

**ANNAMALAI UNIVERSITY**  
(Accredited with Grade 'A' by NAAC)

**FACULTY OF ENGINEERING AND TECHNOLOGY  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**LABORATORY MANUAL**

**CSCP608 – SOFTWARE ENGINEERING LABORATORY**

**NAME:** .....

**REG NO:** .....

**ANNAMALAI UNIVERSITY**  
(Accredited with Grade 'A' by NAAC)

**FACULTY OF ENGINEERING AND TECHNOLOGY  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**FACULTY OF ENGINEERING AND TECHNOLOGY  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CSCP608 – SOFTWARE ENGINEERING LABORATORY**

Certified that this is the Bonafide record of work done by  
Mr/Miss.....  
Register Number ..... of B.E. (Computer Science and  
Engineering), VI Semester in **Software Engineering Lab** during the year 2022-  
2023.

**Annamalai Nagar**  
**Date:**

**Staff in-Charge**

**Internal Examiner**

**External Examiner**

## CONTENTS

s.no	Experiment	Page no	Marks	Sign
1	Write a C program for matrix multiplication to understand the causes of failures			
2	Write a C program for Binary Search - Path Testing			
3	Write a C program to derive test cases based on boundary value analysis			
4	Write a C program for cause effect graph to check whether defect is found in the program			
5	Write a C program for cause effect graph to check whether defect is found in the program			
6	Write a C program to perform data flow testing for the given code and find out all d-use Pairs			
7	Write a C program to demonstrate the working of the looping constructs			
8	Write and test a program to provide total number of objects available on the page using selenium tool			
9	Write and test a program to login a specific web page using selenium tool			
10	Write and test a program to select the number of students who have scored more than 60 in any one subject ( or all subjects)			

## **LIST OF EXERCISES**

1. Write a C program for matrix multiplication to understand the causes of failures.
2. Write a C program for Binary Search - Path Testing.
3. Write a C program to derive test cases based on boundary value analysis.
4. Write a C program for cause effect graph to check whether defect is found in the program.
5. Write a C program to perform data flow testing for the given code and find out all d-use Pairs.
6. Write a C program to demonstrate the working of the looping constructs.
7. Write and test a program to count number of check boxes on the page checked and unchecked count using selenium tool.
8. Write and test a program to provide total number of objects available on the page using selenium tool.
9. Write and test a program to login a specific web page using selenium tool.
10. Write and test a program to select the number of students who have scored more than 60 in any one subject ( or all subjects ).

## **INTRODUCTION TO SOFTWARE TESTING**

The purpose of the laboratory is to evaluate and develop methods of testing software efficiently that aim on discovering security relevant software flaws along with considering core components of quality before the final product is deployed.

The prime goal is to make the students aware about the existing methods of software testing and considering software quality. Many of the software testing techniques available are very expensive and time consuming. Therefore, the aim of the lab is to understand, which existing testing techniques are most effective for vulnerability detection, in order to provide software engineers guidelines for the selection of testing methods using software quality methods.

## **LAB REQUIREMENTS:**

Software Detail

Windows & Turbo C/C++,  
Eclipse, Selenium Tool


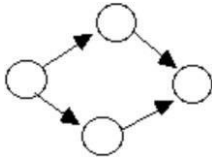
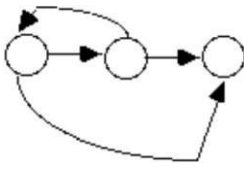
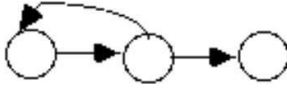
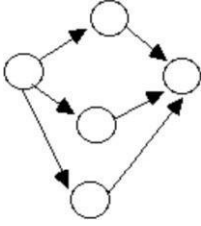
## 1) PATH TESTING:

The basis path method allows for the construction of test cases that are guaranteed to execute every statement in the program at least once. This method can be applied to detailed procedural design or source code.

### Method:

- 1) Draw the flow graph corresponding to the procedural design or code.
- 2) Determine the cyclomatic complexity of the flow graph.
- 3) Determine the basis set of independent paths. (The cyclomatic complexity indicates the number of paths required.)
- 4) Determine a test case that will force the execution of each path.

### Flow graphs:

Sequence	
IF	
WHILE	
REPEAT	
CASE	

### Cyclomatic Complexity:

Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measures the number of linearly-independent paths through a program module.

Lower the Program's cyclomatic complexity, lower the risk to modify and easier to understand. It can be represented using the below formula:

$$\text{Cyclomatic complexity} = E - N + 2 * P$$

where,

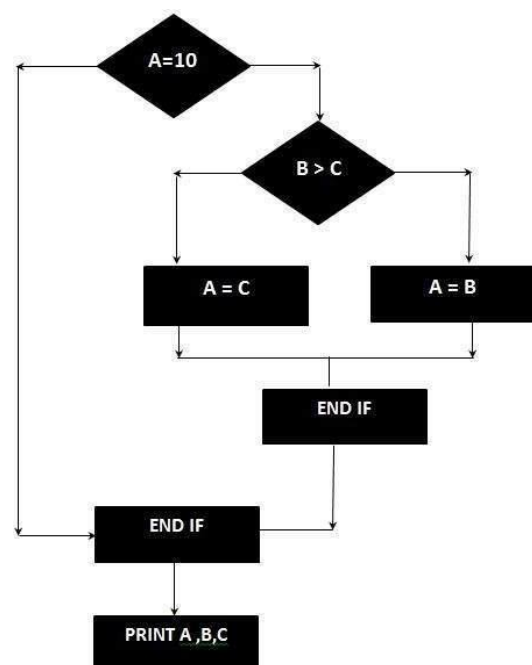
E = number of edges in the flow graph.

N = number of nodes in the flow graph.

P = number of nodes that have exit points

### Example:

```
IF A = 10 THEN
  IF B > C THEN
    A = B
  ELSE
    A = C
  ENDIF
ENDIF
Print A
Print B
Print C
```



The Cyclomatic complexity is calculated using the above control flow diagram that shows seven nodes(shapes) and eight edges (lines), hence the cyclomatic complexity is  $8 - 7 + 2 = 3$

## 2) **BOUNDARY VALUE ANALYSIS:**

Boundary value analysis is complementary to equivalence partitioning. Rather than selecting arbitrary input values to partition the equivalence class, the test case designer chooses values at the extremes of the class. Furthermore, boundary value analysis also encourages test case designers to look at output conditions and design test cases for the extreme conditions in output.

### Guidelines for boundary value analysis:

- 1) If an input condition specifies a range bounded by values ***a*** and ***b***, test cases should be designed with values ***a*** and ***b***, and values just above and just below ***b***.
- 2) If an input condition specifies a number of values, test cases should be developed that exercise the minimum and maximum numbers. Values above and below the minimum and maximum are also tested.
- 3) Apply the above guidelines to output conditions. For example, if the requirement specifies the production of a table as output then you want to choose input conditions that produce the largest and smallest possible table.
- 4) For internal data structures be certain to design test cases to exercise the data structure at its boundary. For example, if the software includes the maintenance of a personnel list, then you should ensure the software is tested with conditions where the list size is 0, 1 and maximum (if constrained).

Data Item	Input Condition	Remarks
Area Code	range	Values between 200 and 999 with area codes requiring 0, 1 in second position <b>Test Cases:</b> 200, 910 (valid end-points) 199, 912 (value below and above)
Prefix	range	Specified value > 200 <b>Test Cases:</b> 201 (minimum) 999 (maximum) 200 (invalid, just below) 199, 912 (value below and above)


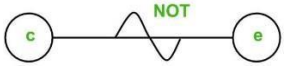
Prefix	range	Specified value > 200 <b>Test Cases:</b> 201 (minimum) 999 (maximum) 200 (invalid, just below)
Suffix	value	Four-digit length <b>Test Cases:</b> 0000 (minimum) 9999 (maximum)
Password	value	Six-character string <b>Test Cases:</b>
Commands	set	<b>Test Cases:</b>

### 3) CAUSE-EFFECT GRAPHS:

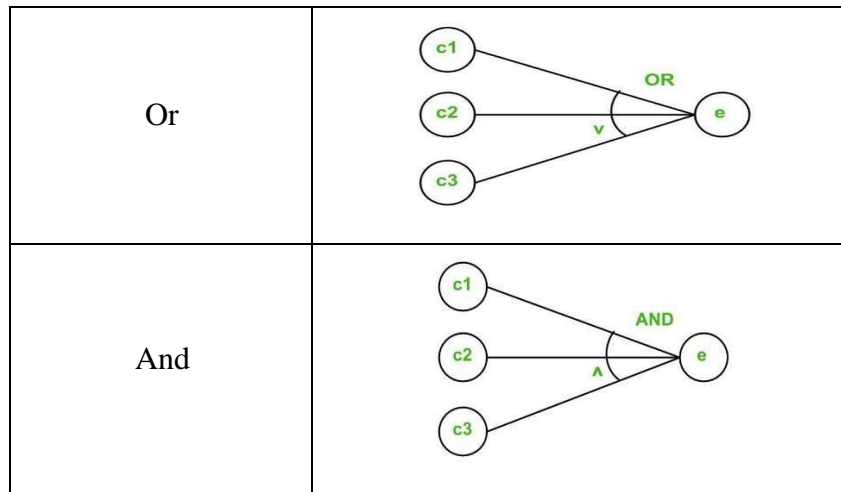
A weakness of the two methods is that do not consider potential combinations of input/output conditions. Cause-effect graphs connect input classes (**causes**) to output classes (**effects**) yielding a directed graph.

Guidelines for cause-effect graphs:

- 1) Decompose the specification into workable pieces.
- 2) Identify causes and their effects.
- 3) Create a (Boolean) cause-effect graph (special symbols are required).
- 4) Annotate the graph with constraints describing combinations of causes and /or effects that are impossible.
- 5) The graph is converted to a decision table by methodically tracing state conditions in the graph. Each column in the table represents a test case.
- 6) Decision table rules are converted to test cases.

Identify	
Not	





#### 4) DATA FLOW TESTING:

**Data Flow Testing** is a type of structural testing. It is a method that is used to find the test paths of a program according to the locations of definitions and uses of variables in the program. It has nothing to do with data flow diagrams.

It is concerned with:

- Statements where variables receive values,
- Statements where these values are used or referenced.

To illustrate the approach of data flow testing, assume that each statement in the program assigned a unique statement number. For a statement number S-

$DEF(S) = \{X \mid \text{statement } S \text{ contains the definition of } X\}$

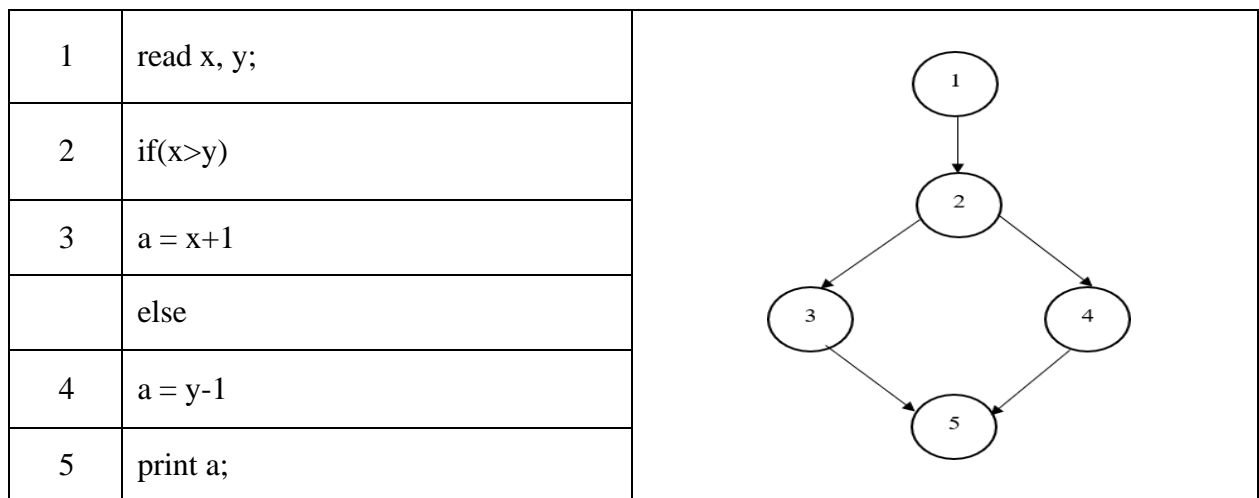
$USE(S) = \{X \mid \text{statement } S \text{ contains the use of } X\}$

If a statement is a loop or if condition then its DEF set is empty and USE set is based on the condition of statement s.

