|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SN** | **Date of**  **Evaluation** | **Name of Experiment/ Program** | **1** | **2** | **3** | **4** | **Total** | **Faculty**  **Signature** |
| Write test cases for the following scenarios | | | | | | | | |
| 1. | 13/10/2021 | ATM System |  |  |  |  |  |  |
| 2. | 20/10/2021 | The Triangle Problem |  |  |  |  |  |  |
| Demonstrate Black box testing techniques using open-source testing tool - JUnit | | | | | | | | |
| 3. | 20/10/2021 | Boundary Value Analysis (BVA) for the NextDate Function |  |  |  |  |  |  |
| 4. | 10/11/2021 | Equivalence Class Partitioning for the NextDate Function |  |  |  |  |  |  |
| Demonstrate White box testing techniques using open-source testing tool - EclEmma | | | | | | | | |
| 5. | 17/11/2021 | The Triangle Problem |  |  |  |  |  |  |
| 6. | 27/11/2021 | The NextDate Function |  |  |  |  |  |  |
| Demonstration of Selenium IDE & Webdriver for conducting test on websites | | | | | | | | |
| 7. | 18/12/2021 | Using Selenium IDE to conduct a test for any web site |  |  |  |  |  |  |
| 8. | 08/12/2021 | Using Selenium Web driver, automate any web page using Java Script |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SN** | **Date of Evaluation** | **Name of Experiment / Program** | **1** | **2** | **3** | **4** | **Total** | **Faculty Signature** |
| 9. | 15/12/2021 | List the total number of objects present on a web page |  |  |  |  |  |  |
| 10. | 22/12/2021 | Demonstrate URL and title check point |  |  |  |  |  |  |
| 11. | 29/12/2021 | Demonstrate selecting and deselecting option from multi select dropdown |  |  |  |  |  |  |
| 12. | 05/01/2022 | Demonstrate Synchronization. |  |  |  |  |  |  |

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

**Exp. No. : 1 Date :**

**ATM SYSTEM**

Consider any ATM system, design and develop a program in a language of your choice for the same. Create the test cases for the following scenarios:

1. Unsuccessful operation due to entering wrong PIN number 3 times.
2. Unsuccessful operation due to invalid account type.
3. Successful selection of amount to be withdrawn.
4. Expected message due to amount to withdraw is greater than possible balance

**IMPLEMENTATION:**

import java.util.\*; class Check{

int checkPin(int p,int i){ if(p==i)

return 1;

System.out.println("Invalid PIN number"); return 0;

}

int checkAccount(int t){ if(t==1){

System.out.println("Opened Savings account"); return 1;

}

else if(t==1){

System.out.println("Opened Current account"); return 1;

}

else{

System.out.println("Invalid Account Type"); return 0;

}

}

void checkBal(int bal){

System.out.println("Your available balance is="+bal);

}

int withdraw(int a,int b){ if(a>b){

System.out.println("Insuffecient funds"); return 0;

}

b=b-a;

System.out.println("Successful transaction"); checkBal(b);

return b;

}

int deposit(int a,int b){ b=b+a;

System.out.println("Successful transaction"); checkBal(b);

return b;

}

}

public class ATM {

public static void main(String[] args) {

// TODO Auto-generated method stub Scanner sc=new Scanner(System.in); Check obj=new Check();

int r,pass=5555,k=0,balance=5000; int inp;

while(k<3){

System.out.println("Enter PIN number"); inp=sc.nextInt();

r=obj.checkPin(pass,inp); if(r==1){

System.out.println("\nWelcome\n"); break;

}

k++;

}

if(k==3){

System.out.println("Ttransaction Terminated"); System.exit(0);

} r=0;

while(r==0){

System.out.println("Enter the account type\n1-Savings\n2-Current\n"); int type=sc.nextInt();

r=obj.checkAccount(type);

}

int trans=0;

while(trans!=4){

System.out.println("Select transaction:\n\n1-Check Balance\n2-Withdraw\n3-Deposit\n4-Exit\n"); trans=sc.nextInt();

switch(trans){ case 1:

obj.checkBal(balance); break;

case 2:

System.out.println("Enter the amount you want to withdraw"); int amt=sc.nextInt();

if(amt%500!=0){

System.out.println("Invalid amount");

}

else{

r=obj.withdraw(amt,balance); balance=r;

};

break; case 3:

System.out.println("Enter the amount you want to deposit"); int amt2=sc.nextInt();

int b=obj.deposit(amt2,balance); balance=b;

break;

}

}

}

}

**TEST CASES:**

**Example:**

**TEST CASE 1:** Unsuccessful operation due to entering wrong PIN number 3 times.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | **Test Information** | | | | |
| Project Name: | ATM | | Test Name: | | Invalid PIN Number | | |
| Project ID: | ATM\_01 | | Original Author: | | Tester QA | | |
| Test Objective: | This test case is to verify the functionality with invalid pin number | | | | | | |
| **Step No.** | **Test Case Description** | **Test Data** | | **Expected Result** | | **Status (Pass/Fail)** | **Remark s** |
| 1 | Insert valid  card in the insertion point of ATM | Valid ATM card | | ATM should display language page with following objects English, Kannada, Hindi | | pass |  |
| 2 | Select the preferred language |  | | ATM should display the PIN number entry screen in selected language | | pass |  |
| 3 | Enter the  invalid pin number | Invalid PIN number | | ATM does not validate PIN and prompts customer to reenter PIN. | | pass |  |
| 4 | Reenter incorrect PIN | Invalid PIN number | | ATM does not validate PIN and prompts customer to reenter PIN | | pass |  |
| 5 | Reenter incorrect PIN | Invalid PIN number | | ATM does not validate PIN | | pass |  |

**TEST CASE 2:** Unsuccessful operation due to invalid account type.

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Information** | | **Test Information** | |
| Project Name: | ATM | Test Name: | Invalid PIN Number |
| Project ID: | ATM\_01 | Original Author: | Tester QA |
| Test Objective: | This testcase is to verify unsuccessful operation due to invalid account type. | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step No.** | **Test Case Description** | **Test Data** | **Expected Result** | **Status (Pass/Fail)** | **Remark s** |
| 1 | Insert valid  card in the insertion point of ATM | Valid ATM card | ATM should display language page with following objects English, Kannada, Hindi | pass |  |
| 2 | Select the preferred language |  | ATM should display the PIN number entry screen in selected language | pass |  |
| 3 | Enter the valid pin number | valid PIN number | ATM does validate PIN and prompts account type option screen. | pass |  |
| 4 | Select invalid account type | Invalid account type | ATM prompts to re-enter the proper account type | pass |  |

