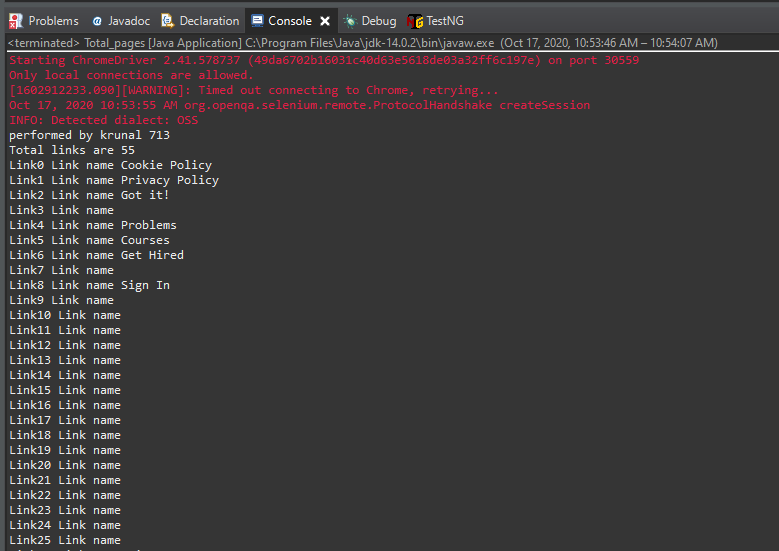
names, class names, etc.. Here we use By.tagName() to select HTML elements through

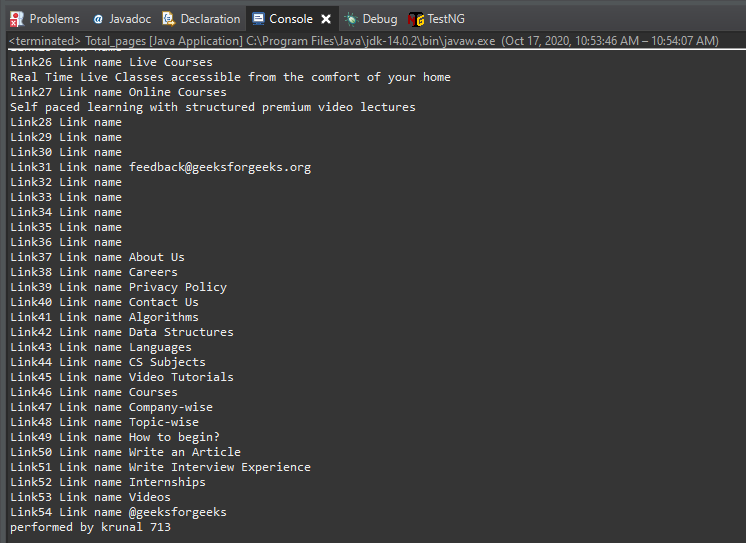
specified tag name.

**Code:**

|  |
| --- |
| package stqapracccc;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.WebElement;  public class Total\_pages {    static String *drivePath* = "E:\\tycs\\stqa prac\\prac2\\chromedriver\_win32\\chromedriver.exe";  public static WebDriver *driver*;  public static void main(String[] args) {  System.*setProperty*("webdriver.chrome.driver" , *drivePath*);  *driver* = new ChromeDriver();  *driver*.get("https://practice.geeksforgeeks.org/");  java.util.List<WebElement> link = *driver*.findElements(By.*tagName*("a"));  System.***out***.println("performed by krunal 713");  System.***out***.println("Total links are " + link.size());  for(int i =0; i < link.size() ; i++)  {  System.***out***.println("Link" + i +" Link name "+ link.get(i).getText());  }  System.***out***.println("performed by krunal 713");  *driver*.close();  }  } |

**Output:-**

****



Date:17/10/2020

**Practical no 7**

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

**Theory**

**WebElement** : Represents an HTML element. Generally, all interesting operations to do with interacting with a page will be performed through this interface. WebElements objects are Selenium(Java) equivalent of HTML elements in the UI such as Form, Button, Select, A, P, H0-6, etc..

**org.openqa.selenium.support.ui.Select** : Models a SELECT tag, providing helper methods to select and deselect options.

**java.util.List** : This data structure allows one two list down WebElements in Java which could be sub-elements or nested elements under the selector. The object has methods such as **object.size()** which returns the size of the list, **Selectobject.getOptions()** which returns a select option element from the object.

**Select.selectByVisibleText()** : Select all options that display text matching the argument. That is, when given ”Bar” this would select an option like: Bar .

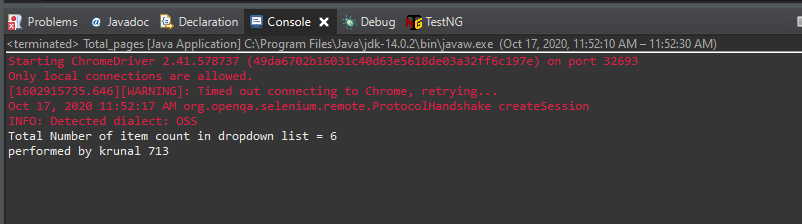
**Select.selectByIndex():** Select the option at the given index. This is done by examining the ”index” attribute of an element, and not merely by counting.

