NEW HORIZON COLLEGE OF ENGINEERING

Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

PRACTICAL RECORD BOOK

Name	SOUMYADIP ROY				
USN	1NH19CS175	**Year**	2022 - 2023		
Program	B.E. in CSE	**Semester**	7	**Section**	C
Course	SOFTWARE TESTING LAB	**Course Code**	20CSL75A		

NEW HORIZON COLLEGE OF ENGINEERING

INSTITUTE VISION AND MISSION

VISION

To emerge as an institute of eminence in the fields of engineering, technology and management in serving the industry and the nation by empowering students with a high degree of technical, managerial and practical competence.

MISSION

- To strengthen the theoretical, practical and ethical dimensions of the learning process by fostering a culture of research and innovation among faculty members and students.
- To encourage long-term interaction between the academia and industry through the involvement of the industry in the design of the curriculum and its hands-on implementation.
- To strengthen and mould students in professional, ethical, social and environmental dimensions by encouraging participation in co-curricular and extracurricular activities.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

To emerge as a department of eminence in Computer Science and Engineering in serving the Information Technology Industry and the nation by empowering students with a high degree of technical and practical competence.

MISSION

To strengthen the theoretical and practical aspects of the learning process by strongly encouraging a culture of research, innovation and hands-on learning in Computer Science and Engineering

To encourage long-term interaction between the department and the IT industry, through the involvement of the IT industry in the design of the curriculum and its hands-on implementation

To widen the awareness of students in professional, ethical, social and environmental dimensions by encouraging their participation in co-curricular and extracurricular activities

QUALITY POLICY

To provide services of the highest quality both curricular and co-curricular, so that our students can integrate their skills and serve the industry and society equally well at the global level.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

Engineering Graduates will be able to:

PEO1: Develop Proficiency as computer scientists with an ability to solve a wide range of computational problems in industry, government, or other work environments.

PEO2: Attain the ability to adapt quickly to new environments and technologies, assimilate new information, and work in multi-disciplinary areas with a strong focus on innovation and entrepreneurship.

PEO3: Possess the ability to think logically and the capacity to understand technical problems with computational systems.

PEO4: Possess the ability to collaborate as team members and team leaders to facilitate cutting-edge technical solutions for computing systems and thereby providing improved functionality.

PROGRAM SPECIFIC OUTCOMES (PSOs)

Engineering Graduates will be able to:

PSO1: Ability to design, develop, implement computer programs and use knowledge in various domains to identify research gaps and hence to provide solution to new ideas and innovations.

PSO2: Work with and communicate effectively with professionals in various fields and pursue lifelong professional development in computing.

NEW HORIZON
COLLEGE OF ENGINEERING

Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

Laboratory Certificate

This is to certify that

Mr **Soumyadip Roy** .....

has satisfactorily completed the experiments prescribed by New Horizon College of Engineering, Bangalore Affiliated to Visvesvaraya Technological University

in ..**Software Testing**.. *Laboratory Course for the* ..**7th**.. *semester of Computer Science and Engineering Program.*

Academic Year: 2022 to 2023 (ODD Semester)

Marks Obtained
Max. Marks

Student Name: Soumyadip Roy

USN: 1NH19CS175

Sem/Sec: 7 - C

Course Code: 20CSL75A

Signature of Student

Signature of the Faculty In-charge

Head of the Department

LABORATORY PERFORMANCE EVALUATION SHEET

Name of Student: Soumyadip Roy
USN: 1NH19CS175
Lab Course: SOFTWARE TESTING LAB
Course Code: 20CSL75A
Sem/Sec: 7 - C
Session: ODD Sem 2022-23

CIE - PART A - Record and Performance (Max Marks: 10)

SN	Date of Evaluation	Name of Experiment/ Program	1	2	3	Total	Faculty Signature
Write test cases for the following scenarios							
1.	01-09-2022	ATM System					
2.	01-09-2022	The Triangle Problem					
Demonstrate Black box testing techniques using open-source testing tool - JUnit							
3.	08-09-2022	Boundary Value Analysis (BVA) for the NextDate Function					
4.	15-09-2022	Equivalence Class Partitioning for the NextDate Function					
Demonstrate White box testing techniques using open-source testing tool - ECLemma							
5.	22-09-2022	The Triangle Problem					
6.	01-10-2022	The NextDate Function					
Demonstration of Selenium IDE & Webdriver for conducting test on websites							
7.	06-10-2022	Using Selenium IDE to conduct a test for any web site					
8.	13-10-2022	Using Selenium Web driver, automate any web page using Java Script					

SN	Date of Evaluation	Name of Experiment / Program	1	2	3	Total	Faculty Signature
9.	27-10-2022	List the total number of objects present on a web page					
10.	10-11-2022	Demonstrate URL and title check point					
11.	17-11-2022	Demonstrate selecting and deselecting option from multi select dropdown					
12.	24-11-2022	Demonstrate Synchronization.					
Average (out of 10 Marks)							

1. **Conduction of Experiment / Writing the Program: 3 Marks**
2. **Specimen Calculation / Execution: 3 Marks**
3. **Result and Record Writing: 4 Marks**

CIE - PART B - Lab Test (Max Marks: 50)

	Date of Lab Test	Procedure and Write Up (15 Marks)	Conduction and Results (25 Marks)	Viva Voce (10 Marks)	Total (50 Marks)	Faculty Signature
Test 1						
Test 2						
Average (out of 15 Marks)						

CIE - Marks Obtained

CIE-Part A Record and Performance (10 Marks)	CIE-Part B Lab Test (15 Marks)	Total (25 Marks)	Faculty Signature

Exp. No. : 1

Date : 01-09-2022

ATM SYSTEM

Consider any ATM system, design and develop a program in a language of your choice for the same. Cr

the test cases for the following scenarios:

i) Unsuccessful operation due to enter wrong PIN number 3 times.

ii) Unsuccessful operation due to invalid account type.

iii) Successful selection of amount to be withdrawn.

iv) Expected message due to amount to withdraw is greater than possible balance

IMPLEMENTATION:

```
import java.util.*;
import java.lang.*;
public class ATMmain {
	private static int pin = 7563, lim = 0, upin, accountType, balance=5000, operation, wAmt, dAmt;
	public static void main(String args[]){
		Scanner in = new Scanner(System.in);
		System.out.println("Welcome to the ICICI Bank ATM!");
		System.out.println("Please insert your ATM card.");
		while(lim<=3){
			System.out.println("Please enter your PIN.");
			upin = in.nextInt();
			if (upin == pin){
				while(true){
					System.out.println("Select your account type.\n1. Savings\n2.
					Current");
					accountType = in.nextInt();
					if (accountType == 1){
						System.out.println("You have selected Savings Account.");
						break;
					}
					else if(accountType == 2){
						System.out.println("You have selected Current Account.");
						break;
					}
					else{
						System.out.println("You have entered an invalid input. Try
						again.");
					}
				}
				while(true){
					System.out.println("Select an operation:\n1. Check Balance\n2.
					Withdrawal\n3. Deposit\n4. Exit");
					operation = in.nextInt();
					if (operation == 1){
						System.out.println("Your current balance is: "+balance);
```

```
                }
                else if(operation == 2){
                        System.out.println("Enter amount to be withdrawn: ");
                        wAmt = in.nextInt();
                        if(wAmt > balance){
                                System.out.println("The amount to be withdrawn is
                                greater than current balance.");
                        }
                        else if (wAmt > 0){
                                balance -= wAmt;
                                System.out.println("Please collect your cash. Thank
                                you.");
                        }
                        else{
                                System.out.println("Please enter an amount greater
                                than zero.");
                        }
                }
                else if(operation == 3){
                        System.out.println("Enter amount to be deposited: ");
                        dAmt = in.nextInt();
                        if (dAmt > 0){
                                balance += dAmt;
                                System.out.println("Thank you.");
                        }
                        else{
                                System.out.println("Please enter an amount greater
                                than zero.");
                        }
                }
                else if(operation == 4){
                        System.out.println("Thank you for banking with ICICI
                        Bank. Have a nice day!");
                        System.exit(0);
                        break;
                }
                else{
                        System.out.println("You have entered an invalid input. Try
                        again.");
                }
        }
}
else{
        lim++;
        if(lim==1){
                System.out.println("Incorrect PIN. You have 2 more chances after
                which your card will be blocked.");
        }
        if(lim==2){
                System.out.println("Incorrect PIN. You have 1 more chance after
                which your card will be blocked.");
        }
        if(lim==3){
```

```
                    System.out.println("Incorrect PIN. Your card has been blocked.");
                    System.out.println("Please take out your card. Thank you.");
                    System.exit(0);
                }
            }
        }
    }
}
```

TEST CASES:

TEST CASE 1: Validity of ATM PIN entered by the user

Project Information		Test Information	
Project Name:	ATM	Test Name:	Invalid PIN Number
Project ID:	ATM_01	Original Author:	XXX
Test Objective:	This test case is to verify the functionality with invalid pin number		

Case No.	Test Case Description	Test Data	Observed Result	Expected Result	Status (Pass/Fail)
1.	Insert valid card in the insertion point of ATM	Valid ATM card	ATM displays the PIN number entry screen	ATM should display the PIN number entry screen	Pass
2.	Enter incorrect PIN (1st Attempt)	Invalid PIN	ATM does not validate PIN and prompts customer to reenter PIN.	ATM should not validate PIN and prompts customer to reenter PIN.	Pass
3.	Reenter incorrect PIN (2nd Attempt)	Invalid PIN	ATM does not validate PIN and prompts customer to reenter PIN	ATM should not validate PIN and prompts customer to reenter PIN	Pass
4.	Reenter incorrect PIN (3rd Attempt)	Invalid PIN	ATM does not validate PIN and blocks the card	ATM should not validate PIN and blocks the card	Pass
5.	Enter Correct PIN	Valid PIN	ATM validates the PIN and displays the Account type selection module	ATM should validate the PIN and display the Account type selection module	Pass

TEST CASE 2: Validity of Account type selection choice

Project Information		Test Information	
Project Name:	ATM	Test Name:	InvalidAccount Type
Project ID:	ATM_02	Original Author:	Soumyadip Roy
Test Objective:	This test case is to verify the functionality with invalid account type		

Case No.	Test Case Description	Test Data	Observed Result	Expected Result	Status (Pass/Fail)
1.	Check working of Account type selection module	Use entering a valid PIN	The module displays the two Account types: Savings and Current	The module should display the two Account types: Savings and Current	Pass

2.	Check if valid input for Savings Account type works	User selects Savings Account by entering '1'	A message saying "You have selected Savings Account." appears.	A message saying "You have selected Savings Account." should appear.	Pass
3.	Check if valid input for Current Account type works	User selects Current Account by entering '2'	A message saying "You have selected Current Account." appears.	A message saying "You have selected Current Account." should appear.	Pass
4.	Check if invalid inputs are handled	User enters an invalid option	A message saying "You have entered an invalid input. Try again." appears.	A message saying "You have entered an invalid input. Try again." should appear.	Pass
5.	Check working of module for valid input	User enters either '1' or '2'	The ATM navigates the user to the Operation selection module.	The ATM should navigate the user to the Operation selection module.	Pass

TEST CASE 3: Successful selection of amount to be withdrawn.

Project Information		**Test Information**	
Project Name:	ATM	Test Name:	Valid Withdrawal Amount
Project ID:	ATM_03	Original Author:	Soumyadip Roy
Test Objective:	This test case is to verify the functionality of the withdrawal operation using valid inputs		

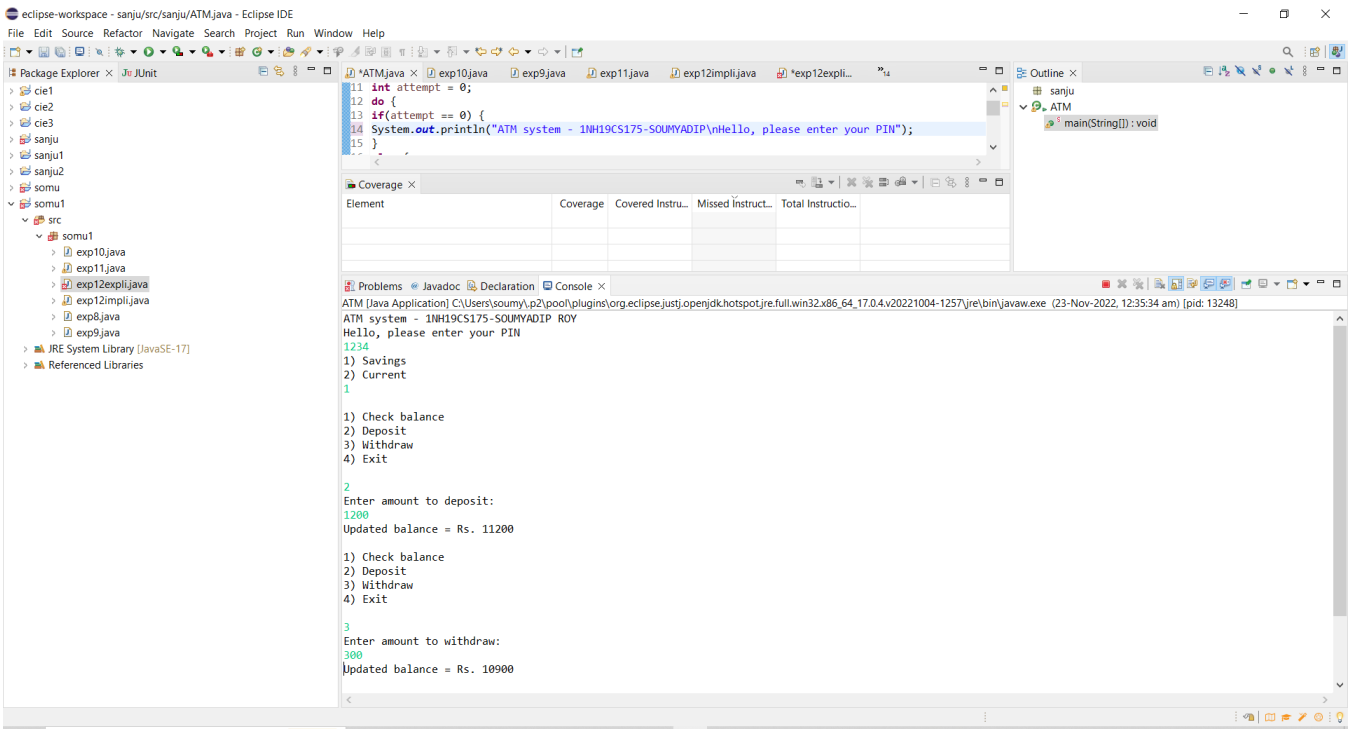
Case No.	**Test Case Description**	**Test Data**	**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
1.	Check working of Withdrawal module	User enters '2' in Operation selection module	A message saying 'Enter amount to be withdrawn:' appears.	A message saying 'Enter amount to be withdrawn:' should appear.	Pass
2.	Check for valid withdrawal amount	User enters '500' as withdrawal amount	A message saying 'Please collect your cash. Thank you.' appears.	A message saying 'Please collect your cash. Thank you.' should appear.	Pass
3.	Check balance for previous Test Case	User enters '1' in Operation selection module	A message saying 'Your current balance is: 4500' appears.	A message saying 'Your current balance is: 4500' should appear.	Pass
4.	Check for valid withdrawal amount	User enters '1000' as withdrawal amount	A message saying 'Please collect your cash. Thank you.' appears.	A message saying 'Please collect your cash. Thank you.' should appear.	Pass
5.	Check balance for previous Test Case	User enters '1' in Operation selection module	A message saying 'Your current balance is: 3500' appears.	A message saying 'Your current balance is: 3500' should appear.	Pass

TEST CASE 4: Expected message due to amount to withdraw is greater than possible balance or is a negative amount.

Project Information		Test Information	
Project Name:	ATM	Test Name:	Invalid Withdrawal Amount
Project ID:	ATM_04	Original Author:	Soumyadip Roy
Test Objective:	This test case is to verify the functionality of the withdrawal operation using invalid inputs		

Case No.	Test Case Description	Test Data	Observed Result	Expected Result	Status (Pass/Fail)
1.	Check for invalid withdrawal amount greater than balance	User enters '6000' as withdrawal amount	A message saying 'The amount to be withdrawn is greater than current balance.' appears.	A message saying 'The amount to be withdrawn is greater than current balance.' should appear.	Pass
2.	Check for invalid negative withdrawal amount	User enters '-500' as withdrawal amount	A message saying 'Please enter an amount greater than zero.' appears.	A message saying 'Please enter an amount greater than zero.' should appear.	Pass
3.	Check for invalid withdrawal amount greater than balance	User enters '7000' as withdrawal amount	A message saying 'The amount to be withdrawn is greater than current balance.' appears.	A message saying 'The amount to be withdrawn is greater than current balance.' should appear.	Pass
4.	Check for invalid negative withdrawal amount	User enters '-100' as withdrawal amount	A message saying 'Please enter an amount greater than zero.' appears.	A message saying 'Please enter an amount greater than zero.' should appear.	Pass
5.	Check for invalid withdrawal amount greater than balance	User enters '8000' as withdrawal amount	A message saying 'The amount to be withdrawn is greater than current balance.' appears.	A message saying 'The amount to be withdrawn is greater than current balance.' should appear.	Pass

EXECUTION



RESULT & DISCUSSION

Test Report:

1. Number of Test Cases Executed : 20
2. Number of Test Cases Passed : 20
3. Number of Test Cases Failed : 0

Exp. No. : 2

Date : 01-09-2022

TRIANGLE PROBLEM

Design and develop a program in a language of your choice to solve the triangle problem defined as follows: Accept three integers which are supposed to be the three sides of triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Create the test cases for the following scenarios:

i) Represents not a triangle

ii) Represents a valid scalene triangle

iii) Represents a valid equilateral triangle

iv) Represents a valid isosceles triangle

Execute the test cases manually and discuss the result.