Data Flow Testing uses the control flow graph to find the situations that can interrupt the flow of the program.

Reference or define anomalies in the flow of the data are detected at the time of associations between values and variables. These anomalies are:

- A variable is defined but not used or referenced,
- A variable is used but never defined,
- A variable is defined twice before it is used

**EXAMPLE:****Define/use of variables of above example:**

Variable	Defined at node	Used at node
X	1	2,3
Y	1	2,4
a	3,4	5

Data flow testing is a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of variables or data objects. Dataflow Testing focuses on the points at which variables receive values and the points at which these values are used.

- A node  $n$  in the program graph is a **defining** node for variable  $v$  – **DEF( $v$ ,  $n$ )** – if the value of  $v$  is defined at the statement fragment in that node.
  - ❖ *Input, assignment, procedure calls*
- A node  $n$  in the program graph is a **usage** node for variable  $v$  – **USE( $v$ ,  $n$ )** – if the value of  $v$  is used at the statement fragment in that node.
  - ❖ *Output, assignment, conditionals*
- A usage node is a **predicate use**, **P-use**, if variable  $v$  appears in a predicate expression.
  - ❖ *Always in nodes with outdegree  $\geq 2$*
- A usage node is a **computation use**, **C-use**, if variable  $v$  appears in a computation.
  - ❖ *Always in nodes with outdegree  $\leq 1$*

- A node  $n$  in the program is a **kill** node for a variable  $v$  – **KILL**( $v, n$ ) – if the variable is deallocated at the statement fragment in that node.

**du-path:**

- A **definition-use path**, **du-path**, with respect to a variable  $v$  is a path whose first node is a defining node for  $v$ , and its last node is a usage node for  $v$ .

**dc-path:**

- A **du-path** with no other defining node for  $v$  is a **definition-clear path**, **dc-path**.

## 5) LOOP TESTING:

Simple Loops	<p>The following set of tests should be applied to simple loops, where <math>n</math> is the maximum number of allowable passes:</p> <ol style="list-style-type: none"> <li>1. Skip the loop entirely.</li> <li>2. Only one pass through the loop.</li> <li>3. Two passes through the loop.</li> <li>4. <math>m</math> passes through the loop where <math>m &lt; n</math>.</li> <li>5. <math>n-1, n, n+1</math> passes through the loop.</li> </ol>
Nested Loops	<ol style="list-style-type: none"> <li>1. Start with the innermost loop. Set all other loops to minimum values.</li> <li>2. Conduct simple loop tests for the innermost loop while holding the outer loops at their minimum iteration values.</li> <li>3. Work outward, conducting tests for the next loop, but keeping all other outer loops at this minimum iteration count.</li> <li>4. 4. Continue until all loop have been tested.</li> </ol>
Concatenated Loops	<p>Concatenated loops can be tested using the approach defined for simple loops, if the loops are independent. If the loop counter from a loop <math>i</math> is used as the initial value for loop <math>i+1</math> then the loops are not independent. When loops are not independent use the concatenated loop strategy.</p>
Unstructured Loops	<p>Redesign the loops so they are one of the above categories.</p>

## **SELENIUM**

### **AUTOMATION:**

Automation is making a process automatic, eliminating the need for human intervention. It is a self-controlling or self-moving process. Automation Software offers automation wizards and commands of its own in addition to providing a task recording and re-play capabilities. Using these programs you can record an IT or business task.

### **Benefits of Automation:**

- Fast
- Reliable
- Repeatable
- Programmable
- Reusable
- Makes Regression testing easy
- Enables 24\*78 Testing Robust verification.

## **INTRODUCTION TO SELENIUM**

### **History of Selenium**

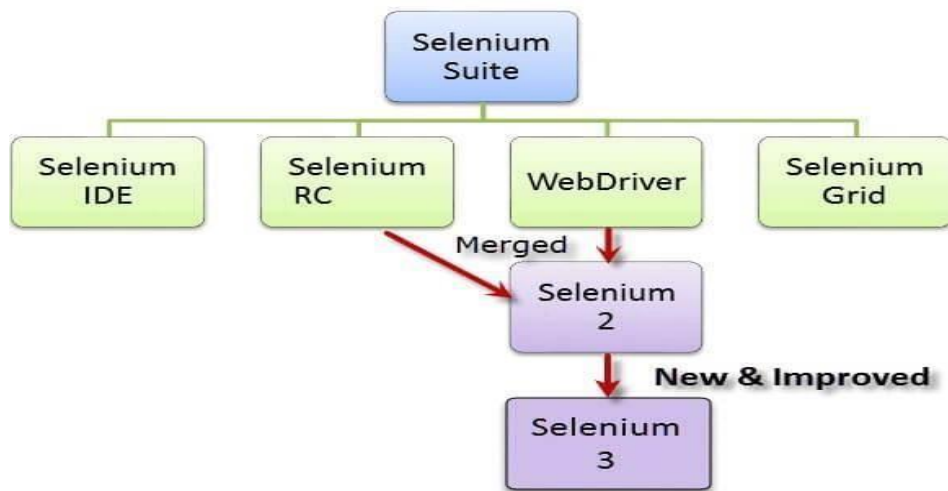
- In 2004 invented by Jason R. Huggins and team.
- Original name is JavaScript Functional Tester [JSFT]
- Open source browser based integration test framework built originally by Thoughtworks.
- 100% JavaScript and HTML
- Web testing tool
- That supports testing Web 2.0 applications
- Supports for Cross-Browser Testing(ON Multiple Browsers)
- And multiple Operating Systems
- Cross browser – IE 6/7, Firefox .8+, Opera, Safari 2.0+

### **What is Selenium?**

- Acceptance Testing tool for web-apps
- Tests run directly in browser
- Selenium can be deployed on Windows, Linux, and Macintosh.
- Implemented entirely using browser technologies -
  - JavaScript
  - DHTML
  - Frames

### **Selenium Components**

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- WebDriver
- Selenium Grid



### INSTALLATION OF SELENIUM:

Selenium installation is a 3 step process:

- Install Java SDK
- Install Eclipse
- Install Selenium Webdriver Files

#### STEP:1: Install Java on your computer:



#### STEP: 2: Install Eclipse:

- Follow the below steps to configure Eclipse on your system:
- Navigate to the following URL – <https://www.eclipse.org/downloads/> and select the

download link depending on your system architecture – (32 Bit or 64 Bit) and download

- Once the download is over, extract the zipped file and save it to any directory. The root folder is the eclipse.

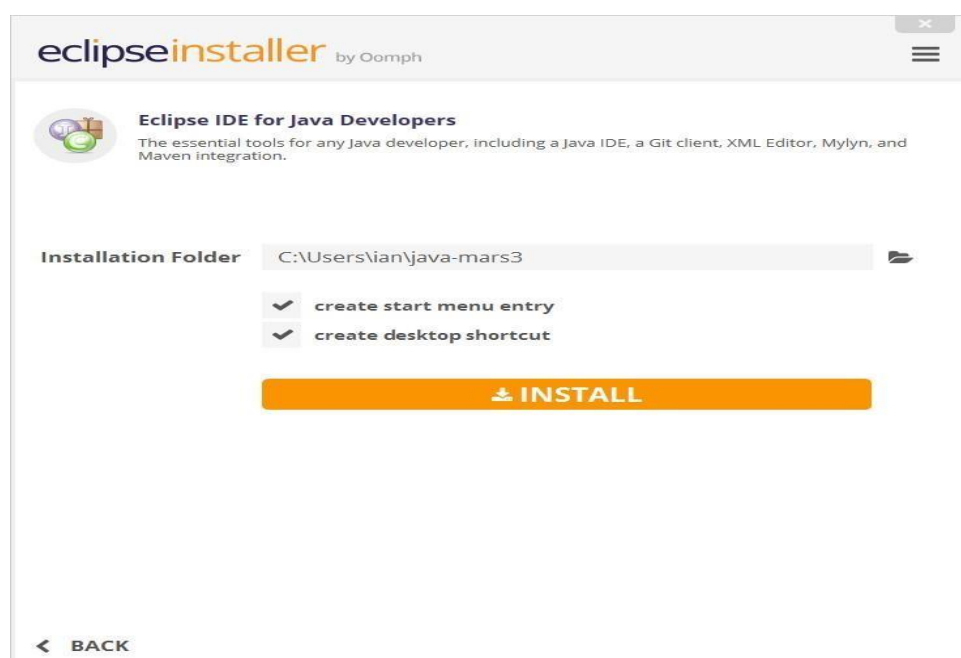
The screenshot shows the Eclipse Foundation website. The header includes the Eclipse Foundation logo, navigation links for Projects, Working Groups, Members, and More, and a Download button. The breadcrumb trail is Home / Downloads / Packages / Eclipse Installer 2020-12 R. The main content area is titled 'Eclipse Installer 2020-12 R' and features a large blue box with the text 'Try the Eclipse Installer 2020-12 R' and 'The easiest way to install and update your Eclipse Development Environment.' It also displays download statistics: 1,418,839 Installer Downloads and 1,634,177 Package Downloads and Updates. To the right, there is a 'Download' section with links for macOS x86\_64, Windows x86\_64, Linux x86\_64, and AArch64. A sidebar on the right contains a Red Hat Developer advertisement and a note about the JRE included in the installer.

- Start the Eclipse Installer executable

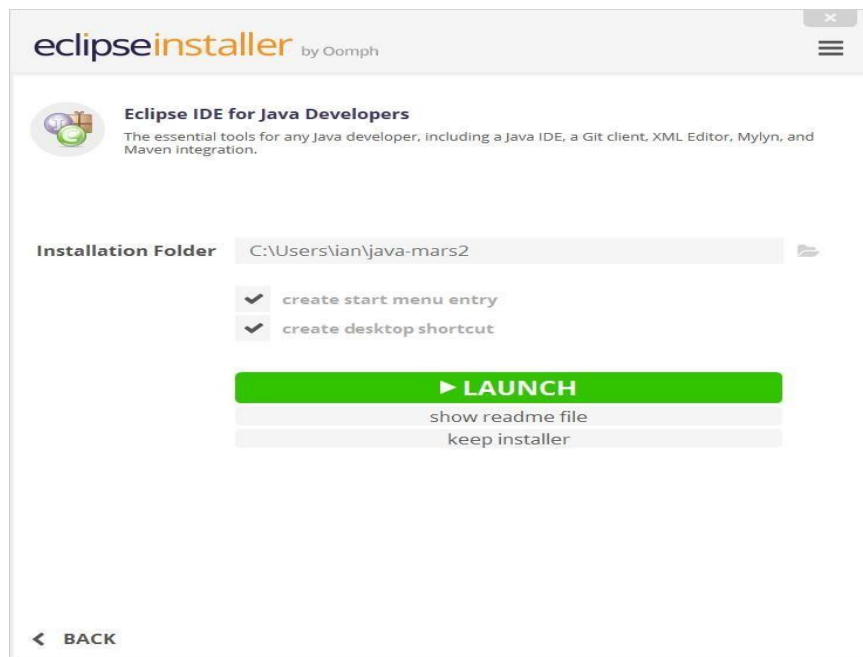




- Select the package to install
- Select your installation folder



- Launch Eclipse



### STEP 3: Download the Selenium Java Client Driver

- Go to the URL: <https://www.selenium.dev/downloads/>
- You will find client drivers for other languages there, but only choose the one for Java.
- This download comes as a ZIP file named "selenium-3.14.0.zip". For simplicity of Selenium installation on Windows 10, extract the contents of this ZIP file on your C drive so that you would have the directory "C:\selenium-3.14.0\". This directory contains all the JAR files that we would later import on Eclipse for Selenium setup.