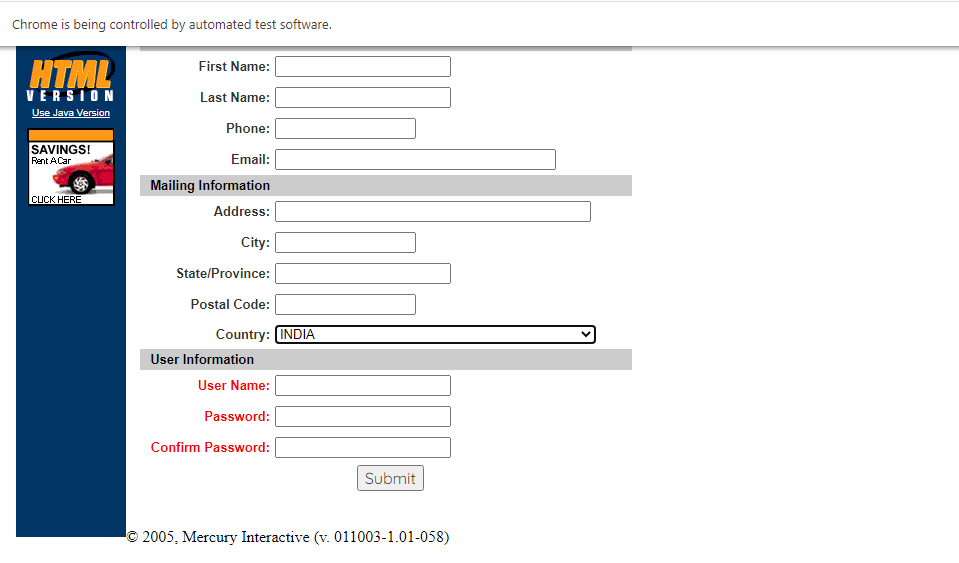
**Code:**

**Part A**

|  |
| --- |
| **package** prac77;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.chrome.ChromeDriver;  **import** org.openqa.selenium.support.ui.Select;  **import** java.util.List;  **import** org.openqa.selenium.WebElement;  **public** **class** Prac7 {  **static** String *driverPath*="E:\\tycs\\stqa prac\\prac2\\chromedriver\_win32\\chromedriver.exe";  **public** **static** **void** main(String[] args) {  System.*setProperty*("webdriver.chrome.driver", *driverPath*);  WebDriver driver= **new** ChromeDriver();  driver.get("E:/tycs/stqa%20prac/prac7/index.html");  Select selectDropdown = **new** Select(driver.findElement(By.*id*("bikes")));  List<WebElement> listOptionDropdown = selectDropdown.getOptions();  **int** dropdownCount = listOptionDropdown.size();  System.***out***.println("Total Number of item count in dropdown list = " + dropdownCount);  driver.close();  System.***out***.println("performed by krunal 713");  }  } |

**Output:-**

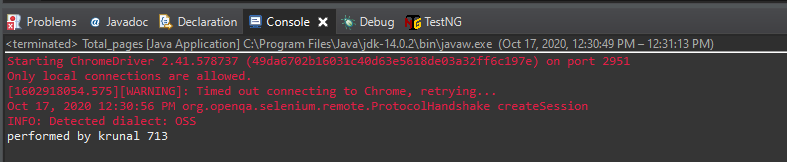


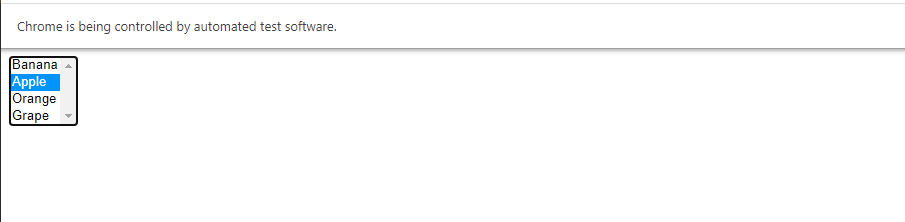
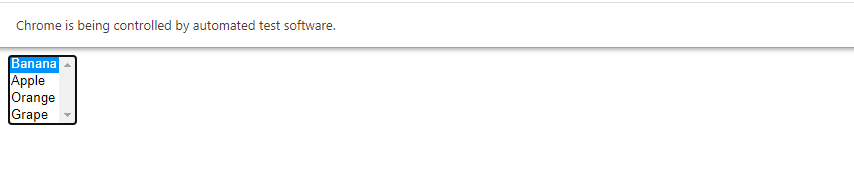


**Part-B**

|  |
| --- |
| **package** stqapracccc;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.chrome.ChromeDriver;  **import** org.openqa.selenium.support.ui.Select;  **public** **class** Total\_pages {  **static** String *driverPath*="E:\\tycs\\stqa prac\\prac2\\chromedriver\_win32\\chromedriver.exe";  **public** **static** **void** main(String[] args) **throws** InterruptedException {  //----PART2A ---------//  System.*setProperty*("webdriver.chrome.driver", *driverPath*);    String baseUrl = "http://demo.guru99.com/test/newtours/register.php";  WebDriver driver= **new** ChromeDriver();  driver.get(baseUrl);  Select drpCountry = **new** Select(driver.findElement(By.*name*("country")));  drpCountry.selectByVisibleText("INDIA");  Thread.*sleep*(8000);    //----PART2B ---------//  driver.get("http://jsbin.com/osebed/2");  Select fruits = **new** Select(driver.findElement(By.*id*("fruits")));  fruits.selectByVisibleText("Banana");  fruits.deselectByIndex(1);  Thread.*sleep*(8000);  driver.close();  System.***out***.println("performed by krunal 713");  }  } |

**Output**





Date:20/10/2020

**Practical no 8**

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

**Theory :**

**Xpath**

XPath stands for XML Path Language. It uses a non-XML syntax to provide a flexible

way of addressing (pointing to) different parts of an XML document. It can also be used

to test addressed nodes within a document to determine whether they match a pattern or not.