IMPLEMENTATION

```
import java.util.*;
public class triangle {

	public static void main(String[] args) {
		Scanner in = new Scanner(System.in);
		int a, b, c;
		while(true){
			System.out.print("Enter value of 1st side: ");
			a = in.nextInt();
			System.out.print("Enter value of 2nd side: ");
			b = in.nextInt();
			System.out.print("Enter value of 3rd side: ");
			c = in.nextInt();
			if(a>=1 && a<=200 && b>=1 && b<=200 && c>=1 && c<=200){
				if((a < b+c) && (b < a+c) && (c < b+a)){
					if(a == b && b == c)
						System.out.println("Given dimensions form an equilateral
						triangle!");
					else if(a==b || b==c || c==a)
						System.out.println("Given dimensions form an isosceles
						triangle!");
					else
						System.out.println("Given dimensions form a scalene
						triangle!");
					break;
				}
				else {
					System.out.println("Given dimensions do not form a triangle!");
					break;
				}
			}
```

```
                else {
                        System.out.println("Enter a valid input!\n");

                }
            }
        }
}
```

TEST CASES

TEST CASE 1: Represents not a triangle

Project Information		Test Information	
Project Name:	Triangle	Test Name:	Not a Triangle
Project ID:	Triangle_01	Original Author:	Soumyadip Roy
Test Objective:	This test case is to verify the functionality using inputs that do not form a triangle or are invalid		

Case No.	Test Case Description	Test Data	Observed Result	Expected Result	Status (Pass/Fail)
1.	Enter invalid inputs	Side a = -10 Side b = 20 Side c = -50	A message saying 'Enter a valid input!' appears.	A message saying 'Enter a valid input!' should appear.	Pass
2.	Enter invalid inputs	Side a = -10 Side b = 20 Side c = 100	A message saying 'Enter a valid input!' appears.	A message saying 'Enter a valid input!' should appear.	Pass
3.	Enter valid inputs that do not form a triangle	Side a = 10 Side b = 10 Side c = 30	A message saying 'Given dimensions do not form a triangle!' appears.	A message saying 'Given dimensions do not form a triangle!' should appear.	Pass
4.	Enter valid inputs that do not form a triangle	Side a = 199 Side b = 1 Side c = 200	A message saying 'Given dimensions do not form a triangle!' appears.	A message saying 'Given dimensions do not form a triangle!' should appear.	Pass
5.	Enter valid inputs that do not form a triangle	Side a = 3 Side b = 2 Side c = 7	A message saying 'Given dimensions do not form a triangle!' appears.	A message saying 'Given dimensions do not form a triangle!' should appear.	Pass

TEST CASE 2: Represents a valid scalene triangle

Project Information		Test Information	
Project Name:	Triangle	Test Name:	Scalene Triangle
Project ID:	Triangle_02	Original Author:	Soumyadip Roy
Test Objective:	This test case is to verify the functionality using inputs that form a scalene triangle		

Case No.	Test Case Description	Test Data	Observed Result	Expected Result	Status (Pass/Fail)
1.	Enter valid inputs	Side a = 10 Side b = 20 Side c = 15	A message saying 'Given dimensions form a scalene triangle!' appears.	A message saying 'Given dimensions form a scalene triangle!' should appear.	Pass

2.	Enter valid inputs	Side a = 150 Side b = 140 Side c = 130	A message saying 'Given dimensions form a scalene triangle!' appears.	A message saying 'Given dimensions form a scalene triangle!' should appear.	Pass
3.	Enter valid inputs	Side a = 90 Side b = 91 Side c = 92	A message saying 'Given dimensions form a scalene triangle!' appears.	A message saying 'Given dimensions form a scalene triangle!' should appear.	Pass
4.	Enter valid inputs	Side a = 199 Side b = 198 Side c = 200	A message saying 'Given dimensions form a scalene triangle!' appears.	A message saying 'Given dimensions form a scalene triangle!' should appear.	Pass
5.	Enter valid inputs	Side a = 3 Side b = 5 Side c = 7	A message saying 'Given dimensions form a scalene triangle!' appears.	A message saying 'Given dimensions form a scalene triangle!' should appear.	Pass

TEST CASE 3: Represents a valid equilateral triangle

Project Information		**Test Information**	
Project Name:	Triangle	Test Name:	Equilateral Triangle
Project ID:	Triangle_03	Original Author:	Soumyadip Roy
Test Objective:	This test case is to verify the functionality using inputs that form an equilateral triangle		

Case No.	**Test Case Description**	**Test Data**	**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
1.	Enter valid inputs	Side a = 10 Side b = 10 Side c = 10	A message saying 'Given dimensions form an equilateral triangle!' appears.	A message saying 'Given dimensions form an equilateral triangle!' should appear.	Pass
2.	Enter valid inputs	Side a = 100 Side b = 100 Side c = 100	A message saying 'Given dimensions form an equilateral triangle!' appears.	A message saying 'Given dimensions form an equilateral triangle!' should appear.	Pass
3.	Enter valid inputs	Side a = 200 Side b = 200 Side c = 200	A message saying 'Given dimensions form an equilateral triangle!' appears.	A message saying 'Given dimensions form an equilateral triangle!' should appear.	Pass
4.	Enter valid inputs	Side a = 1 Side b = 1 Side c = 1	A message saying 'Given dimensions form an equilateral triangle!' appears.	A message saying 'Given dimensions form an equilateral triangle!' should appear.	Pass
5.	Enter valid inputs	Side a = 75 Side b = 75 Side c = 75	A message saying 'Given dimensions form an equilateral triangle!' appears.	A message saying 'Given dimensions form an equilateral triangle!' should appear.	Pass

TEST CASE 4: Represents a valid isosceles triangle

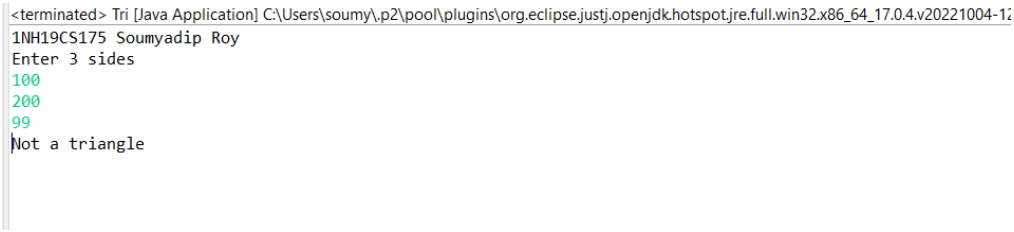
Project Information		**Test Information**	
Project Name:	Triangle	Test Name:	Isosceles Triangle
Project ID:	Triangle_03	Original Author:	Soumyadip Roy
Test Objective:	This test case is to verify the functionality using inputs that form an isosceles triangle		

Case No.	**Test Case Description**	**Test Data**	**Observed Result**	**Expected Result**	**Status (Pass/Fail)**

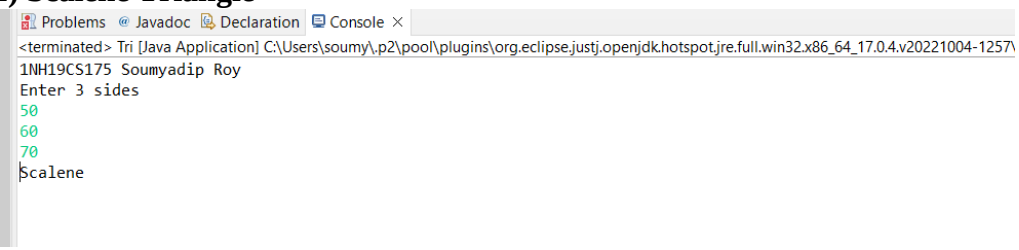
1.	Enter valid inputs	Side a = 10 Side b = 20 Side c = 20	A message saying ‘Given dimensions form an isosceles triangle!’ appears.	A message saying ‘Given dimensions form an isosceles triangle!’ should appear.	Pass
2.	Enter valid inputs	Side a = 50 Side b = 60 Side c = 50	A message saying ‘Given dimensions form an isosceles triangle!’ appears.	A message saying ‘Given dimensions form an isosceles triangle!’ should appear.	Pass
3.	Enter valid inputs	Side a = 200 Side b = 100 Side c = 200	A message saying ‘Given dimensions form an isosceles triangle!’ appears.	A message saying ‘Given dimensions form an isosceles triangle!’ should appear.	Pass
4.	Enter valid inputs	Side a = 199 Side b = 199 Side c = 200	A message saying ‘Given dimensions form an isosceles triangle!’ appears.	A message saying ‘Given dimensions form an isosceles triangle!’ should appear.	Pass
5.	Enter valid inputs	Side a = 1 Side b = 2 Side c = 2	A message saying ‘Given dimensions form an isosceles triangle!’ appears.	A message saying ‘Given dimensions form an isosceles triangle!’ should appear.	Pass

EXECUTION

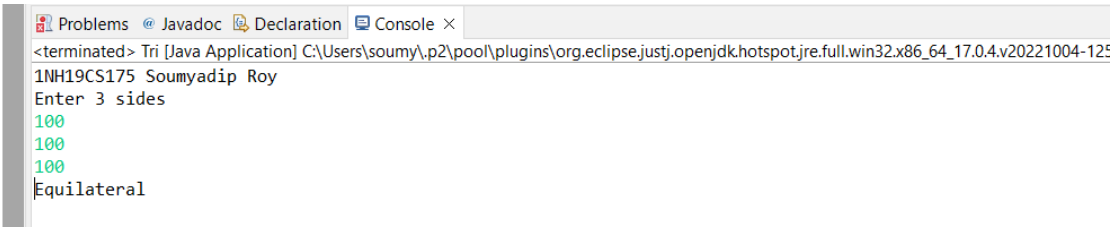
1) Invalid case: Not a Triangle



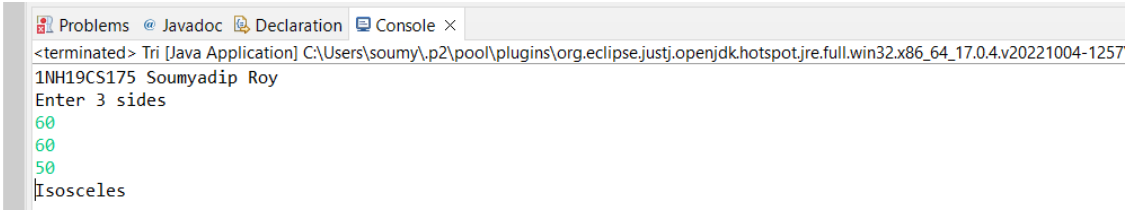
2) Scalene Triangle



3) Equilateral Triangle



4) Isosceles Triangle



RESULT & DISCUSSION

Test Report:

1. Number of Test Cases Executed : 20
2. Number of Test Cases Passed : 20
3. Number of Test Cases Failed : 0

Exp. No. : 3

Date : 08-09-2022

BOUNDARY VALUE ANALYSIS (BVA) FOR NEXTDATE FUNCTION

Design, develop, code and run the program in any suitable language to implement the NextDate function. Analyse it from the perspective boundary value testing. Create different test cases based on the following variants, execute the test cases by using Junit and discuss the test results.

i) Normal Boundary Value Testing

ii) Robust Boundary Value Testing

iii) Worst-Case Boundary Value Testing

iv) Robust Worst-Case Boundary Value Testing

IMPLEMENTATION

1. NextDate function

```
import java.util.*;
public class date {
	public String nextDate (int d, int m, int y) {
		int nd, nm, ny;
		if(d>31 || d<1 || m>12 || m <1 || y<1821 || y>2021){
			return ("Invalid date!");
		}
		else if(m==2 || m==4 || m==6 || m==9 || m==11 ){
			if(d==31)
				return("Invalid date!");
			else if(m==2){
				if(checkLeapYear(y)){
					if(d>29){
						return("Invalid date!");
					}
					if(d == 29) {
						nd = 1;
						nm = 3;
					}
					else {
						nd = ++d;
						nm = 2;
					}
				}
				else {
					if(d>28){
						return("Invalid date!");
					}
					if(d == 28) {
						nd = 1;
						nm = 3;
					}
```

```
                                        else {
                                                nd = ++d;
                                                nm = 2;
                                        }
                                }
                                ny = y;
                        }
                        else {
                                if (d == 30){
                                        nd = 1;
                                        nm = ++m;
                                }
                                else{
                                        nd = ++d;
                                        nm = m;
                                }
                                ny = y;
                        }
                }
                else {
                        if (d == 31 && m != 12){
                                nd = 1;
                                nm = ++m;
                                ny = y;
                        }
                        else if (d == 31 && m == 12){
                                nd = 1;
                                nm = 1;
                                ny = ++y;
                        }
                        else{
                                nd = ++d;
                                nm = m;
                                ny = y;
                        }
                }
                return("The next date is: "+nd+"-"+nm+"-"+ny);
        }

        public static boolean checkLeapYear(int year){
                if(year % 400 == 0)
                        return true;
                else if(year % 100 == 0)
                        return false;
                else if(year % 4 == 0)
                        return true;
                else
                        return false;
        }

}
```

2. Normal BVA - JUnit Test Cases

```
import static org.junit.Assert.*;
import org.junit.Test;

public class NormalBVT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 1821), "The next date is: 16-6-1821");
	}
	@Test
	public void test2() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 1822), "The next date is: 16-6-1822");
	}
	@Test
	public void test3() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 1921), "The next date is: 16-6-1921");
	}
	@Test
	public void test4() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2020), "The next date is: 16-6-2020");
	}
	@Test
	public void test5() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2021), "The next date is: 16-6-2021");
	}
	@Test
	public void test6() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 1, 1921), "The next date is: 16-1-1921");
	}
	@Test
	public void test7() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 2, 1921), "The next date is: 16-2-1921");
	}
	@Test
	public void test8() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 11, 1921), "The next date is: 16-11-1921");
	}
	@Test
	public void test9() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 12, 1921), "The next date is: 16-12-1921");
	}
```

```
	@Test
	public void test10() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 6, 1921), "The next date is: 2-6-1921");
	}
	@Test
	public void test11() {
		date d1 = new date();
		assertEquals(d1.nextDate(2, 6, 1921), "The next date is: 3-6-1921");
	}
	@Test
	public void test12() {
		date d1 = new date();
		assertEquals(d1.nextDate(29, 6, 1921), "The next date is: 30-6-1921");
	}
	@Test
	public void test13() {
		date d1 = new date();
		assertEquals(d1.nextDate(30, 6, 1921), "The next date is: 1-7-1921");
	}
}
```

3. Worst Case BVA - JUnit Test Cases

```
import static org.junit.Assert.*;
import org.junit.Test;

public class WC_BVT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1821), "The next date is: 2-1-1821");
	}
	@Test
	public void test2() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1822), "The next date is: 2-1-1822");
	}
	@Test
	public void test3() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1921), "The next date is: 2-1-1921");
	}
	@Test
	public void test4() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 2020), "The next date is: 2-1-2020");
	}
	@Test
	public void test5() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 2021), "The next date is: 2-1-2021");
```

```
}
@Test
public void test6() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 1821), "The next date is: 3-1-1821");
}
@Test
public void test7() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 1822), "The next date is: 3-1-1822");
}
@Test
public void test8() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 1921), "The next date is: 3-1-1921");
}
@Test
public void test9() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 2020), "The next date is: 3-1-2020");
}
@Test
public void test10() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 2021), "The next date is: 3-1-2021");
}
@Test
public void test11() {
	date d1 = new date();
	assertEquals(d1.nextDate(6, 1, 1821), "The next date is: 7-1-1821");
}
@Test
public void test12() {
	date d1 = new date();
	assertEquals(d1.nextDate(6, 1, 1822), "The next date is: 7-1-1822");
}
@Test
public void test13() {
	date d1 = new date();
	assertEquals(d1.nextDate(6, 1, 1921), "The next date is: 7-1-1921");
}
@Test
public void test14() {
	date d1 = new date();
	assertEquals(d1.nextDate(6, 1, 2020), "The next date is: 7-1-2020");
}
@Test
public void test15() {
	date d1 = new date();
	assertEquals(d1.nextDate(6, 1, 2021), "The next date is: 7-1-2021");
}
@Test
public void test16() {
```

```
		date d1 = new date();
		assertEquals(d1.nextDate(30, 1, 1821), "The next date is: 31-1-1821");
	}
	@Test
	public void test17() {
		date d1 = new date();
		assertEquals(d1.nextDate(30, 1, 1822), "The next date is: 31-1-1822");
	}
	@Test
	public void test18() {
		date d1 = new date();
		assertEquals(d1.nextDate(30, 1, 1921), "The next date is: 31-1-1921");
	}
	@Test
	public void test19() {
		date d1 = new date();
		assertEquals(d1.nextDate(30, 1, 2020), "The next date is: 31-1-2020");
	}
	@Test
	public void test20() {
		date d1 = new date();
		assertEquals(d1.nextDate(30, 1, 2021), "The next date is: 31-1-2021");
	}
	@Test
	public void test21() {
		date d1 = new date();
		assertEquals(d1.nextDate(31, 1, 1821), "The next date is: 1-2-1821");
	}
	@Test
	public void test22() {
		date d1 = new date();
		assertEquals(d1.nextDate(31, 1, 1822), "The next date is: 1-2-1822");
	}
	@Test
	public void test23() {
		date d1 = new date();
		assertEquals(d1.nextDate(31, 1, 1921), "The next date is: 1-2-1921");
	}
	@Test
	public void test24() {
		date d1 = new date();
		assertEquals(d1.nextDate(31, 1, 2020), "The next date is: 1-2-2020");
	}
	@Test
	public void test25() {
		date d1 = new date();
		assertEquals(d1.nextDate(31, 1, 2021), "The next date is: 1-2-2021");
	}
}
```