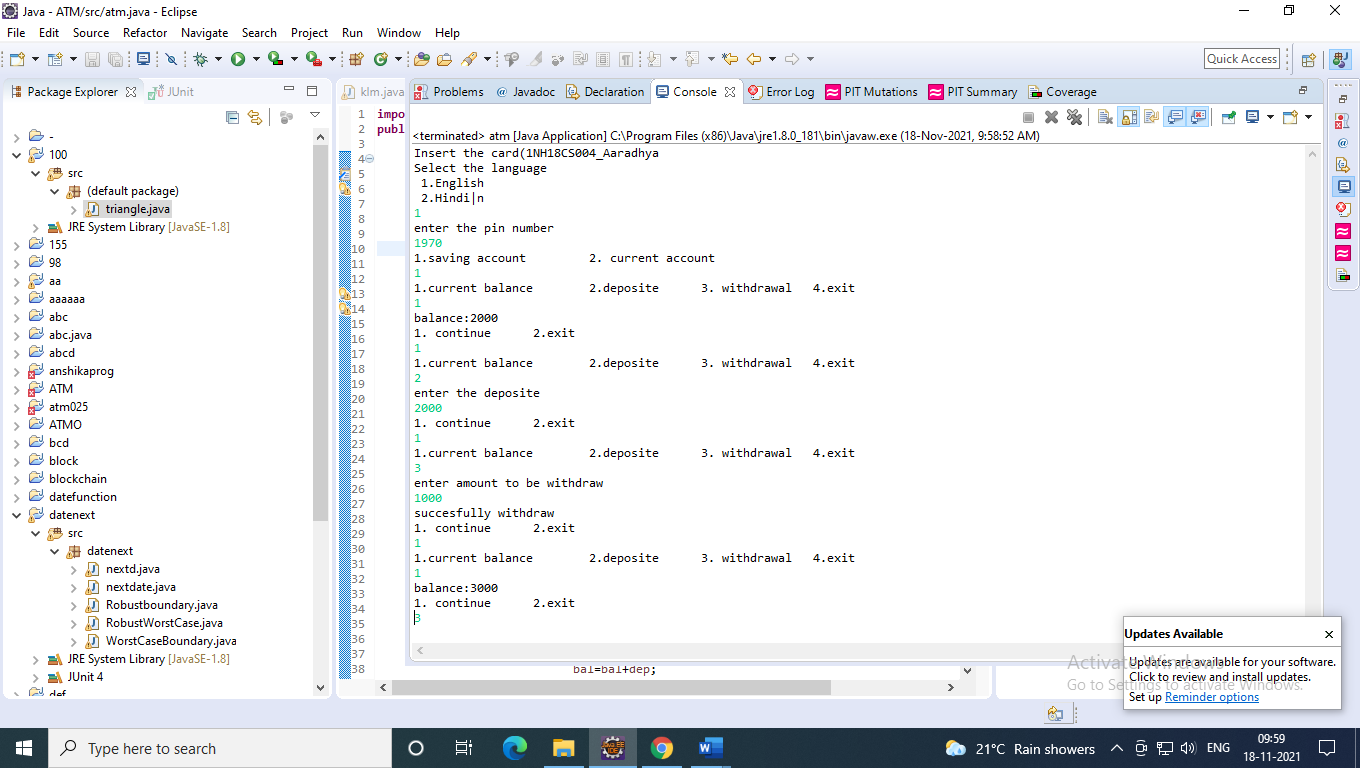
**TEST CASE 3:** Successful selection of amount to be withdrawn.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | **Test Information** | | | | |
| Project Name: | ATM | | Test Name: | | Invalid PIN Number | | |
| Project ID: | ATM\_01 | | Original Author: | | Tester QA | | |
| Test Objective: | Successful selection of amount to be withdrawn. | | | | | | |
| **Step No.** | **Test Case Description** | **Test Data** | | **Expected Result** | | **Status (Pass/Fail)** | **Remark s** |
| 1 | Insert valid  card in the insertion point of ATM | Valid ATM card | | ATM should display language page with following objects English, Kannada, Hindi | | pass |  |
| 2 | Select the preferred language |  | | ATM should display the PIN number entry screen in selected language | | pass |  |
| 3 | Enter the valid pin number | valid PIN number | | ATM does validate PIN and prompts account type option screen. | | pass |  |
| 4 | Select valid account type | valid account type | | ATM does validate account type and it should display different functions deposit, withdraw, balance check | | pass |  |
| 5 | Select the with- draw option | Valid option | | ATM prompts the screen for the user to enter the amount | | pass |  |
| 6 | Select the  amount to withdraw | Amount to withdraw | | Successful transaction | | pass |  |

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | **Test Information** | | | | |
| Project Name: | ATM | | Test Name: | | Invalid PIN Number | | |
| Project ID: | ATM\_01 | | Original Author: | | Tester QA | | |
| Test Objective: | Expected message due to amount to withdraw is greater than possible balance. | | | | | | |
| **Step No.** | **Test Case Description** | **Test Data** | | **Expected Result** | | **Status (Pass/Fail)** | **Remark s** |
| 1 | Insert valid  card in the insertion point of ATM | Valid ATM card | | ATM should display language page with following objects English, Kannada, Hindi | | pass |  |
| 2 | Select the preferred language |  | | ATM should display the PIN number entry screen in selected language | | pass |  |
| 3 | Enter the valid pin number | valid PIN number | | ATM does validate PIN and prompts account type option screen. | | pass |  |
| 4 | Select valid account type | valid account type | | ATM does validate account type and it should display different functions deposit, withdraw, balance check | | pass |  |
| 5 | Select the with- draw option | Valid option | | ATM prompts the screen for the user to enter the amount | | pass |  |
| 6 | Select the amount greater than the  balance amount to withdraw | Amount greater then balance to withdraw | | ATM prompts a message saying selected amount is greater than available balance | | pass |  |

**EXECUTION**



**RESULT & DISCUSSION**

Test Report:

1. Number of Test Cases Executed : 4
2. Number of Test Cases Passed : 4
3. Number of Test Cases Failed : 0

**Exp. No. : 2 Date :**

**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:

* 1. Represents not a triangle
  2. Represents a valid scalene triangle
  3. Represents a valid equilateral triangle
  4. Represents a valid isosceles triangle

Execute the test cases manually and discuss the result.

**IMPLEMENTATION**

import java.util.Scanner;

public class TRIANGLE {

public String check(int a,int b,int c){

if((a>=1 && a<=200) && (b>=1 && b<=200) && (c>=1 && c<=200))

{

if((a+b>c)&&(b+c>a)&&(c+a>b))

{

if(a==b && b==c)

{

return ("Equilateral");

}

else if(a==b||b==c||c==a){

return ("Isosceles");

}

else

return ("Scalene");

}

return ("Not a Triangle");

}

return ("Invalid Values");

}

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in); TRIANGLE t= new TRIANGLE();

System.out.println("enter 3 sides");

int a=sc.nextInt(); int b=sc.nextInt();

int c=sc.nextInt();

System.out.println(t.check(a,b,c));

}

}

/\* code this pgm seperately using junit test case \*/ import static org.junit.Assert.\*;

import org.junit.Test; public class triangle\_test1 {

@Test

public void test1()

{

TRIANGLE t=new TRIANGLE();

assertEquals(t.check(100,100,1),"Isosceles");

}

@Test

public void test2()

{

TRIANGLE t=new TRIANGLE();

assertEquals(t.check(100,100,2),"Isosceles");

}

@Test

public void test3()

{

TRIANGLE t=new TRIANGLE();

assertEquals(t.check(100,100,100),"Equilateral");

}

@Test

public void test4()

{

TRIANGLE t=new TRIANGLE();

assertEquals(t.check(100,100,199),"Isosceles");

}

@Test

public void test5()

{

TRIANGLE t=new TRIANGLE();

assertEquals(t.check(100,100,200),"Not a Triangle");

}

}

**TEST CASES**

**Example:**

**TEST CASE 1:** Represents not a triangle

|  |  |
| --- | --- |
| **Project Information** | **Test Information** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Project Name: | Triangle | | Test Name: | | test1 | | |
| Project ID: | 1 | | Original Author: | | XYZ | | |
| Test Objective: | To validate the outcome of the program for not a triangle inputs. | | | | | | |
| **Step No.** | **Test Case Description** | **Test Data** | | **Expected Result** | | **Status (Pass/Fail)** | **Remark s** |
| 1 | Enter 3 sides of the triangle to form not a  triangle | 1,2,3 | | Not a Triangle | | pass |  |
| 2 | Enter 3 sides of the triangle to form not a triangle | 5,5,10 | | Not a Triangle | | pass |  |
| 3 | Enter 3 sides of the triangle to form not a triangle | 5,1,9 | | Not a Triangle | | pass |  |
| 4 | Enter 3 sides of the triangle to form not a triangle | 100,50,150 | | Not a Triangle | | pass |  |
| 5 | Enter 3 sides of the triangle to form not a triangle | 100,200,99 | | Not a Triangle | | pass |  |

**TEST CASE 2:** Represents a valid scalene triangle

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | **Test Information** | | | | |
| Project Name: | Triangle | | Test Name: | | test2 | | |
| Project ID: | 1 | | Original Author: | | XYZ | | |
| Test Objective: | To validate the outcome of the program for scalene triangle inputs | | | | | | |
| **Step No.** | **Test Case Description** | **Test Data** | | **Expected Result** | | **Status (Pass/Fail)** | **Remark s** |
| 1 | Enter 3 sides of the triangle to form a scalene  triangle | 3,4,5 | | Scalene Triangle | | pass |  |
| 2 | Enter 3 sides of the triangle to form a scalene triangle | 6,8,10 | | Scalene Triangle | | pass |  |
| 3 | Enter 3 sides of the triangle to form a scalene triangle | 8,15,17 | | Scalene Triangle | | pass |  |
| 4 | Enter 3 sides of the triangle to form a scelene triangle | 15,20,25 | | Scalene Triangle | | pass |  |
| 5 | Enter 3 sides of the triangle to form a scelene triangle | 9,40,41 | | Scalene Triangle | | pass |  |