XPath is mainly used in XSLT, but can also be used as a much more powerful way of

navigating through the DOM of any XML-like language document using XPathExpression, such

as HTML and SVG, instead of relying on the Document.getElementById() or

ParentNode.querySelectorAll() methods, the Node.childNodes properties, and other

DOM Core features.

XPath uses a path notation (as in URLs) for navigating through the hierarchical

structure of an XML document. It uses a non-XML syntax so that it can be used in

URIs and XML attribute values.

**Xpath Syntax**

XPath contains the path of the element situated at the web page. Standard syntax for

creating XPath is.

Xpath = //tagname[@attribute =0 value0 ]

|  |  |
| --- | --- |
| Literal | Description |
| // : | Select current node. |
| Tagname: | Tagname of the particular node. |
| @: | Select attribute. |
| Attribute: | Attribute name of the node |
| Value: | Value of the attribute. |

**Types of Xpath**

There are two types of xpath:

**1. Absolute xpath** : It is the direct way to find the element, but the disadvantage of the absolute XPath is

that if there are any changes made in the path of the element then that XPath gets failed. The key

characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the

element from the root node.

**Example:** /html/body/div[2]/div[1]/div/h4[1]/b/html[1]/body[1]/div[2]/div[1]/div[1]/h4[1]/b[1]

**2. Relative xpath :** Relative Xpath starts from the middle of HTML DOM structure. It starts with double

forward slash (//). It can search elements anywhere on the webpage, means no need to write a long xpath

and you can start from the middle of HTML DOM structure. Relative Xpath is always preferred as it is not

a complete path from the root element. Below is the example of a relative XPath expression of the same

element shown in the below screen. This is the common format used to find element through a relative

XPath.

**Example**: //div[@class =0 featured − boxcloumnsize1 0 ]//h4[1]//b[1]

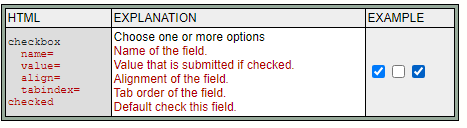
**Code:**

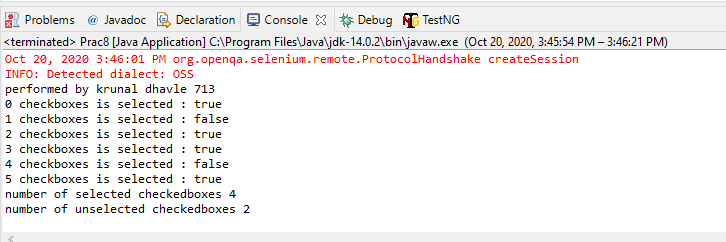
**Part A**

|  |
| --- |
| **import** java.util.List;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;  **public** **class** Prac8 {  **static** String *driverPath*="E:\\tycs\\stqa prac\\prac2\\chromedriver\_win32\\chromedriver.exe";  **public** **static** **void** main(String[] args) **throws** InterruptedException {  System.*setProperty*("webdriver.chrome.driver", *driverPath*);  WebDriver driver= **new** ChromeDriver();  //driver.get("http://www.ironspider.ca/forms/checkradio.htm");  driver.get("http://www.echoecho.com/htmlforms09.htm");  //driver.get("file:///E:/tycs/stqa%20prac/prac8/radio.html");  List<WebElement> checkboxes = driver.findElements(By.*xpath*("//input[@type = 'checkbox']"));  **for**(**int** i = 0 ; i<checkboxes.size() ; i=i+1)  {  checkboxes.get(i).click();  }  System.***out***.println("performed by krunal dhavle 713");  **int** checkedCount = 0 , uncheckedCount =0;  **for**(**int** i =0 ; i < checkboxes.size() ; i++)  {  System.***out***.println(i + " " + "checkboxes is selected : "+checkboxes.get(i).isSelected());  **if**(checkboxes.get(i).isSelected())  checkedCount++;  **else**  uncheckedCount++;  }  Thread.*sleep*(5000);  System.***out***.println("number of selected checkedboxes " + checkedCount);  System.***out***.println("number of unselected checkedboxes " + uncheckedCount);  driver.close();  }  } |