4. Robust BVA - JUnit Test Cases

```
import static org.junit.Assert.*;
import org.junit.Test;

public class RobustBVT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 1820), "Invalid date!");
	}
	@Test
	public void test2() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 1821), "The next date is: 16-6-1821");
	}
	@Test
	public void test3() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 1822), "The next date is: 16-6-1822");
	}
	@Test
	public void test4() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 1921), "The next date is: 16-6-1921");
	}
	@Test
	public void test5() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2020), "The next date is: 16-6-2020");
	}
	@Test
	public void test6() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2021), "The next date is: 16-6-2021");
	}
	@Test
	public void test7() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2022), "Invalid date!");
	}
	@Test
	public void test8() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 0, 1921), "Invalid date!");
	}
	@Test
	public void test9() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 1, 1921), "The next date is: 16-1-1921");
	}
```

```
	@Test
	public void test10() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 2, 1921), "The next date is: 16-2-1921");
	}
	@Test
	public void test11() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 11, 1921), "The next date is: 16-11-1921");
	}
	@Test
	public void test12() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 12, 1921), "The next date is: 16-12-1921");
	}
	@Test
	public void test13() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 13, 1921), "Invalid date!");
	}
	@Test
	public void test14() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 6, 1921), "Invalid date!");
	}
	@Test
	public void test15() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 6, 1921), "The next date is: 2-6-1921");
	}
	@Test
	public void test16() {
		date d1 = new date();
		assertEquals(d1.nextDate(2, 6, 1921), "The next date is: 3-6-1921");
	}
	@Test
	public void test17() {
		date d1 = new date();
		assertEquals(d1.nextDate(29, 6, 1921), "The next date is: 30-6-1921");
	}
	@Test
	public void test18() {
		date d1 = new date();
		assertEquals(d1.nextDate(30, 6, 1921), "The next date is: 1-7-1921");
	}
	@Test
	public void test19() {
		date d1 = new date();
		assertEquals(d1.nextDate(31, 6, 1921), "Invalid date!");
	}
}
```

5. Robust Worst Case - JUnit Test Cases

```
import static org.junit.Assert.*;
import org.junit.Test;

public class Robust_WC_BVT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 1821), "Invalid date!");
	}
	@Test
	public void test2() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 1822), "Invalid date!");
	}
	@Test
	public void test3() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 1921), "Invalid date!");
	}
	@Test
	public void test4() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 2020), "Invalid date!");
	}
	@Test
	public void test5() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 1, 2021), "Invalid date!");
	}
	@Test
	public void test6() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1821), "The next date is: 2-1-1821");
	}
	@Test
	public void test7() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1822), "The next date is: 2-1-1822");
	}
	@Test
	public void test8() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 1921), "The next date is: 2-1-1921");
	}
	@Test
	public void test9() {
		date d1 = new date();
		assertEquals(d1.nextDate(1, 1, 2020), "The next date is: 2-1-2020");
	}
```

```
@Test
public void test10() {
	date d1 = new date();
	assertEquals(d1.nextDate(1, 1, 2021), "The next date is: 2-1-2021");
}
@Test
public void test11() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 1821), "The next date is: 3-1-1821");
}
@Test
public void test12() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 1822), "The next date is: 3-1-1822");
}
@Test
public void test13() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 1921), "The next date is: 3-1-1921");
}
@Test
public void test14() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 2020), "The next date is: 3-1-2020");
}
@Test
public void test15() {
	date d1 = new date();
	assertEquals(d1.nextDate(2, 1, 2021), "The next date is: 3-1-2021");
}
@Test
public void test16() {
	date d1 = new date();
	assertEquals(d1.nextDate(6, 1, 1821), "The next date is: 7-1-1821");
}
@Test
public void test17() {
	date d1 = new date();
	assertEquals(d1.nextDate(6, 1, 1822), "The next date is: 7-1-1822");
}
@Test
public void test18() {
	date d1 = new date();
	assertEquals(d1.nextDate(6, 1, 1921), "The next date is: 7-1-1921");
}
@Test
public void test19() {
	date d1 = new date();
	assertEquals(d1.nextDate(6, 1, 2020), "The next date is: 7-1-2020");
}
@Test
public void test20() {
	date d1 = new date();
```

```
        assertEquals(d1.nextDate(6, 1, 2021), "The next date is: 7-1-2021");
}
@Test
public void test21() {
        date d1 = new date();
        assertEquals(d1.nextDate(30, 1, 1821), "The next date is: 31-1-1821");
}
@Test
public void test22() {
        date d1 = new date();
        assertEquals(d1.nextDate(30, 1, 1822), "The next date is: 31-1-1822");
}
@Test
public void test23() {
        date d1 = new date();
        assertEquals(d1.nextDate(30, 1, 1921), "The next date is: 31-1-1921");
}
@Test
public void test24() {
        date d1 = new date();
        assertEquals(d1.nextDate(30, 1, 2020), "The next date is: 31-1-2020");
}
@Test
public void test25() {
        date d1 = new date();
        assertEquals(d1.nextDate(30, 1, 2021), "The next date is: 31-1-2021");
}
@Test
public void test26() {
        date d1 = new date();
        assertEquals(d1.nextDate(31, 1, 1821), "The next date is: 1-2-1821");
}
@Test
public void test27() {
        date d1 = new date();
        assertEquals(d1.nextDate(31, 1, 1822), "The next date is: 1-2-1822");
}
@Test
public void test28() {
        date d1 = new date();
        assertEquals(d1.nextDate(31, 1, 1921), "The next date is: 1-2-1921");
}
@Test
public void test29() {
        date d1 = new date();
        assertEquals(d1.nextDate(31, 1, 2020), "The next date is: 1-2-2020");
}
@Test
public void test30() {
        date d1 = new date();
        assertEquals(d1.nextDate(31, 1, 2021), "The next date is: 1-2-2021");
}
```

```
	@Test
	public void test31() {
		date d1 = new date();
		assertEquals(d1.nextDate(32, 1, 1821), "Invalid date!");
	}
	@Test
	public void test32() {
		date d1 = new date();
		assertEquals(d1.nextDate(32, 1, 1822), "Invalid date!");
	}
	@Test
	public void test33() {
		date d1 = new date();
		assertEquals(d1.nextDate(32, 1, 1921), "Invalid date!");
	}
	@Test
	public void test34() {
		date d1 = new date();
		assertEquals(d1.nextDate(32, 1, 2020), "Invalid date!");
	}
	@Test
	public void test35() {
		date d1 = new date();
		assertEquals(d1.nextDate(32, 1, 2021), "Invalid date!");
	}
}
```

TEST CASES

Test Case Name : BVA testing for NextDate function
Test Data : Day, Month and Year
Pre-condition : Day {1 <= d <= 31}, Month {1 <= m <= 12}, Year {1821 <= y <= 2021}
Test Objective : To find the next date for the given valid date.

i) TEST CASES FOR NORMAL BOUNDARY VALUE TESTING

Project Information					Test Information		
Project Name:	NextDate				Test Name:	ND_BVA	
Project ID:	NextDate_01				Original Author:	Soumyadip Roy	
Test Objective:	To find the next date for the given date using Normal BVA.						
Case No.	**Test Case Description**	**Test Data**			**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
		d	**m**	**y**			
ND_BVA_01	Date: nom Month: nom Year: min	15	6	1821	A message is displayed as "The next date is: 16-6-1821"	A message must be displayed as "The next date is: 16-6-1821"	Pass
ND_BVA_02	Date: nom Month: nom Year: min+	15	6	1822	A message is displayed as "The next date is: 16-6-1822"	A message must be displayed as "The next date is: 16-6-1822"	Pass
ND_BVA_03	Date: nom Month: nom Year: nom	15	6	1921	A message is displayed as "The next date is: 16-6-1921"	A message must be displayed as "The next date is: 16-6-1921"	Pass

ND_BVA_04	Date: nom Month: nom Year: max-	15	6	2020	A message is displayed as "The next date is: 16-6-2020"	A message must be displayed as "The next date is: 16-6-2020"	Pass
ND_BVA_05	Date: nom Month: nom Year: max	15	6	2021	A message is displayed as "The next date is: 16-6-2021"	A message must be displayed as "The next date is: 16-6-2021"	Pass
ND_BVA_06	Date: nom Month: min Year: nom	15	1	1921	A message is displayed as "The next date is: 16-1-1921"	A message must be displayed as "The next date is: 16-1-1921"	Pass
ND_BVA_07	Date: nom Month: min+ Year: nom	15	2	1921	A message is displayed as "The next date is: 16-2-1921"	A message must be displayed as "The next date is: 16-2-1921"	Pass
ND_BVA_08	Date: nom Month: max- Year: nom	15	11	1921	A message is displayed as "The next date is: 16-11-1921"	A message must be displayed as "The next date is: 16-11-1921"	Pass
ND_BVA_09	Date: nom Month: max Year: nom	15	12	1921	A message is displayed as "The next date is: 16-12-1921"	A message must be displayed as "The next date is: 16-12-1921"	Pass
ND_BVA_10	Date: min Month: nom Year: nom	1	6	1921	A message is displayed as "The next date is: 2-6-1921"	A message must be displayed as "The next date is: 2-6-1921"	Pass
ND_BVA_11	Date: min+ Month: nom Year: nom.	2	6	1921	A message is displayed as "The next date is: 3-6-1921"	A message must be displayed as "The next date is: 3-6-1921"	Pass
ND_BVA_12	Date: max- Month: nom Year: nom	30	6	1921	A message is displayed as "The next date is: 31-6-1921"	A message must be displayed as "The next date is: 31-6-1921"	Pass
ND_BVA_13	Date: max Month: nom Year: nom	31	6	1921	A message is displayed as "The next date is: 1-7-1921"	A message must be displayed as "The next date is: 1-7-1921"	Pass

ii) TEST CASES FOR ROBUST BOUNDARY VALUE TESTING

Project Information					**Test Information**		
Project Name:	NextDate				Test Name:	ND_R_BVA	
Project ID:	NextDate_01				Original Author:	Soumyadip Roy	
Test Objective:	To find the next date for the given date using Robust BVA.						
Case No.	**Test Case Description**	**Test Data**			**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
		d	**m**	**y**			
ND_R_BVA_01	Date: nom Month: nom Year: min-	15	6	1820	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_R_BVA_02	Date: nom Month: nom Year: min	15	6	1821	A message is displayed as "The next date is: 16-6-1821"	A message must be displayed as "The next date is: 16-6-1821"	Pass
ND_R_BVA_03	Date: nom Month: nom Year: min+	15	6	1822	A message is displayed as "The next date is: 16-6-1822"	A message must be displayed as "The next date is: 16-6-1822"	Pass

ND_R_BVA_04	Date: nom Month: nom Year: nom	15	6	1921	A message is displayed as "The next date is: 16-6-1921"	A message must be displayed as "The next date is: 16-6-1921"	Pass
ND_R_BVA_05	Date: nom Month: nom Year: max-	15	6	2020	A message is displayed as "The next date is: 16-6-2020"	A message must be displayed as "The next date is: 16-6-2020"	Pass
ND_R_BVA_06	Date: nom Month: nom Year: max	15	6	2021	A message is displayed as "The next date is: 16-6-2021"	A message must be displayed as "The next date is: 16-6-2021"	Pass
ND_R_BVA_07	Date: nom Month: nom Year: max+	15	6	2022	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_R_BVA_08	Date: nom Month: min- Year: nom	15	0	1921	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_R_BVA_09	Date: nom Month: min Year: nom	15	1	1921	A message is displayed as "The next date is: 16-1-1921"	A message must be displayed as "The next date is: 16-1-1921"	Pass
ND_R_BVA_10	Date: nom Month: min+ Year: nom	15	2	1921	A message is displayed as "The next date is: 16-2-1921"	A message must be displayed as "The next date is: 16-2-1921"	Pass
ND_R_BVA_11	Date: nom Month: max- Year: nom	15	11	1921	A message is displayed as "The next date is: 16-11-1921"	A message must be displayed as "The next date is: 16-11-1921"	Pass
ND_R_BVA_12	Date: nom Month: max Year: nom	15	12	1921	A message is displayed as "The next date is: 16-12-1921"	A message must be displayed as "The next date is: 16-12-1921"	Pass
ND_R_BVA_13	Date: nom Month: max+ Year: nom	15	13	1820	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_R_BVA_14	Date: min- Month: nom Year: nom	0	6	1921	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_R_BVA_15	Date: min Month: nom Year: nom	1	6	1921	A message is displayed as "The next date is: 2-6-1921"	A message must be displayed as "The next date is: 2-6-1921"	Pass
ND_R_BVA_16	Date: min+ Month: nom Year: nom.	2	6	1921	A message is displayed as "The next date is: 3-6-1921"	A message must be displayed as "The next date is: 3-6-1921"	Pass
ND_R_BVA_17	Date: max- Month: nom Year: nom	30	6	1921	A message is displayed as "The next date is: 31-6-1921"	A message must be displayed as "The next date is: 31-6-1921"	Pass
ND_R_BVA_18	Date: max Month: nom Year: nom	31	6	1921	A message is displayed as "The next date is: 1-7-1921"	A message must be displayed as "The next date is: 1-7-1921"	Pass
ND_R_BVA_19	Date: max+ Month: nom Year: nom	32	6	1921	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass

iii)TEST CASES FOR WORST CASE BOUNDARY VALUE TESTING

Project Information					Test Information		
Project Name:	NextDate				Test Name:	ND_WC_BVA	
Project ID:	NextDate_01				Original Author:	Soumyadip Roy	
Test Objective:	To find the next date for the given date using Worst Case BVA.						
Case No.	**Test Case Description**	**Test Data**			**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
		d	**m**	**y**			
ND_WC_BVA_01	Date: min Month: min Year: min	1	1	1821	A message is displayed as "The next date is: 2-1-1821"	A message must be displayed as "The next date is: 2-1-1821"	Pass
ND_WC_BVA_02	Date: min Month: min Year: min+	1	1	1822	A message is displayed as "The next date is: 2-1-1822"	A message must be displayed as "The next date is: 2-1-1822"	Pass
ND_WC_BVA_03	Date: min Month: min Year: nom	1	1	1921	A message is displayed as "The next date is: 2-1-1921"	A message must be displayed as "The next date is: 2-1-1921"	Pass
ND_WC_BVA_04	Date: min Month: min Year: max-	1	1	2020	A message is displayed as "The next date is: 2-1-2020"	A message must be displayed as "The next date is: 2-1-2020"	Pass
ND_WC_BVA_05	Date: min Month: min Year: max	1	1	2021	A message is displayed as "The next date is: 2-1-2021"	A message must be displayed as "The next date is: 2-1-2021"	Pass
ND_WC_BVA_06	Date: min+ Month: min Year: min	2	1	1821	A message is displayed as "The next date is: 3-1-1821"	A message must be displayed as "The next date is: 3-1-1821"	Pass
ND_WC_BVA_07	Date: min+ Month: min Year: min+	2	1	1822	A message is displayed as "The next date is: 3-1-1822"	A message must be displayed as "The next date is: 3-1-1822"	Pass
ND_WC_BVA_08	Date: min+ Month: min Year: nom	2	1	1921	A message is displayed as "The next date is: 3-1-1921"	A message must be displayed as "The next date is: 3-1-1921"	Pass
ND_WC_BVA_09	Date: min+ Month: min Year: max-	2	1	2020	A message is displayed as "The next date is: 3-1-2020"	A message must be displayed as "The next date is: 3-1-2020"	Pass
ND_WC_BVA_10	Date: min+ Month: min Year: max	2	1	2021	A message is displayed as "The next date is: 3-1-2021"	A message must be displayed as "The next date is: 3-1-2021"	Pass
ND_WC_BVA_11	Date: nom Month: min Year: min	15	1	1821	A message is displayed as "The next date is: 16-1-1821"	A message must be displayed as "The next date is: 16-1-1821"	Pass
ND_WC_BVA_12	Date: nom Month: min Year: min+	15	1	1822	A message is displayed as "The next date is: 16-1-1822"	A message must be displayed as "The next date is: 16-1-1822"	Pass
ND_WC_BVA_13	Date: nom Month: min Year: nom	15	1	1921	A message is displayed as "The next date is: 16-1-1921"	A message must be displayed as "The next date is: 16-1-1921"	Pass