**Selenium Client & WebDriver Language Bindings**

In order to create scripts that interact with the Selenium Server (Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers.

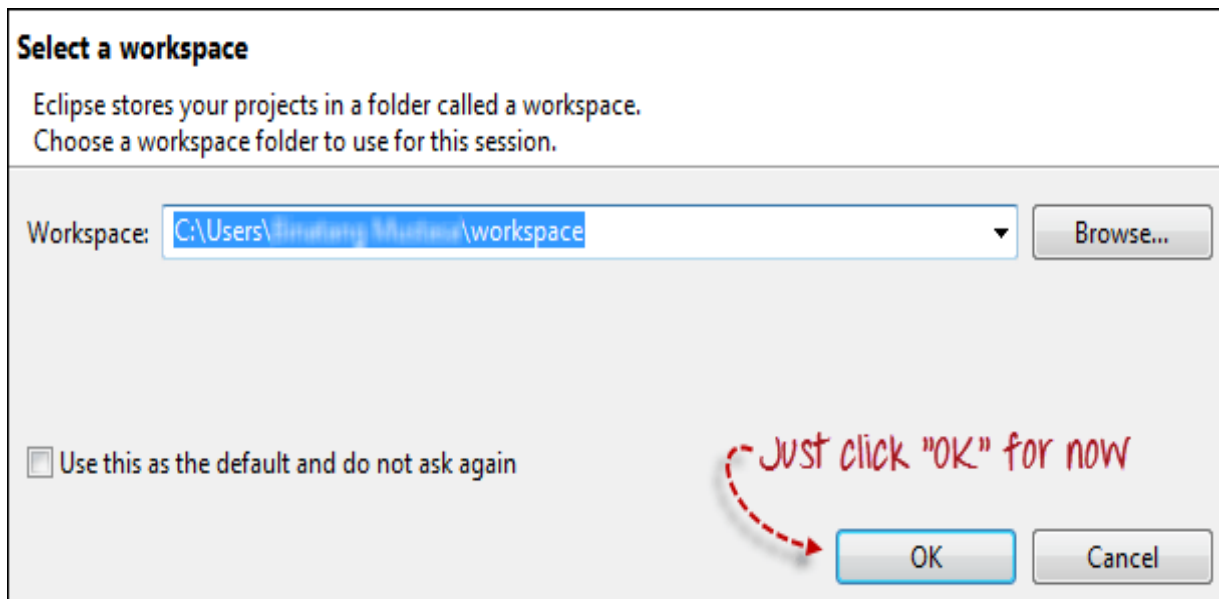
While language bindings for **other languages exist**, these are the core ones that are supported by the main project hosted on GitHub.

LANGUAGE	VERSION	RELEASE DATE	
Ruby	3.142.6	October 04, 2019	<a href="#">Download</a>
JavaScript	4.0.0-alpha.5	September 08, 2019	<a href="#">Download</a>
Java	3.141.59	November 14, 2018	<a href="#">Download</a>
Python	3.141.0	November 01, 2018	<a href="#">Download</a>
C#	3.14.0	August 02, 2018	<a href="#">Download</a>

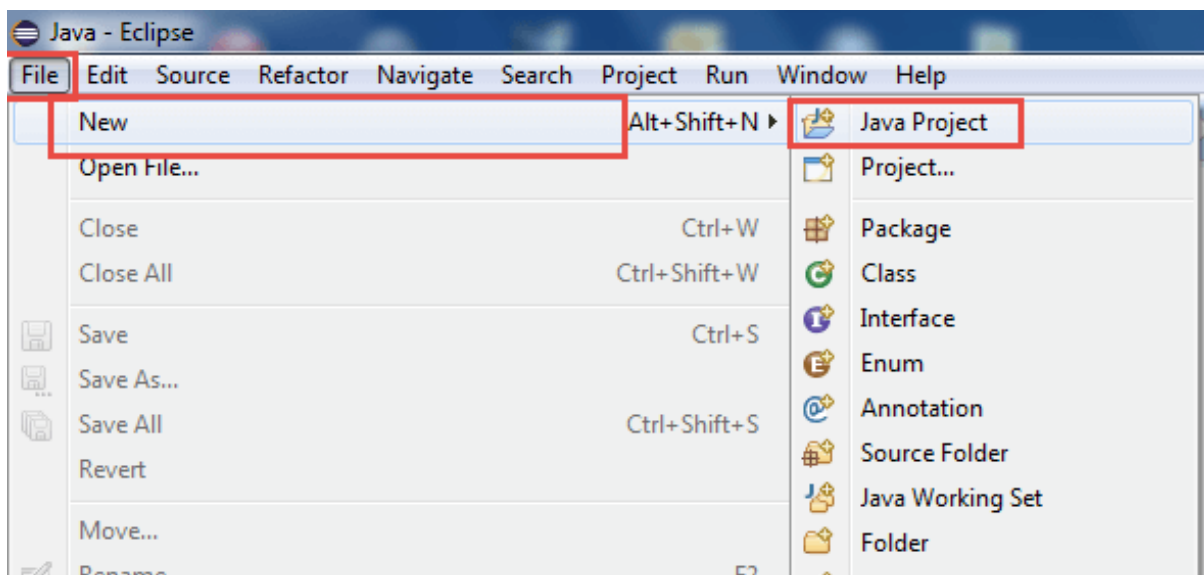
### Configure Eclipse IDE with WebDriver

1. Launch the "eclipse.exe" file inside the "eclipse" folder that we extracted in step 2. If you followed step 2 correctly, the executable should be located on C:\eclipse\eclipse.exe.
2. When asked to select for a workspace, just accept the default location.



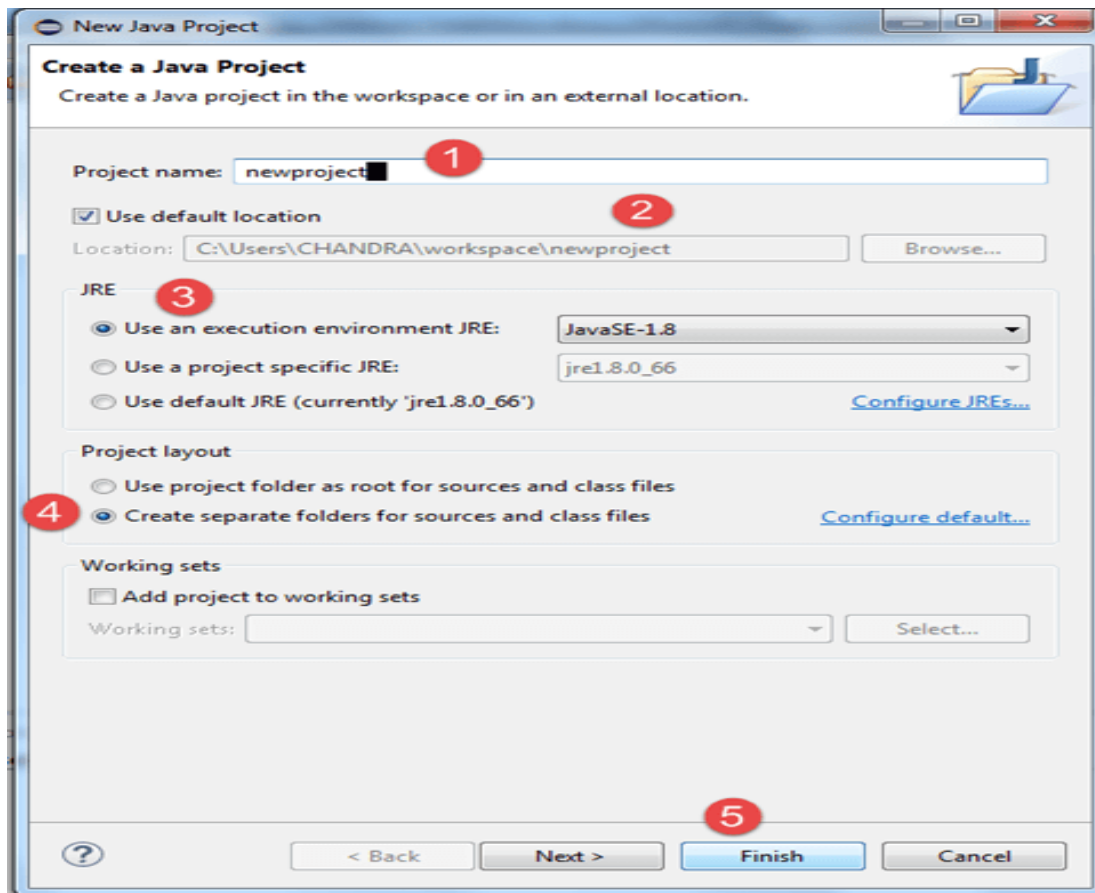


3. Create a new project through File > New > Java Project. Name the project as "newproject".

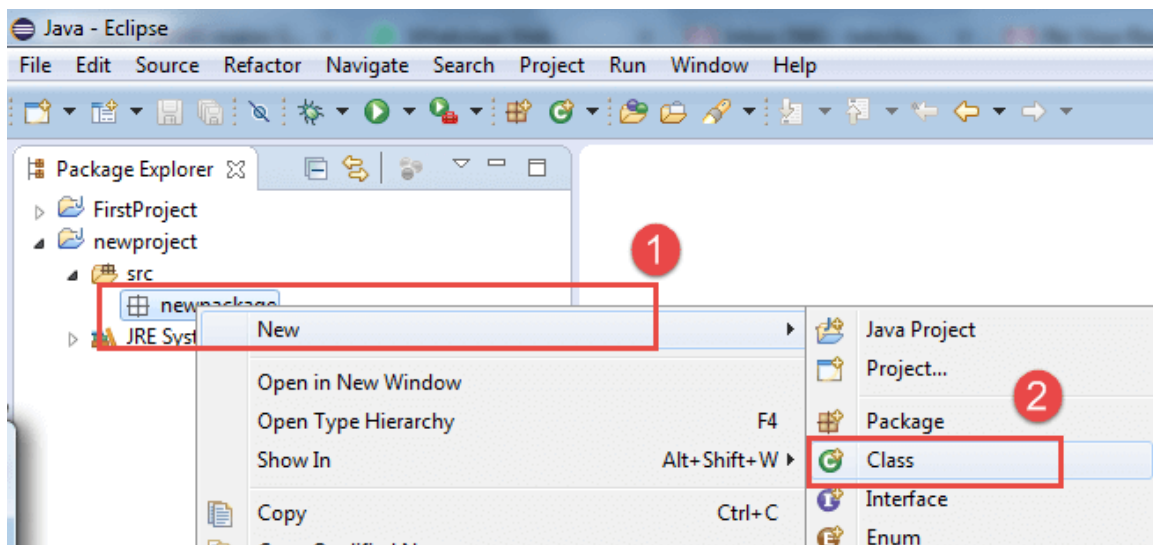


A new pop-up window will open enter details as follow

1. Project Name
2. Location to save project
3. Select an execution JRE
4. Select layout project option
5. Click on Finish button

