ERODE SENGUNTHAR
ENGINEERING COLLEGE

(APPROVED BY AICTE, NEW DELHI & PERMANENTLY AFFILIATED TO ANNA UNIVERSITY, CHENNAI.
ACCREDITED BY NBA, NEW DELHI, NAAC WITH GRADE "A" & IE(I),KOLKATA)

PERUNDURAI, ERODE – 638 057

An Autonomous Institution

BONAFIDE CERTIFICATE

Register No: .....

Certified that this is the Bonafide Record of Work Done By

Name of the Student :.....

Branch :.....

Lab Code/Name :.....

Year/Semester :.....

Faculty Incharge Head of the Department

Submitted for the End Semester Practical Examination held on .....

Internal Examiner *External Examiner*

Ex.No:1	Understand The Automation Testing Approach
Date:	

Automation: Transforming Efficiency in Modern Workflows

Automation is the process of making tasks self-controlling and self-moving, eliminating the need for human intervention. It streamlines operations, increasing efficiency, consistency, and productivity across various domains, from IT and business processes to manufacturing and service industries.

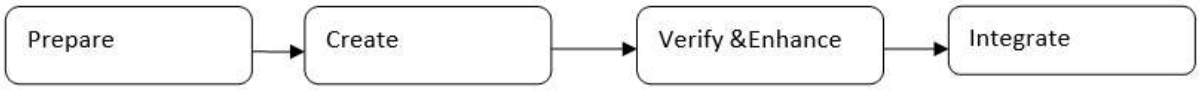
Types of Automation

1. **Fixed Automation:** Best for high-volume production, such as assembly lines.
2. **Programmable Automation:** Allows adjustments for different processes, commonly used in batch production.
3. **Robotic Process Automation (RPA):** Software robots handling repetitive digital tasks in business operations.
4. **Cognitive Automation:** Uses AI and machine learning for intelligent decision-making beyond simple task execution.

Benefits of Automation

- **Speed:** Automates tasks at a much faster rate than human execution.
- **Reliability:** Ensures accuracy with minimal risk of human error.
- **Repeatability:** Allows the same task to be executed identically every time.
- **Programmability:** Can be tailored for specific functions across different industries.
- **Reusability:** Automation scripts or processes can be adapted for multiple applications.
- **Improved Testing:** Makes regression testing more efficient and enables continuous testing cycles (e.g., 24/7 testing).
- **Robust Verification:** Ensures quality control with automated validation and monitoring.

Automation Test Workflow



Create Basic Test
Record user actions Confirm successful playback

Enhance Basic Test
Add Synchronization Insert check points Data drive the test

Integrate Tests
Pass Data Build integrated test sets

Introduction to Selenium

1. History of Selenium

Selenium was invented in 2004 by **Jason R. Huggins** and his team while they were working at ThoughtWorks. The tool was initially named **JavaScript Functional Tester (JSFT)**.

Selenium is an open-source **browser-based integration testing framework** originally developed to automate web application testing. It is built entirely using **JavaScript and HTML** and provides comprehensive support for testing **Web 2.0 applications** across multiple browsers and operating systems.

Supported Browsers:

- Internet Explorer 6/7
- Mozilla Firefox 0.8+
- Opera
- Safari 2.0+

Supported Operating Systems:

- Windows
- Linux
- Macintosh

2. What is Selenium?

Selenium is a powerful **acceptance testing tool** designed specifically for **web applications**. The tests run **directly in the browser**, ensuring real-time validation of web functionalities across multiple platforms.

Key Features:

- Enables **cross-browser testing** to ensure application compatibility.
- Works across multiple operating systems for broader accessibility.
- Implemented entirely using browser technologies such as:
    - **JavaScript**
    - **DHTML**
    - **Frames**

3. Selenium Components

Selenium is divided into several key components, each serving a specific purpose in web automation.

Selenium IDE

- **Integrated Development Environment (IDE)** for creating test cases.
- Functions as a **Firefox plug-in** designed for quick recording and playback of test scripts.
- Automates commands, but assertions must be entered manually.
- Creates **the simplest possible locators** for elements.
- Uses **Selenese**, Selenium's scripting language.

Selenium Core

- The foundation of Selenium that directly interacts with the browser.
- Acts as the testing framework to run JavaScript automation within browsers.

Selenium Remote Control (RC)

- Allows execution of tests in different environments remotely.
- Supports multiple programming languages like Java, C#, Python, and Ruby.

Selenium Grid

- Enables **parallel test execution** across multiple machines and browsers.
- Improves testing efficiency by distributing test cases over multiple systems.

Overview of Selenium IDE:

A. Test Case Pane

B. Toolbar

C. Menu Bar

D. Log/Reference/UI-Element/Rollup Pane

A. Test Case Pane:

- Your script is displayed in the test case pane.
- It has two tabs.
- one for displaying the command (source)
- and their parameters in a readable “table” format.

B. Toolbar: The toolbar contains buttons for controlling the execution of your test cases.

C. Menu Bar:

- File Menu: The File menu allows you to create, open and save test case and test suite files.
- Edit Menu: The Edit menu allows copy, paste, delete, undo and select all operations for editing the commands in your test case.
- Options Menu: The Options menu allows the changing of settings. You can set the timeout value for certain commands, add user-defined user extensions to the base set of Selenium commands, and specify the format(language) used when saving your test cases.

D. Help Menu

Introducing Selenium Commands

The command set is often called selenese. Selenium commands come in three "flavors":
Actions, Accessory and Assertions.

a. Actions: user actions on application / Command the browser to do something.

   Actions are commands that generally manipulate the state of the application.

   1. Click link- click / Clickandwait

   2. Selecting items

b. Accessors: Accessors examine the state of the application and store the results in variables, e.g. "storeTitle".

c. Assertions: For validating the application we are using Assertions

   1. For verifying the web pages

   2. For verifying the text

   3. For verifying alerts

Assertions can be used in 3 modes:

- assert
- verify
- waitFor

Example: "assertText","verifyText" and "waitForText".

NOTE:

1. When an "assert" fails, the test is aborted.

2. When a "verify" fails, the test will continue execution

3. "waitFor" commands wait for some condition to become

4. trueCommonly Used Selenium Commands

These are probably the most commonly used commands for building test.

open - opens a page using a URL.

click/clickAndWait - performs a click operation, and optionally waits for a new page to load.

verifyTitle/assertTitle - verifies an expected page title.

verifyTextPresent- verifies expected text is somewhere on the page.

verifyElementPresent -verifies an expected UI element, as defined by its HTML tag, is present on the page.

verifyText - verifies expected text and it's corresponding HTML tag are present onthe page.

verifyTable - verifies a table's expected contents.

waitForPageToLoad -pauses execution until an expected new page loads. Called automatically when clickAndWait is used.

waitForElementPresent -pauses execution until an expected UI element, as definedby its HTML tag, is present on the page.