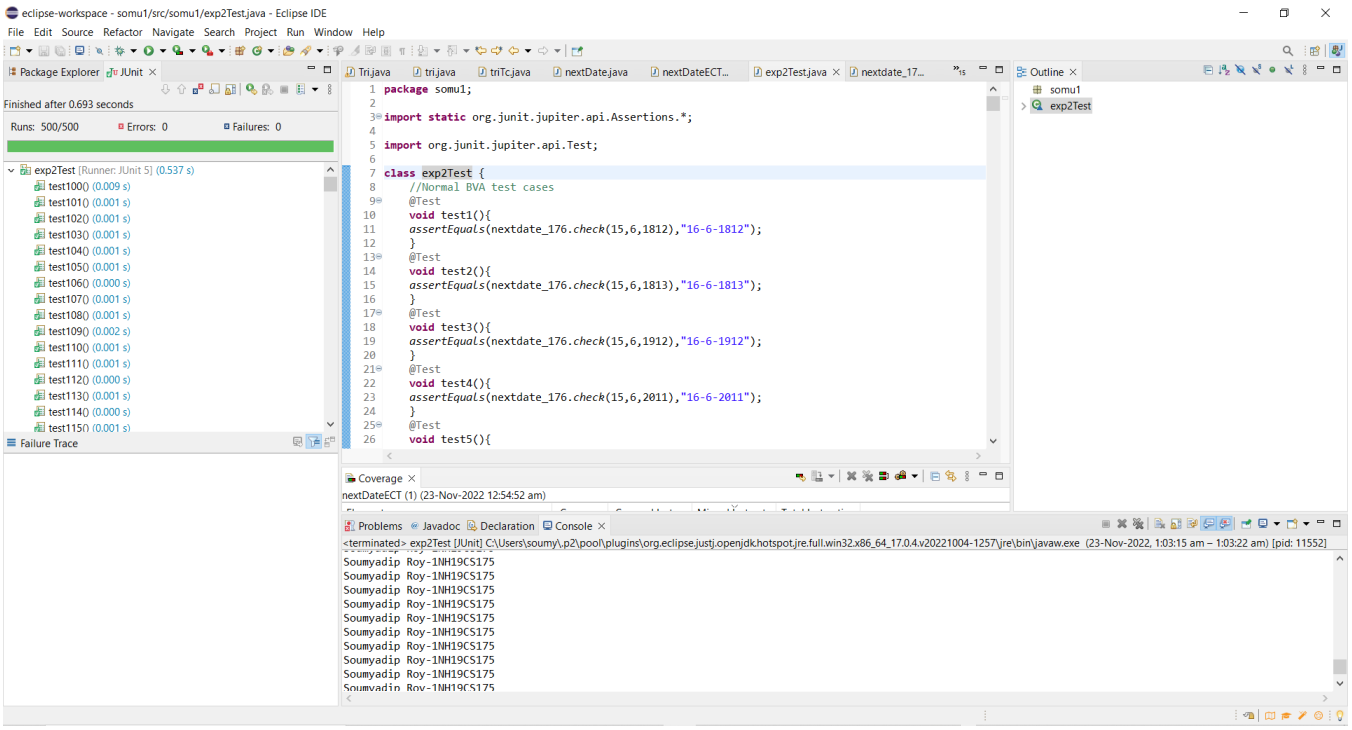
ND_WC_BVA _14	Date: nom Month: min Year: max-	15	1	2020	A message is displayed as "The next date is: 16-1-2020"	A message must be displayed as "The next date is: 16-1-2020"	Pass
ND_WC_BVA _15	Date: nom Month: min Year: max	15	1	2021	A message is displayed as "The next date is: 16-1-2021"	A message must be displayed as "The next date is: 16-1-2021"	Pass
ND_WC_BVA _16	Date: max- Month: min Year: min	30	1	1821	A message is displayed as "The next date is: 31-1-1821"	A message must be displayed as "The next date is: 31-1-1821"	Pass
ND_WC_BVA _17	Date: max- Month: min Year: min+	30	1	1822	A message is displayed as "The next date is: 31-1-1822"	A message must be displayed as "The next date is: 31-1-1822"	Pass
ND_WC_BVA _18	Date: max- Month: min Year: nom	30	1	1921	A message is displayed as "The next date is: 31-1-1921"	A message must be displayed as "The next date is: 31-1-1921"	Pass
ND_WC_BVA _19	Date: max- Month: min Year: max-	30	1	2020	A message is displayed as "The next date is: 31-1-2020"	A message must be displayed as "The next date is: 31-1-2020"	Pass
ND_WC_BVA _20	Date: max- Month: min Year: max	30	1	2021	A message is displayed as "The next date is: 31-1-2021"	A message must be displayed as "The next date is: 31-1-2021"	Pass

iv) TEST CASES FOR ROBUST WORST CASE BOUNDARY VALUE TESTING

Project Information					Test Information		
Project Name:	NextDate				Test Name:	ND_RWC_BVA	
Project ID:	NextDate_01				Original Author:	Soumyadip Roy	
Test Objective:	To find the next date for the given date using Robust Worst Case BVA.						
Case No.	**Test Case Description**	**Test Data**			**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
		d	**m**	**y**			
ND_RWC_ BVA_01	Date: min- Month: min Year: min	0	1	1821	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_RWC_ BVA_02	Date: min- Month: min Year: min+	0	1	1822	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_RWC_ BVA_03	Date: min- Month: min Year: nom	0	1	1921	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_RWC_ BVA_04	Date: min- Month: min Year: max-	0	1	2020	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_RWC_ BVA_05	Date: min- Month: min Year: max	0	1	2021	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_RWC_ BVA_06	Date: min Month: min Year: min	1	1	1821	A message is displayed as "The next date is: 2-1-1821"	A message must be displayed as "The next date is: 2-1-1821"	Pass

ND_RWC_ BVA_07	Date: min Month: min Year: min+	1	1	1822	A message is displayed as "The next date is: 2-1-1822"	A message must be displayed as "The next date is: 2-1-1822"	Pass
ND_RWC_ BVA_08	Date: min Month: min Year: nom	1	1	1921	A message is displayed as "The next date is: 2-1-1921"	A message must be displayed as "The next date is: 2-1-1921"	Pass
ND_RWC_ BVA_09	Date: min Month: min Year: max-	1	1	2020	A message is displayed as "The next date is: 2-1-2020"	A message must be displayed as "The next date is: 2-1-2020"	Pass
ND_RWC_ BVA_10	Date: min Month: min Year: max	1	1	2021	A message is displayed as "The next date is: 2-1-2021"	A message must be displayed as "The next date is: 2-1-2021"	Pass
ND_RWC_ BVA_11	Date: min+ Month: min Year: min	2	1	1821	A message is displayed as "The next date is: 3-1-1821"	A message must be displayed as "The next date is: 3-1-1821"	Pass
ND_RWC_ BVA_12	Date: min+ Month: min Year: min+	2	1	1822	A message is displayed as "The next date is: 3-1-1822"	A message must be displayed as "The next date is: 3-1-1822"	Pass
ND_RWC_ BVA_13	Date: min+ Month: min Year: nom	2	1	1921	A message is displayed as "The next date is: 3-1-1921"	A message must be displayed as "The next date is: 3-1-1921"	Pass
ND_RWC_ BVA_14	Date: min+ Month: min Year: max-	2	1	2020	A message is displayed as "The next date is: 3-1-2020"	A message must be displayed as "The next date is: 3-1-2020"	Pass
ND_RWC_ BVA_15	Date: min+ Month: min Year: max	2	1	2021	A message is displayed as "The next date is: 3-1-2021"	A message must be displayed as "The next date is: 3-1-2021"	Pass
ND_RWC_ BVA_16	Date: nom Month: min Year: min	15	1	1821	A message is displayed as "The next date is: 16-1-1821"	A message must be displayed as "The next date is: 16-1-1821"	Pass
ND_RWC_ BVA_17	Date: nom Month: min Year: min+	15	1	1822	A message is displayed as "The next date is: 16-1-1822"	A message must be displayed as "The next date is: 16-1-1822"	Pass
ND_RWC_ BVA_18	Date: nom Month: min Year: nom	15	1	1921	A message is displayed as "The next date is: 16-1-1921"	A message must be displayed as "The next date is: 16-1-1921"	Pass
ND_RWC_ BVA_19	Date: nom Month: min Year: max-	15	1	2020	A message is displayed as "The next date is: 16-1-2020"	A message must be displayed as "The next date is: 16-1-2020"	Pass
ND_RWC_ BVA_20	Date: nom Month: min Year: max	15	1	2021	A message is displayed as "The next date is: 16-1-2021"	A message must be displayed as "The next date is: 16-1-2021"	Pass

EXECUTION



RESULT & DISCUSSION

Test Report:

1. Number of Test Cases Executed : 92
2. Number of Test Cases Passed : 92
3. Number of Test Cases Failed : 0

Exp. No. : 4

Date : 15-09-2022

EQUIVALENCE CLASS PARTITIONING (ECP) FOR NEXTDATE FUNCTION

Design, develop, code and run the program in any suitable language to implement the NextDate function. Analyse it from the perspective equivalence class testing. Create different test cases, execute these test cases by using JUnit and discuss the test results.

i) Weak Normal Equivalence Class Testing

ii) Strong Normal Equivalence Class Testing

iii) Weak Robust Equivalence Class Testing

iv) Strong Robust Equivalence Class Testing

IMPLEMENTATION

1. NextDate function

```
import java.util.*;
public class date {
	public String nextDate (int d, int m, int y) {
		int nd, nm, ny;
		if(d>31 || d<1 || m>12 || m <1 || y<1821 || y>2021){
			return ("Invalid date!");
		}
		else if(m==2 || m==4 || m==6 || m==9 || m==11 ){
			if(d==31)
				return("Invalid date!");
			else if(m==2){
				if(checkLeapYear(y)){
					if(d>29){
						return("Invalid date!");
					}
					if(d == 29) {
						nd = 1;
						nm = 3;
					}
					else {
						nd = ++d;
						nm = 2;
					}
				}
				else {
					if(d>28){
						return("Invalid date!");
					}
					if(d == 28) {
						nd = 1;
						nm = 3;
					}
```

```
                                        else {
                                                nd = ++d;
                                                nm = 2;
                                        }
                                }
                                ny = y;
                        }
                        else {
                                if (d == 30){
                                        nd = 1;
                                        nm = ++m;
                                }
                                else{
                                        nd = ++d;
                                        nm = m;
                                }
                                ny = y;
                        }
                }
                else {
                        if (d == 31 && m != 12){
                                nd = 1;
                                nm = ++m;
                                ny = y;
                        }
                        else if (d == 31 && m == 12){
                                nd = 1;
                                nm = 1;
                                ny = ++y;
                        }
                        else{
                                nd = ++d;
                                nm = m;
                                ny = y;
                        }
                }
                return("The next date is: "+nd+"-"+nm+"-"+ny);
        }

        public static boolean checkLeapYear(int year){
                if(year % 400 == 0)
                        return true;
                else if(year % 100 == 0)
                        return false;
                else if(year % 4 == 0)
                        return true;
                else
                        return false;
        }

}
```

2. Weak Normal Equivalence Class - JUnit Test Cases

```
import static org.junit.Assert.*;
import org.junit.Test;

public class WN_ECT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(14, 6, 2000), "The next date is: 15-6-2000");
	}
}
```

3. Strong Normal Equivalence Class - JUnit Test Cases

```
import static org.junit.Assert.*;
import org.junit.Test;

public class SN_ECT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(14, 6, 2000), "The next date is: 15-6-2000");
	}
}
```

4. Weak Robust Equivalence Class - JUnit Test Cases

```
import static org.junit.Assert.*;
import org.junit.Test;

public class WR_ECT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2001), "The next date is: 16-6-2001");
	}
	@Test
	public void test2() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 6, 1822), "Invalid date!");
	}
	@Test
	public void test3() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 13, 1921), "Invalid date!");
	}
	@Test
	public void test4() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2028), "Invalid date!");
	}
}
```

5. Strong Robust Equivalence Class - JUnit Test Cases

```
import static org.junit.Assert.*;
import org.junit.Test;

public class SR_ECT {

	@Test
	public void test1() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2001), "The next date is: 16-6-2001");
	}
	@Test
	public void test2() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 6, 1822), "Invalid date!");
	}
	@Test
	public void test3() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 13, 1921), "Invalid date!");
	}
	@Test
	public void test4() {
		date d1 = new date();
		assertEquals(d1.nextDate(15, 6, 2028), "Invalid date!");
	}
	@Test
	public void test5() {
		date d1 = new date();
		assertEquals(d1.nextDate(0, 13, 2021), "Invalid date!");
	}
	@Test
	public void test6() {
		date d1 = new date();
		assertEquals(d1.nextDate(41, 1, 1785), "Invalid date!");
	}
	@Test
	public void test7() {
		date d1 = new date();
		assertEquals(d1.nextDate(5, 15, 2112), "Invalid date!");
	}
	@Test
	public void test8() {
		date d1 = new date();
		assertEquals(d1.nextDate(46, 19, 1512), "Invalid date!");
	}
}
```

TEST CASES

Test Case Name : Equivalence Class Testing for NextDate function
Test Data : Day, Month and Year
Pre-condition : Day {1 <= d <= 31}, Month {1 <= m <= 12}, Year {1821 <= y <= 2021}
Test Objective : To find the next date for the given valid date.

i) TEST CASES FOR WEAK NORMAL ECP TESTING

Project Information					Test Information		
Project Name:	NextDate				Test Name:	ND_WN_ECP	
Project ID:	NextDate_01				Original Author:	Soumyadip Roy	
Test Objective:	To find the next date for the given date using Weak Normal ECP.						
Case No.	**Test Case Description**	**Test Data**			**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
		d	**m**	**y**			
ND_WN_ECP _01	A date that contains 1 valid input each from date, month and year.	14	6	2000	A message is displayed as "The next date is: 15-6-2000"	A message must be displayed as "The next date is: 15-6-2000"	Pass

ii) TEST CASES FOR STRONG NORMAL ECP TESTING

Project Information					Test Information		
Project Name:	NextDate				Test Name:	ND_SN_ECP	
Project ID:	NextDate_01				Original Author:	Soumyadip Roy	
Test Objective:	To find the next date for the given date using Strong Normal ECP.						
Case No.	**Test Case Description**	**Test Data**			**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
		d	**m**	**y**			
ND_SN_ECP_ 01	A date that contains 1 valid input each from date, month and year.	14	6	2000	A message is displayed as "The next date is: 15-6-2000"	A message must be displayed as "The next date is: 15-6-2000"	Pass

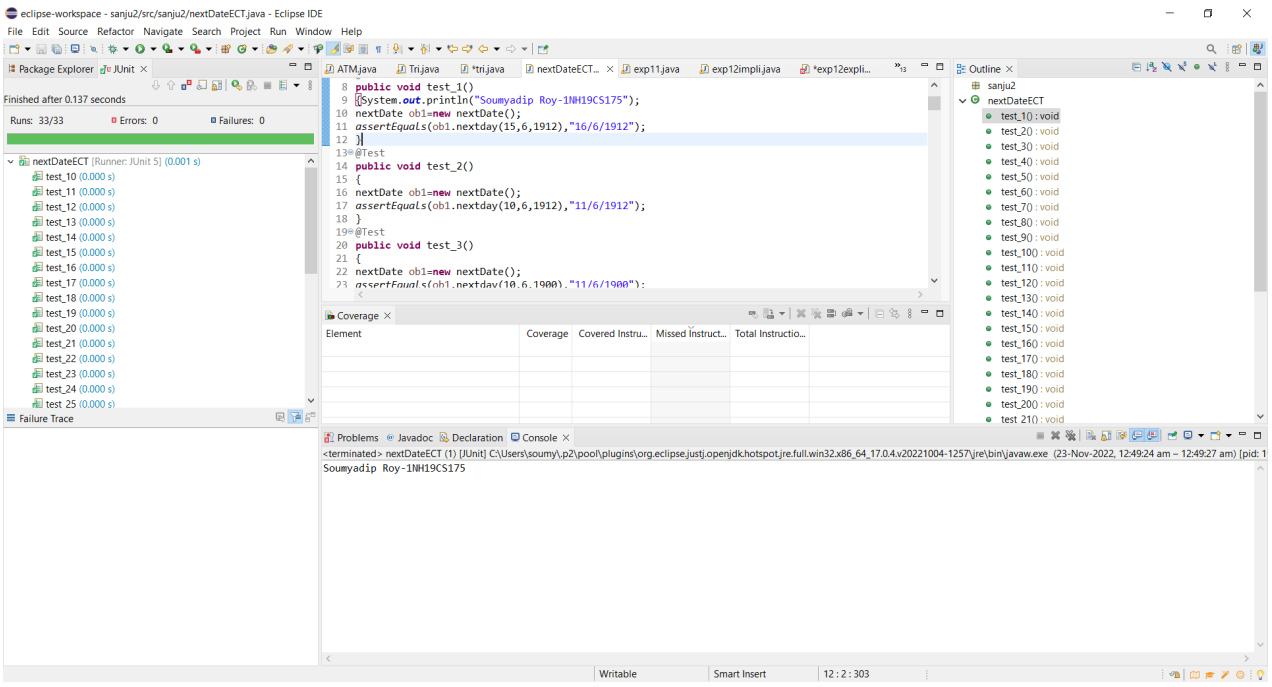
iii) TEST CASES FOR WEAK ROBUST ECP TESTING