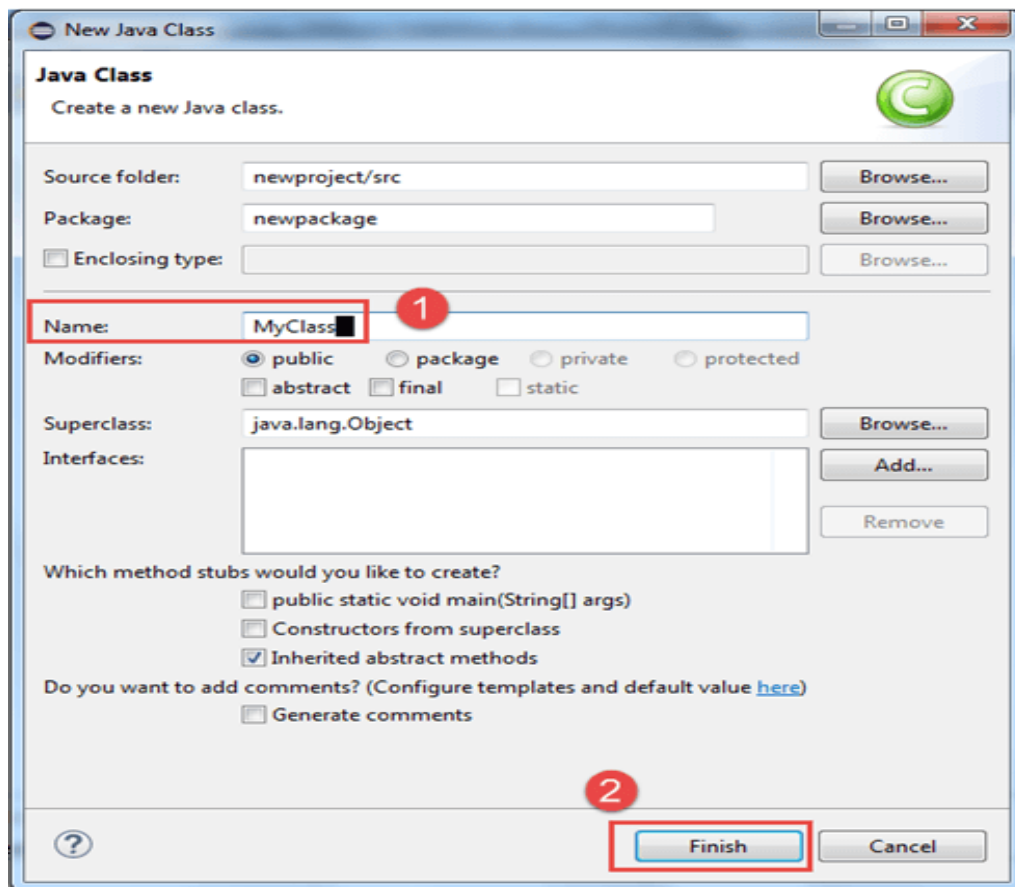


5. Create a new Java class under “new project” by right-clicking on it and then selecting- New > Class, and then name it as "MyClass". Your Eclipse IDE should look like the image below.

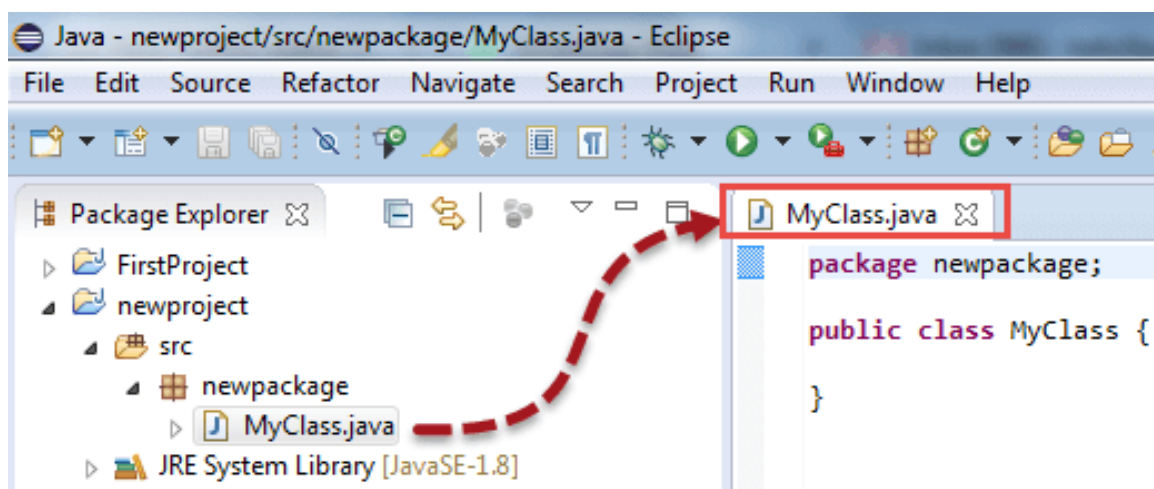


When you click on Class, a pop-up window will open, enter details as

1. Name of the class
2. Click on Finish button



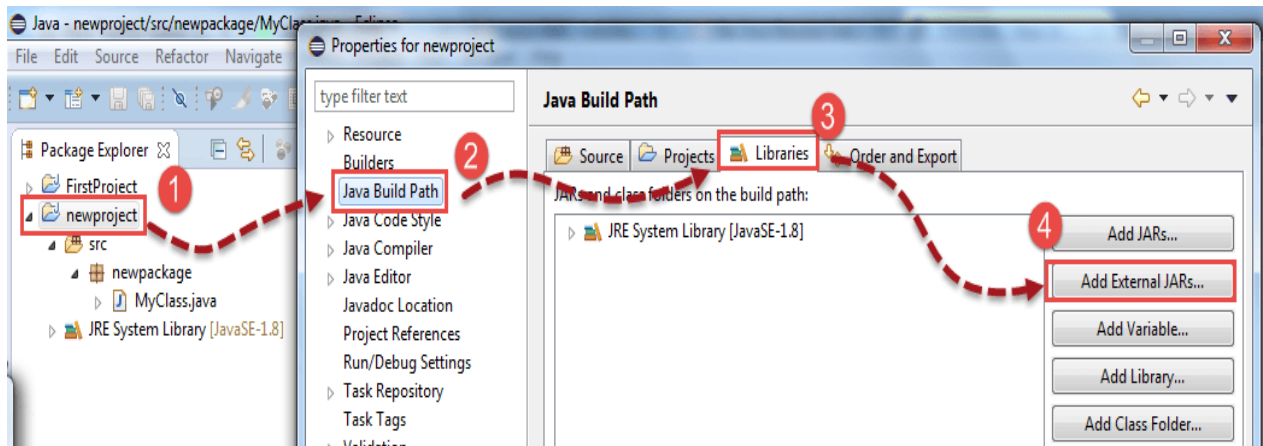
- This is how it looks like after creating class.



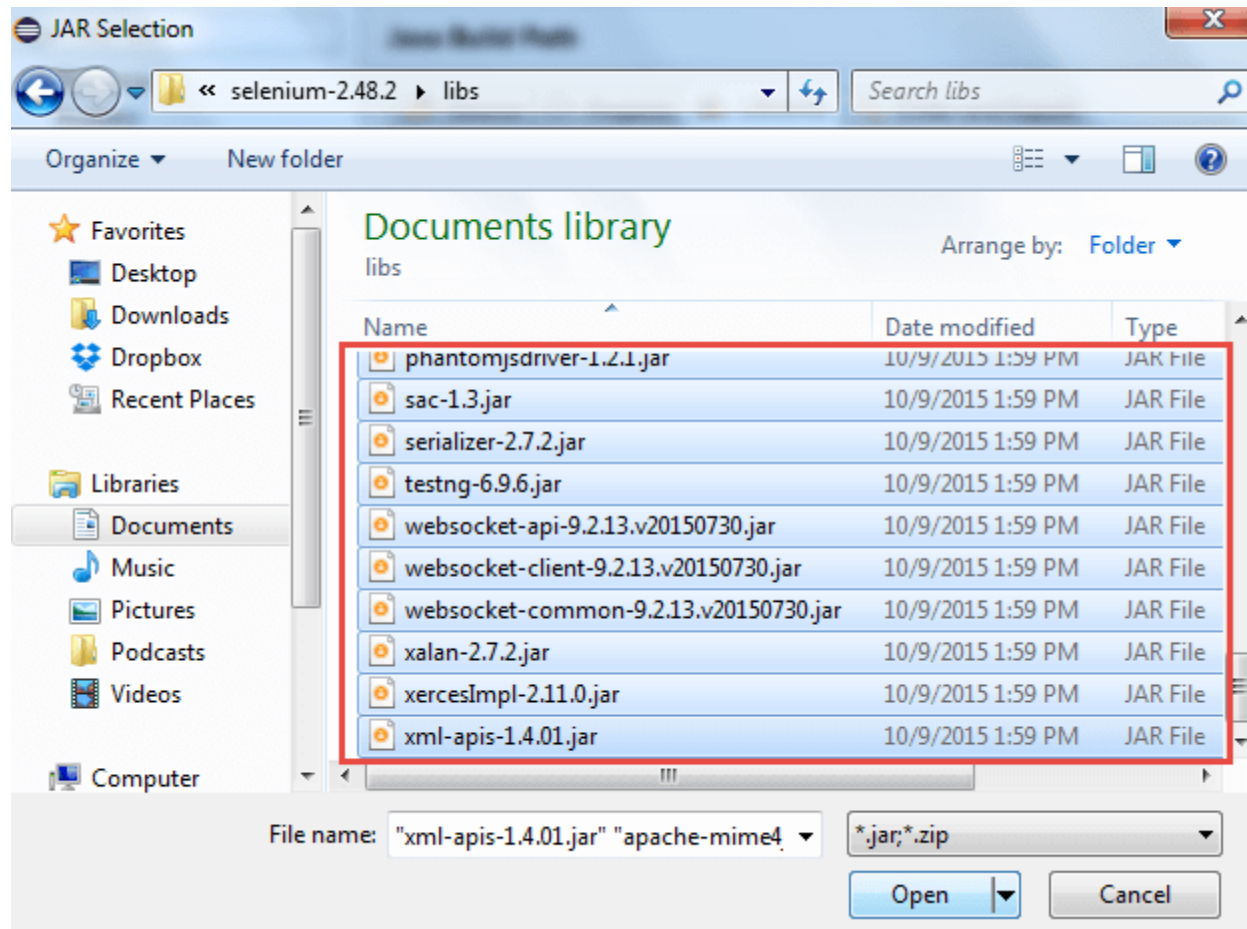
Now selenium WebDriver's into Java Build Path

In this step,

1. Right-click on "newproject" and select **Properties**.
2. On the Properties dialog, click on "Java Build Path".
3. Click on the **Libraries** tab, and then Click on "Add External JARs.."



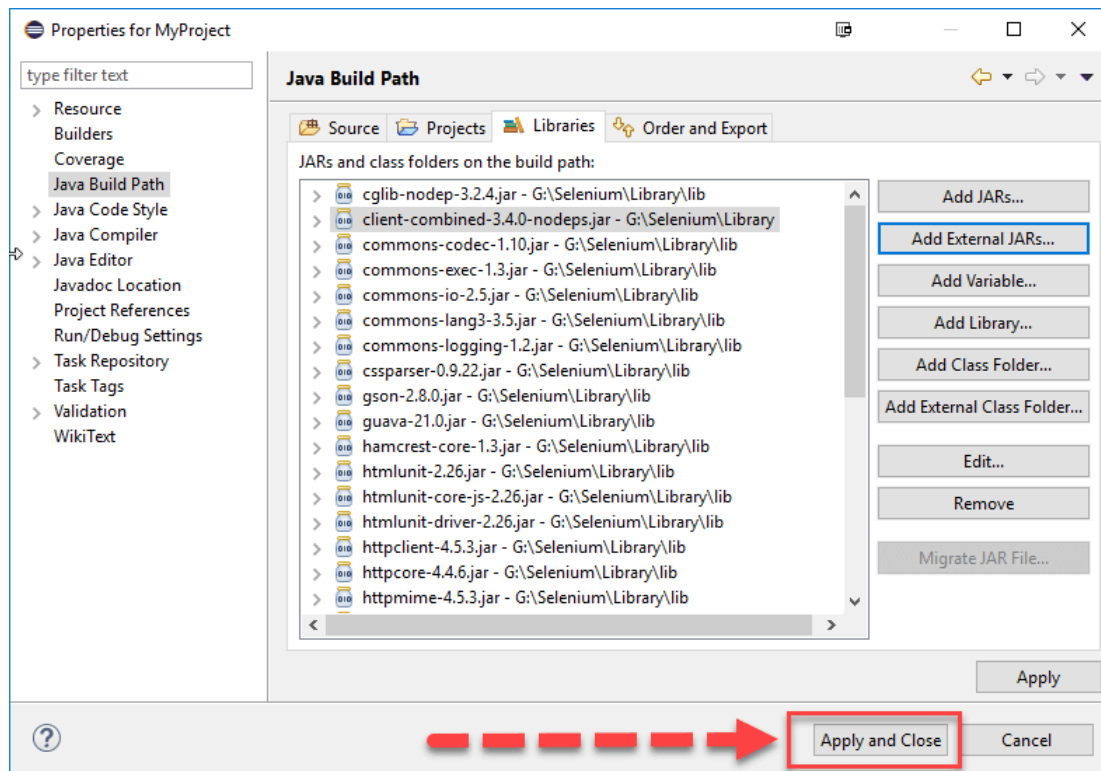
When you click on "Add External JARs.." It will open a pop-up window. Select the JAR files you want to add.