Recording and Run settings

When Selenium-IDE is first opened, the record button is ON by default.

During recording, Selenium-IDE will automatically insert commands into yourtest case based on your actions.

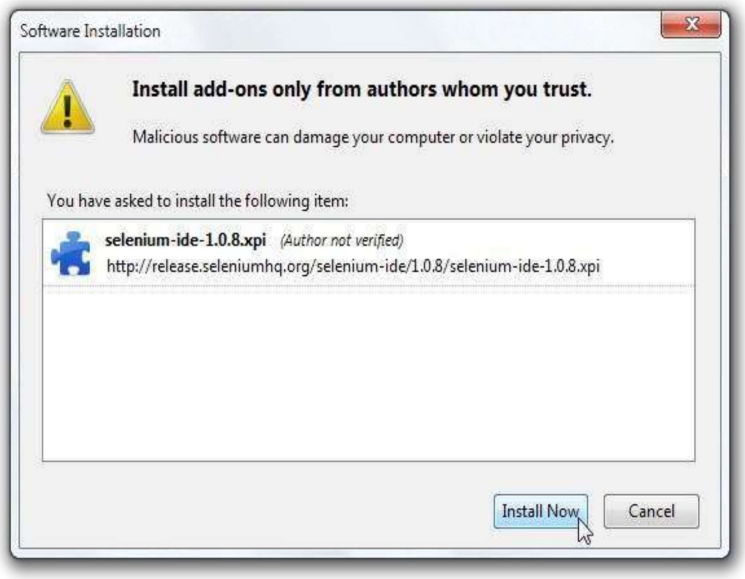
a. Remember Base URL MODE - Using Base URL to Run Test Cases in Different Domains

b. Record Absolute recording mode – Run Test Cases in Particular Domain.

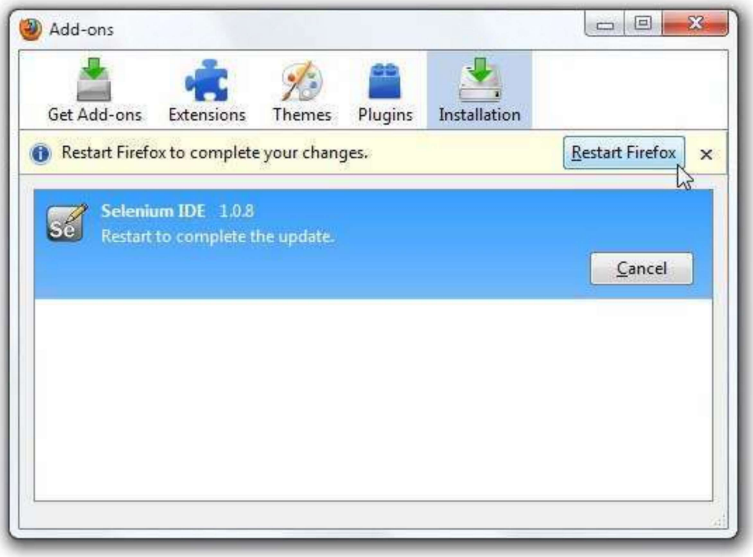
Running Test Cases

Run a Test Case Click the Run button to run the currently displayed test case.
Run a Test Suite Click the Run All button to run all the test cases in the currently loaded test suite.
Stop and Start The Pause button can be used to stop the test case while it is running. The icon of this button then changes to indicate the Resume button. To continue click Resume.
Stop in the Middle You can set a breakpoint in the test case to cause it to stop on a particular command. This is useful for debugging your test case. To set a breakpoint, select a command, right-click, and from the context menu select Toggle Breakpoint.
Start from the Middle You can tell the IDE to begin running from a specific command in the middle of the test case. This also is used for debugging. To set a startpoint, select a command, right-click, and from the context menu select Set/Clear Start Point.
Run Any Single Command Double-click any single command to run it by itself. This is useful when writing a single command. It lets you immediately test a command you are constructing, when you are not sure if it is correct. You can double-click it to see if it runs correctly. This is also available from the context menu.
Test Suite:

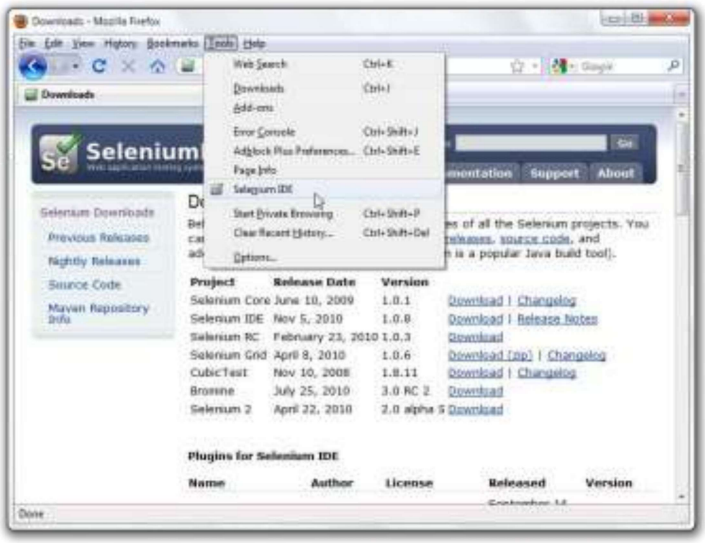
A test suite is a collection of tests. Often one will run all the tests in a test suite as one continuous batch-job.
When using Selenium-IDE, test suites also can be defined using a simple HTML file. The syntax again is simple. An HTML table defines a list of tests where each row defines the filesystem path to each test.
Installing the IDE
Using Firefox, first, download the IDE from the SeleniumHQ downloads page
Firefox will protect you from installing addons from unfamiliar locations, so you will need to click 'Allow' to proceed with the installation, as shown in the following screenshot.
When downloading from Firefox, you'll be presented with the following windo



Select Install Now. The Firefox Add-ons window pops up, first showing a progress bar, and when the download is complete, displays the following.

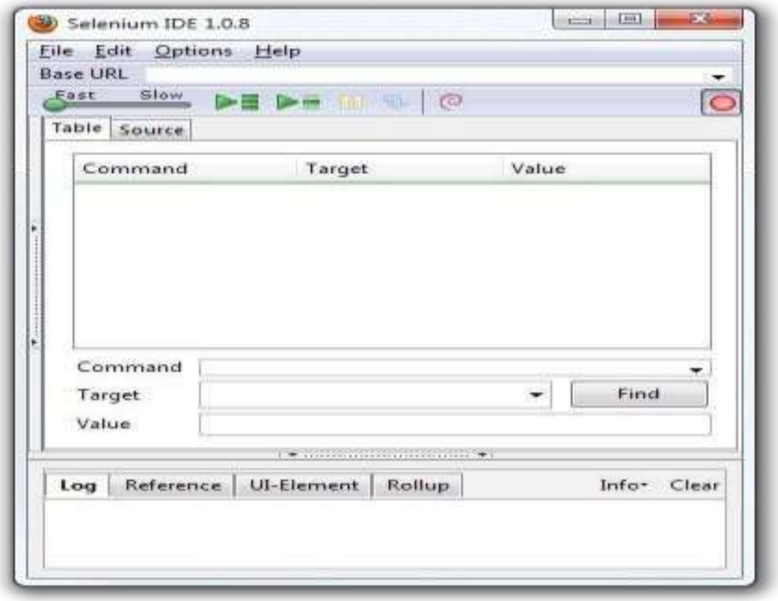


Restart Firefox. After Firefox reboots you will find the Selenium-IDE listed under the Firefox Tools menu.