Project Information					Test Information		
Project Name:	NextDate				Test Name:	ND_WR_ECP	
Project ID:	NextDate_01				Original Author:	Soumyadip Roy	
Test Objective:	To find the next date for the given date using Weak Robust ECP.						
Case No.	**Test Case Description**	**Test Data**			**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
		d	**m**	**y**			
ND_WR_ECP _01	A date that contains 1 valid input each from date, month and year.	15	6	2001	A message is displayed as "The next date is: 16-6-2001"	A message must be displayed as "The next date is: 16-6-2001"	Pass
ND_WR_ECP _02	Invalid day.	0	6	1822	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_WR_ECP _03	Invalid month.	15	13	1921	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_WR_ECP _04	Invalid year.	15	6	2028	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass

iv) TEST CASES FOR STRONG ROBUST ECP TESTING

Project Information					Test Information		
Project Name:	NextDate				Test Name:	ND_SR_ECP	
Project ID:	NextDate_01				Original Author:	Soumyadip Roy	
Test Objective:	To find the next date for the given date using Strong Robust ECP.						
Case No.	**Test Case Description**	**Test Data**			**Observed Result**	**Expected Result**	**Status (Pass/Fail)**
		d	**m**	**y**			
ND_SR_ECP_01	A date that contains 1 valid input each from date, month and year.	15	6	2001	A message is displayed as "The next date is: 16-6-2001"	A message must be displayed as "The next date is: 16-6-2001"	Pass
ND_SR_ECP_02	Invalid day.	0	6	1822	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_SR_ECP_03	Invalid month.	15	13	1921	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_SR_ECP_04	Invalid year.	15	6	2028	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_SR_ECP_05	Invalid day and month.	0	13	2021	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_SR_ECP_06	Invalid day and year.	41	1	1785	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_SR_ECP_07	Invalid month and year.	5	15	2112	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass
ND_SR_ECP_08	Invalid day, month and year.	46	19	1512	A message is displayed as "Invalid date!"	A message must be displayed as "Invalid date!"	Pass

EXECUTION



RESULT & DISCUSSION

Test Report:

1. Number of Test Cases Executed : 14
2. Number of Test Cases Passed : 14
3. Number of Test Cases Failed : 0

Exp. No. : 5

Date : 22-09-2022

DEMONSTRATION OF WHITE BOX TESTING TECHNIQUE USING ECLEMMA

Demonstrate white box testing techniques using open-source testing tool JUnit and ECLEMMA. Implement and execute test cases for achieving full statement coverage, decision/branch coverage and condition coverage for the triangle problem.

IMPLEMENTATION

1. Triangle problem

```
import java.util.*;
public class triangle {
	public String check (int a, int b, int c){
		while(true){
			if(a>=1 && a<=200 && b>=1 && b<=200 && c>=1 && c<=200){
				if((a < b+c) && (b < a+c) && (c < b+a)){
					if(a == b && b == c)
						return ("Given dimensions form an equilateral triangle!");
					else if(a==b || b==c || c==a)
						return("Given dimensions form an isosceles triangle!");
					else
						return("Given dimensions form a scalene triangle!");
				}
				else {
					return("Given dimensions do not form a triangle!");
				}
			}
			else{
				return("Enter a valid input!");
			}
		}
	}
}
```

2. JUnit Test Cases for Complete Coverage

```
import static org.junit.Assert.*;
import org.junit.Test;

import org.junit.Test;

public class triangleTest {

	@Test
	public void test1() {
		triangle t1 = new triangle();
		assertEquals(t1.check(1, 2, 3), "Given dimensions do not form a triangle!");
	}
```

```
@Test
public void test2() {
	triangle t1 = new triangle();
	assertEquals(t1.check(2, 2, 2), "Given dimensions form an equilateral triangle!");
}

@Test
public void test3() {
	triangle t1 = new triangle();
	assertEquals(t1.check(2, 2, 3), "Given dimensions form an isosceles triangle!");
}

@Test
public void test4() {
	triangle t1 = new triangle();
	assertEquals(t1.check(4, 5, 6), "Given dimensions form a scalene triangle!");
}

@Test
public void test5() {
	triangle t1 = new triangle();
	assertEquals(t1.check(-4, 5, 6), "Enter a valid input!");
}

@Test
public void test6() {
	triangle t1 = new triangle();
	assertEquals(t1.check(4, 5, 4), "Given dimensions form an isosceles triangle!");
}

@Test
public void test7() {
	triangle t1 = new triangle();
	assertEquals(t1.check(5, 4, 4), "Given dimensions form an isosceles triangle!");
}

@Test
public void test8() {
	triangle t1 = new triangle();
	assertEquals(t1.check(7, 4, 2), "Given dimensions do not form a triangle!");
}

@Test
public void test9() {
	triangle t1 = new triangle();
	assertEquals(t1.check(4, 7, 2), "Given dimensions do not form a triangle!");
}

@Test
public void test10() {
	triangle t1 = new triangle();
	assertEquals(t1.check(4, -5, 6), "Enter a valid input!");
}
```

```
	@Test
	public void test11() {
		triangle t1 = new triangle();
		assertEquals(t1.check(4, 5, -6), "Enter a valid input!");
	}

	@Test
	public void test12() {
		triangle t1 = new triangle();
		assertEquals(t1.check(4, 205, -6), "Enter a valid input!");
	}

	@Test
	public void test13() {
		triangle t1 = new triangle();
		assertEquals(t1.check(204, 205, 209), "Enter a valid input!");
	}

	@Test
	public void test14() {
		triangle t1 = new triangle();
		assertEquals(t1.check(5, 5, 209), "Enter a valid input!");
	}
}
```

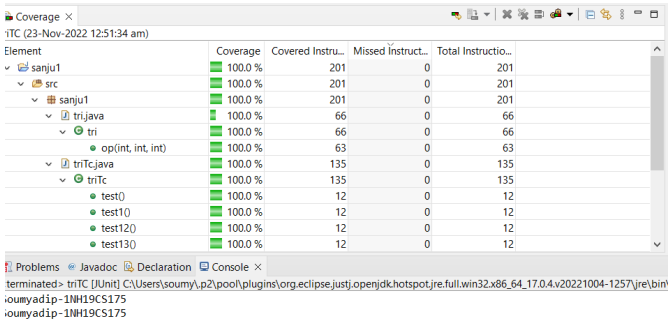
TEST CASES FOR TRIANGLE PROGRAM

Project Information		Test Information	
Project Name:	Triangle	Test Name:	Triangle Code Coverage
Project ID:	Triangle_02	Original Author:	Soumyadip Roy
Test Objective:	To check whether three given sides forms a scalene or equilateral or isosceles triangle or if given sides do not form a triangle or if given sides are invalid inputs.		

Case No.	Test Case Description	Test Data			Observed Result	Expected Result	Status (Pass/Fail)
		a	b	c			
Triangle _Cov_1	Check for dimensions that do not form a triangle.	1	2	3	A message saying 'Given dimensions do not form a triangle!' appears.	A message saying 'Given dimensions do not form a triangle!' should appear.	Pass
Triangle _Cov_2	Check for dimensions that form an equilateral triangle.	2	2	2	A message saying 'Given dimensions form an equilateral triangle!' appears.	A message saying 'Given dimensions form an equilateral triangle!' should appear.	Pass
Triangle _Cov_3	Check for dimensions that form an isosceles triangle.	4	4	5	A message saying 'Given dimensions form an isosceles triangle!' appears.	A message saying 'Given dimensions form an isosceles triangle!' should appear.	Pass
Triangle _Cov_4	Check for dimensions that form a scalene triangle.	4	5	6	A message saying 'Given dimensions form a scalene triangle!' appears.	A message saying 'Given dimensions form a scalene triangle!' should appear.	Pass
Triangle _Cov_5	Check for dimensions that are not valid inputs.	-4	5	6	A message is displayed as "Enter a valid input!"	A message must be displayed as "Enter a valid input!"	Pass
Triangle _Cov_6	Check for dimensions that form an isosceles triangle.	4	5	4	A message saying 'Given dimensions form an isosceles triangle!' appears.	A message saying 'Given dimensions form an isosceles triangle!' should appear.	Pass

Triangle _Cov_7	Check for dimensions that form an isosceles triangle.	5	4	4	A message saying ‘Given dimensions form an isosceles triangle!’ appears.	A message saying ‘Given dimensions form an isosceles triangle!’ should appear.	Pass
Triangle _Cov_8	Check for dimensions that do not form a triangle.	7	4	2	A message saying ‘Given dimensions do not form a triangle!’ appears.	A message saying ‘Given dimensions do not form a triangle!’ should appear.	Pass
Triangle _Cov_9	Check for dimensions that do not form a triangle.	4	7	2	A message saying ‘Given dimensions do not form a triangle!’ appears.	A message saying ‘Given dimensions do not form a triangle!’ should appear.	Pass
Triangle _Cov_10	Check for dimensions that are not valid inputs.	4	-5	6	A message is displayed as “Enter a valid input!”	A message must be displayed as “Enter a valid input!”	Pass
Triangle _Cov_11	Check for dimensions that are not valid inputs.	4	5	-6	A message is displayed as “Enter a valid input!”	A message must be displayed as “Enter a valid input!”	Pass
Triangle _Cov_12	Check for dimensions that are not valid inputs.	4	205	-6	A message is displayed as “Enter a valid input!”	A message must be displayed as “Enter a valid input!”	Pass
Triangle _Cov_13	Check for dimensions that are not valid inputs.	204	205	209	A message is displayed as “Enter a valid input!”	A message must be displayed as “Enter a valid input!”	Pass
Triangle _Cov_14	Check for dimensions that are not valid inputs.	5	5	209	A message is displayed as “Enter a valid input!”	A message must be displayed as “Enter a valid input!”	Pass

EXECUTION



RESULT & DISCUSSION: Thus, the above programs are written and executed using JUnit and ECLEMMA, and 100% coverage is achieved.

Exp. No. : 6

Date : 01-10-2022

DEMONSTRATION OF WHITE BOX TESTING TECHNIQUE USING ECLEMMA

Demonstrate white box testing techniques using open-source testing tool JUnit and ECLEMMA. Implement and execute test cases for achieving full statement coverage, decision/branch coverage and condition coverage for the NextDate problem.

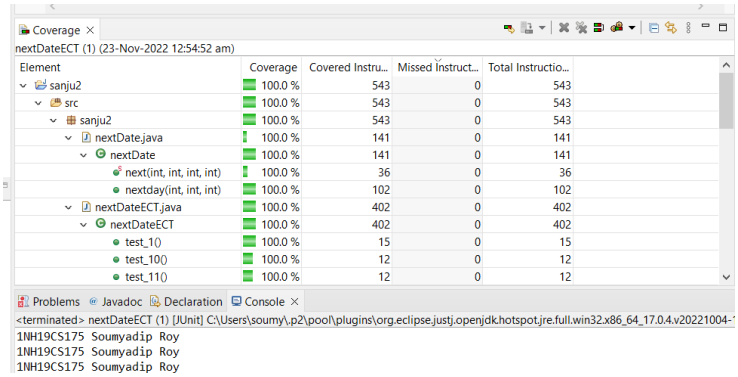
IMPLEMENTATION

1. NextDate function

```
import java.util.Scanner;

public class date {

	public String nextDate (int d, int m, int y) {
		int nd, nm, ny;
		if(d>31 || d<1 || m>12 || m <1 || y<1821 || y>2021){
			return ("Invalid date!");
		}
		else if(m==2 || m==4 || m==6 || m==9 || m==11 ){
			if(d==31){
				return("Invalid date!");
			}
			else if(m==2){
				if(checkLeapYear(y)){
					if(d>29){
						return("Invalid date!");
					}
					if(d == 29) {
						nd = 1;
						nm = 3;
					}
					else {
						nd = ++d;
						nm = 2;
					}
				}
				else {
					if(d>28){
						return("Invalid date!");
					}
					if(d == 28) {
						nd = 1;
						nm = 3;
					}
					else {
						nd = ++d;
						nm = 2;
					}
```

```
                        }
                        ny = y;
                    }
                    else {
                        if (d == 30){
                            nd = 1;
                            nm = ++m;
                        }
                        else{
                            nd = ++d;
                            nm = m;
                        }
                        ny = y;
                    }
                }
                else {
                    if (d == 31){
                        if (m!=12){
                            nd = 1;
                            nm = ++m;
                            ny = y;
                        }
                        else {
                            nd = 1;
                            nm = 1;
                            ny = ++y;
                        }
                    }
                    else{
                        nd = ++d;
                        nm = m;
                        ny = y;
                    }
                }
                return("The next date is: "+nd+"-"+nm+"-"+ny);
        }

        public static boolean checkLeapYear(int year){
            if(year % 400 == 0)
                return true;
            else if(year % 100 == 0)
                return false;
            else if(year % 4 == 0)
                return true;
            else
                return false;
        }

}
```

EXECUTION



RESULT & DISCUSSION: Thus, the above programs are written and executed using JUnit and ECLEMMA, and 100% coverage is achieved.

Exp. No. : 7

Date : 06-10-2022

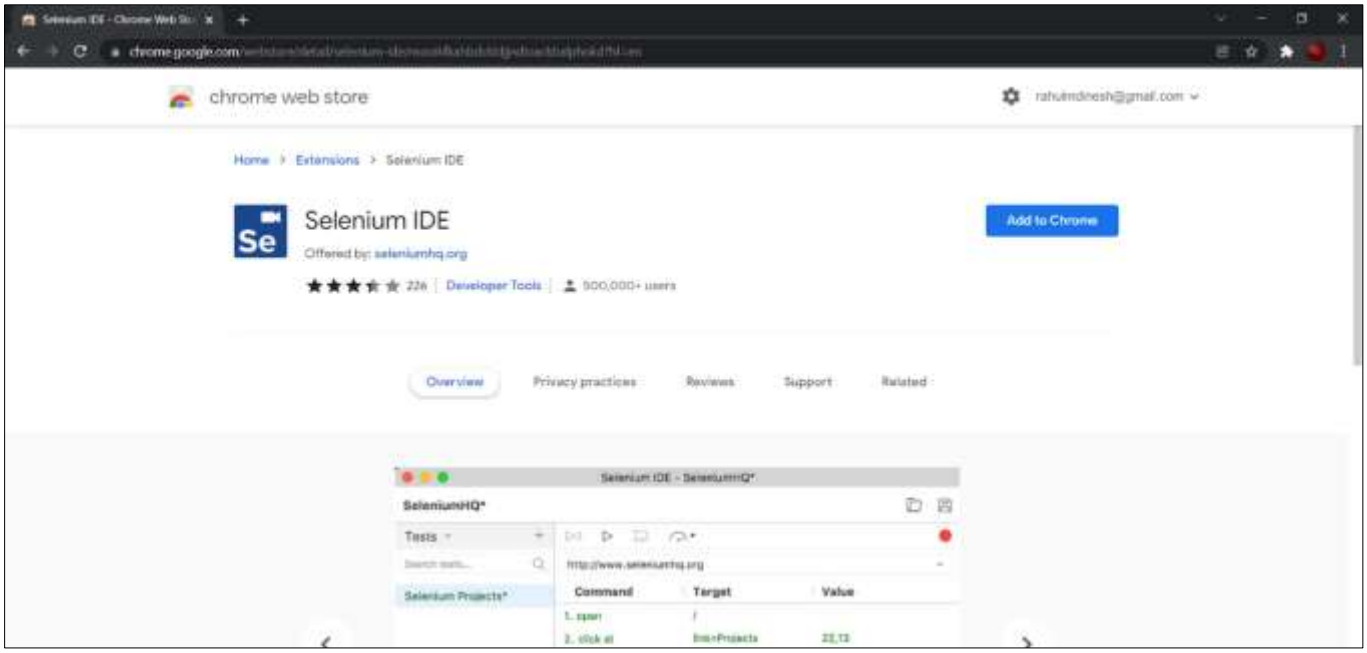
DEMONSTRATION OF SELENIUM IDE FOR CONDUCTING TEST ON WEBSITE(S)

Designing Test Cases using Selenium IDE.