**TEST CASE 3:** Represents a valid equilateral triangle

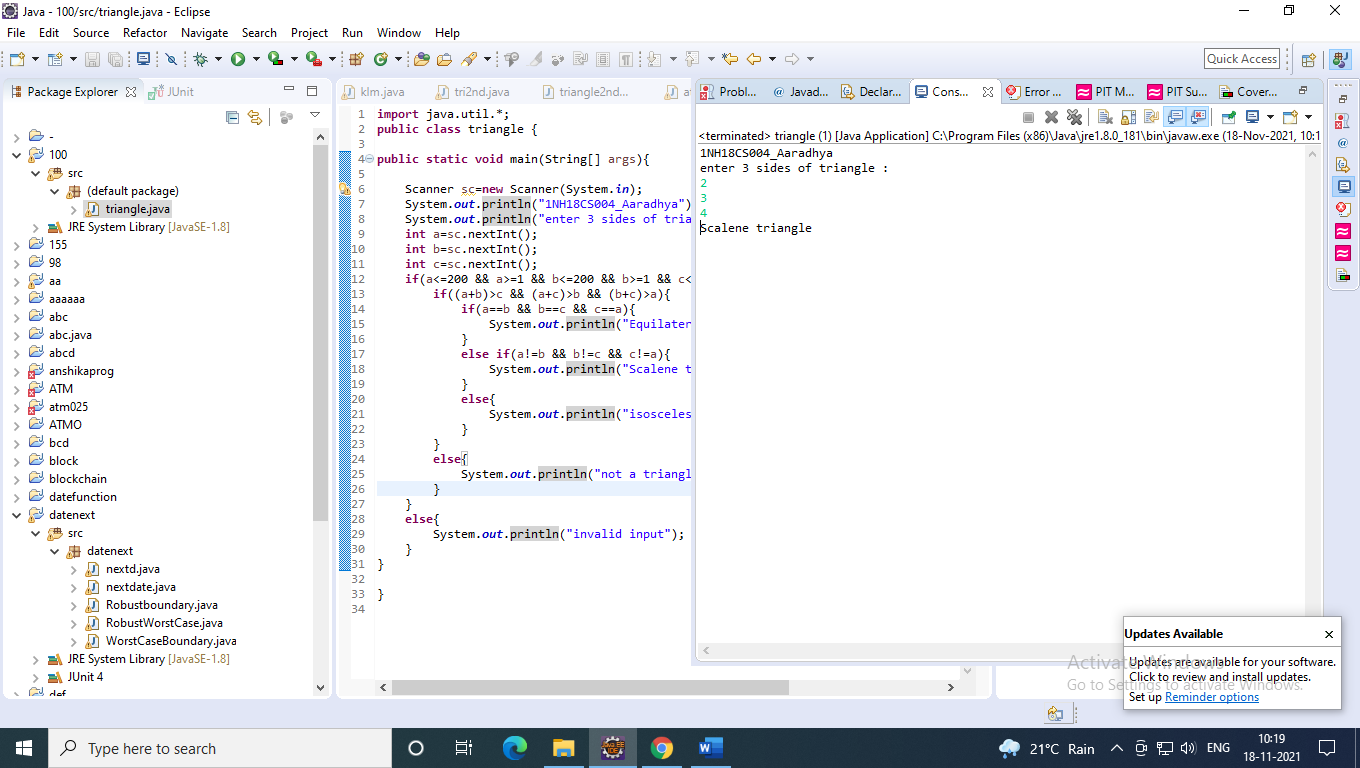
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | **Test Information** | | | | |
| Project Name: | Triangle | | Test Name: | | test3 | | |
| Project ID: | 1 | | Original Author: | | XYZ | | |
| Test Objective: | To validate the output of the program for equilateral triangle input | | | | | | |
| **Step No.** | **Test Case Description** | **Test Data** | | **Expected Result** | | **Status (Pass/Fail)** | **Remark s** |
| 1 | Enter 3 sides of the triangle to form an equilateral triangle | 5,5,5 | | Equilateral Triangle | | pass |  |
| 2 | Enter 3 sides of the triangle to form an equilateral triangle | 8,8,8 | | Equilateral Triangle | | pass |  |
| 3 | Enter 3 sides of the triangle to form an equilateral triangle | 15,15,15 | | Equilateral Triangle | | pass |  |
| 4 | Enter 3 sides of the triangle to form an equilateral triangle | 20,20,20 | | Equilateral Triangle | | pass |  |
| 5 | Enter 3 sides of the triangle to form an equilateral triangle | 10,10,10 | | Equilateral Triangle | | pass |  |

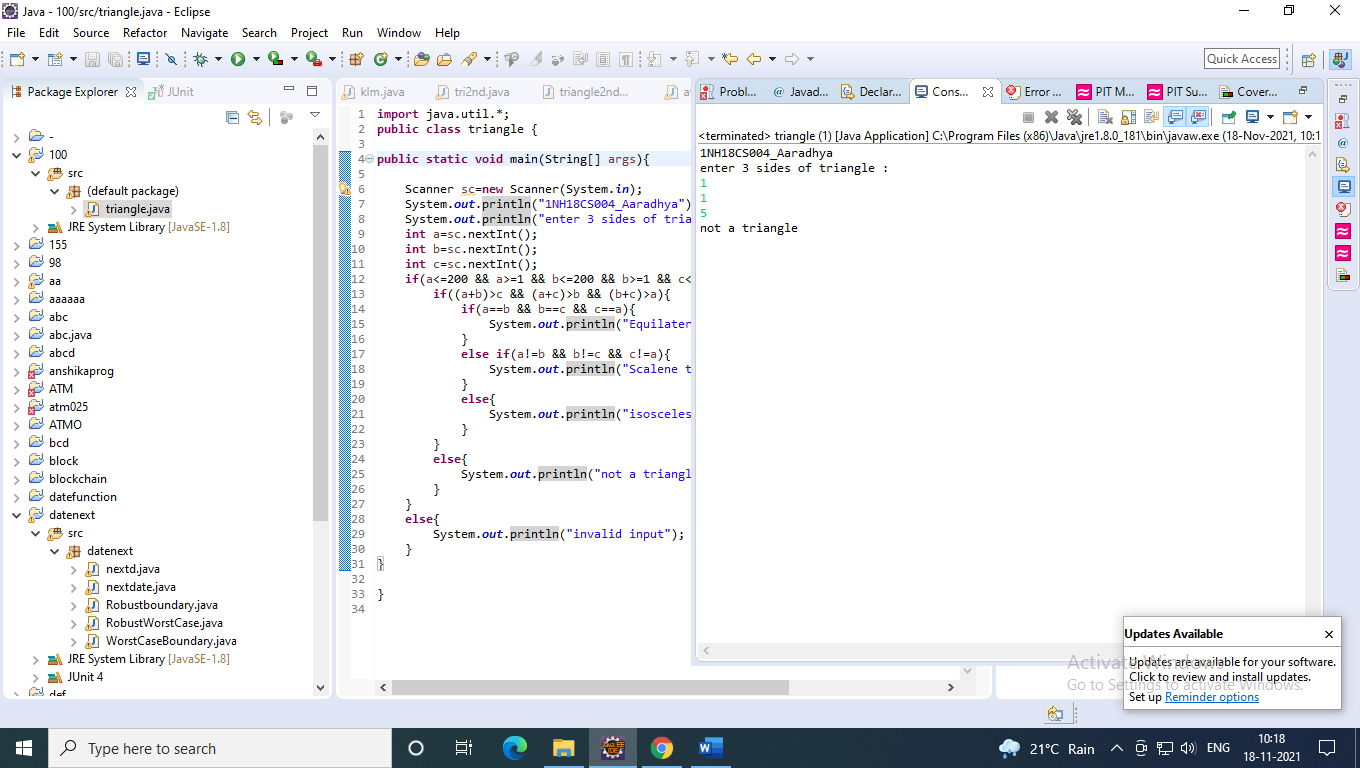
**TEST CASE 4:** Represents a valid isosceles triangle