**Output:-**

****

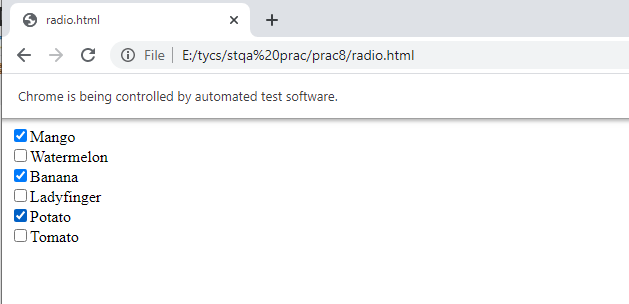


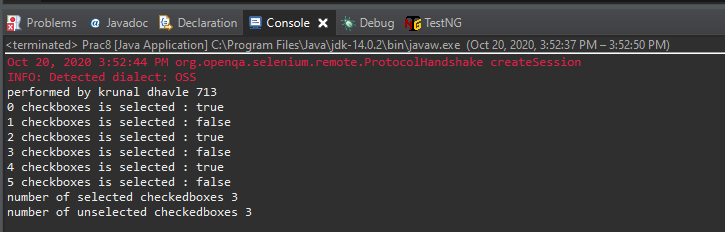


**Part -2**

|  |
| --- |
| **import** java.util.List;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;  **public** **class** Prac8 {  **static** String *driverPath*="E:\\tycs\\stqa prac\\prac2\\chromedriver\_win32\\chromedriver.exe";  **public** **static** **void** main(String[] args) **throws** InterruptedException {  System.*setProperty*("webdriver.chrome.driver", *driverPath*);  WebDriver driver= **new** ChromeDriver();  driver.get("file:///E:/tycs/stqa%20prac/prac8/radio.html");  List<WebElement> checkboxes = driver.findElements(By.*xpath*("//input[@type = 'checkbox']"));  **for**(**int** i = 0 ; i<checkboxes.size() ; i=i+2)  {  checkboxes.get(i).click();  }  System.***out***.println("performed by krunal dhavle 713");  **int** checkedCount = 0 , uncheckedCount =0;  **for**(**int** i =0 ; i < checkboxes.size() ; i++)  {  System.***out***.println(i + " " + "checkboxes is selected : "+checkboxes.get(i).isSelected());  **if**(checkboxes.get(i).isSelected())  checkedCount++;  **else**  uncheckedCount++;  }  Thread.*sleep*(5000);  System.***out***.println("number of selected checkedboxes " + checkedCount);  System.***out***.println("number of unselected checkedboxes " + uncheckedCount);  driver.close();  }  } |

**Output:-**





Date:03/11/2020

**Practical no 9**

**AIM:** Load Testing using JMeter.

**Theory:-**

**What is JMeter ?**

Apache JMeter may be used to test performance both on static and dynamic resources,

Web dynamic applications. It can be used to simulate a heavy load on a server, group

of servers, network or object to test its strength or to analyze overall performance under

different load types.

**1. Thread Group :** Thread group elements are the beginning points of any test plan.

All controllers and samplers must be under a thread group. Other elements, e.g.

Listeners, may be placed directly under the test plan, in which case they will apply

to all the thread groups. As the name implies, the thread group element controls

the number of threads JMeter will use to execute your test. The controls for a

thread group allow you to:

**1.1 Number of Threads** : Each thread will execute the test plan in its entirety

and completely independently of other test threads. Multiple threads are used

to simulate concurrent connections to your server application.

**1.2 Ramp-up Period** : The ramp-up period tells JMeter how long to take to

”ramp-up” to the full number of threads chosen. If 10 threads are used, and

the ramp-up period is 100 seconds, then JMeter will take 100 seconds to get all

10 threads up and running. Each thread will start 10 (100/10) seconds after

the previous thread was begun. If there are 30 threads and a ramp-up period

of 120 seconds, then each successive thread will be delayed by 4 seconds.

**1.3 Loops :** By default, the thread group is configured to loop once through its

elements. However one can change it repeat the tests.

**2. Controllers** : JMeter has two types of Controllers: Samplers and Logical Controllers.

These drive the processing of a test.

**2.1 Samplers** : Samplers tell JMeter to send requests to a server and wait for a

response. They are processed in the order they appear in the tree. Controllers

can be used to modify the number of repetitions of a sampler. ”HTTP Request” is one of those samplers which we use to interact with HTTP protocol

to our server. **2.2 Logic Controllers :** Logic Controllers let you customize the logic that JMeter

uses to decide when to send requests. Logic Controllers can change the order

of requests coming from their child elements. They can modify the requests

themselves, cause JMeter to repeat requests, etc

**3. Listeners** : Listeners provide access to the information JMeter gathers about the test

cases while JMeter runs. The Graph Results listener plots the response times on

a graph. The ”View Results Tree” Listener shows details of sampler requests and

responses, and can display basic HTML and XML representations of the response.

Other listeners provide summary or aggregation information.

**3.1 Results Tree** : The View Results Tree shows a tree of all sample responses,

allowing you to view the response for any sample. In addition to showing the

response, you can see the time it took to get this response, and some response

codes. Note that the Request panel only shows the headers added by JMeter.

It does not show any headers (such as Host) that may be added by the HTTP

protocol implementation.

**3.2 Graph Results** : The Graph Results listener generates a simple graph that

plots all sample times. Along the bottom of the graph, the current sample (black), the

current average of all samples (blue), the current standard

deviation (red), and the current throughput rate (green) are displayed in milliseconds.

**Pre-Installation Requirements**

**JDK/JRE Installation is required(Java 8 or higher)**

Running the batch file may not work directly so one needs to add path of the java

bin folder through the terminal and then try running the batch file.

For Linux use OpenJDK package(latest available) and mark the jar file as executable

through file properties or use chmod to set execution permission.