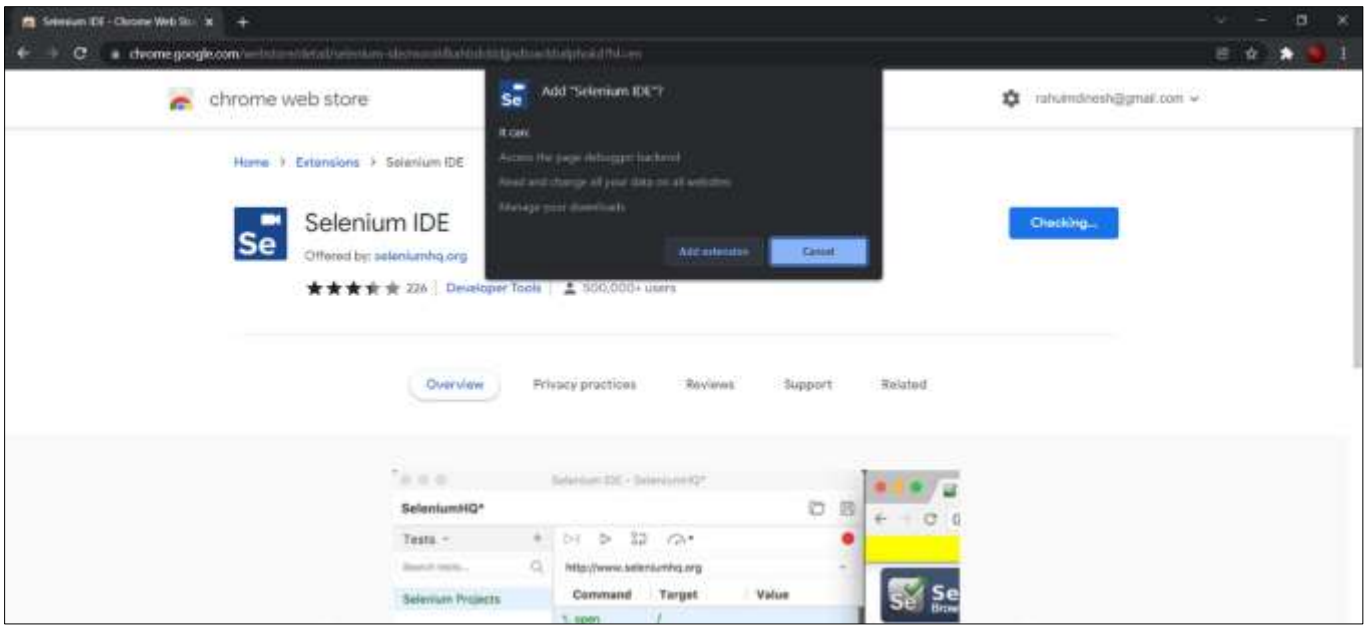
IMPLEMENTATION

Installing Selenium IDE

Step 1: Using Chrome, first, download the Selenium IDE extension from the Chrome Web Store by clicking on the 'Add to Chrome' button.

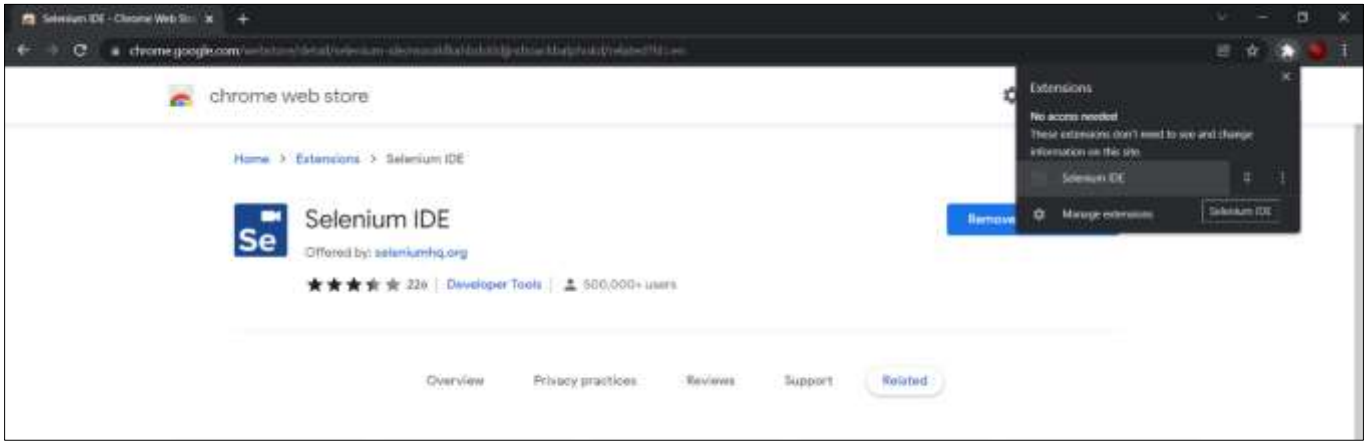


Step 2: This will lead to a dialog box appearing at the top of the screen. Read the permissions that the extension will have and click on 'Add extension'.

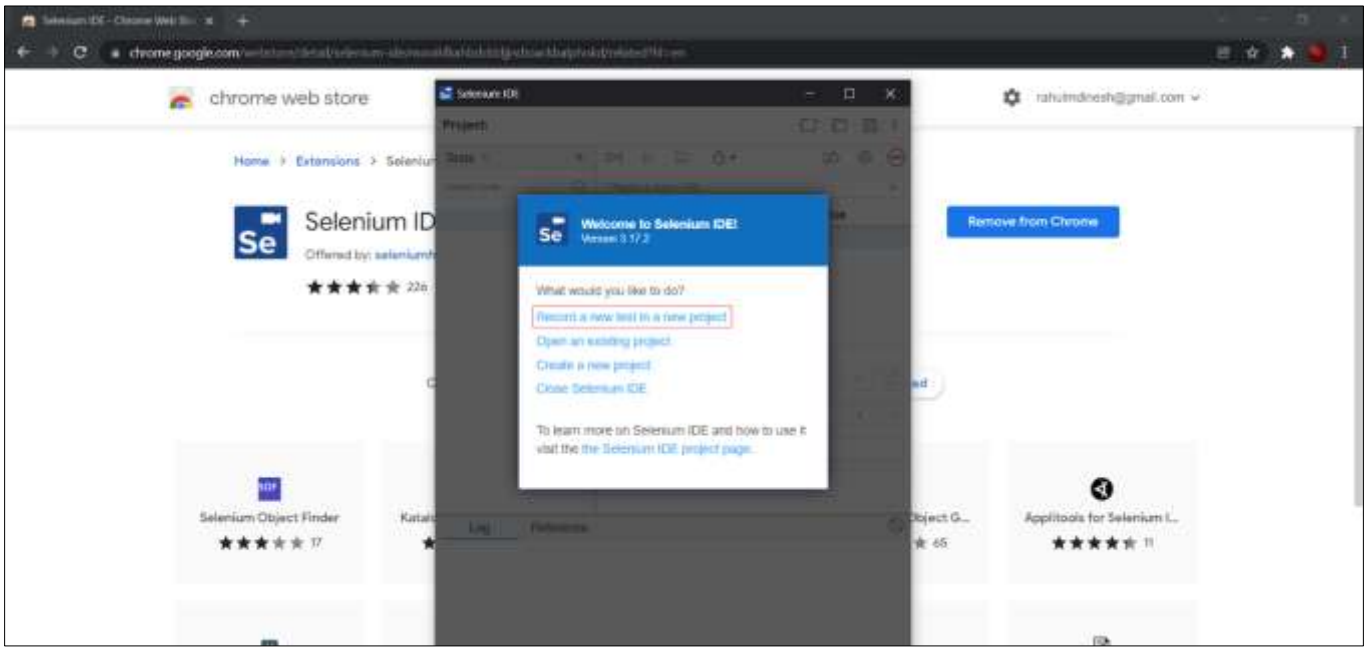


Launching Selenium IDE and Creating a new Project

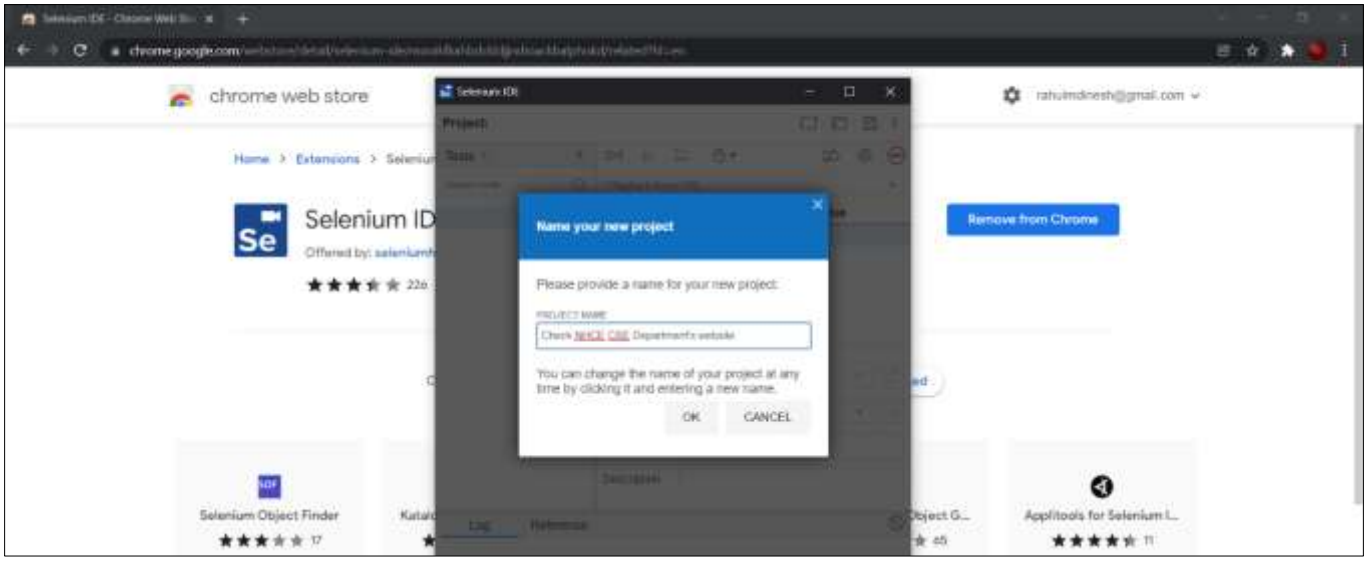
Step 3: Click on the Extension icon which looks like a puzzle piece, located in the top-right corner of the screen. Then click on 'Selenium IDE'.



Step 4: After the Selenium IDE extension launches, select 'Record a new test in a new project'.



Step 5: Give a suitable project name.



Recording and Running an Automated Test

Step 6: Provide the base URL. Then click 'Start Recording'.

Step 7: Now, it will take you to the base URL. Perform some action like googling something or visiting a website. These actions are being recorded by Selenium IDE. Once you are done, go to the Selenium IDE window and click on 'Stop Recording'. Then enter a suitable test name.

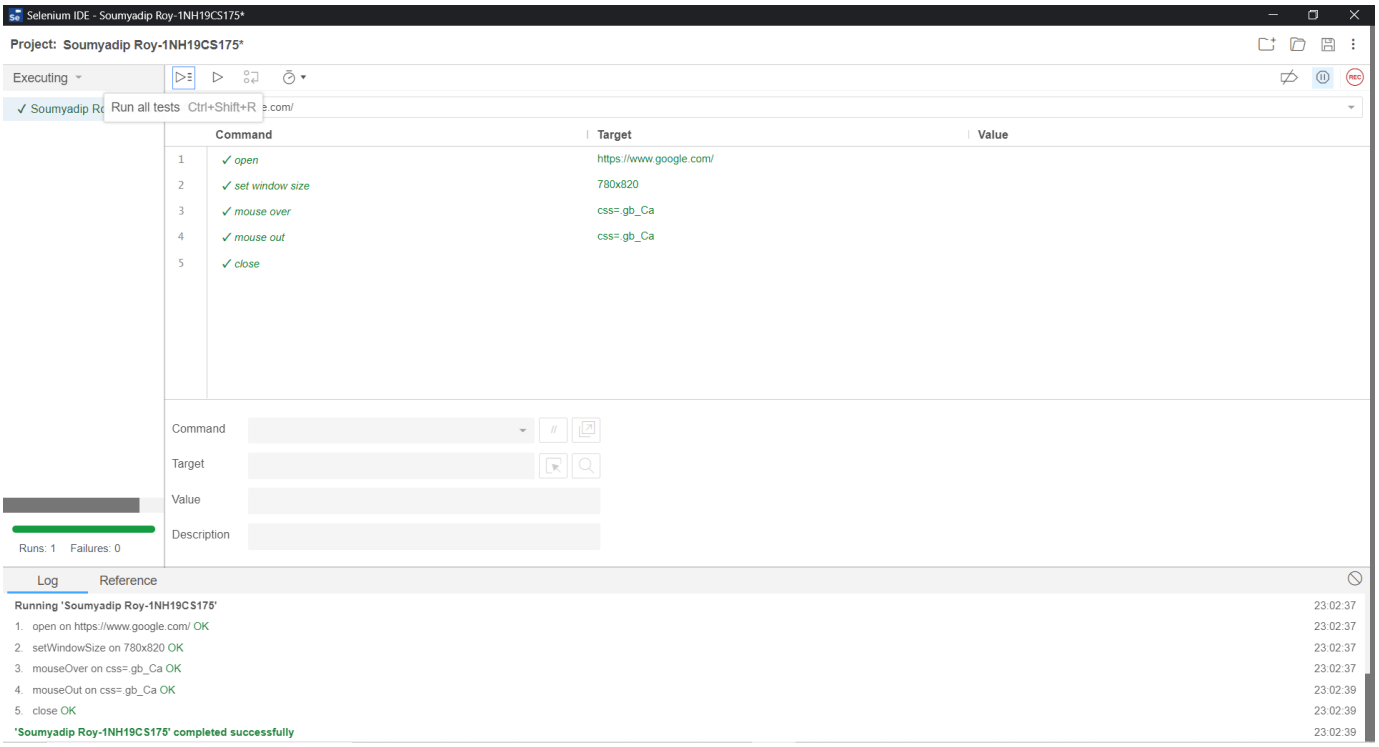
Step 8: Run the recorded steps as an automated test by clicking on the 'Run current test' button (▷)

TEST CASES

Test Case 1 - Manual Steps:

- Open (http://www.google.com)
- Type "nhce cse" in the Google Search Input Box
- Hit the 'Enter' key or click on 'Search' button
- Scroll down to find the intended website and click on it.
- Once the website loads, click on an link in the website (here, 'Syllabus')
- Click on any button in the newly opened tab.

EXECUTION



RESULT: Thus, the demonstration of Selenium IDE for conducting test on a website is done successfully.

Exp. No. : 8

Date : 13-10-2022

DEMONSTRATION OF SELENIUM WEBDRIVER FOR CONDUCTING TEST ON WEBSITE(S)

Write an automated selenium script to login into a web page by using Selenium Web driver, automate any website using Java based Selenium Script.

IMPLEMENTATION

Installation

Step 1:

i) Go to the Chromium Driver website: https://chromedriver.chromium.org
ii) Download the latest stable release
i) Ensure that your Chrome browser is updated to the latest version

Step 2:

iii) Go to the Selenium website's download page: https://www.selenium.dev/downloads/
iv) Download the latest stable version of Selenium Server

Step 3:

i) Extract the jar file of Selenium Server Standalone and add it to the Eclipse project
ii) Right click on the Project in the Project Explorer ⟶ Build Path ⟶ Configure Build Path ⟶ 'Libraries' tab ⟶ Add External Jar
iii) Now, navigate to the Selenium Server Standalone jar, downloaded earlier and add it.

Java based Selenium Script

```
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
public class exp8 {
	public static void main(String args[]){
		System.setProperty("webdriver.chrome.driver", "R:\\ST Lab \\Jar\\chromedriver.exe");
		ChromeDriver driver = new ChromeDriver();
		driver.manage().window().maximize();

		driver.get("file:///R:\\ST Lab\\login738.html");

		driver.findElement(By.name("username")).sendKeys("rahulmd");
		driver.findElement(By.name("password")).sendKeys("12345");
		driver.findElement(By.name("submit")).click();
	}
}
```

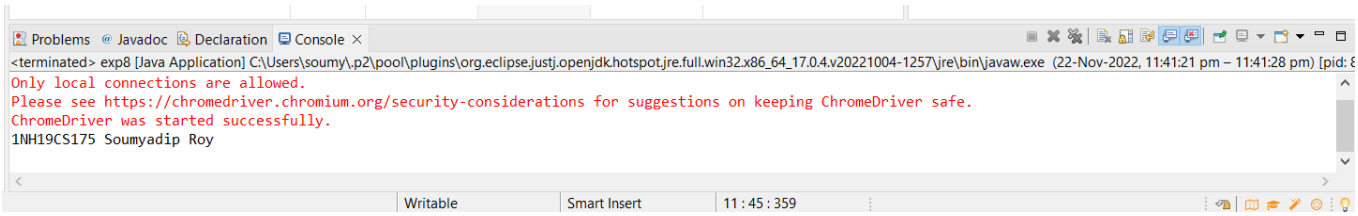
HTML code for Login Page

```
<form action="loginSuccess.html">
    <label><b>Username</b></label>
    <input name="username" type="text" required>
    <br/>
    <label><b>Password</b></label>
    <input name="password" type="password" required>
    <br/>
    <button name="submit" type="submit">Login</button>
</form>
```

HTML code for Login Sucess Page

```
<h1>Login Successful!</h1>
```

EXECUTION



RESULT: Thus, the above program, written and executed using selenium web driver has successfully tested the login functionality of a sample web page.