Opening the IDE

To run the Selenium-IDE, simply select it from the Firefox Tools menu. It opens as follows with an empty script-editing window and a menu for loading, or creating new test cases.



.

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt about the basics of automation testing and the selenium.

Ex.No:2	Using Selenium IDE, Write a test suite containing minimum 4 test cases
Date:	

Test Case #1: Search for "energy efficient"
Manual Steps:

1. Open **Google** (Type www.google.com)
2. Type "energy efficient" in the **Google Search Input Box**
3. Click outside on an empty spot
4. Click the **Search Button**
5. Verify the **Text Present** is "energy efficient"
6. Assert the **Title** as "energy efficient - Google Search"
7. Save the **test case** with .html extension

Test Case #2: Search for "Selenium RC"
Manual Steps:

1. Open **Google** (Type www.google.com)
2. Type "Selenium RC" in the **Google Search Input Box**
3. Click outside on an empty spot
4. Click the **Search Button**
5. Verify the **Text Present** is "Selenium RC"
6. Assert the **Title** as "Selenium RC - Google Search"
7. Save the **test case** with .html extension

Test Case #3: Search for "Automated Testing"
Manual Steps:

1. Open **Google** (Type www.google.com)
2. Type "Automated Testing" in the **Google Search Input Box**
3. Click outside on an empty spot
4. Click the **Search Button**
5. Verify the **Text Present** is "Automated Testing"
6. Assert the **Title** as "Automated Testing - Google Search"
7. Save the **test case** with .html extension

Test Case #4: Search for "Selenium Grid"

Manual Steps:

1. Open **Google** (Type www.google.com)
2. Type "Selenium Grid" in the **Google Search Input Box**
3. Click outside on an empty spot
4. Click the **Search Button**
5. Verify the **Text Present** is "Selenium Grid"
6. Assert the **Title** as "Selenium Grid – Google Search"
7. Save the **test case** with .html extension

Steps to Create and Run the Test Suite in Selenium IDE:

1. Create **four test cases** and **save** each with the .html extension.
2. Open **Firefox**.
3. Open **Tools → Selenium IDE**.
4. Go to **File → Open → New Test Suite**.
5. Click **File → Open → Add Test Cases**.
6. Add more test cases as needed.
7. Save the **Test Suite** with a .html extension.
8. **Run the Test Suite** in Selenium IDE to execute all four test cases.

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt how to design a test suite with 4 test cases.

Ex.No:3	Conduct a test suite for any two web sites
Date:	

Test Case #1: Google Search for "Energy Efficient"
Manual Steps:

1. Open **Google** (Type www.google.com).
2. Type "energy efficient" in the **Google Search Input Box**.
3. Click outside on an empty spot.
4. Click the **Search Button**.
5. Verify that the text **"energy efficient"** appears in the search results.
6. Assert that the **Title** is "energy efficient - Google Search".
7. Save the **test case** with the .html extension.

Test Case #2: Yahoo Search for "Energy Efficient"
Manual Steps:

1. Open **Firefox Web Browser**.
2. Type http://www.yahoo.com in the address bar.
3. In the **search input box**, type "energy efficient".
4. Click on the **"Web Search"** submit button.
5. Wait for search results to load at "http:/search.yahoo.com".
6. Verify that **"energy efficient"** appears somewhere in the search results.
7. Assert that the browser's **Title** is "energy efficient - Yahoo! Search Results".
8. Save the **test case** with the .html extension.

Steps to Create and Run the Test Suite in Selenium IDE:

1. Create **two test cases** and **save** each with the .html extension.
2. Open **Firefox**.
3. Open **Tools → Selenium IDE**.
4. Go to **File → Open → New Test Suite**.
5. Click **File → Open → Add Test Cases**.
6. Add more test cases as needed.
7. Save the **Test Suite** with a .html extension.
8. **Run the Test Suite** in Selenium IDE to execute all test cases.

Selenium-RC: The Advanced Solution for Web Application Testing

Introduction to Selenium-RC

Selenium-RC (Remote Control) is a powerful solution for **complex web application testing** that extends beyond simple browser interactions. It enables automation using **full programming logic**, allowing for tasks such as database queries, file operations, and test result reporting.

Why Use Selenium-RC Instead of Selenium IDE?

Selenium-RC is recommended when a test requires functionality **beyond Selenium IDE**, such as:

- **Conditional Statements** (if-else, switch-case)
- **Iteration & Loops** (for, while)
- **Logging & Reporting** of test results
- **Error Handling**, especially for unexpected errors
- **Database Testing** for SQL queries
- **Test Case Grouping** for better organization
- **Re-execution of Failed Tests** for debugging
- **Test Case Dependencies** to maintain execution order
- **Capturing Screenshots** of test failures

Programming Languages Supported:

Selenium-RC supports multiple languages through specific client libraries, such as:

- **Java**
- **C#**
- **Python**
- **Ruby**
- **Perl**
- **PHP**

By integrating Selenium-RC with these programming languages, testers gain full control over complex testing workflows, making automated web testing more efficient and reliable.

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt how to design a test suite to test websites with 2 test cases.

Ex.No:4	Install Selenium server and demonstrate it using a script in Java/PHP
Date:	

Installation of Selenium RC and Eclipse

1. Downloading and Installing Eclipse

Follow these steps to install Eclipse and configure it properly for Selenium RC:

1. Navigate to the Eclipse official website: Eclipse Downloads
2. Select **Eclipse IDE for Java Developers** and choose the appropriate **Windows 64-bit version**.
3. Click **OK**, then **save the file** to a local drive (e.g., C: or D:).
4. Unzip the downloaded zip file and rename the extracted folder to **"Eclipse"**.
5. Create an additional **workspace folder** in the same drive where Eclipse is installed:
    - Example: C:\Eclipse-Workspace

Setting Up Eclipse for Selenium RC

6. Create a desktop shortcut for Eclipse:
    - Navigate to C:\Eclipse, right-click on Eclipse.exe, and select **Create Shortcut**.
7. Now, create a **specific workspace** for Selenium projects:
    - Example: C:\Eclipse-Workspace\SeleniumTests
8. Launch Eclipse by **double-clicking the shortcut** on your desktop.
9. Close the default **Eclipse Welcome Screen**
10. Select **File → Switch Workspace → Other**, then choose C:\Eclipse-Workspace\SeleniumTests.