**Steps:**

**Step-1** :- Download and Unzip Jmeter package from Apache Website.

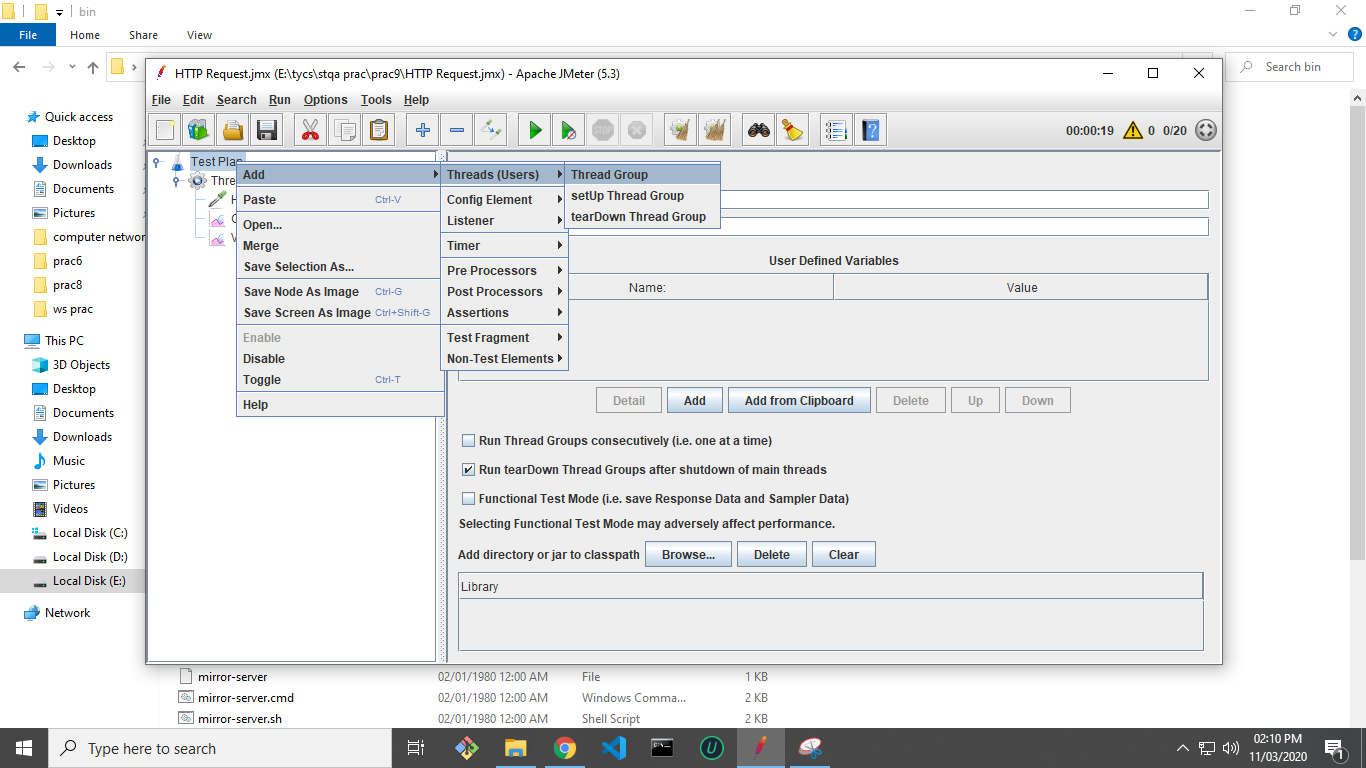
<https://jmeter.apache.org/download_jmeter.cgi>

**Step-2** :- Navigate to bin folder and run the ”ApacheJMeter.jar” or ”jmeter.bat” file

and it will launch a window. In case of error/failure refer the Pre-Installation Requirements.

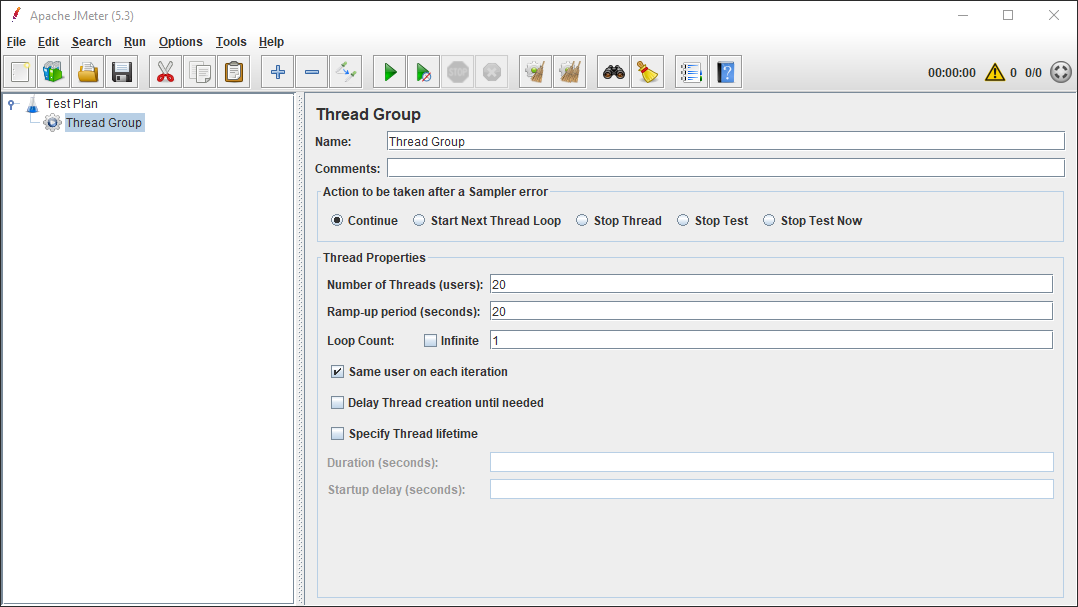
**Step-3** :- Now First step is to add an Thread Group, RightclickonT estPlan →