Exp. No. : 9

Date : 27-10-2022

DEMONSTRATION OF SELENIUM WEBDRIVER FOR CONDUCTING TEST ON WEBSITE(S)

Write a test program to list the total number of objects present on a web page

IMPLEMENTATION

Installation

Step 1:

i) Go to the Chromium Driver website: https://chromedriver.chromium.org
ii) Download the latest stable release
iii) Ensure that your Chrome browser is updated to the latest version

Step 2:

i) Go to the Selenium website's download page: https://www.selenium.dev/downloads/
ii) Download the latest stable version of Selenium Server

Step 3:

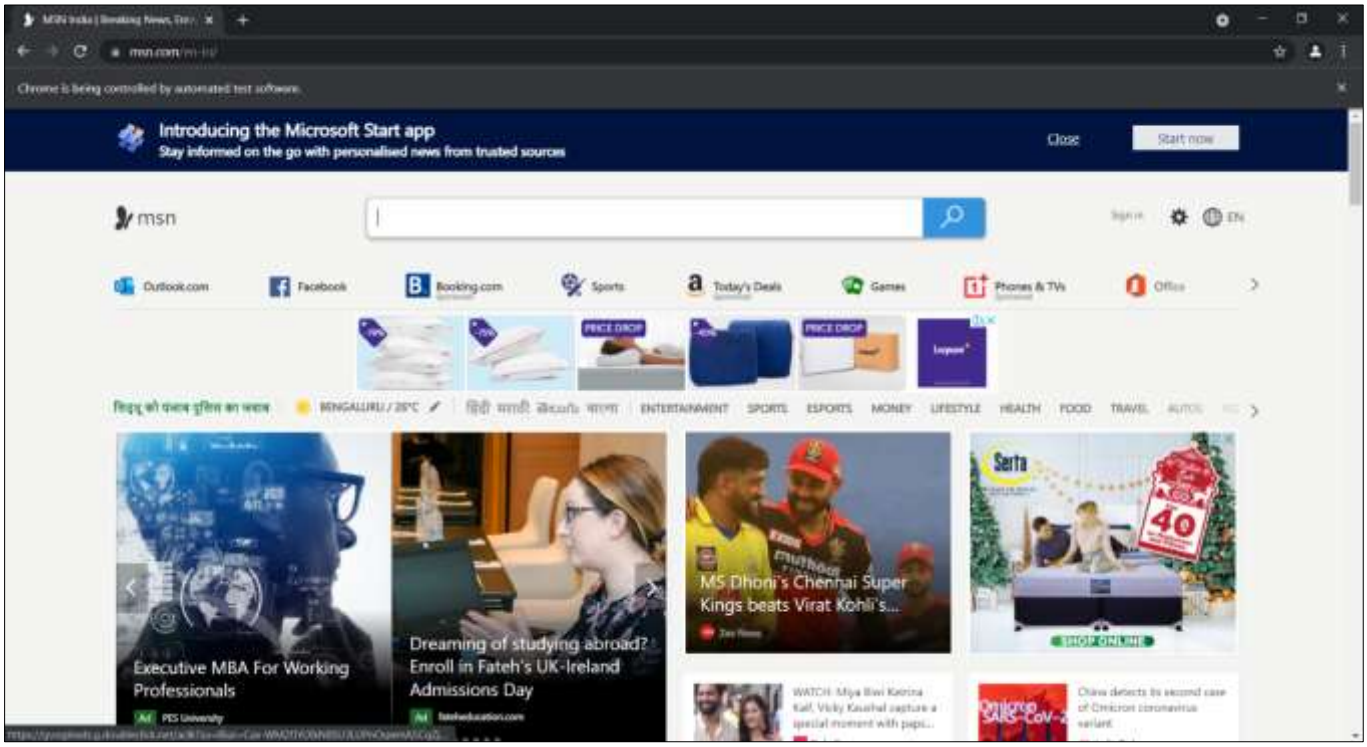
i) Extract the jar file of Selenium Server Standalone and add it to the Eclipse project
ii) Right click on the Project in the Project Explorer ⟶ Build Path ⟶ Configure Build Path ⟶ 'Libraries' tab ⟶ Add External Jar
iii) Now, navigate to the Selenium Server Standalone jar, downloaded earlier and add it.

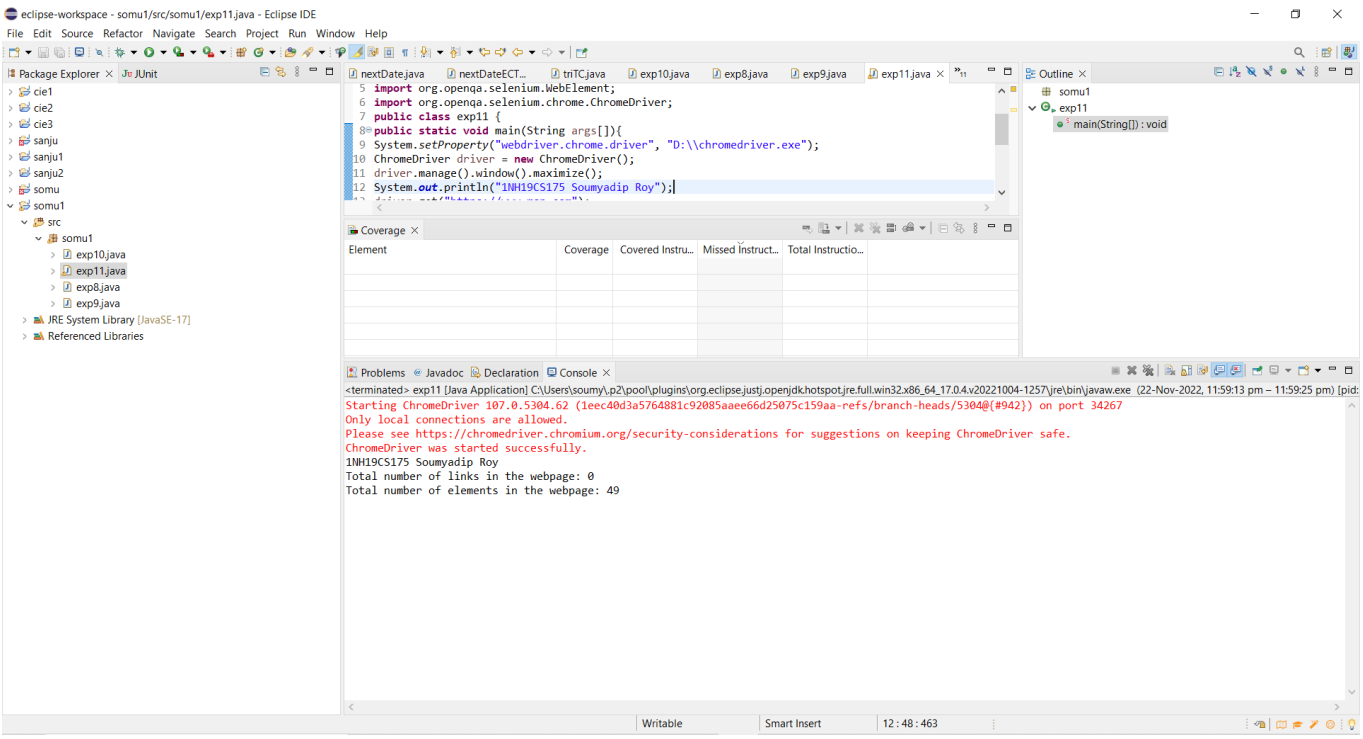
Java based Selenium Script

```
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;

public class exp9 {
	public static void main(String args[]){
		System.setProperty("webdriver.chrome.driver", "R:\\ST Lab \\Jar\\chromedriver.exe");
		ChromeDriver driver = new ChromeDriver();
		driver.manage().window().maximize();

		driver.get("https://www.msn.com");

		List<WebElement> linksList = driver.findElements(By.xpath("//a"));
		int linkCount = linksList.size();
		System.out.println("Total number of links in the webpage: "+linkCount);

		List<WebElement> elementsList = driver.findElements(By.xpath("//*"));
		int elementsCount = elementsList.size();
```

```
            System.out.println("Total number of elements in the webpage: "+elementsCount);
        }
}
```

EXECUTION





RESULT: Thus, the above program, written and executed using selenium web driver has successfully displayed the total number of links and elements in the given webpage.

Exp. No. : 10

Date : 10-11-2022

DEMONSTRATION OF SELENIUM WEBDRIVER FOR CONDUCTING TEST ON WEBSITE(S)

Write a test program to demonstrate URL and title check point

IMPLEMENTATION

Installation

Step 1:

i) Go to the Chromium Driver website: https://chromedriver.chromium.org
ii) Download the latest stable release
iii) Ensure that your Chrome browser is updated to the latest version

Step 2:

i) Go to the Selenium website's download page: https://www.selenium.dev/downloads/
ii) Download the latest stable version of Selenium Server

Step 3:

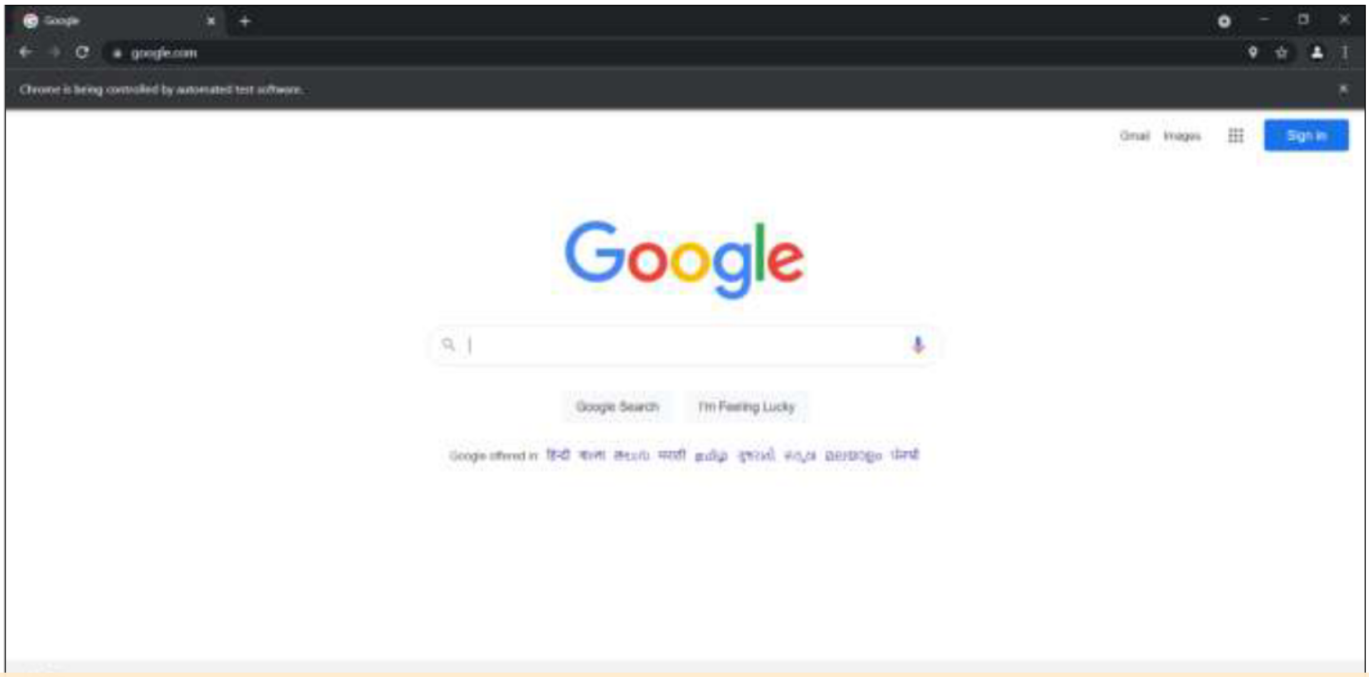
i) Extract the jar file of Selenium Server Standalone and add it to the Eclipse project
ii) Right click on the Project in the Project Explorer ⟶ Build Path ⟶ Configure Build Path ⟶ 'Libraries' tab ⟶ Add External Jar
iii) Now, navigate to the Selenium Server Standalone jar, downloaded earlier and add it.