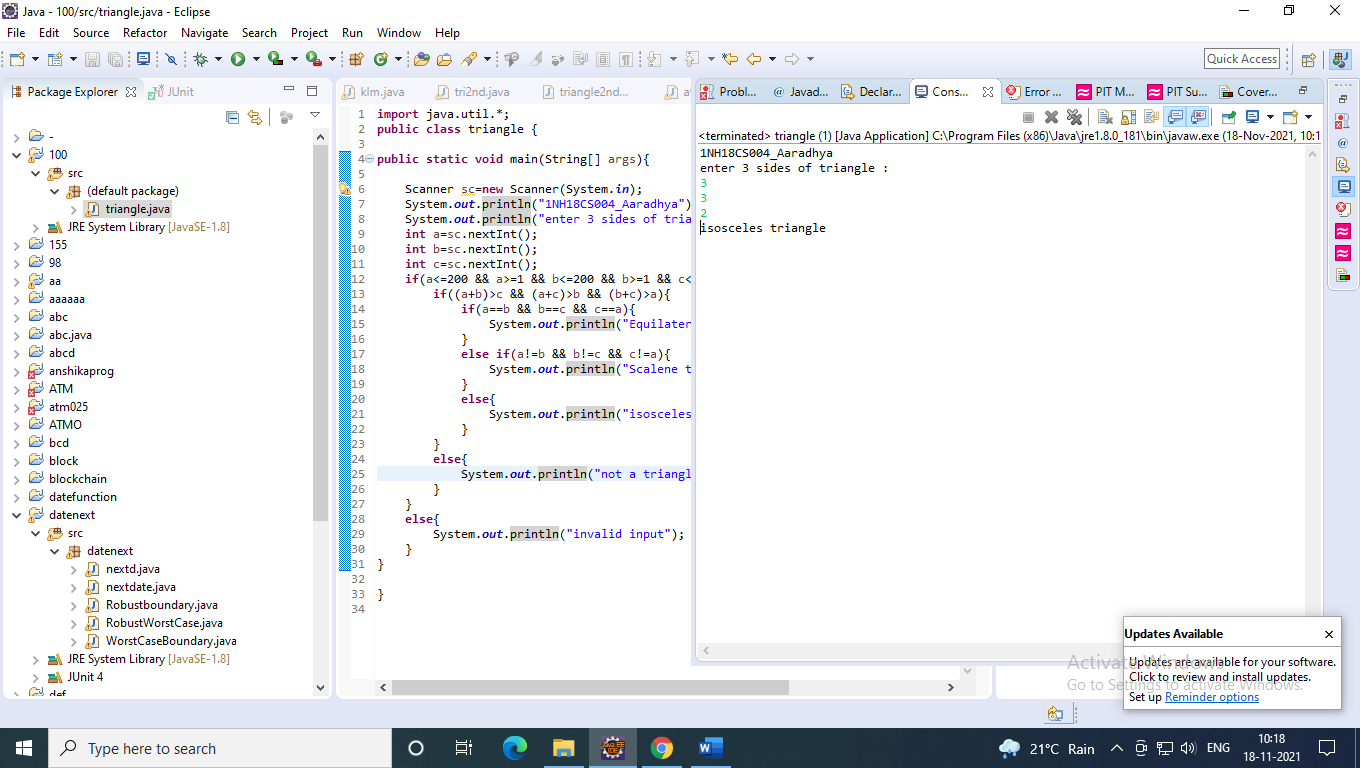
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | **Test Information** | | | | |
| Project Name: | Triangle | | Test Name: | | test4 | | |
| Project ID: | 1 | | Original Author: | | XYZ | | |
| Test Objective: | To validate the output of the program for equilateral triangle input | | | | | | |
| **Step No.** | **Test Case Description** | **Test Data** | | **Expected Result** | | **Status (Pass/Fail)** | **Remark s** |
| 1 | Enter 3 sides of the triangle to form an isosceles  triangle | 6,6,10 | | Isosceles Triangle | | pass |  |
| 2 | Enter 3 sides of the triangle to form an isosceles triangle | 5,3,3 | | Isosceles Triangle | | pass |  |
| 3 | Enter 3 sides of the triangle to form an isosceles triangle | 15,10,10 | | Isosceles Triangle | | pass |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | Enter 3 sides of the triangle to form an isosceles triangle | 2,2,3 | Isosceles Triangle | pass |  |
| 5 | Enter 3 sides of the triangle to form an isosceles triangle | 8,8,14 | Isosceles Triangle | pass |  |

**EXECUTION**







**RESULT & DISCUSSION**

Test Report:

1. Number of Test Cases Executed : 4
2. Number of Test Cases Passed 4
3. Number of Test Cases Failed 0

**Exp. No. : 3 Date :**

**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.

* 1. Normal Boundary Value Testing
  2. Robust Boundary Value Testing
  3. Worst-Case Boundary Value Testing
  4. Robust Worst-Case Boundary Value Testing

**IMPLEMENTATION**

import java.util.\*; public class Next {

public String nextd(int day,int month, int year) { if((month>12)||((year<1812)||(year>2020))||(day>31))

{

}

else

{

month%2==1))

return("Enter valid dates");

if((day==31 && month%2!=1 && month<8)||(day==31 && month>7 &&

{

return("Enter valid dates");

}

else

{

if((month%2==1)||((month>7)&&(month%2==0)))

{

if(day==31)

{

if(month==12){

day=1; month=1; year+=1;

}

else

}

else

{

day=1; month+=1;

}

{

}

}

else

{

day+=1;

if(month==2 && day==28)

{

if((year%4==0 && year%100!=0)||(year%400==0))

{

}

else

{

}

}

day+=1;

month+=1; day=1;

else if(day==30)

{

if(month==12){

day=1; month=1; year+=1;

}

else

{

}

}

}

}

}

else

{

day=1; month+=1;

}

day+=1;

return(day+"/"+month+"/"+year);

}

}

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

public class Normalbva { @Test

public void test1()

{

}

@Test Next d1 = new Next(); assertEquals(d1.nextd(12,3,1812),"13/3/1812");

public void test2()

{

Next d1 = new Next(); assertEquals(d1.nextd(30,3,1813),"31/3/1813");

}

@Test

public void test3()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,1912),"1/1/1913");

}

@Test

public void test4()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(12,3,2019),"13/3/2019");

public void test5()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(12,3,2020),"13/3/2020");

public void test6()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(15,1,2020),"16/1/2020");

public void test7()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(15,2,2020),"16/2/2020");

public void test8()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(15,11,2020),"16/11/2020");

public void test9()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(15,12,2020),"16/12/2020");

public void test10()

{}

@Test

public void test11()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(1,6,2020),"2/6/2020");

public void test12()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(2,6,2020),"3/6/2020");

public void test13()

{

Next d1 = new Next(); assertEquals(d1.nextd(15,6,2020),"16/6/2020");

}

@Test

public void test14()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(30,6,2020),"1/7/2020");

public void test15()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,3,2020),"1/4/2020");

}

}

\*Robust BVA

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

public class robustbva {

@Test

public void test()

{

Next d1 = new Next(); assertEquals(d1.nextd(25,3,2019),"26/3/2019");

}

@Test

public void test1()

{

Next d1 = new Next(); assertEquals(d1.nextd(12,3,1950),"13/3/1950");

Next d1 = new Next(); assertEquals(d1.nextd(15,6,2020),"16/6/2020");

}

@Test

public void test3()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,1915),"1/1/1916");

}

@Test

public void test6()

{

Next d1 = new Next(); assertEquals(d1.nextd(12,3,1915),"13/3/1915");

}

@Test

public void test4()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(32,3,1914),"Enter valid dates");

public void test5()

{

Next d1 = new Next(); assertEquals(d1.nextd(12,13,2021),"Enter valid dates");

}

@Test

public void test7()

{

Next d1 = new Next(); assertEquals(d1.nextd(12,3,2020),"13/3/2020");

}

}

\*Worst-case BVA

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

public class worstcase {

@Test

public void test()

{

Next d1 = new Next(); assertEquals(d1.nextd(25,3,2012),"26/3/2012");

}

@Test

public void test1()

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(12,3,1925),"13/3/1925");

public void test2()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(30,3,1950),"31/3/1950");

public void test3()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2010),"1/1/2011");

}

public void test4()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2010),"1/1/2010");

}

public void test5()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2010),"1/1/2010");

}

@Test

public void test6()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(12,3,1915),"13/3/1915");

public void test7()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(12,3,1920),"13/3/1920");

public void test8()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2009),"1/1/2010");

public void test9()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2000),"1/1/2001");

}

}

\*Robust worst-case BVA

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

public class robustworstcase {

@Test

public void test()

{

Next d1 = new Next(); assertEquals(d1.nextd(25,3,2012),"26/3/2012");

}

@Test

public void test1()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(12,3,1925),"13/3/1925");

public void test2()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(30,3,1950),"31/3/1950");

public void test3()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2010),"1/1/2011");

}

public void test4()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2010),"1/1/2010");

}

public void test5()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2010),"1/1/2010");

}

@Test

public void assertEquals(d1.nextd(12,3,1915),"13/3/1915");

}

@Test

public void test7()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(12,3,1920),"13/3/1920");

public void test8()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2009),"1/1/2010");

public void test9()

{

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2000),"1/1/2001");

}

public void test12()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(31,12,2019),"1/1/2020");

public void test13()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(31,12,1999),"1/1/2000");

public void test10()

{

}

@Test

Next d1 = new Next(); assertEquals(d1.nextd(32,3,1914),"Enter valid dates");

public void test11()

{

Next d1 = new Next(); assertEquals(d1.nextd(12,13,2021),"Enter valid dates");

}

}

test6()

{

Next d1 = new Next();

**TEST CASES**

Test Case Name: Equivalence Class testing for next problem Test Data: Enter the 3 Integer Value (m, d and y)

Pre-condition: month {1<=m<=12}, day {1<=d<=31}, year {1812<=y<=2012} Test Objective: To find the next date to the given valid date.