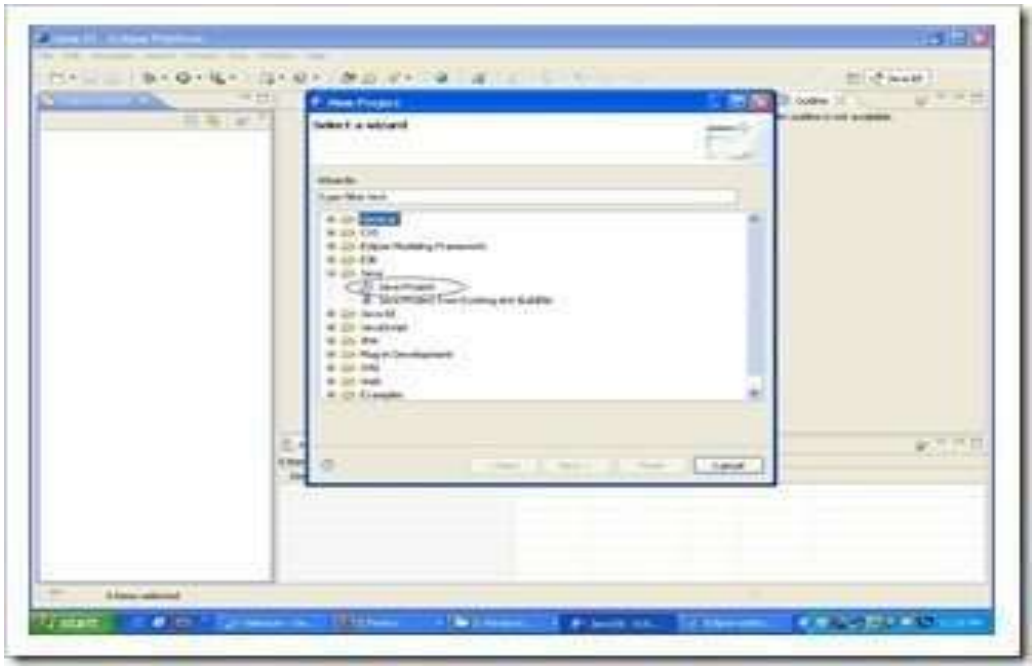
2. Downloading and Configuring Selenium RC Server and Client Drivers

Now, configure Selenium RC in Eclipse:

1. **Download Selenium RC Server**: Selenium HQ
2. Download **Selenium Client Driver for Java** from the "Selenium Client Drivers" section.
3. Create a dedicated **"Selenium" folder** in C:\ drive:
    - Copy the **Selenium-server.jar** file into this folder.
    - Unzip the **Selenium Client Driver** package inside C:\Selenium.

Configuring Selenium Client Driver in Eclipse

4. Open **Eclipse** → Click **File** → **New** → **Project**.
5. In the **Select Wizard**, choose **Java** → **Java Project** → Click **Next**.
6. Name the project (e.g., "SeleniumAutomation"), then click **Finish**.
7. Right-click on the project (SeleniumAutomation) → Click **Properties** → **Java Build Path**.
8. Navigate to the **Libraries** tab and click **Add External JARs**.
9. Select the **Selenium Client Drivers** in C:\Selenium.
10. Click **OK**—the **Referenced Libraries** now contain Selenium JAR files.



Writing the First Selenium RC Script Using JUnit in Java

Selenium RC allows writing automated tests in multiple programming languages. Below are **two methods** for scripting with Selenium RC.

Method 1: Extending the Selenese Test Case

This method extends the **SeleneseTestCase** class and runs tests using JUnit.

1. Open **Eclipse** → Create a new **Java package** inside your Selenium project.
2. Inside the package, create a **Java Class**, and paste the following code:

```
import com.thoughtworks.selenium.SeleneseTestCase;

public class SeleniumTest1 extends SeleneseTestCase {
   public void setUp() throws Exception {
      setUp("http://www.hexbytes.com/", "*firefox");
   }

   public void testNew() throws Exception {
      selenium.open("/");
      selenium.type("searchbox", "selenium rc");
      selenium.click("css=input[type='submit'][value='Search']");
      selenium.waitForPageToLoad("30000");
      assertTrue(selenium.isTextPresent("selenium rc"));
   }
}
```

3. Start **Selenium Server** (Selenium-server.jar).
4. Click **Run As → JUnit Test**.
5. If JUnit isn't installed, install the **JUnit plugin** for Eclipse.

Method 2: Extending the DefaultSelenium Class

This method helps when running tests on **different machines or ports**.

1. Create a package named "test" in Eclipse.
2. Create a new Java Class inside this package, and paste the code below:

```
package test;
import junit.framework.*;
import com.thoughtworks.selenium.DefaultSelenium;

public class DefaultSeleniumTest extends TestCase {
   private DefaultSelenium selenium;

   public void setUp() {
      selenium = new DefaultSelenium("localhost", 4444, "*iexplore",
"http://hexbytes.com");
      selenium.start();
   }

   public void testSearch() {
      selenium.open("/");
      selenium.type("searchbox", "selenium rc");
      selenium.click("css=input[type='submit'][value='Search']");
      selenium.waitForPageToLoad("30000");
      Assert.assertTrue(selenium.isTextPresent("selenium rc"));
   }

   public void tearDown() {
      selenium.stop();
   }
}
```

3. Start the **Selenium Server** (selenium-server.jar).
4. Click **Run As → JUnit Test** to execute the script.

Additional Updates and Enhancements

Why Use Selenium RC Instead of Selenium IDE?

Selenium RC provides more flexibility than Selenium IDE by allowing:

- **Conditional Statements** (if-else, switch-case).
- **Looping** (for, while).
- **Logging & Reporting** of test results.
- **Error Handling** (handling unexpected errors).
- **Database Connectivity** (for database validation).
- **Test Case Grouping** and dependencies.
- **Screenshot Capture** for failed test cases.

Supported Programming Languages:

Selenium RC works with multiple languages:

- **Java**
- **Python**
- **C#**
- **Ruby**
- **Perl**
- **PHP**

Final Steps: Running the Test Automation

Once everything is set up, you can execute tests using Selenium RC:

1. **Start Selenium Server** (selenium-server.jar).
2. Open Eclipse and navigate to your Selenium test script.
3. Click **Run As → JUnit Test**.
4. Review test execution results in the Eclipse console.
5. Capture failures and errors using **assertions & logging**.

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt how to install selenium server and implemented it in java.

Ex.No:5	Write and test a program to login a specific web page
Date:	

AIM:

To Write and test a program to login a specific web page.