Add → T hreads(Users) → T hreadGroup



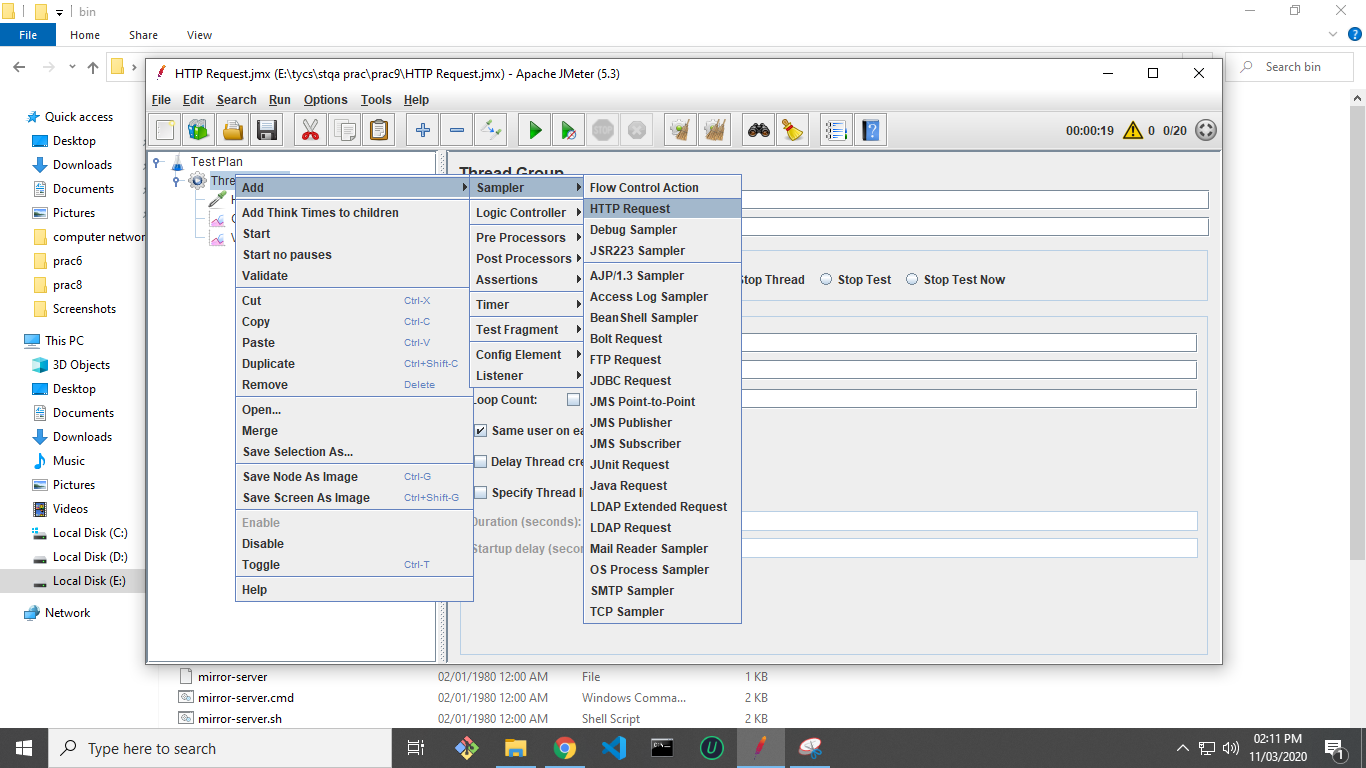
**Step-4 :-** Next rename the Thread Group and set ”Number of Threads (Users):”

as 20 and ”Ramp-up period(seconds):” as 20. Keep rest options as default.



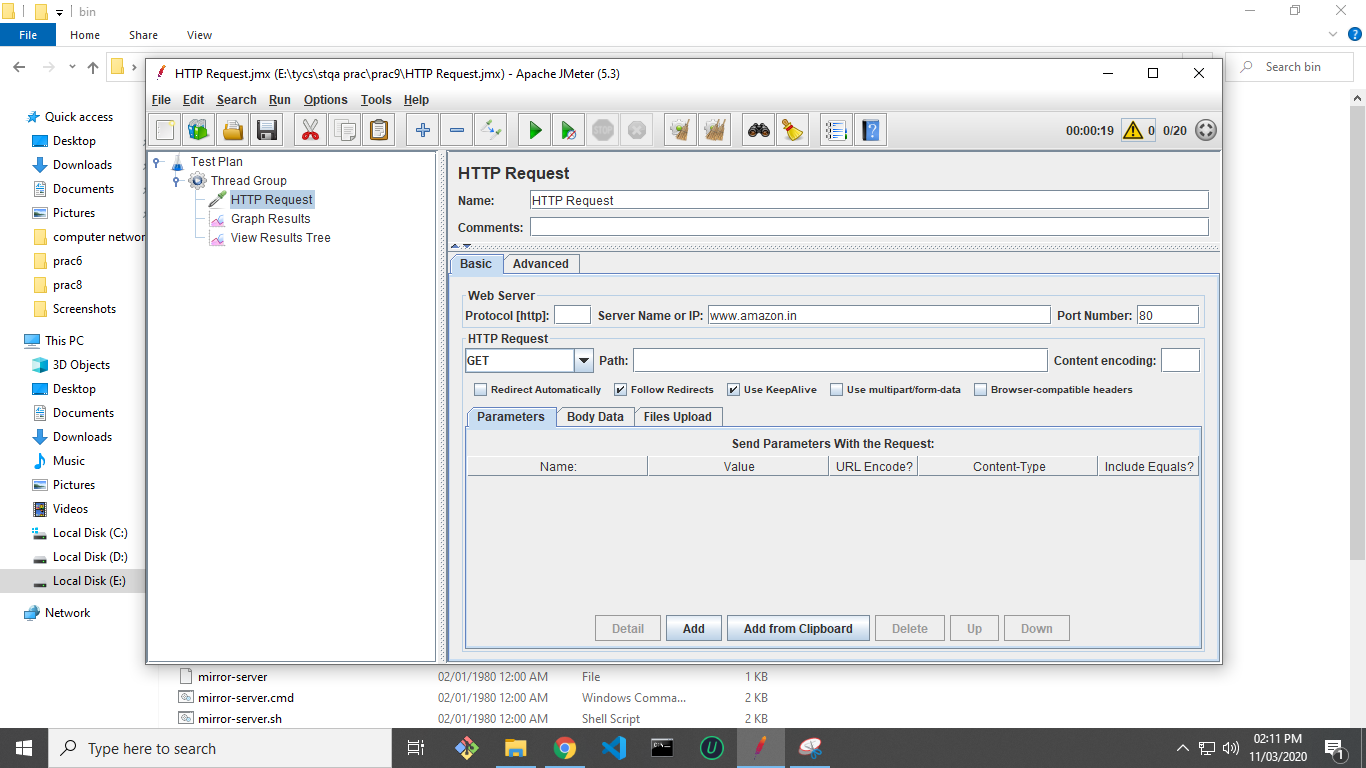
**Step-5** :- We add a HTTP Request Sampler to the Thread Group Rightclick T hreadGroup →