1. **TEST CASES FOR NORMAL BOUNDARY VALUE TESTING**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | | | **Test Information** | | | | | |
| Project Name: | NEXTDATE | | | | Project Name: | | | | NEXT DATE | |
| Project ID: | NEXTDATE\_01 | | | | Original Author: | | | | xyz | |
| Test  Objective: | Find out the next date for a given date (Normal BVA) | | | | | | | | | |
| **Test Case ID** | **Test Case Description** | **Test Data** | | | | **Expected Result** | | **Status (Pass/**  **Fail)** | | **Remark** |
| **a** | **b** | **c** | |
| NXTDATE2b  \_n1 | Enter the nominal values for m& d, y changes | 6 | 15 | 181  2 | | Message must displayed “16.6.1812” | be as | Pass | |  |
| NXTDATE2b  \_n2 | Enter the nominal values for m& d, y changes | 6 | 15 | 181  3 | | Message must displayed “16.6.1813” | be as | Pass | |  |
| NXTDATE2b  \_n3 | Enter the nominal values for m& d, y changes | 6 | 15 | 191  2 | | Message must displayed “16.6.1912” | be as | Pass | |  |
| NXTDATE2b  \_n4 | Enter the nominal values for m& d, y changes | 6 | 15 | 201  1 | | Message must displayed “16.6.2011” | be as | Pass | |  |
| NXTDATE2b  \_n5 | Enter the nominal values for m& d, y changes | 6 | 15 | 201  2 | | Message must displayed “16.6.2012” | be as | Pass | |  |
| NXTDATE2b  \_n6 | Enter the nominal values for m& y, dchanges | 6 | 1 | 191  2 | | Message must displayed “2.6.1912” | be as | Pass | |  |
| NXTDATE2b  \_n7 | Enter the nominal values for m& y, dchanges | 6 | 2 | 191  2 | | Message must displayed “3.6.1912” | be as | Pass | |  |
| NXTDATE2b  \_n8 | Enter the nominal values for m& y, dchanges | 6 | 30 | 191  2 | | Message must displayed “1.7.1912” | be as | Pass | |  |
| NXTDATE2b  \_n9 | Enter the nominal values for m& y, dchanges | 6 | 31 | 191  2 | | Message must displayed “Invalid values” | be as | Pass | |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NXTDATE2b  \_n10 | Enter the nominal values for m changes, d,&y | 1 | 15 | 191  2 | Message must be displayed as “16.1.1912” | Pass |  |
| NXTDATE2b  \_n11 | Enter the nominal values for m changes, d,&y | 2 | 15 | 191  2 | Message must be displayed as “16.2.1912” | Pass |  |
| NXTDATE2b  \_n12 | Enter the nominal values for m changes, d,&y | 11 | 15 | 191  2 | Message must be displayed as “16.11.1912” | Pass |  |
| NXTDATE2b  \_n13 | Enter the nominal values for m changes, d,&y | 12 | 15 | 191  2 | Message must be displayed as “16.12.2012” | Pass |  |

1. **TEST CASES FOR ROBUST BOUNDARY VALUE TESTING**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | | | **Test Information** | | | | |
| Project Name: | NEXTDATE | | | | Project Name: | | | NEXT DATE | |
| Project ID: | NEXTDATE\_01 | | | | Original Author: | | | XYZ | |
| Test  Objective: | Find out the next date for a given date (Normal BVA) | | | | | | | | |
| **Test Case ID** | **Test Case Description** | **Test Data** | | | | **Expected Result** | **Status (Pass/**  **Fail)** | | **Remark** |
| **a** | **b** | **c** | |
| NXTDATE2b  \_n1 | Enter the nominal values for m& d, y changes | 6 | 15 | 181  2 | | Message must be displayed as “16.6.1812” | Pass | |  |
| NXTDATE2b  \_n2 | Enter the nominal values for m& d, y changes | 6 | 15 | 181  3 | | Message must be displayed as “16.6.1813” | Pass | |  |
| NXTDATE2b  \_n3 | Enter the nominal values for m& d, y changes | 6 | 15 | 191  2 | | Message must be displayed as “16.6.1912” | Pass | |  |
| NXTDATE2b  \_n4 | Enter the nominal values for m& d, y changes | 6 | 15 | 201  1 | | Message must be displayed as “16.6.2011” | Pass | |  |
| NXTDATE2b  \_n5 | Enter the nominal values for m& d, y changes | 6 | 15 | 201  2 | | Message must be displayed as “16.6.2012” | Pass | |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NXTDATE2b  \_n6 | Enter the nominal values for m& y, dchanges | 6 | 1 | 191  2 | Message must be displayed as “2.6.1912” | Pass |  |
| NXTDATE2b  \_n7 | Enter the nominal values for m& y, dchanges | 6 | 2 | 191  2 | Message must be displayed as “3.6.1912” | Pass |  |
| NXTDATE2b  \_n8 | Enter the nominal values for m& y, dchanges | 6 | 30 | 191  2 | Message must be displayed as “1.7.1912” | Pass |  |
| NXTDATE2b  \_n9 | Enter the nominal values for m& y, dchanges | 6 | 31 | 191  2 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n10 | Enter the nominal values for m changes, d,&y | 1 | 15 | 191  2 | Message must be displayed as “16.1.1912” | Pass |  |
| NXTDATE2b  \_n11 | Enter the nominal values for m changes, d,&y | 2 | 15 | 191  2 | Message must be displayed as “16.2.1912” | Pass |  |
| NXTDATE2b  \_n12 | Enter the nominal values for m changes, d,&y | 11 | 15 | 191  2 | Message must be displayed as “16.11.1912” | Pass |  |
| NXTDATE2b  \_n13 | Enter the nominal values for m changes, d,&y | 12 | 15 | 191  2 | Message must be displayed as “16.12.2012” | Pass |  |
| NXTDATE2b  \_n14 | Enter the nominal values for m& y, dchanges | 6 | 0 | 191  2 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n15 | Enter the nominal values for m& y, dchanges | 6 | 32 | 191  2 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n16 | Enter the nominal values for m changes, d,&y | 0 | 15 | 191  2 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n17 | Enter the nominal values for m changes, d,&y | 13 | 15 | 191  2 | Message must be displayed as “Invalid values” | Pass |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NXTDATE2b  \_n18 | Enter the nominal values for m changes, d,&y | 11 | 15 | 181  1 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n19 | Enter the nominal values for m changes, d,&y | 12 | 15 | 201  3 | Message must be displayed as “Invalid values” | Pass |  |