ALGORITHM:

1. Setup: Install Eclipse, Selenium RC, TestNG, and Jxl.jar. . Add necessary libraries in Eclipse.
2. Initialize Selenium RC: Start Selenium Server, launch browser, and open the test website.
3. Run Tests: Open login page, read credentials from Excel, enter details, submit, and validate login.
4. Excel Handling: Open Excel file, fetch test data dynamically, and execute tests.
5. Test Execution: Use TestNG (@Test), run test cases, log results, and capture failures.
6. Cleanup: Stop Selenium, generate reports, and log results for debugging.

PROGRAM:

```
import
com.thoughtworks.selenium.*;
import org.junit.After;
import
org.junit.Before;
import org.junit.Test;
import java.util.regex.Pattern;

public class exp5 extends SeleneseTestCase
        {@Before
        public void setUp() throws Exception {
                selenium = new DefaultSelenium("localhost", 4444, "*chrome",
"http://demo.opensourcecms.com/");
```

```
        selenium.start();
    }

    @Test
    public void testExp5() throws Exception {
        selenium.open("/wordpress/wp-
        login.php");
        selenium.type("id=user_login", "admin");
        selenium.type("id=user_pass",
        "demo123"); selenium.click("id=wp-
        submit");
        selenium.waitForPageToLoad("30000");
    }

    @After
    public void tearDown() throws Exception {
        selenium.stop();
    }
}
```

TestNG

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use, such as:

- Annotations.
- Run your tests in arbitrarily big thread pools with various policies available(all methods in their own thread, one thread per test class, etc...).
- Test that your code is multithread safe.
- Flexible test configuration.
- Support for data-driven testing (with @DataProvider).
- Support for parameters.
- Powerful execution model (no more TestSuite).
- Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
- Embeds BeanShell for further flexibility.
- Default JDK functions for runtime and logging (no dependencies).
- Dependent methods for application server testing.

TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc...

Installing TestNG in eclipse

- Select Help / Software updates / Find and Install.
- Search for new features to install.
  - New remote site.
- For Eclipse 3.4 and above, enter http://beust.com/eclipse.
- For Eclipse 3.3 and below, enter http://beust.com/eclipse1.
- Make sure the check box next to URL is checked and click Next.
- Eclipse will then guide you through the process.

Launching your tests in Eclipse

- We finished writing our tests, now how can we run them?
- You can launch TestNG from the command line, using a Eclipse plugin oreven programatically. We are going to use the Eclipse plugin. Follow the steps described on the official TestNG documentation over here
- If you installed TestNG correctly, you will see this menu when you right click on the XML file:

- Click on "Run as TestNG Suite" and your test will start running. You willthen see this nice results tree:

Selenium Tests with Microsoft Excel

Parameterizing a test from external sources such as Microsoft Excel is always recommended in order to handle large amount of test data. To read data from Excel, we need APIs which support opening file, reading data, and writing data into Excel. We should know various classes and methods which support above mentioned operations. In this post, let us try to figure out which is the API that supports all the activities we need to do during execution of a test.

Jxl.jar is an open source Java API which supports read Excel spreadsheets and towrite into Excel spreadsheets. Below are some of the operations that we can handle with this API.

1. Read data from Excel spreadsheet
2. Read and write formulas into spreadsheets

3. Generate spreadsheets
4. Supports formatting of font, number, and date
5. Supports coloring of cells

To access the methods and classes provided by this API inside Eclipse we need to add this JAR file to the Java Build Path. (I have explained steps to add external Jarfiles to Java Build Path in previous posts)

Download the jxl.jar from "http://jexcelapi.sourceforge.net/"
Add the JAR file to Java Build Path

Add import statements to the .java file as below to read from an Excel Spreadsheet

```
import jxl.Cell;
import jxl.Sheet;
importjxl.Workbook;
import jxl.read.biff.BiffException
```

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt how to write and test a program to login specific web page.

Ex.No:6	Write and test a program to update 10 student records into table into Excel file
Date:	

AIM:

To Write and test a program to update 10 student records into tableinto Excel file .

ALGORITHM:

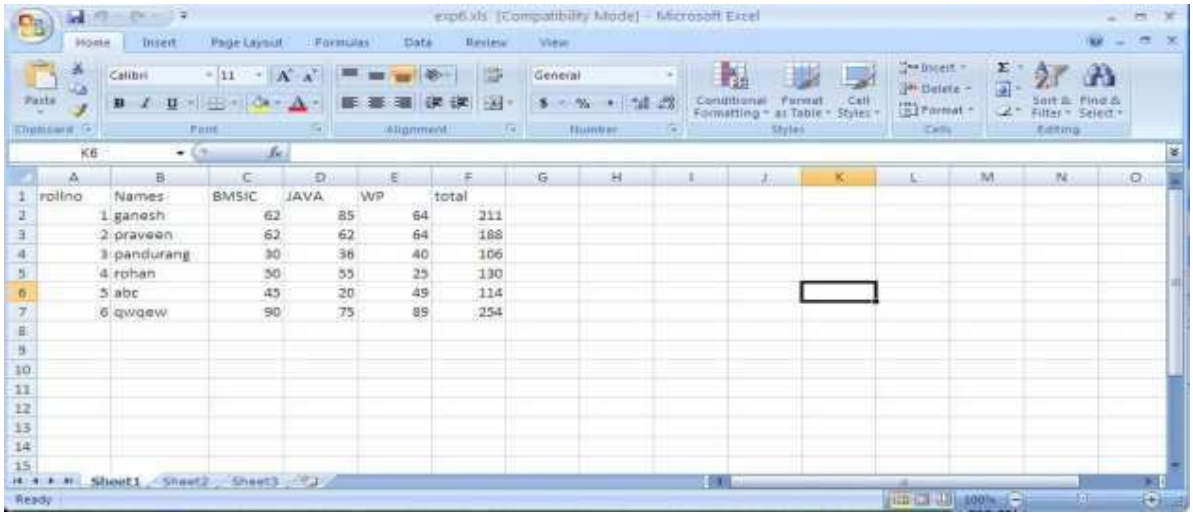
1. **Setup**: Initialize the test environment using @BeforeClass.
2. **Read Excel File**: Open exp6.xls and extract data into a 2D array.
3. **Create Output Excel**: Generate exp6Result.xls and create a writable sheet.
4. **Copy Data**: Loop through rows and columns to transfer data to the new sheet.
5. **Evaluate Scores**: Check if values are greater than 35; mark "pass" or "fail" in the result column.
6. **Write & Close**: Save the modified Excel file and close the stream.