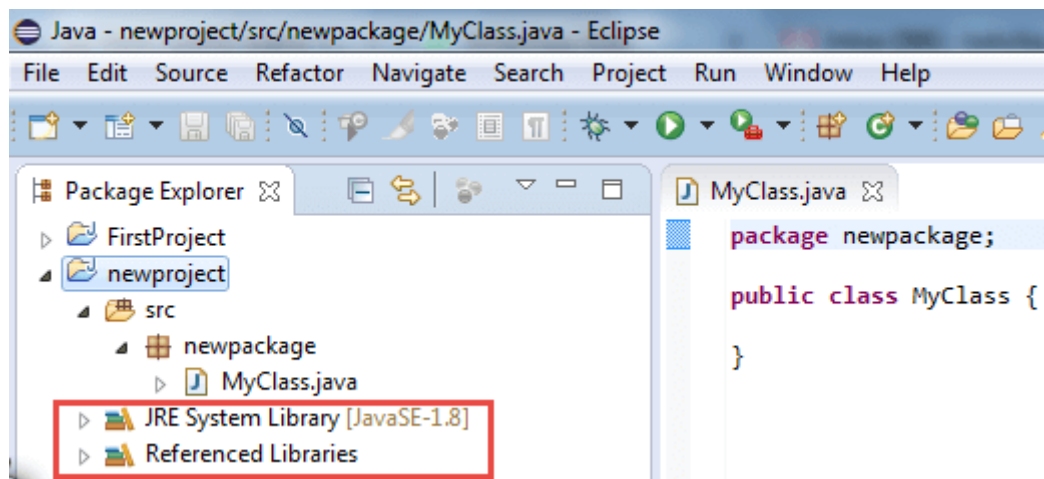
4. After selecting jar files, click on OK button.

Select all files inside the lib folder.

Once done, click "Apply and Close" button



5. Add all the JAR files inside and outside the "libs" folder. Your Properties dialog should now look similar to the image below.



6. Finally, click OK and we are done importing Selenium libraries into our project.

## FINAL STEP:

- Download – ChromeDriver – Webdriver for Chrome:
- Select your Chrome driver according to your chrome version.
- Extract and save it in C:\\ChromeDriver\\chromedriver.exe.

# ChromeDriver - WebDriver for Chrome

CHROMEDRIVER

CAPABILITIES & CHROME OPTIONS

CHROME EXTENSIONS

CHROMEDRIVER CANARY

CONTRIBUTING

▼ DOWNLOADS

VERSION SELECTION

▼ GETTING STARTED

ANDROID

CHROME OS

▼ LOGGING

PERFORMANCE LOG

MOBILE EMULATION

## Downloads

### Current Releases

- If you are using Chrome version 89, please download [ChromeDriver 89.0.4389.23](#)
- If you are using Chrome version 88, please download [ChromeDriver 88.0.4324.96](#)
- If you are using Chrome version 87, please download [ChromeDriver 87.0.4280.88](#)
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.

If you are using Chrome from Dev or Canary channel, please following instructions on the [ChromeDriver Canary](#) page.

For more information on selecting the right version of ChromeDriver, please see the [Version Selection](#) page.

## Matrix Multiplication

**Ex no: 01**

**Date:**

**AIM:**

To write a C program for matrix multiplication to understand the causes of failures.

**ALGORITHM:**

The matrix multiplication can only be performed, if it satisfies this condition. Suppose two matrices are A and B, and their dimensions are A (m x n) and B (p x q) the resultant matrix can be found if and only if n = p. Then the order of the resultant matrix C will be (m x q).

Given two Matrices A[ ][ ] and B[ ][ ]. Your task is to compute the product matrix that multiplies the given two matrices and stores the multiplied matrix in a new matrix C[ ][ ].

$$C[i,j] = C[i,j] + A[i,k] + B[k,j]$$

**Possible reasons for Matrix multiplication failure:**

**Failure case: 1:** The main reason is for failure is, the number of columns in first matrix is not equal to the number of rows in second matrix.

**Failure case:2:** One or more values entered is/are not an integer.

**PROGRAM:**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
void main ()
{
    int a[10][10], b[10][10], m[10][10], i, j, p, q, r, s, k;
    char x[10], y[10];
    int t = 0, fl = 0, l, ain = 0, bin = 0;
    printf ("Enter the size of AMatrix\n");
    scanf ("%d%d", &p, &q);
    printf ("Enter the size of B matrix:")
    scanf ("%d %d", &r, &s);
    if (q == r)
    {
        printf ("Test case 1: Matrix multiplication can be
        done\n");
        printf ("Case 1: Success\n");
        printf ("Enter the elements
        of matrix A:\n"); for (i = 0; i < p; i++)
        {
            for (j = 0; j < q; j++)
            {
                fl = 0;
                ain = 0;
```

```

scanf ("%s", x);
l = strlen (x);
while (x[ain++] != '\0')
{
    if (x[ain] == '.' )
    {
        fl = 1;
        break;
    }
}
if (fl == 0)
{
    a[i][j] = atoi (x);
}
else
{
    t = 1;
}
}
}

```

```

printf ("Enter the elements of matrix B:\n");
for (i = 0; i < r; i++)
{
    for (j = 0; j < s; j++)
    {
        fl = 0;
        bin = 0;
        scanf ("%s", y);
        l = strlen (y);
        while (y[bin++] != '\0')
        {
            if (y[bin] == '.')
            {
                fl = 1;
                break;
            }
        }
        if (fl == 0)
        {
            b[i][j] = atoi (y);
        }
        else
        {
            t = 1;
        }
    }
}

```



```

    }
}
if (t == 0)
{
    for (i = 0; i < p; i++)
    {
        for (j = 0; j < s; j++)
        {
            m[i][j] = 0;
            for (k = 0; k < q; k++)
            {
                m[i][j] = m[i][j] + a[i][k] * b[k][j];
            }
        }
    }

    printf ("The result of Matrix Multiplication is:\n");
    for (i = 0; i < p; i++)
    {
        for (j = 0; j < s; j++)
        {
            printf ("%d\t", m[i][j]);
        }
        printf ("\n");
    }
}
else
{
    printf
        ("Test case 3: Matrix multiplication is not possible\nOne or more values is not an
integer\nCase 3: Failure\n");
    getch ();
}
}
else
printf
    ("Test case 2: Matrix multiplication is not possible\nThe columns of A matrix is not equal
to rows of B matrix\nCase 2: Failure\n");
getch ();
}

```

**OUTPUT:**

```
Enter the size of A Matrix
2 3
Enter the size of B Matrix
2 1
Test case 2: Matrix multiplication is not possible
The columns of A matrix is not equal to rows of B matrix
Case 2: Failure
```

```
Enter the size of A Matrix
3 2
Enter the size of B Matrix
2 2
Test case 1: Matrix multiplication can be done
Case 1: Success
Enter the elements of matrix A:
1 5
3 4
7 1
Enter the elements of matrix b:
2 3
4 1
matrix multiplication is:
6      20
18     16
42      4
```

```
Enter the size of A Matrix
2 2
Enter the size of B Matrix
2 3
Test case 1: Matrix multiplication can be done
Case 1: Success
Enter the elements of matrix A:
1 2
1.5 2
Enter the elements of matrix B:
2 1 3
2 6 1.2
Test case 3: Matrix multiplication is not possible
One or more values is not an integer
Case 3: Failure
```

**Result:**

Thus the c program for Matrix Multiplication to understand the causes of failure is written and executed successfully.

## Binary Search

**Ex no:02**

**Date:**

**AIM:**

To write a C program for binary search – Path Testing

**ALGORITHM:**

***Binary Search:***

Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

***Path Testing:***

- The objective of path testing is to ensure that the set of test cases is such that each path through the program is executed at least once.
- The starting point for path testing is a program flow graph that shows nodes representing program decisions and arcs representing the flow of control.
- Statements with conditions are therefore nodes in the flow graph.
- Cyclomatic complexity = Number of edges - Number of nodes +2.

**PROGRAM:**

```
#include<stdio.h>
int binsrc(int x[], int low, int high, int key)
{
    int mid;
    while(low<=high)
    {
        mid = (low+high)/2;
        if(x[mid]==key)
            return mid;
        if(x[mid]<key)
            low=mid+1;
        else
            high =mid1;
    }
}
```

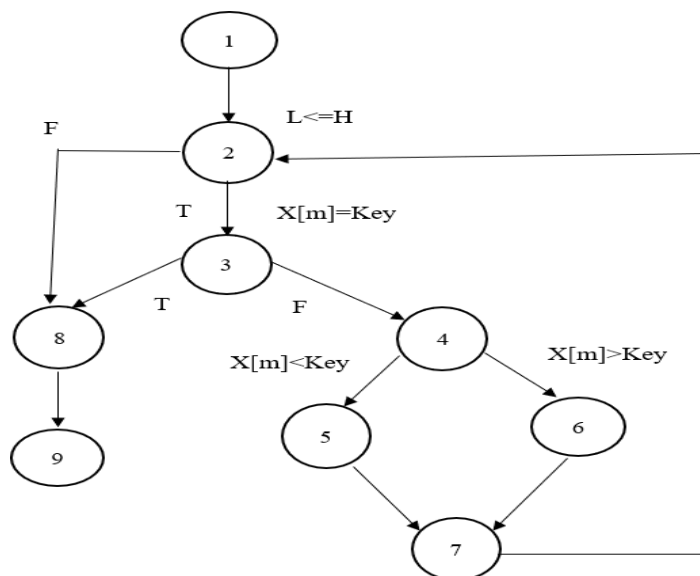
```

        return -1;
    }
int main()
{
    int a[200], key, i, n, succ;
    printf("Enter the n value");
    scanf("%d", &n);
    if(n>0)
    {
        printf("Enter the elements in ascending order\n");
        for(i=0; i<n; i++)
            scanf("%d", &a[i]);
        printf("Enter the key element to be searched\n");
        scanf("%d", &key);
        succ = binsrc(a, 0, n-1, key);
        if(succ>=0)
            printf("Element found in position = %d\n", succ+1);
        else
            result
    }
    else
        printf("number of element should be greater than zero \n ")
        return 0;
}

```

**BINARY SEARCH FUNCTION WITH LINE NUMBER:**

Program	Line.no.
int binsrc(int x[], int low, int high, int key)	
{	
int mid;	1
while(low<=high)	2
{	
mid = (low+high)/2;	
if(x[mid]==key)	3
return mid;	8
if(x[mid]<key)	4
low=mid+1;	5
else	
high=mid-1;	6
}	7
return -1;	8
}	9

**GRAPH FOR BINARY SEARCH:**

**Independent Paths:**

#Edges = 11

#Nodes = 9

#P = 1

 $V(G) = E - N + 2P$  $= 11 - 9 + 2$  $= 4$ **P1:** 1-2-3-8-9**P2:** 1-2-3-4-5-7-2**P3:** 1-2-3-4-6-7-2**P4:** 1-2-8-9**Pre-Conditions / Issues:**

- Array has elements in Ascending order: **T/F**
- Key element is in the Array: **T/F**
- Array has ODD number of elements: **T/F**

**Test Cases – Binary Search:**

Paths	Inputs		Expected Output	Remarks
	X[ ]	Key		
<b>P1:</b> 1-2-3-8-9	{ 10,20,30,40,50}	30	Success	Key $\in$ X[ ] and Key == X[mid]
<b>P2:</b> 1-2-3-4-5-7-2	{ 10,20,30,40,50}	20	Repeat and Success	Key < X[mid] Search 1 <sup>st</sup> Half
<b>P3:</b> 1-2-3-4-6-7-2	{ 10,20,30,40,50}	40	Repeat and Success	Key > X[mid] Search 2 <sup>nd</sup> Half
<b>P4:</b> 1-2-8-9	{ 10,20,30,40,50}	60 or 05	Repeat and Failure	Key $\notin$ X[ ]
<b>P4:</b> 1-2-8-9	Empty	Any Key	Failure	Empty List

**Result:**

Thus the c program for Binary Search is written and executed successfully.

## Boundary Value Analysis

**Ex no:03**

**Date:**

**AIM:**

To write a C program to derive test cases based on Boundary Value Analysis.

**ALGORITHM:**

**Our objective is to design the boundary value test cases:**

Boundary value analysis is a software testing technique in which tests are designed to include representatives of boundary values in a range. A boundary value analysis has a total of  $4*n+1$  distinct test cases, where  $n$  is the number of variables in a problem.

Here we have to consider all the three variables and design all the distinct possible test cases. We will have a total of 13 test cases as  $n = 3$ .

**Quadratic equation will be of type:  $ax^2+bx+c=0$**

Roots are **real** if  $(b^2 - 4ac) > 0$

Roots are **imaginary** if  $(b^2 - 4ac) < 0$

Roots are **equal** if  $(b^2 - 4ac) = 0$

Equation is **not quadratic** if  $a = 0$

**Designing the test cases:**

For each variable we consider below 5 cases:

$a_{min} = 0$

$a_{min+1} = 1$

$a_{nominal} = 50$

$a_{max-1} = 99$

$a_{max} = 100$

When we are considering these 5 cases for a variable, rest of the variables have the nominal values, like in the above case where the value of 'a' is varying from 0 to 100, the value of 'b' and 'c' will be taken as the nominal or average value. Similarly, when the values of variable 'b' are changing from 0 to 100, the values of 'a' and 'c' will be nominal or average i.e., 50.

**PROGRAM:**