1. **TEST CASES FOR WORST-CASE BOUNDAR VALUE TESTING**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | | | **Test Information** | | | | | |
| Project Name: | NEXTDATE | | | | Project Name: | | | | NEXT DATE | |
| Project ID: | NEXTDATE\_01 | | | | Original Author: | | | | XYZ | |
| Test  Objective: | Find out the next date for a given date (Normal BVA) | | | | | | | | | |
| **Test Case ID** | **Test Case Description** | **Test Data** | | | | **Expected Result** | | **Status (Pass/**  **Fail)** | | **Remark** |
| **a** | **b** | **c** | |
| NXTDATE2b  \_n1 | Enter the nominal values for m& d, y changes | 6 | 15 | 181  2 | | Message must displayed “16.6.1812” | be as | Pass | |  |
| NXTDATE2b  \_n2 | Enter the nominal values for m& d, y changes | 6 | 15 | 181  3 | | Message must displayed “16.6.1813” | be as | Pass | |  |
| NXTDATE2b  \_n3 | Enter the nominal values for m& d, y changes | 6 | 15 | 191  2 | | Message must displayed “16.6.1912” | be as | Pass | |  |
| NXTDATE2b  \_n4 | Enter the nominal values for m& d, y changes | 6 | 15 | 201  1 | | Message must displayed “16.6.2011” | be as | Pass | |  |
| NXTDATE2b  \_n5 | Enter the nominal values for m& d, y changes | 6 | 15 | 201  2 | | Message must displayed “16.6.2012” | be as | Pass | |  |
| NXTDATE2b  \_n6 | Enter the nominal values for m y, dchanges | 6 | 1 | 181  2 | | Message must displayed “2.6.1812” | be as | Pass | |  |
| NXTDATE2b  \_n7 | Enter the nominal values for m y, dchanges | 6 | 1 | 181  3 | | Message must displayed “2.6.1813” | be as | Pass | |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NXTDATE2b  \_n8 | Enter the nominal values for m y, dchanges | 6 | 1 | 191  2 | Message must be displayed as “2.6.1912” | Pass |  |
| NXTDATE2b  \_n9 | Enter the nominal values for m y, dchanges | 6 | 1 | 201  1 | Message must be displayed as “2.6.2011” | Pass |  |
| NXTDATE2b  \_n10 | Enter the nominal values for m changes, d,&y | 6 | 1 | 201  2 | Message must be displayed as “2.6.2012” | Pass |  |
| NXTDATE2b  \_n11 | Enter the nominal values for m changes, d,&y | 6 | 2 | 181  2 | Message must be displayed as “3.6.1812” | Pass |  |
| NXTDATE2b  \_n12 | Enter the nominal values for m changes, d,&y | 6 | 2 | 181  3 | Message must be displayed as “3.6.1813” | Pass |  |
| NXTDATE2b  \_n13 | Enter the nominal values for m changes, d,&y | 6 | 2 | 191  2 | Message must be displayed as “3.6.1912” | Pass |  |
| NXTDATE2b  \_n14 | Enter the nominal values for m& y, dchanges | 6 | 2 | 201  1 | Message must be displayed as “3.6.2012” | Pass |  |
| NXTDATE2b  \_n15 | Enter the nominal values for m& y, dchanges | 6 | 2 | 201  2 | Message must be displayed as “3.6.2012” | Pass |  |
| NXTDATE2b  \_n16 | Enter the nominal values for m changes, d,&y | 6 | 30 | 181  2 | Message must be displayed as “1.7.1812” | Pass |  |
| NXTDATE2b  \_n17 | Enter the nominal values for m changes, d,&y | 6 | 30 | 181  13 | Message must be displayed as “1.7.1813” | Pass |  |
| NXTDATE2b  \_n18 | Enter the nominal values for m changes, d,&y | 6 | 30 | 191  2 | Message must be displayed as “1.7.1912” | Pass |  |
| NXTDATE2b  \_n19 | Enter the nominal values for m changes, d,&y | 6 | 30 | 201  1 | Message must be displayed as “1.7.2011” | Pass |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NXTDATE2b  \_n20 | Enter the nominal values for m changes, d,&y | 6 | 30 | 201  2 | Message must be displayed as “1.7.2012” | Pass |  |

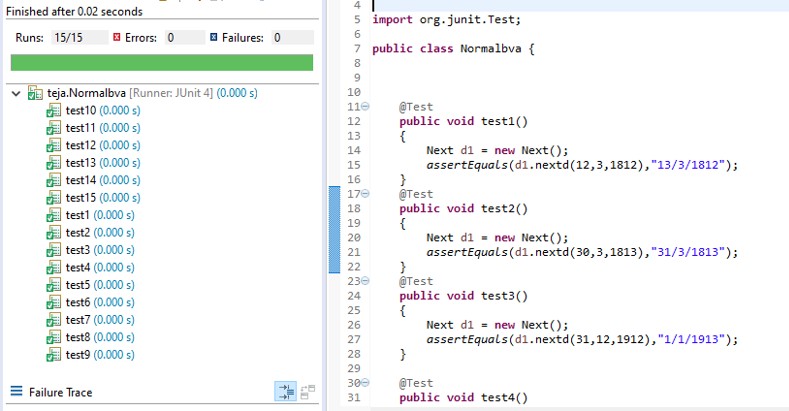
1. **TEST CASES FOR ROBUST WORST-CASE BOUNDARY VALUE TESTING**

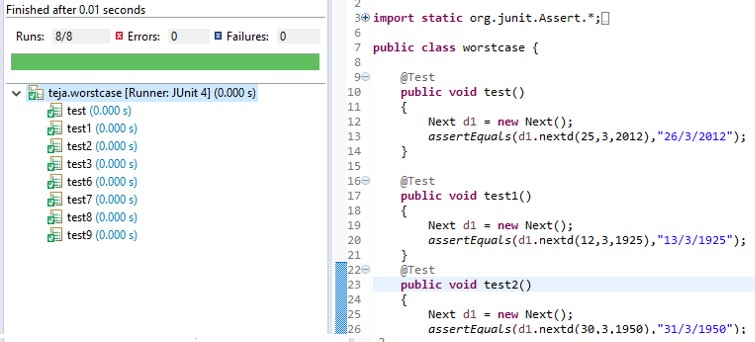
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | | | **Test Information** | | | | | |
| Project Name: | NEXTDATE | | | | Project Name: | | | | NEXT DATE | |
| Project ID: | NEXTDATE\_01 | | | | Original Author: | | | | XYZ | |
| Test  Objective: | Find out the next date for a given date (Normal BVA) | | | | | | | | | |
| **Test Case ID** | **Test Case Description** | **Test Data** | | | | **Expected Result** | | **Status**  **(Pass/ Fail)** | | **Remark** |
| **a** | **b** | **c** | |
| NXTDATE2b  \_n1 | Enter the nominal values for m& d, y changes | 6 | 15 | 181  2 | | Message must displayed “16.6.1812” | be as | Pass | |  |
| NXTDATE2b  \_n2 | Enter the nominal values for m& d, y changes | 6 | 15 | 181  3 | | Message must displayed “16.6.1813” | be as | Pass | |  |
| NXTDATE2b  \_n3 | Enter the nominal values for m& d, y changes | 6 | 15 | 191  2 | | Message must displayed “16.6.1912” | be as | Pass | |  |
| NXTDATE2b  \_n4 | Enter the nominal values for m& d, y changes | 6 | 15 | 201  1 | | Message must displayed “16.6.2011” | be as | Pass | |  |
| NXTDATE2b  \_n5 | Enter the nominal values for m& d, y changes | 6 | 15 | 201  2 | | Message must displayed “16.6.2012” | be as | Pass | |  |
| NXTDATE2b  \_n6 | Enter the nominal values for m y, dchanges | 6 | 1 | 181  2 | | Message must displayed “2.6.1812” | be as | Pass | |  |
| NXTDATE2b  \_n7 | Enter the nominal values for m y, dchanges | 6 | 1 | 181  3 | | Message must displayed “2.6.1813” | be as | Pass | |  |
| NXTDATE2b  \_n8 | Enter the nominal values for m y, dchanges | 6 | 1 | 191  2 | | Message must displayed “2.6.1912” | be as | Pass | |  |

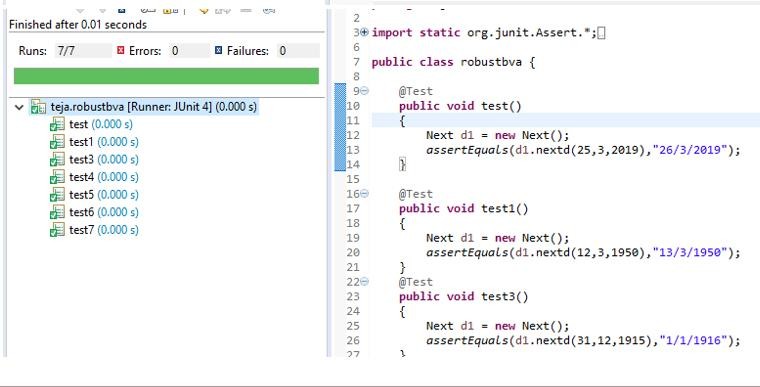
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NXTDATE2b  \_n9 | Enter the nominal values for m y, dchanges | 6 | 1 | 201  1 | Message must be displayed as “2.6.2011” | Pass |  |
| NXTDATE2b  \_n10 | Enter the nominal values for m changes, d,&y | 6 | 1 | 201  2 | Message must be displayed as “2.6.2012” | Pass |  |
| NXTDATE2b  \_n11 | Enter the nominal values for m changes, d,&y | 6 | 2 | 181  2 | Message must be displayed as “3.6.1812” | Pass |  |
| NXTDATE2b  \_n12 | Enter the nominal values for m changes, d,&y | 6 | 2 | 181  3 | Message must be displayed as “3.6.1813” | Pass |  |
| NXTDATE2b  \_n13 | Enter the nominal values for m changes, d,&y | 6 | 2 | 191  2 | Message must be displayed as “3.6.1912” | Pass |  |
| NXTDATE2b  \_n14 | Enter the nominal values for m y, dchanges | 6 | 15 | 181  1 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n15 | Enter the nominal values for m y, dchanges | 6 | 15 | 201  3 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n16 | Enter the nominal values for m y, dchanges | 6 | 1 | 181  1 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n17 | Enter the nominal values for m changes, d,&y | 6 | 1 | 201  3 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n18 | Enter the nominal values for m changes, d,&y | 6 | 2 | 181  1 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n19 | Enter the nominal values for m changes, d,&y | 6 | 2 | 201  3 | Message must be displayed as “Invalid values” | Pass |  |
| NXTDATE2b  \_n20 | Enter the nominal values for m changes, d,&y | 6 | 29 | 181  1 | Message must be displayed as “Invalid values” | Pass |  |