PROGRAM:

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import jxl.Sheet;
import jxl.Workbook;
import jxl.write.Label;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import org.testng.annotations.*;
public class ExcelTest {
   @BeforeClass
   public void setUp() {
      System.out.println("Test setup initialized.");
   }
   @Test
   public void testImportExport() {
      try {
         // Read data from the input Excel file
         FileInputStream fi = new FileInputStream("D:\\exp6.xls");
         Workbook workbook = Workbook.getWorkbook(fi);
         Sheet sheet = workbook.getSheet(0);
```

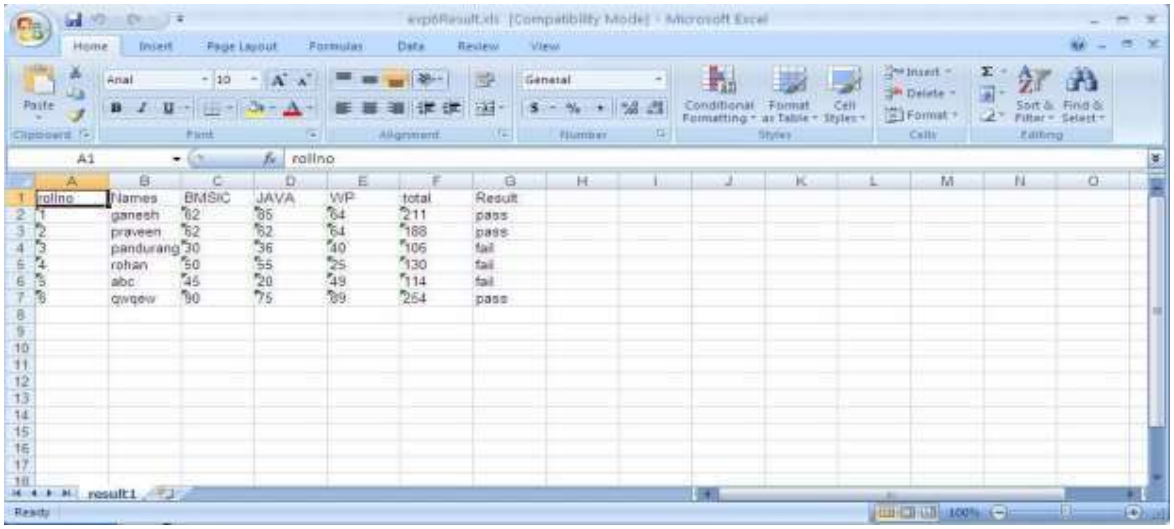
```
            int rows = sheet.getRows();
            int columns = sheet.getColumns();
            String[][] data = new String[rows][columns];

            // Create a new Excel file for results
            FileOutputStream fo = new FileOutputStream("D://exp6Result.xls");
            WritableWorkbook writableWorkbook = Workbook.createWorkbook(fo);
            WritableSheet writableSheet =
    writableWorkbook.createSheet("ResultSheet", 0);

            // Copy data and add column headers
            Label header = new Label(columns, 0, "Result");
            writableSheet.addCell(header);

            for (int i = 0; i < rows; i++) {
               for (int j = 0; j < columns; j++) {
                  data[i][j] = sheet.getCell(j, i).getContents();
                  writableSheet.addCell(new Label(j, i, data[i][j]));
               }
            }
            // Evaluate test results
            for (int i = 1; i < rows; i++) {
               int score = Integer.parseInt(sheet.getCell(2, i).getContents());
               Label resultLabel = (score > 35)
                  ? new Label(columns, i, "Pass")
                  : new Label(columns, i, "Fail");

               writableSheet.addCell(resultLabel);
            }
            // Save and close the workbook
            writableWorkbook.write();
            writableWorkbook.close();
            System.out.println("Excel processing complete.");

         } catch (Exception e) {
            System.err.println("Error processing Excel file: " + e.getMessage());
         }
      }
   }
```

INPUT:



rollno	Names	BMSIC	JAVA	WP	total
1	ganesh	62	85	64	211
2	praveen	62	62	64	188
3	pandurang	30	36	40	106
4	rohan	50	55	25	130
5	abc	45	20	49	114
6	qwqew	90	75	89	254

OUTPUT:



rollno	Names	BMSIC	JAVA	WP	total	Result
1	ganesh	62	85	64	211	pass
2	praveen	62	62	64	188	pass
3	pandurang	30	36	40	106	fail
4	rohan	50	55	25	130	fail
5	abc	45	20	49	114	fail
6	qwqew	90	75	89	254	pass

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt how to write and test a program to update 10 student records into table into excel file .

Ex.No:7	Write and test a program to select the number of students who have scored more than 60 in any one subject
Date:	

AIM:

To Write and test a program to select the number of students who have scored more than 60 in any one subject (or all subjects).

ALGORITHM:

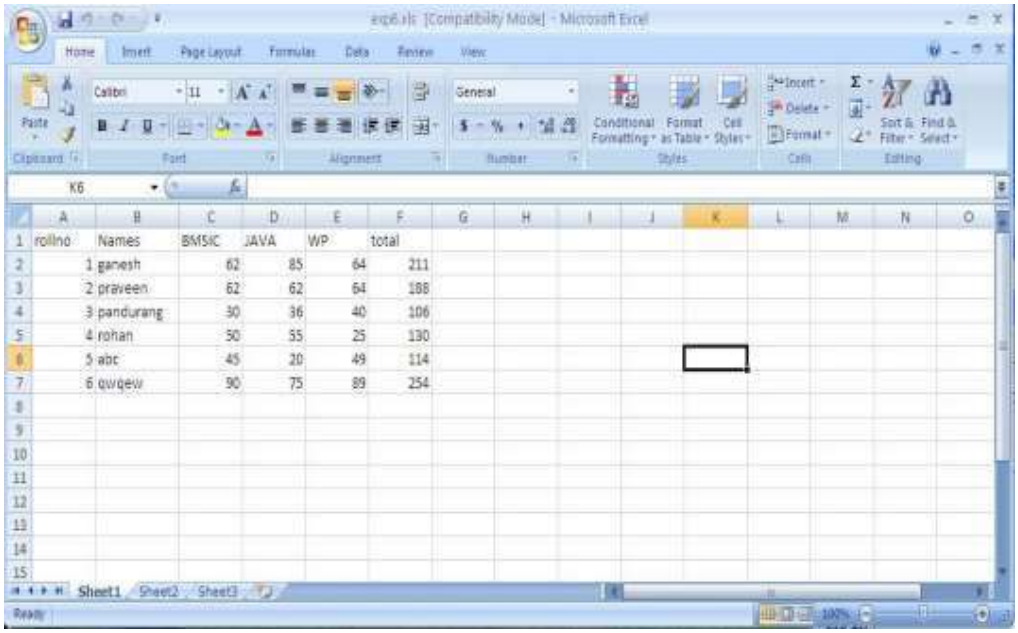
1. Setup: Initialize the test environment using @BeforeClass.
2. Read Excel File: Open exp6.xls, retrieve sheet data, and store it in a 2D array.
3. Create Output File: Generate exp7Result.xls and create a writable sheet.
4. Filter Data: If the value in column 3 is less than 60, mark the row for exclusion.
5. Write Data: Copy filtered data to the output file, adjusting row positions.
6. Save & Close: Write modifications and close the workbook streams.

PROGRAM:

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import jxl.Sheet;
import jxl.Workbook;
import jxl.write.Label;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import org.testng.annotations.*;

public class Exp7 {

   @BeforeClass
   public void setUp() {
      System.out.println("Test setup initialized.");
   }
```