```
#include <stdio.h>
#include<stdlib.h>
#include<math.h>
int D;
int a,b,c;
int testcase=1;
void nature_of_roots(int a, int b, int c)
{
    if (a == 0) {
        printf("Not a Quadratic Equation\n");
        return;
    }
```



```

D = b * b - 4 * a * c;
if (D > 0) {
    printf("Real Roots\n");
    else if (D == 0) {
        printf("Equal Roots\n");
    }
    else {
        printf("Imaginary Roots\n");
    }
}
// Function to check for all testcases
void checkForAllTestCase()
{
    printf("Testcase\ta\tb\tc\tActual Output\n");
    while (testcase <= 13) {
        if (testcase == 1) {
            a = 0;
            b = 50;
            c = 50;
        }
        else if (testcase == 2) {
            a = 1;
            b = 50;
            c = 50;
        }
        else if (testcase == 3) {
            a = 50;
            b = 50;
            c = 50;
        }
        else if (testcase == 4) {
            a = 99;
            b = 50;
            c = 50;
        }
        else if (testcase == 5) {
            a = 100;
            b = 50;
            c = 50;
        }
        else if (testcase == 6) {
            a = 50;
            b = 0;
            c = 50;
        }
        else if (testcase == 7) {
            a = 50;
            b = 1;
            c = 50;
        }
        else if (testcase == 8) {

```

```

        a = 50;
        b = 99;
        c = 50;
    }
    else if (testcase == 9) {
        a = 50;
        b = 100;
        c = 50;
    }
    else if (testcase == 10) {
        a = 50;
        b = 50;
        c = 0;
    }
    else if (testcase == 11) {
        a = 50;
        b = 50;
        c = 1;
    }
    else if (testcase == 12) {
        a = 50;
        b = 50;
        c = 99;
    }
    else if (testcase == 13) {
        a = 50;
        b = 50;
        c = 100;
    }
    printf("%d\\t\\t%d\\t\\t%d\\t\\t", testcase,a,b,c);
    nature_of_roots(a, b, c);
    testcase++;
}
}
int main()
{
    checkForAllTestCase();
    return 0;
}

```

### OUTPUT:

Testcase	a	b	c	Actual Output
1	0	50	50	Not a Quadratic Equation
2	1	50	50	Real Roots
3	50	50	50	Imaginary Roots
4	99	50	50	Imaginary Roots
5	100	50	50	Imaginary Roots
6	50	0	50	Imaginary Roots
7	50	1	50	Imaginary Roots
8	50	99	50	Imaginary Roots
9	50	100	50	Equal Roots
10	50	50	0	Real Roots
11	50	50	1	Real Roots
12	50	50	99	Imaginary Roots
13	50	50	100	Imaginary Roots

### Result :

Thus the c program for Boundary Value Analysis is written and Executed successfully.

## Cause Effect Graph

**Ex no :04**

**Date :**

**AIM:**

Write a C program for cause effect graph to check whether defect is found in the program.

**ALGORITHM:**

Cause-Effect graph is a testing technique that aids in choosing test cases that logically relate Causes (inputs) to Effects (outputs) to produce test cases.

**STEP: 1:** Recognize and describe the input conditions (causes) and actions (effect).

**STEP: 2:** Build up a cause-effect graph

**STEP: 3:** Convert cause-effect graph into a decision table

**STEP: 4:** 11 test cases according to the 11 rules.

**PROGRAM:**

```
#include<conio.h>
#include<stdio.h>
void main()
{
    int a,b,c,result;
    printf(" Enter the values of a, b and c : = ");
    scanf("%d %d %d", &a,&b,&c);
    if(((a+b)>c)&&((b+c)>a)&&((c+a)>b))
    {
        if((a==b)&&(b==c))
            printf("\n It is an Equilateral Triangle");
        else if((a==b)||((b==c)||((c==a)))
            printf("\n It is an Isosceles Triangle");
        else
            printf("\n It is a Scalene Triangle");
    }
    else
        printf("\n Not a Triangle");
    getch();
}
```

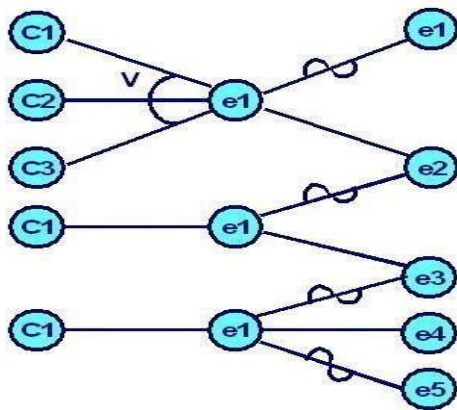
**STEP: 1:** Recognize and describe the input conditions (causes) and actions (effect).

The <b>causes</b> designated by letter “C” are as under		The <b>effects</b> designated by letter “e” are as under	
<b>C1</b>	Side “x” is less than sum of “y” and “z”	<b>e1</b>	Not a triangle

<b>C2</b>	Side “y” is less than sum of “x” and “z”	<b>e2</b>	Scalene triangle
<b>C3</b>	Side “z” is less then sum of “x” and “y”	<b>e3</b>	Isosceles triangle
<b>C4</b>	Side “x” is equal to side “y”	<b>e4</b>	Equilateral Triangle
<b>C5</b>	Side “x” is equal to side “z”	<b>e5</b>	Impossible
<b>C6</b>	Side “y” is equal to side “z”		

**STEP: 2:** Build up a cause-effect graph

*Cause – Effect graph for Triangle:*



**STEP: 3:** Convert cause-effect graph into a decision table

Conditions	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
<b>C1: X &lt; Y+Z?</b>	0	1	1	1	1	1	1	1	1	1	1
<b>C2: X &lt; Y+Z?</b>	X	0	1	1	1	1	1	1	1	1	1
<b>C3: X &lt; Y+Z?</b>	X	X	0	1	1	1	1	1	1	1	1
<b>C3: X=Y?</b>	X	X	X	1	1	1	1	0	0	0	0
<b>C4: X=Y?</b>	X	X	X	1	1	0	0	1	1	0	0
<b>C5: X=Y?</b>	X	X	X	1	1	0	0	1	1	0	0
<b>C6: X=Y?</b>	X	X	X	1	0	1	0	1	0	1	0
<b>e1: Not a Triangle</b>	1	1	1								
<b>e2: Scalene</b>											1
<b>e3: Isosceles</b>							1		1	1	
<b>e4: Equilateral</b>				1							
<b>e5: Impossible</b>					1	1		1			

**STEP: 4:** 11 test cases according to the 11 rules.

Test case	X	Y	Z	Expected Output
1	4	1	2	Not a triangle
2	1	4	2	Not a triangle
3	1	2	4	Not a triangle
4	5	5	5	Equilateral
5	?	?	?	Impossible
6	?	?	?	Impossible
7	2	2	3	Isosceles
8	?	?	?	Impossible
9	2	3	2	Isosceles
10	3	2	2	Isosceles
11	3	4	5	Scalene

**Result:**

Thus the c program for Cause Effect Graph is written and executed successfully .

## Dynamic Data Flow Testing

**Ex no:05**

**Date:**

**AIM:**

Write a C program to perform data flow testing for the given code and find out all d-use Pairs.

**ALGORITHM:**

**Dynamic Data Flow Testing:**

Dynamic data-flow testing is performed with the intention to uncover possible bugs in data usage during the execution of the code. The test cases are designed in such a way that every definition of data variable to each of its use is traced and every use is traced to each of its definition. Various strategies are employed for the creation of test cases. All these strategies are defined below.

**All-du Paths (ADUP):**

It states that every du-path from every definition of every variable to every use of that definition should be exercised under some test. It is the strongest data flow testing strategy since it is a superset of all other data flow testing strategies. Moreover, this strategy requires the maximum number of paths for testing.

**DU-PATHS:**

In a path segment if the variable is defined earlier and the last link has a computational use of that variable then that path is termed as du path.

A sub-path in the flow is defined to go from a point where a variable is “defined”, to a point where it is “referenced, that is, where it is “used” – whatever kind of usage it is. Such a sub-path is called a “definition-use pair” or “du-pair”. The pair is made up of a “definition” of a variable and a “use” of the variable,

**DC-PATHS:**

- A path  $(i, n_1, \dots, n_m, j)$  is called a *definition-clear path* with respect to  $x$  from node  $i$  to node  $j$  if it contains no definitions of variable  $x$  in nodes  $(n_1, \dots, n_m, j)$ .
- The family of data flow criteria requires that the test data execute definition-clear paths from each node containing a definition of a variable to specified nodes containing *c-use* and edges containing *p-use* of that variable.

**PROGRAM CODE:**

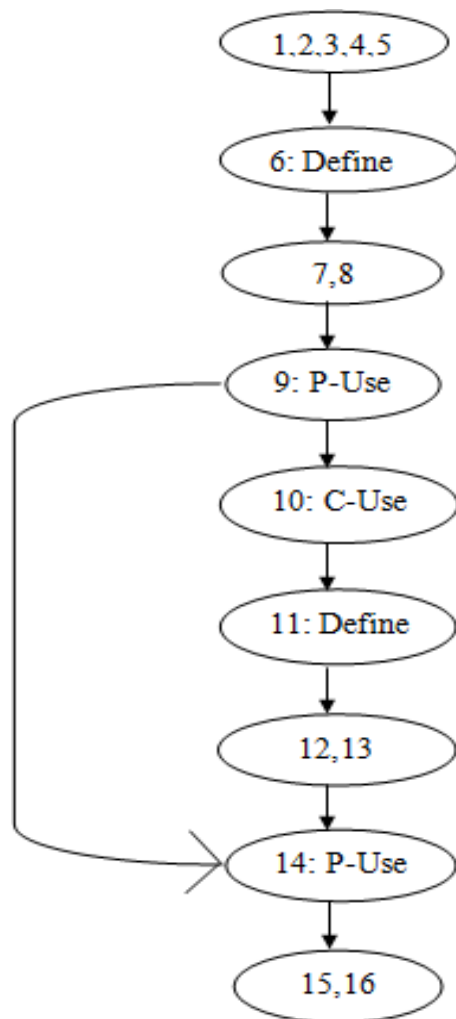
```
#include <stdio.h>
#include <conio.h>
    i. void main()
    ii. {
    iii. int a,b, t;
    iv. clrscr();
```