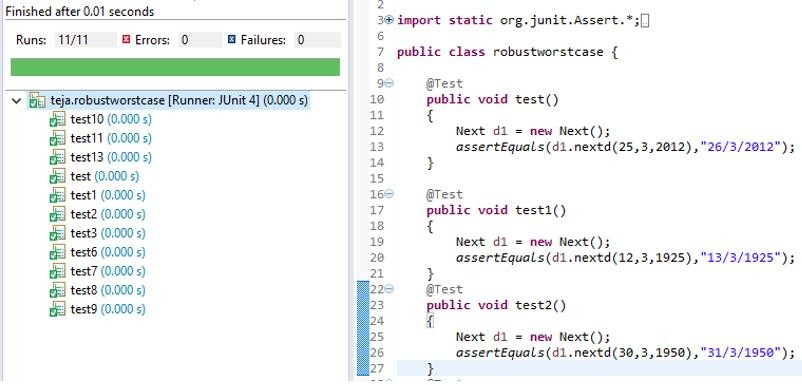
**EXECUTION**

**.**









**RESULT & DISCUSSION**

Test Report:

1. Number of Test Cases Executed : 4
2. Number of Test Cases Passed 4
3. Number of Test Cases Failed 0

**Exp. No. : 4 Date :**

**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.

* 1. Weak Normal Equivalence Class Testing
  2. Strong Normal Equivalence Class Testing
  3. Weak Robust Equivalence Class Testing
  4. Strong Robust Equivalence Class Testing

**IMPLEMENTATION**

**public class** nextdate

{

**public static** String next(**int** d, **int** m, **int** y, **int** cc)

{

**if**(d==cc)

{

d=1;

**if**(m==12)

{

}

**else**

{

}

}

**else**

{

}

d++;

y++; m=1;

m++;

**return**(String.*valueOf*(d)+"/"+String.*valueOf*(m)+"/"+String.*valueOf*(y));

}

**public** String nextday(**int** d, **int** m, **int** y)

{

**if**(d>=1 && d<=31 && m>=1 && m<=12 && y>=1812 && y<=2012)

{

**switch**(m)

{

**case** 1:

**case** 3:**return**(*next*(d,m,y,31));

**case** 5:**return**(*next*(d,m,y,31));

**case** 7:**return**(*next*(d,m,y,31));

**case** 8:**return**(*next*(d,m,y,31)); **case** 10:**return**(*next*(d,m,y,31)); **case** 12: **return**(*next*(d,m,y,31));

**case** 4: **return**(*next*(d,m,y,30));

**case** 6: **return**(*next*(d,m,y,30));

**case** 9: **return**(*next*(d,m,y,30));

**case** 11: **return**(*next*(d,m,y,30));

**default**: **return**(*next*(d,m,y,((y%4==0 && y%100!=0) || y%400==0)?29:28));

}

}

**return** "Invalid Values";

}

}

\***Junit code**

package nd2;

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

public class equind2pgm {

//weak and strong normal test case @Test

public void test\_1()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(15,6,1912),"16/6/1912");

}

@Test

public void test\_2()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(10,6,1912),"11/6/1912");

public void test\_3()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(10,6,1900),"11/6/1900");

}

@Test

public void test\_4()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(10,5,1912),"11/5/1912");

}

@Test

public void test\_5()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(20,10,2010),"21/10/2010");

}

//weak robust test cases

@Test

public void test3()

{

nextdate ob1=new nextdate();

assertEquals(ob1.nextday(-1,10,1912),"Invalid Values");

}

@Test

public void test31()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(12,7,1912),"13/7/1912");

public void test32()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(12,8,1912),"13/8/1912");

public void test33()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(12,4,1912),"13/4/1912");

public void test34()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(12,9,1912),"13/9/1912");

public void test35()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(12,1,1912),"13/1/1912");

public void test36()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(12,2,1912),"13/2/1912");

public void test37()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(12,3,1912),"13/3/1912");

public void test30()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(10,3,1912),"11/3/1912");

}

@Test

public void test4()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(15,13,1912),"Invalid Values");

public void test5()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(1,6,2200),"Invalid Values");

public void test6()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(32,6,1912),"Invalid Values");

public void test7()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(15,6,1811),"Invalid Values");

public void test8()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(15,6,2013),"Invalid Values");

}

//strong robust test cases @Test

public void test9()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(2,1,1912),"3/1/1912");

public void test10()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(-1,3,1900),"Invalid Values");

public void test11()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(15,0,1811),"Invalid Values");

public void test12()

{

}

@Test nextdate ob1=new nextdate(); assertEquals(ob1.nextday(33,12,1912),"Invalid Values");

public void test13()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(15,-1,-1),"Invalid Values");

public void test14()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(-1,6,-1),"Invalid Values");

public void test15()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(-1,-1,-1),"Invalid Values");

public void test16()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(31,12,2010),"1/1/2011");

public void test17()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(30,11,2010),"1/12/2010");

}

//////leap @Test

public void test18()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(3,2,2010),"4/2/2010");

}

@Test

public void test19()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(28,2,2010),"1/3/2010");

public void test20()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(20,2,2008),"21/2/2008");

}@Test

public void test21()

{

}

@Test

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(29,2,2000),"1/3/2000");

public void test22()

{

nextdate ob1=new nextdate(); assertEquals(ob1.nextday(28,2,1900),"1/3/1900");

}

}

**TEST CASES**

Test Case Name: Equivalence Class testing for next problem Test Data: Enter the 3 Integer Value (m, d and y)

Pre-condition: month{1<=m<=12}, day{1<=d<=31}, year{1812<=y<=2012} Test Objective: To find the next date to the given valid date.

**I) TEST CASES FOR WEAK NORMAL EQUIVALENCE CLASS TESTING**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | | | **Test Information** | | | | |
| Project Name: | NEXTDATE | | | | Project Name: | | | NEXTDATE | |
| Project ID: | NEXTDATE\_01 | | | | Original Author: | | | XYZ | |
| Test  Objective: | Check if valid date input gives next date (Weak normal equivalence class testing) | | | | | | | | |
| **Test Case ID** | **Test Case Description** | **Test Data** | | | | **Expected Result** | **Status (Pass/**  **Fail)** | | **Remark** |
| **m** | **d** | **y** | |
| TEST2d\_wn1 | Enter the values for m, d, y arbitrarily chosen from equivalence class | 6 | 14 | 200  0 | | Message must be displayed as “15.6.2000” | Pass | |  |

1. **TEST CASES FOR STRONG NORMAL EQUIVALENCE CLASS TESTING**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | | | **Test Information** | | | | |
| Project Name: | NEXTDATE | | | | Project Name: | | | NEXTDATE | |
| Project ID: | NEXTDATE\_01 | | | | Original Author: | | | XYZ | |
| Test  Objective: | Check if valid date input gives next date (Strong normal equivalence class testing) | | | | | | | | |
| **Test Case ID** | **Test Case Description** | **Test Data** | | | | **Expected Result** | **Status (Pass/**  **Fail)** | | **Remark** |
| **m** | **d** | **y** | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TEST2d\_wn1 | Enter the values for m, d, y arbitrarily chosen from equivalence class | 6 | 19 | 200  0 | Message must be displayed as “20.6.2000” | Pass |  |
| TEST2d\_wn 2 | Enter the values for m, d, y arbitrarily chosen from equivalence class | 7 | 31 | 195  6 | Message must be displayed as “1.8.1957” | Pass |  |