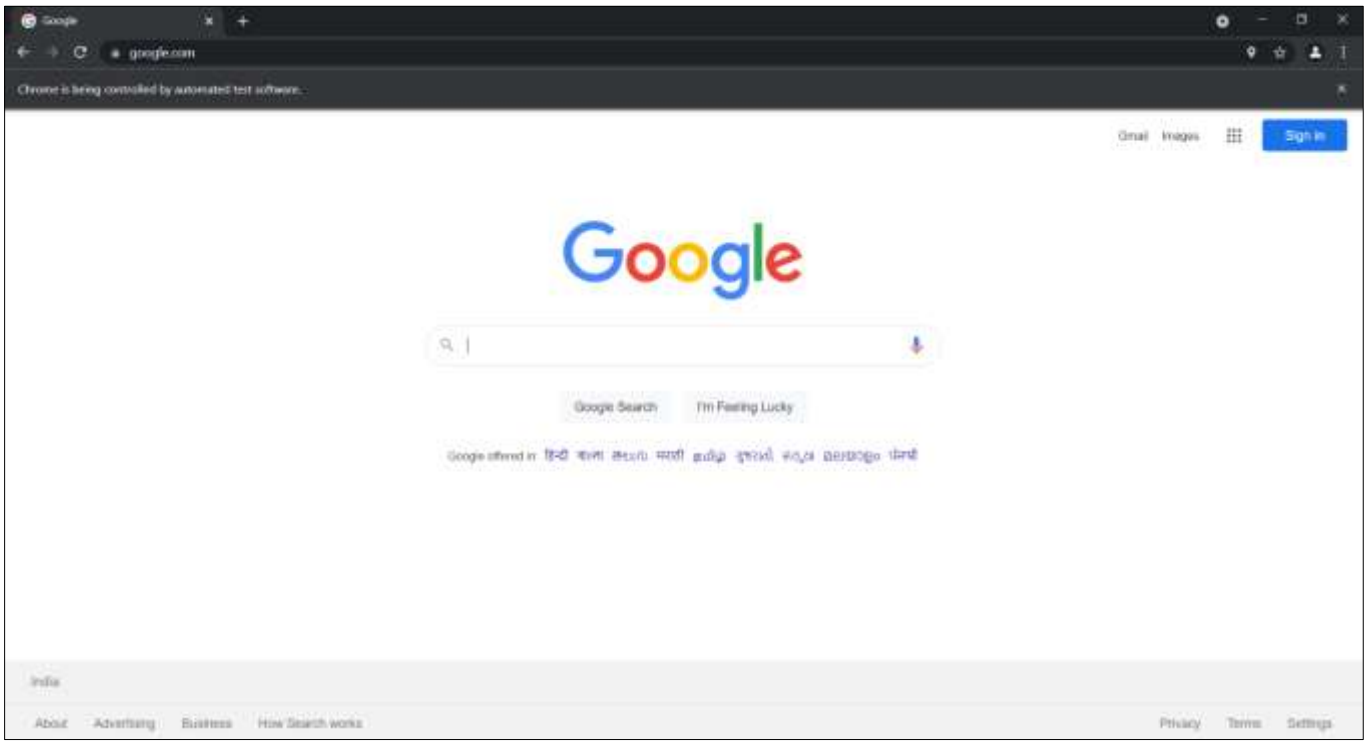
Java based Selenium Script

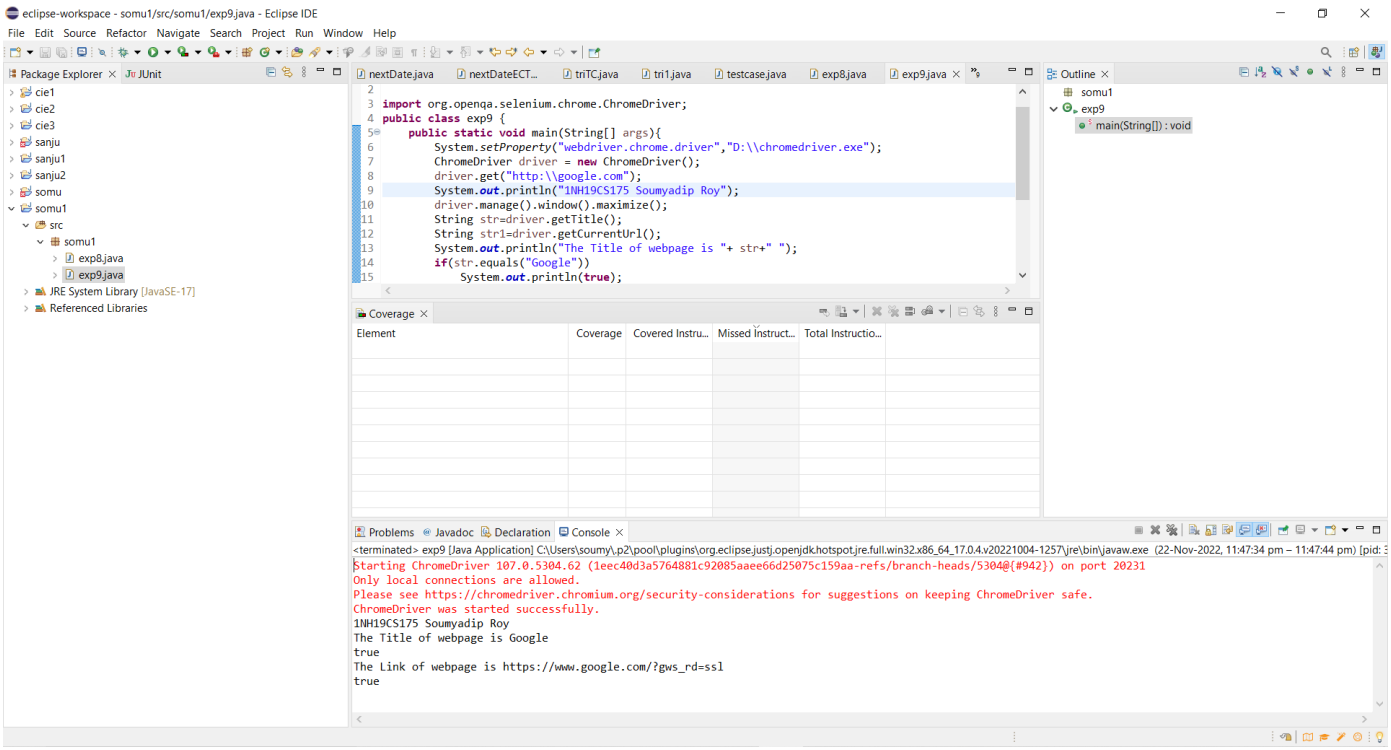
```
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
public class exp10 {
	public static void main(String args[]){
		System.setProperty("webdriver.chrome.driver", "R:\\ST Lab \\Jar\\chromedriver.exe");
		ChromeDriver driver = new ChromeDriver();
		driver.manage().window().maximize();

		driver.get("https://en.wikipedia.org/wiki/Wikipedia");
		String url = driver.getCurrentUrl();
		System.out.println("Current URL: " + url);
		if(url.equals("https://en.wikipedia.org/wiki/Wikipedia"))
			System.out.println("URL matches!");
		else
			System.out.println("URL doesn't match.");

		driver.get("https://www.google.com");
```

```
            String title = driver.getTitle();
            System.out.println("Current Title: " + title);
            if(title.equals("Google"))
                System.out.println("Title matches!");
            else
                System.out.println("Title doesn't match.");
        }
}
```

EXECUTION







RESULT: Thus, the above program, written and executed using selenium web driver has successfully displayed and verified the URL and title of two different websites using check point.

Exp. No. : 11

Date : 17-11-2022

DEMONSTRATION OF SELENIUM IDE & WEBDRIVER FOR CONDUCTING TEST ON WEBSITE(S)

Write a test program to demonstrate selecting and deselecting option from multi select dropdown

IMPLEMENTATION

Installation

Step 1:

i) Go to the Chromium Driver website: https://chromedriver.chromium.org
ii) Download the latest stable release
ii) Ensure that your Chrome browser is updated to the latest version

Step 2:

i) Go to the Selenium website's download page: https://www.selenium.dev/downloads/
ii) Download the latest stable version of Selenium Server

Step 3:

i) Extract the jar file of Selenium Server Standalone and add it to the Eclipse project
ii) Right click on the Project in the Project Explorer ⟶ Build Path ⟶ Configure Build Path ⟶ 'Libraries' tab ⟶ Add External Jar
iii) Now, navigate to the Selenium Server Standalone jar, downloaded earlier and add it.

Java based Selenium Script

```
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;
public class exp11 {

    public static void main(String[] args) throws InterruptedException  {
        System.setProperty("webdriver.chrome.driver", "R:\\ST Lab \\Jar\\chromedriver.exe");
        ChromeDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

        driver.get("file:///R:\\ST Lab\\multiselect.html");
        Select select = new Select(driver.findElement(By.id("depts")));
```

```
System.out.println("The multiselect options are: ");
List<WebElement> options = select.getOptions();
for(WebElement option: options)
        System.out.println(option.getText());

 System.out.println("Is the selected element a multiselect element?: "+ select.isMultiple());

 if(select.isMultiple()){
        System.out.println("Selecting option ECE using its index.");
        select.selectByIndex(2);
        Thread.sleep(4000);

        System.out.println("Selecting option ISE using its value.");
        select.selectByValue("ise");
        Thread.sleep(4000);

        System.out.println("Selecting option CSE using its visible text.");
        select.selectByVisibleText("CSE");
        Thread.sleep(4000);

        System.out.println("The selected options are: ");
        options = select.getAllSelectedOptions();
        for(WebElement option: options)
                System.out.println(option.getText());

        System.out.println("Deselecting option ECE using its index.");
        select.deselectByIndex(2);
        Thread.sleep(4000);

        System.out.println("Deselecting option ISE using its value.");
        select.deselectByValue("ise");
        Thread.sleep(4000);

        System.out.println("The selected values after deselecting some options are: ");
        options = select.getAllSelectedOptions();
        for(WebElement option: options)
                System.out.println(option.getText());
```

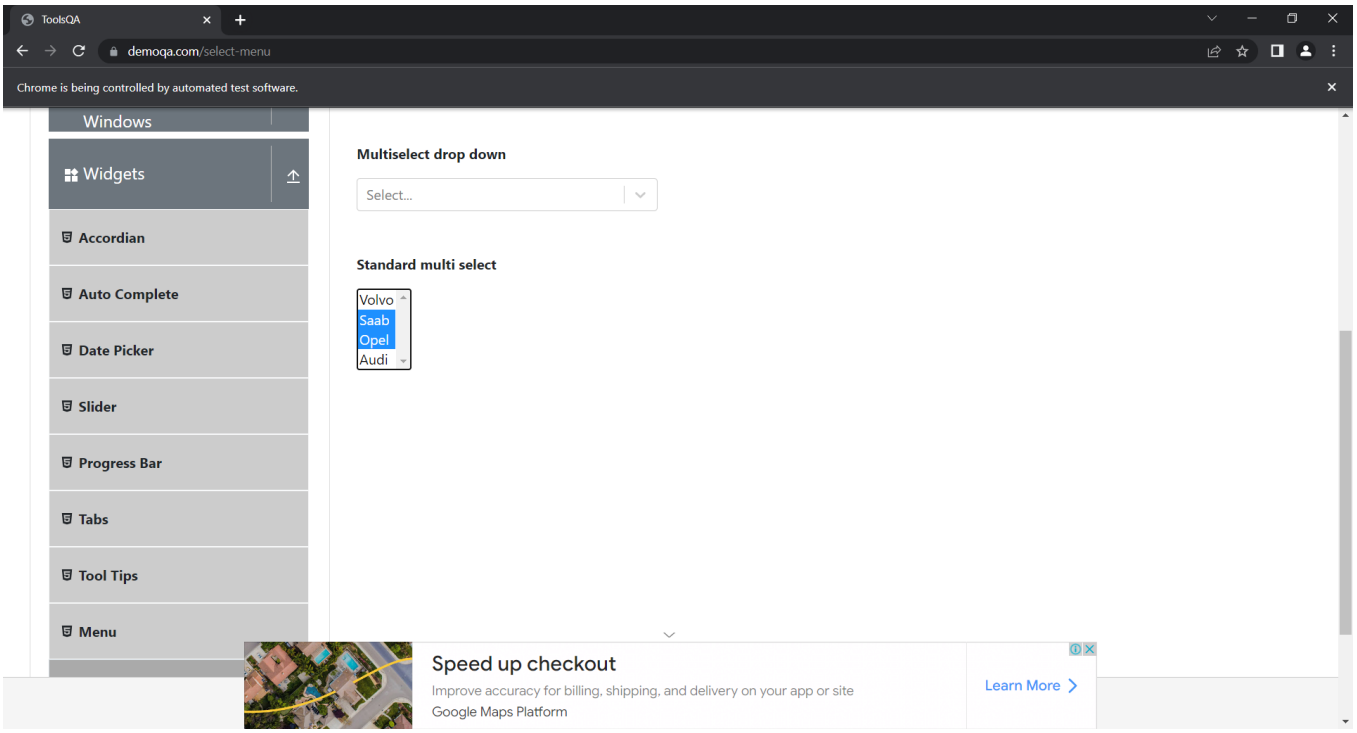
```
                System.out.println("Deselecting all options.");
                select.deselectAll();
            }
        }
}
```

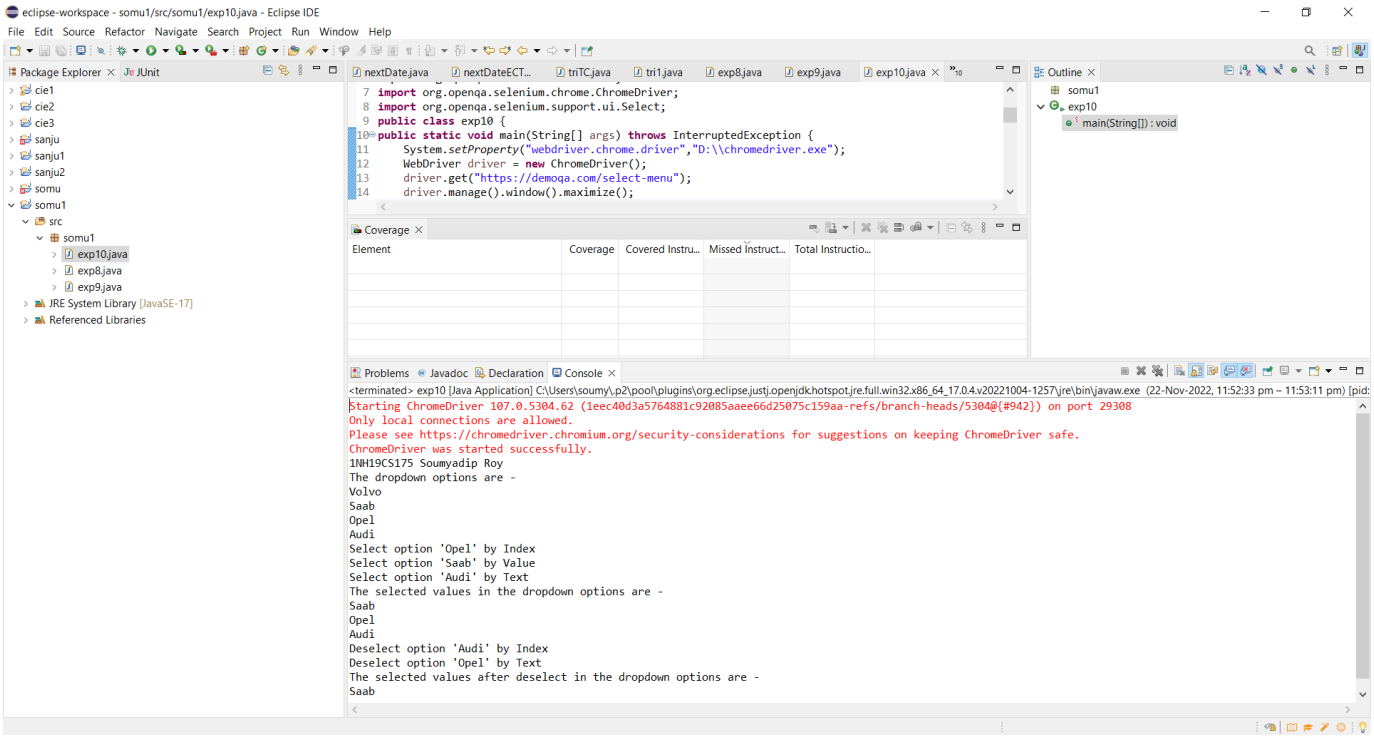
HTML code for Multiselect Page

```
<h1>
    Select and Deselect Departments
</h1>
<form>
    <select multiple name="depts" id="depts">
        <option value="cse">CSE</option>
        <option value="ise">ISE</option>
        <option value="ece">ECE</option>
        <option value="eee">EEE</option>
    </select>
</form>
```

EXECUTION

The HTML page having a Multiselect Element





RESULT: Thus, the above program, written and executed using selenium web driver has successfully tested the functionality of a multiselect element by selecting and deselecting various options.

Exp. No. : 12

Date : 24-11-2022

DEMONSTRATION OF SELENIUM IDE & WEBDRIVER FOR CONDUCTING TEST ON WEBSITE(S)

Write a test program to demonstrate Synchronization

IMPLEMENTATION

Installation

Step 1:

i) Go to the Chromium Driver website: https://chromedriver.chromium.org
ii) Download the latest stable release
iii) Ensure that your Chrome browser is updated to the latest version

Step 2:

i) Go to the Selenium website's download page: https://www.selenium.dev/downloads/
ii) Download the latest stable version of Selenium Server

Step 3:

i) Extract the jar file of Selenium Server Standalone and add it to the Eclipse project
ii) Right click on the Project in the Project Explorer ⟶ Build Path ⟶ Configure Build Path ⟶ 'Libraries' tab ⟶ Add External Jar
iii) Now, navigate to the Selenium Server Standalone jar, downloaded earlier and add it.

Java based Selenium Script (Implicit Wait)

```
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;

public class exp12_implicit {
	public static void main(String[] args){
		System.setProperty("webdriver.chrome.driver", "R:\\ST Lab \\Jar\\chromedriver.exe");
		ChromeDriver driver = new ChromeDriver();
		driver.manage().window().maximize();

		String eText = "Welcome";		//Expected Text
		String aText_1="";			//Actual Text (Test 1)
		String aText_2="";			//Actual Text (Test 2)
```

```
		//***Using Implicit Wait***
		driver.get("file:///R:\\\\ST Lab\\sample.html");
		driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

		//***Test 1 - Will Pass as element exists***
		try{
			aText_1 = driver.findElement(By.id("welcome")).getText();
			if (aText_1.equals(eText))
				System.out.println("Test 1 Passed using Implicit Wait");
		}
		catch (NoSuchElementException e){
			System.out.println("Test 1 Failed using Implicit Wait");
		}

		//***Test 2 - Will Fail after waiting for 10s as element doesn't exist****
		try{
			aText_2 = driver.findElement(By.id("abcd")).getText();
			if (aText_2.equals(eText))
				System.out.println("Test 2 Passed using Implicit Wait");
		}
		catch (NoSuchElementException e){
			System.out.println("Test 2 Failed using Implicit Wait");
		}
	}
}
```

Java based Selenium Script (Explicit Wait)

```
import org.openqa.selenium.By;
import org.openqa.selenium.TimeoutException;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class exp12_explicit {
	public static void main(String[] args){
		System.setProperty("webdriver.chrome.driver", "R:\\ST Lab \\Jar\\chromedriver.exe");
```

```
        ChromeDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

        String eText = "Welcome";           //Expected Text
        String aText_1="";                  //Actual Text (Test 1)
        String aText_2="";                  //Actual Text (Test 2)

        //***Using Explicit Wait***
        driver.get("file:///R:\\ST Lab\\sample.html");
        WebDriverWait wait = new WebDriverWait(driver, 10);

        //***Test 1 - Will Pass as element exists***
        try{
            WebElement text_1 =
            wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("welcome")));
            aText_1 = text_1.getText();
            if (aText_1.equals(eText))
                System.out.println("Test 1 Passed using Explicit Wait");
        }
        catch (TimeoutException e){
            System.out.println("Test 1 Failed using Explicit Wait");
        }

        //***Test 2 - Will Fail after waiting for 10s as element doesn't exist****
        try{
            WebElement text_2 =
            wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("abcd")));
            aText_2 = text_2.getText();
            if (aText_2.equals(eText))
                System.out.println("Test 2 Passed using Explicit Wait");
        }
        catch (TimeoutException e){
            System.out.println("Test 2 Failed using Explicit Wait");
        }
    }
}
```

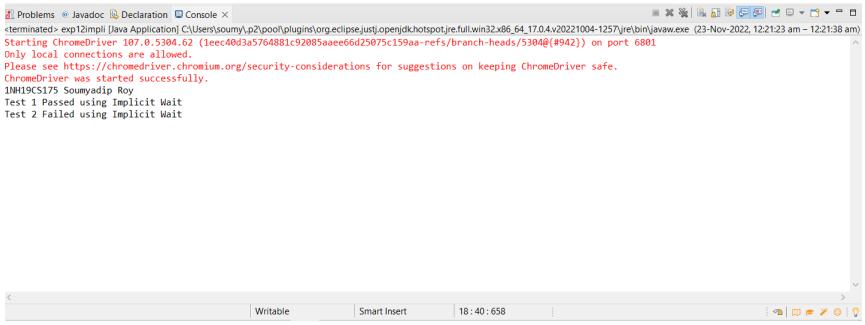
HTML code for Sample Page

```
<h1 id="welcome">Welcome</h1>
```

EXECUTION

The Sample HTML page

Implicit Wait



Explicit Wait

RESULT: Thus, the above program, written and executed using selenium web driver has successfully demonstrated synchronization using implicit and explicit waits.