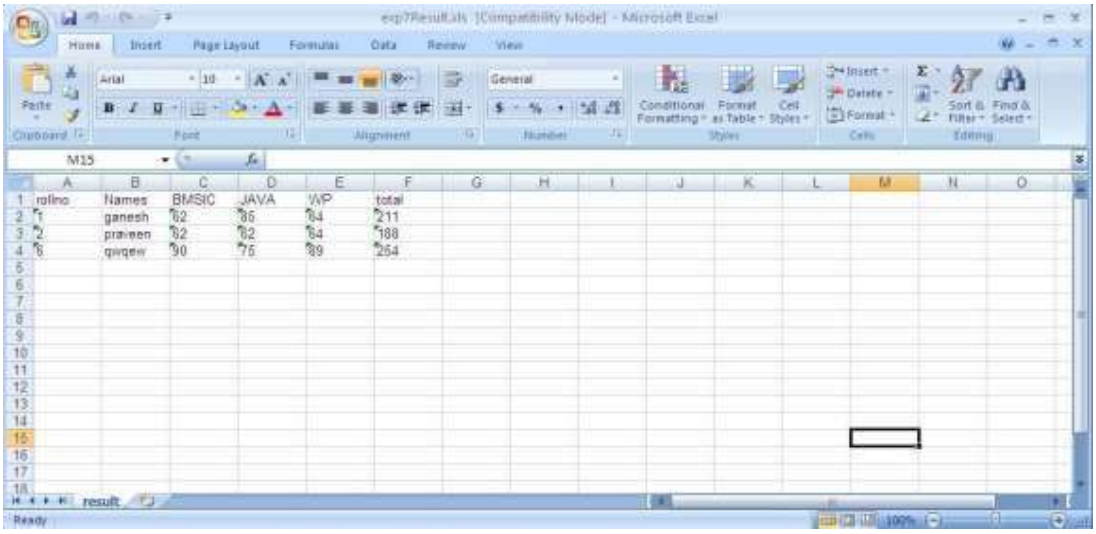
```
@Test
  public void testImportExport() {
    try {
      FileInputStream fi = new FileInputStream("D:\\exp6.xls");
      Workbook workbook = Workbook.getWorkbook(fi);
      Sheet sheet = workbook.getSheet(0);
      int rows = sheet.getRows();
      int columns = sheet.getColumns();
      String[][] data = new String[rows][columns];
      FileOutputStream fo = new FileOutputStream("D://exp7Result.xls");
      WritableWorkbook writableWorkbook = Workbook.createWorkbook(fo);
      WritableSheet writableSheet =
writableWorkbook.createSheet("FilteredResults", 0);
      int adjustedRow = 0;
      for (int i = 0; i < rows; i++) {
        boolean excludeRow = false;
        if (i >= 1) {
          int value = Integer.parseInt(sheet.getCell(3, i).getContents());
          if (value < 60) {
            excludeRow = true;
          }
        }
        if (!excludeRow) {
          for (int j = 0; j < columns; j++) {
            data[i][j] = sheet.getCell(j, i).getContents();
            writableSheet.addCell(new Label(j, adjustedRow, data[i][j]));
          }
          adjustedRow++;
        }
      }
      writableWorkbook.write();
      writableWorkbook.close();
      System.out.println("Excel processing completed successfully.");

    } catch (Exception e) {
      System.err.println("Error processing Excel file: " + e.getMessage());
    }
  }
}
```

INPUT:



rollno	Names	BMSIC	JAVA	WP	total
1	ganesh	62	85	64	211
2	praveen	62	62	64	188
3	pandurang	30	36	40	106
4	rohan	50	55	25	130
5	abc	45	20	49	114
6	qwqew	90	75	89	254

OUTPUT:



rollno	Names	BMSIC	JAVA	WP	total
1	ganesh	62	85	64	211
2	praveen	62	62	64	188
6	qwqew	90	75	89	254

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt how to write and test a program to select the number of students who have scored more than 60 in any one subject

Ex.No:8	Write and test a program to provide total number of objects available on the page
Date:	

AIM:

To Write and test a program to provide total number of objects available on the page .

ALGORITHM:

Setup Selenium RC: Start Selenium Server and initialize browser (Firefox).

Run Test: Open Google, maximize window, retrieve all links, buttons, and input fields.

Log Information: Print the count of links, buttons, and fields on the webpage.

Cleanup: Stop Selenium session and server after execution.

PROGRAM:

```
package testscripts;

import com.thoughtworks.selenium.*;
import org.openqa.selenium.server.*;
import org.testng.annotations.*;

public class Exp8 {

    private Selenium selenium;
    private SeleniumServer seleniumServer;

    @BeforeClass
    public void setUp() {
        try {
            RemoteControlConfiguration rc = new RemoteControlConfiguration();
            seleniumServer = new SeleniumServer(rc);
            selenium = new DefaultSelenium("localhost", 4444, "*firefox", "http://");

            seleniumServer.start();
            selenium.start();
```

```
        } catch (Exception e) {
            System.err.println("Error starting Selenium Server: " + e.getMessage());
        }
    }

    @Test
    public void testDefaultTNG() {
        try {
            selenium.open("http://www.google.co.in/");
            selenium.windowMaximize();

            String[] links = selenium.getAllLinks();
            System.out.println("Total number of links: " + links.length);

            String[] buttons = selenium.getAllButtons();
            System.out.println("Total number of buttons: " + buttons.length);

            String[] fields = selenium.getAllFields();
            System.out.println("Total number of input fields: " + fields.length);

        } catch (Exception e) {
            System.err.println("Error during test execution: " + e.getMessage());
        }
    }

    @AfterClass
    public void tearDown() {
        try {
            selenium.stop();
            seleniumServer.stop();
        } catch (Exception e) {
            System.err.println("Error stopping Selenium Server: " + e.getMessage());
        }
    }
}
```

OUTPUT:

TOTAL NO OF LINKS=41 TOTAL
NO OF BUTTONS=1 TOTAL NO
OF INPUT FIELDS=3

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt how to write and test a program to provide total number of objects available on the page .