Add → Sampler → HT T P Request



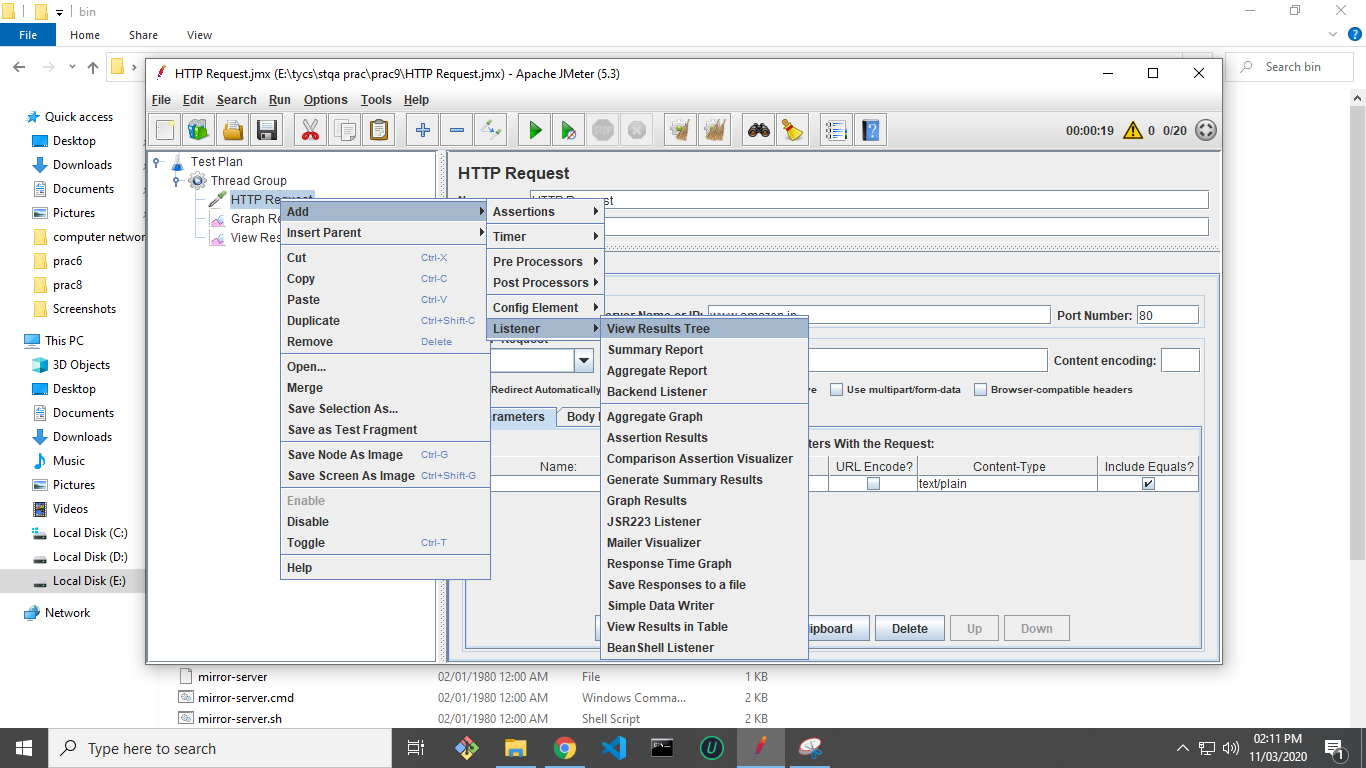
**Step-6** :- Next, Edit the ”Server Name or IP:” to the specific website or Server IP address, optionally one

configure port number and Request parameters and path as well.