```

v.   printf("Enter the first number");
vi.  scanf("%d", &b);
vii. printf("Enter the second number");
viii. scanf("%d",&b);
ix.  if (a<b) {
x.    t=a;
xi.   a=b;
xii.  b=t;
xiii. }
xiv.  printf("%d%d", a,b);
xv.   getch();
xvi.  }

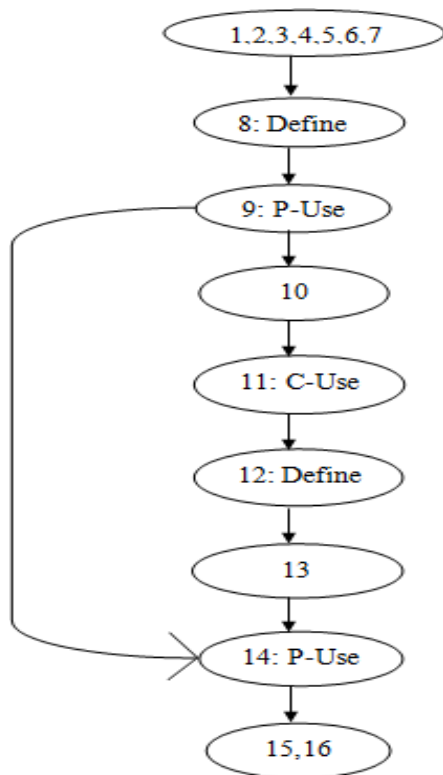
```

**Variable 'a' Data Flow Graph:**

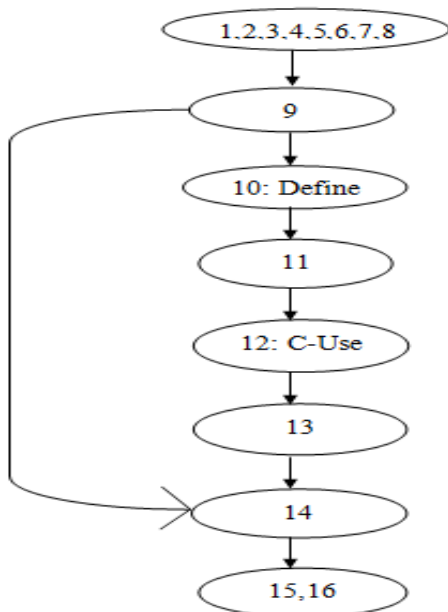




**Variable 'b' Data Flow Graph:**



**Variable 't' Data Flow Graph:**



VARIABLE	DEFINED AT	USED AT
a	6,11	9,10,14
b	8,12	9,11,14
t	10	12

**Result:**

Thus the c program for Dynamic Data Flow is written and executed successfully.

## Loop testing

**Ex no:06**

**Date:**

**AIM:**

Write a C program to demonstrate the working of the looping constructs.

**ALGORITHM:**

**1. While Loops:**

*Syntax :*

while( condition )

body;

**2. Do While loops**

*Syntax:*

do {

body;

} while( condition );

**PROGRAM:**

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
int i = 0;
```

```
printf("Before loop, i=%d\n", i);
```

```
while( i < 5 )
```

```
printf( "i = %d\n", i++ );
```

```
printf( "After loop, i = %d\n", i );
```

```
return 0;
```

```
}
```

**Output:**

```
Before loop, i=0
```

```
i = 0
```

```
i = 1
```

```
i = 2
```

```
i = 3
```

```
i = 4
```

```
After loop, i = 5
```

**Do While loops in C:**

**Code:**

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
int month;
```

```

do {
    printf( "Please enter the month of your birth > " );
    scanf( "%d", &month );
} while ( month < 1 || month > 12 );
return 0
}

```

### OUTPUT:

Code runs till a month between 1-12 is entered

```

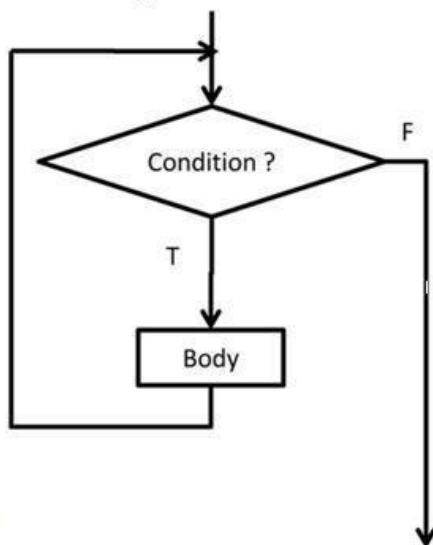
Please enter the month of your birth > 13
Please enter the month of your birth > -1
Please enter the month of your birth > 15
Please enter the month of your birth > -3
Please enter the month of your birth > 12

...Program finished with exit code 0
Press ENTER to exit console.

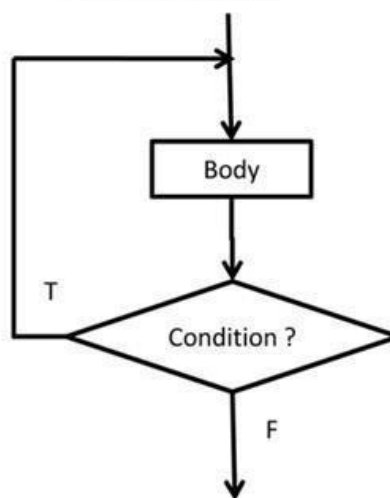
```

### While Vs Do-While Constructs:

while( condition )  
body;



do {  
body;  
} while( condition );



### Result:

Thus a c program for understand the working of loop is written and executed successfully

## SELENIUM JAVA

### Checkbox counter

**Ex no:07**

**Date:**

**AIM:**

To write and test a program to count number of check boxes on the page checked and unchecked count using selenium tool.

**ALGORITHM:**

- We can count the total number of checkboxes in a page in Selenium with the help of **find\_elements** method. While working on any checkboxes, we will always find an attribute type in the html code and its value should be checkbox.
- This characteristic is only applicable to checkboxes on that particular page and to no other types of UI elements like edit box, link and so on.
- To retrieve all the elements with attribute type = '**checkbox**', we will use **find\_elements\_by\_xpath()** method. This method returns a list of web elements with the type of xpath specified in the method argument. In case there are no matching elements, an empty list will be returned.
- After the list of checkboxes are fetched, in order to count its total numbers, we need to get the size of that list. The size of the list can be obtained from the **len()** method of the list data structure.
- Finally, this length is printed on the console.

**Syntax:**

```
driver.find_elements_by_xpath("//input[@type='checkbox']")
```

Boolean **isSelected()**:

- This method determines if an element is *selected* or *not*. It *returns true* if the element is *selected* and *false* if it is *not*. It is widely used on *check boxes*, *radio buttons* and *options* in a select.

**PROGRAM CODE:**

```
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
public class Checkbox_multi {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driverC
        :\\chromedriver\\chromedriver.exe");
        ChromeDriver driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(10,
        TimeUnit.SECONDS);driver.manage().window().maximize();
        driver.get("C:\\Users\\Dr.PD\\Desktop\\New.html");
        List<org.openqa.selenium.WebElement>CheckBoxes =
            driver.findElements(By.xpath("//input[@ty
            pe='checkbox']"));
        System.out.println("Number of Check boxes : "+
```

```

        Integer.toString(CheckBoxes.size()));
    for(int i=0; i<CheckBoxes.size(); i=i+4)
    {
        CheckBoxes get(i).click();
    }
int checkedCount=0,uncheckedCount=0;
for(int i=0; i<CheckBoxes.size(); i++){
    System.out.println(i+" checkbox is selected
    "+CheckBoxes.get(i).isSelected());if(CheckBoxes.get(i).isSelected()){
        checkedCount++;
    }else{
        uncheckedCount++;
    }
}
System.out.println("number of selected checkbox: "+checkedCount);
System.out.println("number of unselected checkbox: "+uncheckedCount);
}
}

```

## **OUTPUT:**



```

Starting ChromeDriver 88.0.4324.27 (6347fe8b1e48bd0c54d07dc55ca011cf40861c9-refs/branch-heads/4324@{#450}) on port 47282
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Jan 22, 2021 11:05:44 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Number of Check boxes : 2
0 checkbox is selected true
1 checkbox is selected false
number of selected checkbox: 1
number of unselected checkbox: 1

```

### **HTML CODE TO CREATE A WEBPAGE CONTAINING CHECKBOX:**

```
<!DOCTYPE HTML>
<html lang = "en">
<head>
<title>formDemo.html</title>
<meta charset = "UTF-8" />
</head><body>
<h1>Form Demo</h1>
<form>
<fieldset>
<legend>Selecting elements</legend>
<p>
<label>Check boxes</label>
<input type = "checkbox"
      id = "chkEggs"
      value = "greenEggs" />
<label for = "chkEggs">Green Eggs</label>
<input type = "checkbox"
      id = "chkHam"
      value = "ham" />
<label for = "chkHam">Ham</label>
</p>
</fieldset>
</form>
</body>
</html>
```

### **Result:**

Thus a java program to count a number of check boxes on the web page checked and unchecked count using selenium tool is written and executed successfully.

## Objects on the web page

**Ex no:08**

**Date:**

**AIM:**

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

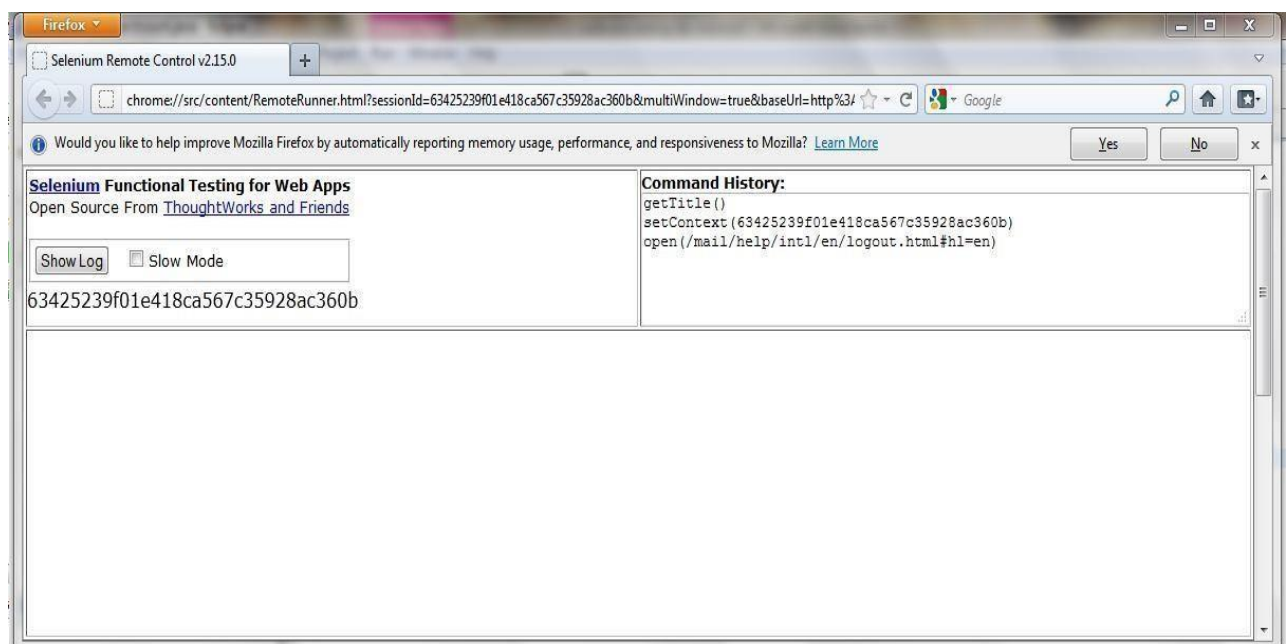
**PROGRAM:**

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.By;

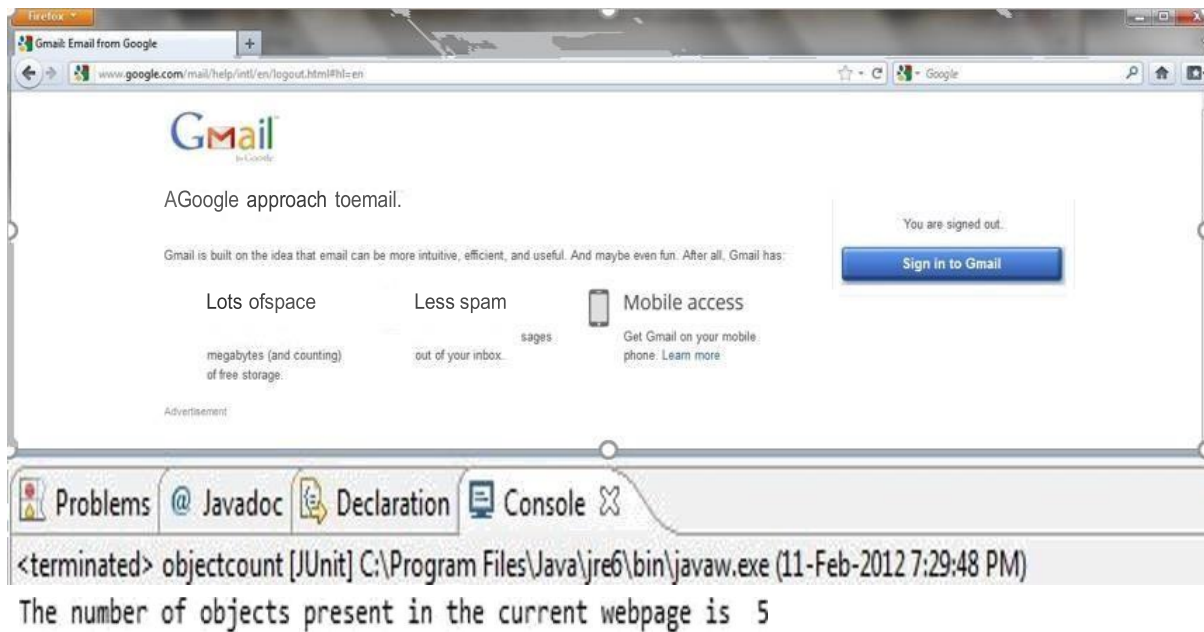
public class ObjectCount {
    public static void main(String[] args) {
        System.setProperty("webdriver.gecko.driver", "path/to/geckodriver.exe");
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com/mail/help/intl/en/logout.html#hl=en");
        driver.manage().window().maximize();
        int num = driver.findElements(By.tagName("p")).size();
        System.out.println("The number of p elements present are " + num);
        driver.quit();
    }
}
```