Ex.No:9	Write and test a program to get the number of list items in a list /combo box
Date:	

AIM:

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

ALGORITHM:

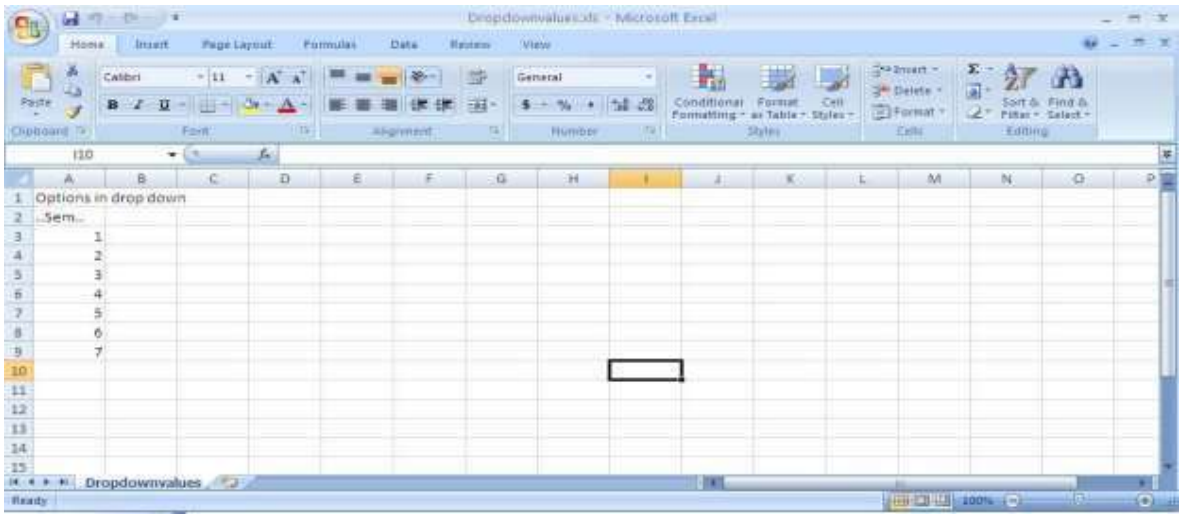
1. Setup Selenium RC: Start Selenium Server and initialize Firefox browser.
2. Open Webpage: Navigate to http://www.schools9.com/kdiploma1102.html.
3. Fetch Dropdown Options: Retrieve values from the dropdown menu.
4. Store Data in Files: Write dropdown options to Dropdownvalues.xls (Excel) and Dropdownvalues.txt (Notepad).
5. Close Files: Save the output files and close buffered writers.
6. Cleanup: Stop Selenium session and server.

PROGRAM:

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import com.thoughtworks.selenium.*;
import org.openqa.selenium.server.*;
import org.testng.annotations.*;
public class DropDownGoogle {
   private Selenium selenium;
   private SeleniumServer seleniumServer;
   @BeforeClass
   public void setUp() {
      try {
         RemoteControlConfiguration rc = new RemoteControlConfiguration();
         seleniumServer = new SeleniumServer(rc);
         selenium = new DefaultSelenium("localhost", 4444, "*firefox", "http://")
         seleniumServer.start();
         selenium.start();
      } catch (Exception e) {
         System.err.println("Error starting Selenium Server: " + e.getMessage());
      }
```

```
    }
    @Test
    public void testDropdownGoogle() {
        try {
            selenium.open("http://www.schools9.com/kdiploma11022011.htm");
            selenium.waitForPageToLoad("3000");
            String[] options = selenium.getSelectOptions("name=course");
            File excelFile = new File("C:/Dropdownvalues.xls");
            File textFile = new File("C:/Dropdownvalues.txt");
            try (BufferedWriter excelWriter = new BufferedWriter(new
FileWriter(excelFile));
                 BufferedWriter textWriter = new BufferedWriter(new
FileWriter(textFile))) {
                excelWriter.write("Options in drop-down\n");
                textWriter.write("Options in drop-down\n")
                for (int i = 0; i < options.length; i++) {
                    System.out.println("Option " + i + ": " + options[i]);
                    excelWriter.write(options[i]);
                    excelWriter.newLine();
                    textWriter.write(options[i]);
                    textWriter.newLine();
                }
            }
            System.out.println("Dropdown values successfully saved.");
        } catch (Exception e) {
            System.err.println("Error processing dropdown values: " +
e.getMessage());
        }
    }
    @AfterClass
    public void tearDown() {
        try {
            selenium.stop();
            seleniumServer.stop();
        } catch (Exception e) {
            System.err.println("Error stopping Selenium Server: " + e.getMessage());
        }
  }
}
```

OUTPUT:



Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt how to write and test a program to get the number of list items in a list /combo box.

Ex.No:10	Write and test a program to count number of check boxes on thepage checked and unchecked count
Date:	

AIM:

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

ALGORITHM:

1. Setup Selenium RC: Start Selenium Server and initialize Firefox browser.
2. Open Webpage: Navigate to http://browsershots.org/.
3. Count Elements: Retrieve count of all checkboxes, checked checkboxes, and unchecked checkboxes.
4. Log Information: Print the counts for debugging and verification.
5. Cleanup: Stop Selenium session and server.

PROGRAM:

```
package testscripts;

import com.thoughtworks.selenium.*;
import org.openqa.selenium.server.*;
import org.testng.annotations.*;

public class Exp8 {

   private Selenium selenium;
   private SeleniumServer seleniumServer;

   @BeforeClass
   public void setUp() {
      try {
         RemoteControlConfiguration rc = new RemoteControlConfiguration();
         seleniumServer = new SeleniumServer(rc);
         selenium = new DefaultSelenium("localhost", 4444, "*firefox",
```

```
"http://");

            seleniumServer.start();
            selenium.start();
        } catch (Exception e) {
            System.err.println("Error starting Selenium Server: " +
e.getMessage());
        }
    }

    @Test
    public void testDefaultTNG() {
        try {
            selenium.open("http://browsershots.org/");
            selenium.windowMaximize();

            // Count all checkboxes
            Number totalCheckboxes =
selenium.getXpathCount("//input[@type='checkbox']");
            System.out.println("Total number of checkboxes: " +
totalCheckboxes);
            // Count checked checkboxes
            Number checkedCheckboxes =
selenium.getXpathCount("//input[@type='checkbox' and @checked]");
            System.out.println("Number of checked checkboxes: " +
checkedCheckboxes);
            // Count unchecked checkboxes
            Number uncheckedCheckboxes =
selenium.getXpathCount("//input[@type='checkbox' and not(@checked)]");
            System.out.println("Number of unchecked checkboxes: " +
uncheckedCheckboxes);
        } catch (Exception e) {
            System.err.println("Error during test execution: " + e.getMessage());
        }
    }
    @AfterClass
    public void tearDown() {
        try {
            selenium.stop();
            seleniumServer.stop();
```

```
        } catch (Exception e) {
            System.err.println("Error stopping Selenium Server: " +
e.getMessage());
        }
    }
}
```

OUTPUT:

Count of check boxes 127

Count of Checked check boxes 91

Count of Checked check boxes 36

Department of M.Tech(CSE)		
Description	**Max Marks**	**Awarded**
Aim	**5**	
Software/Tools Required & Algorithm	**10**	
Coding/Program & Execution	**20**	
Record	**20**	
Viva-Voice	**10**	
Result	**10**	
TOTAL	**75**	

RESULT:

Thus,we have learnt how to write and test a program to count number of check boxes on thepage checked and unchecked count.