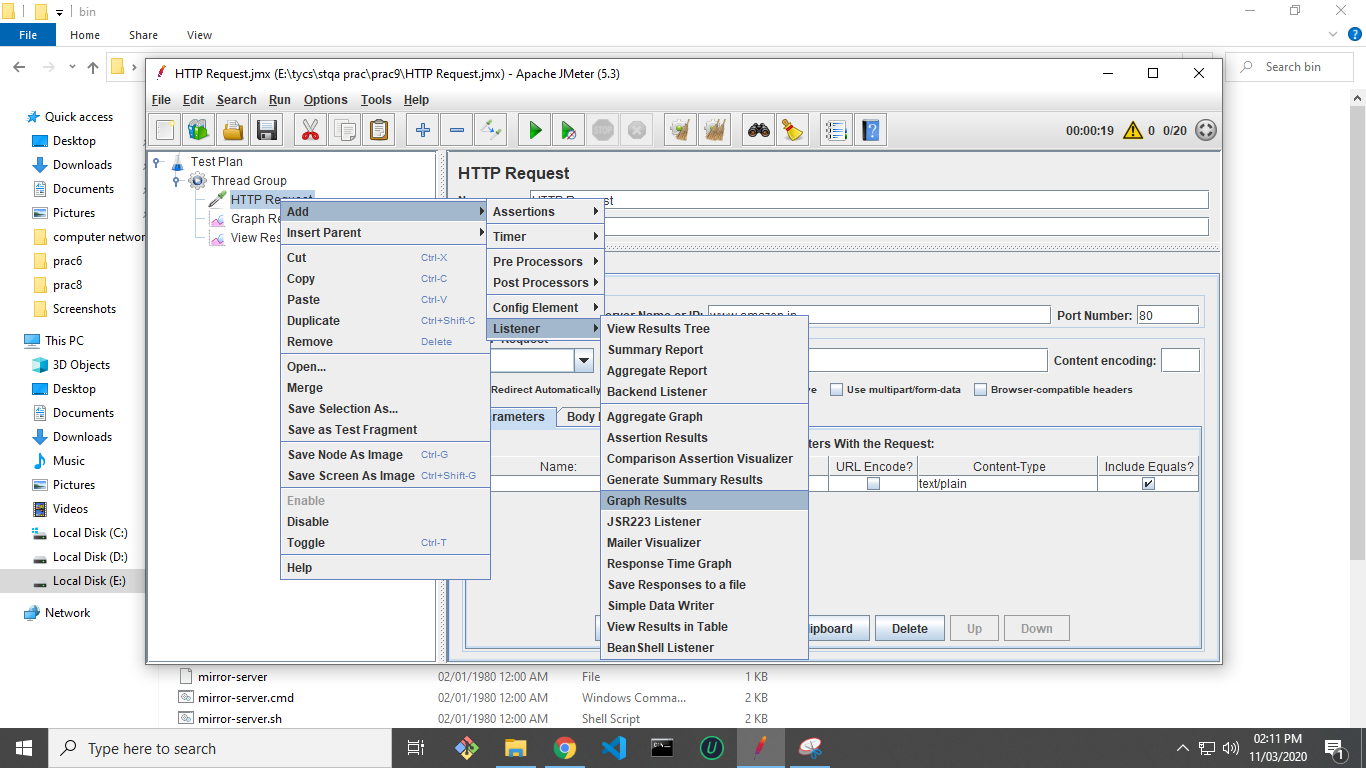
**Step-7 :-** To view the log data add a ”View Results Tree Listener” Rightclick T estPlan → Add

→ Listener → ViewResultsTree.



**Step-8 :-** To visualize the data we add a ”Graph Results Listener” Rightclick T estPlan →

Add → Listener → GraphResults

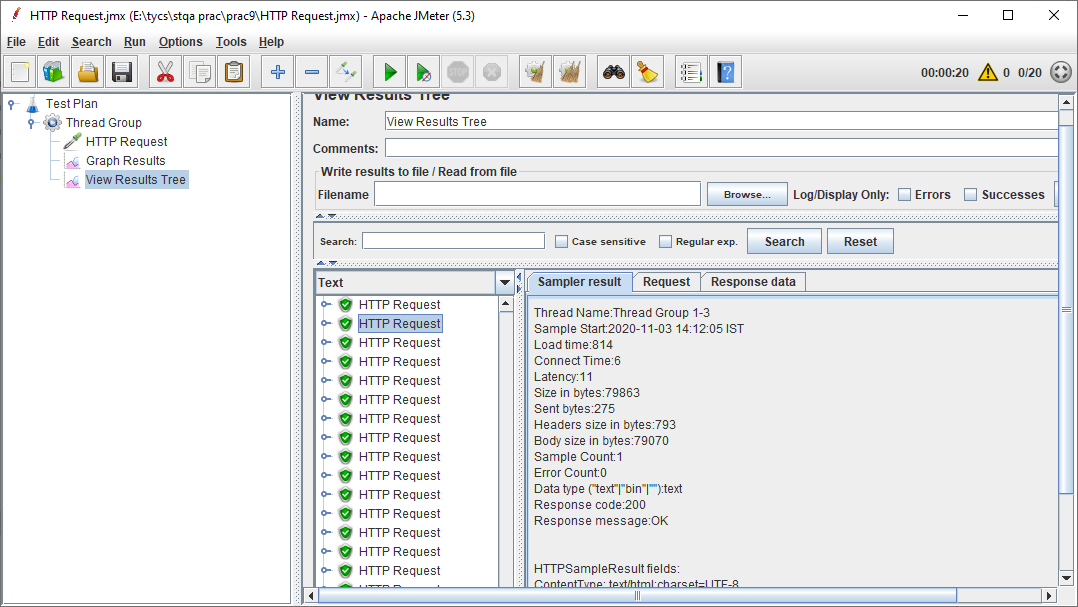


**Step-9 :-** Next save the project as some filename into the hard drive by clicking Ctrl-S.

**Step-10** :- Now Run the tests by clicking on the Green Play Button .

**Step-11** :- After the test completes successfully you can view the logged results of the test in ”View

Results Tree”



**Step-12** :- One can also visualize the parameters by looking at the ”Graph Results”.