**OUTPUT:**

Before running the code makes sure your selenium RC server is running.







**Result:**

Thus a java program to provide total number of objects present on the web page is written and executed successfully.

## Login a web page

**Ex no:09**

**Date:**

**AIM:**

To write and test a program to login a specific web page using selenium tool.

**ALGORITHM:**

Before we perform automation testing for login validation using Selenium and Java, there are some basic steps that need to be followed for whichever test case you intend to write. If you follow them, you will never have incomplete test cases in your automation suite:

1. Create a Selenium WebDriver instance.
2. Configure your browser if required (for example, maximize browser, disable browser notifications, etc.).
3. Navigate to the required URL (webpage).
4. Locate the HTML element.
5. Perform the action on the located HTML element.
6. Verify and validate the action (concluded step).
7. Take screenshots and generate the report using a framework for the test cases.

**PROGRAM CODE:**

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class LoginUsingSelenium {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver",
"C:\\chromedriver\\chromedriver.exe");
        ChromeDriver driver=new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.linkedin.com/login
");

        WebElement username=driver.findElement(By.id("username"));
        WebElement password=driver.findElement(By.id("password"));
        WebElement login=driver.findElement(By.xpath("//button[text()='Sign in']"));

        username.sendKeys("preethi92.mit@gmail.com");
        password.sendKeys("password");
        login.click();

        String actualUrl="https://www.linkedin.com/feed/";
        String expectedUrl= driver.getCurrentUrl();

        if(actualUrl.equalsIgnoreCase(expectedUrl))
        {
```

```

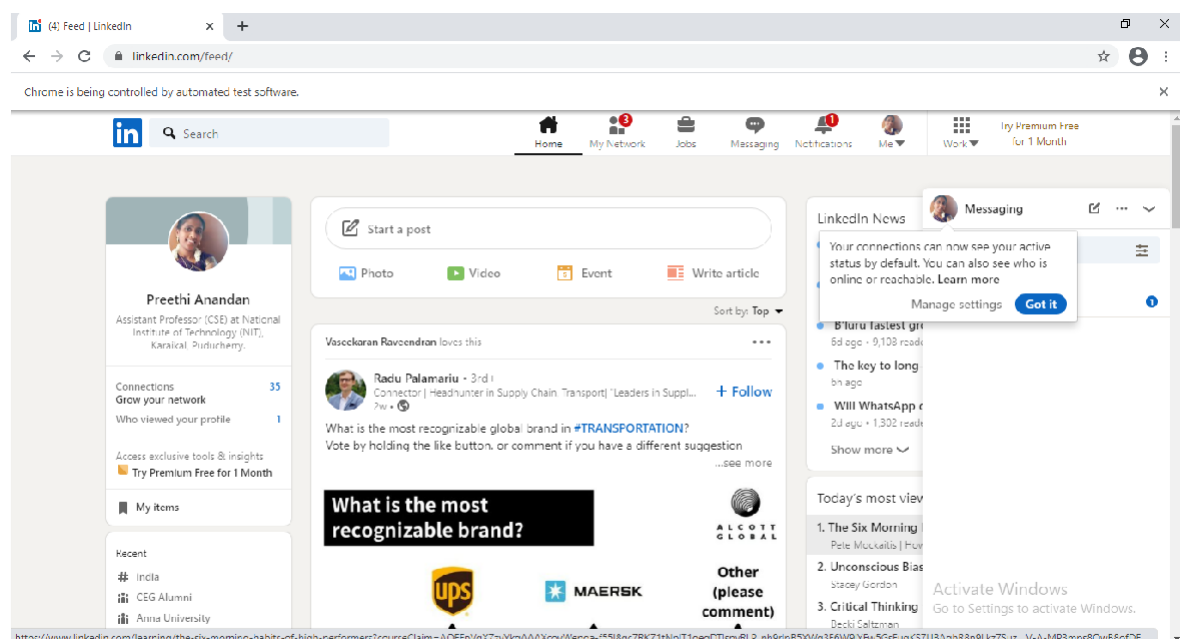
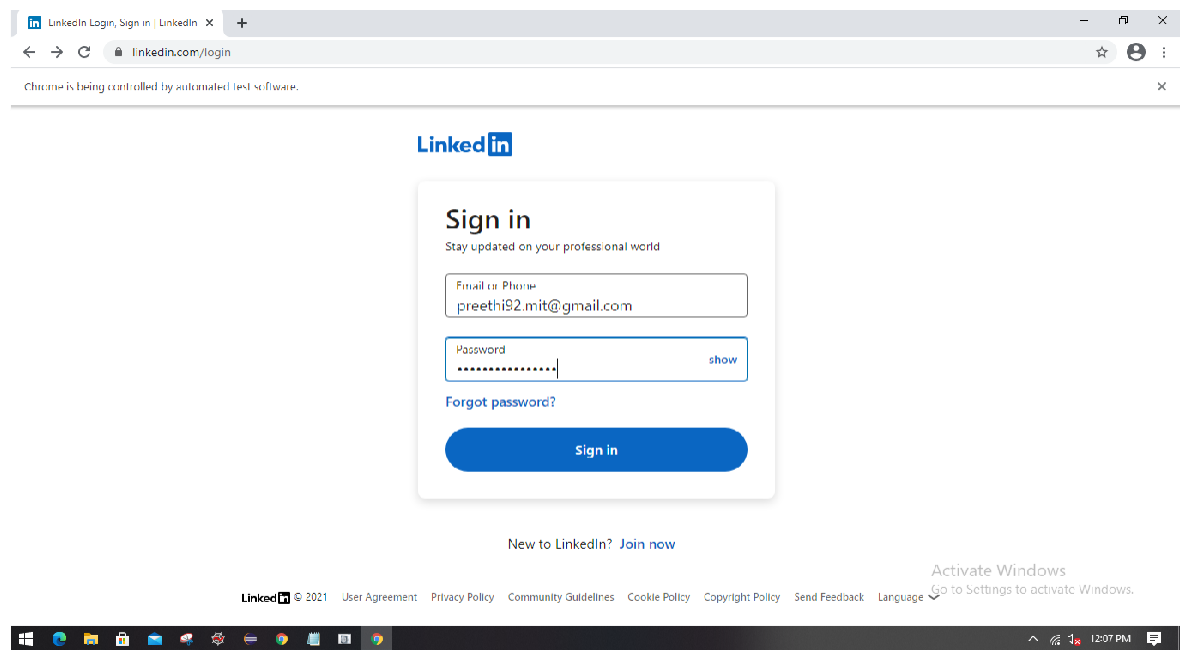
    System.out.println("Test passed");
}
else
{
    System.out.println("Test failed");
}

}

}

```

## OUTPUT:



```

Starting ChromeDriver 88.0.4324.27 (6347fe8b1e48bd0c54d07dc55ca011cf40861c9-refs/branch-heads/4324@{#450}) on port 43331
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Jan 22, 2021 12:11:59 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Test passed

```

**Result:**

Thus a java program to login to a web page is written and executed successfully.

## Read a Excel file using selenium

**Ex no:10**

**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:**

- This program reads an excel file so Create an Excel file Student.xls under the desktop as follows.
- Since we are dealing with the excel, we need to add the external jar file jxl-2.6 in to the java project.( download jxl-2.6 and Follow the same steps as we used to add selenium server and selenium java driver to our java project.)

**PROGRAM CODE:**

```
import java.io.File;
import java.io.IOException;
import jxl.Cell;
import jxl.CellType;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;

public class student_excel_read {
    private String inputFile;
    public void setInputFile(String inputFile) {
        this.inputFile = inputFile;
    }
    public void read() throws IOException {
        File inputWorkbook = new File(inputFile);
        Workbook w;
        boolean flag=false;
        int count=0;
        try {
            w = Workbook.getWorkbook(inputWorkbook);
            // Get the first sheet
            Sheet sheet = w.getSheet(0);
            // Loop over first 10 column and lines
            for (int j = 0; j < sheet.getRows(); j++) {
                for (int i = 0; i < sheet.getColumns(); i++)
                {
                    Cell cell = sheet.getCell(i, j);
                    if (cell.getType() == CellType.NUMBER) {
                        if(Integer.parseInt(cell.getContents())>60){ flag = true;
                        if(flag == true){
                            count++; flag=false;
                        }
                    }
                }
            }
        }
    }
}
```

```

        break;
    }
}
}
}
System.out.println("Total number of students who scored more than 60 in one or
more subjects is: " +count);
    } catch (BiffException e) { e.printStackTrace();
    }
}
    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        // TODO Auto-generated method stub
        student_excel_read test = new student_excel_read();
        test.setInputFile("C:\\Users\\nehru\\Desktop\\Studentnew.xls");
        test.read();
    }
}

```

## **OUTPUT:**

## **INPUT FILE:**

	A	B	C	D
1	Student Name	Subject1	Subject2	Subject3
2	Student1	35	67	60
3	Student2	36	46	57
4	Student3	59	48	58
5	Student4	80	80	60
6	Student5	35	29	28
7	Student6	46	40	39
8	Student7	59	53	52
9	Student8	74	68	67
10	Student9	91	85	84

## **AFTER EXECUTION:**

```

Console
<terminated> student_excel_read [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (05-Feb-2021, 6:19:39 pm - 6:19:40 pm)
Total number of students who scored more than 60 in one or more subjects is: 4

```

**Result:**

Thus a java program to select the number of students who have scored more than 60 in any one subject is written and executed successfully.