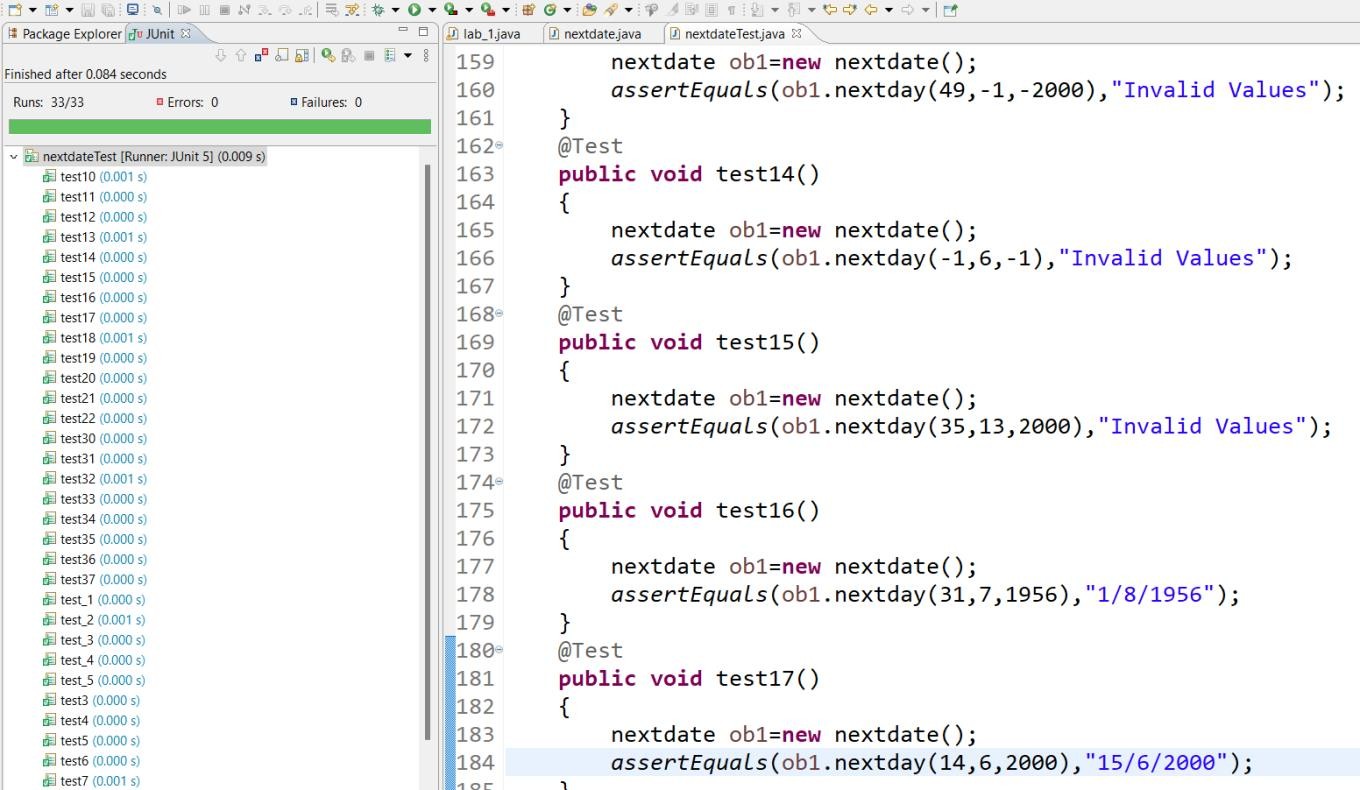
1. **TEST CASES FOR WEAK ROBUST EQUIVALENCE CLASS TESTING**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | | | **Test Information** | | | | |
| Project Name: | NEXTDATE | | | | Project Name: | | | NEXTDATE | |
| Project ID: | NEXTDATE\_01 | | | | Original Author: | | | XYZ | |
| Test  Objective: | Check if valid date input gives next date (Weak robust equivalence class testing) | | | | | | | | |
| **Test Case ID** | **Test Case Description** | **Test Data** | | | | **Expected Result** | **Status**  **(Pass/ Fail)** | | **Remark** |
| **m** | **d** | **y** | |
| TEST2d\_wn1 | Enter the values for m, d, y arbitrarily chosen from equivalence class | 31 | 14 | 200  0 | | Message must be displayed as not a valid date | Pass | |  |

1. **TEST CASES FOR STRONG ROBUST EQUIVALENCE CLASS TESTING**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | | | **Test Information** | | | | |
| Project Name: | NEXTDATE | | | | Project Name: | | | NEXTDATE | |
| Project ID: | NEXTDATE\_01 | | | | Original Author: | | | XYZ | |
| Test  Objective: | Check if valid date input gives next date (Weak normal equivalence class testing) | | | | | | | | |
| **Test Case ID** | **Test Case Description** | **Test Data** | | | | **Expected Result** | **Status (Pass/**  **Fail)** | | **Remark** |
| **m** | **d** | **y** | |
| TEST2d\_wn1 | Enter the values for m, d, y arbitrarily chosen from equivalence class | 13 | 3  5 | 2000 | | Message must be displayed as not a valid date | Pass | |  |
| TEST2d\_wn1 | Enter the values for m, d, y arbitrarily chosen from equivalence class | -1 | 4  9 | - 2000 | | Message must be displayed as not a valid date | Pass | |  |

**EXECUTION**



**RESULT & DISCUSSION**

Test Report:

1. Number of Test Cases Executed : 33
2. Number of Test Cases Passed 33
3. Number of Test Cases Failed : 0

**Exp. No. : 5 Date :**

**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**

**public class** triangle {

**public** String op(**int** a,**int** b,**int** c)

{

**if**(a>=1 && a<=200 && b>=1 && b<=200 && c>=1 && c<=200)

{

**if**(a+b>c && b+c>a && c+a>b)

{

**if**(a==b && b==c)

{

**return** "Equilateral Triangle";

}

**else if**(a==b||b==c)

{

}

**else**

{

}

**else**

{

}

**return** "Isosceles Triangle";

**return** "Scalen Triangle";

}

}

**else**

{

**return** "Not a Triangle";

**return** "Invalid";

}

}

}

\***Junit code**

package cs067;

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

import cs067.triangle;

public class triangleTest { @Test

public void test() {

triangle t1=new triangle(); assertEquals(t1.op(1, 2, 3),"Not a Triangle");

}

@Test

public void test12() {

triangle t1=new triangle(); assertEquals(t1.op(2, 1, 1),"Not a Triangle");

}

@Test

public void test13() {

triangle t1=new triangle(); assertEquals(t1.op(2, 4, 2),"Not a Triangle");

}

@Test

public void test1() {

triangle t1=new triangle();

assertEquals(t1.op(100, 100, 100),"Equilateral Triangle");

}

@Test

public void test2() {

triangle t1=new triangle(); assertEquals(t1.op(4, 5, 6),"Scalen Triangle");

}

@Test

public void test3() {

triangle t1=new triangle();

assertEquals(t1.op(4, 6, 6),"Isosceles Triangle");

}

@Test

public void test4() {

triangle t1=new triangle(); assertEquals(t1.op(201, 201, 201),"Invalid");

}

@Test

public void test5() {

triangle t1=new triangle();

assertEquals(t1.op(6, 6, 4),"Isosceles Triangle");

}

@Test

public void test6() {

triangle t1=new triangle(); assertEquals(t1.op(4, 201, 7),"Invalid");

}

@Test

public void test7() {

triangle t1=new triangle(); assertEquals(t1.op(4, 7, 201),"Invalid");

}

@Test

public void test8() {

triangle t1=new triangle(); assertEquals(t1.op(0, 7, 201),"Invalid");

}

@Test

public void test9() {

triangle t1=new triangle(); assertEquals(t1.op(7, 0, 201),"Invalid");

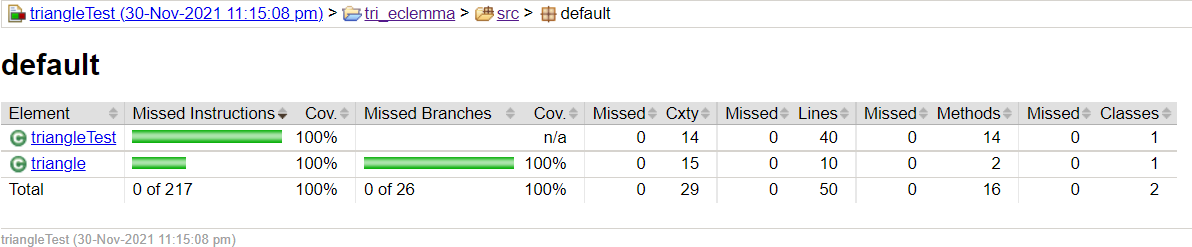
}

@Test

public void test11() {

triangle t1=new triangle(); assertEquals(t1.op(7, 9, 0),"Invalid");

**EXECUTION**



**TEST CASES FOR TRIANGLE PROGRAM**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Information** | | | | | **Test Information** | | | |
| Project Name: | TRIANGLE | | | | Project Name: | | TRIANGLE | |
| Project ID: | TRIANGLE\_01 | | | | Original Author: | |  | |
| Test Objective: | Check whether given value for a equilateral, isosceles, Scalene triangle or can't from a triangle | | | | | | | |
| **Test Case ID** | **Test Case Description** | **Test Data** | | | | **Expected Result** | **Status (Pass/ Fail)** | **Remark** |
| **a** | **b** | **c** | |
| TEST2c\_1 | Enter the values for a, b, c arbitrarily chosen from equivalenceclass | 5 | 5 | 5 | | Message must be displayed as “the triangle is Equilateral” | Pass |  |
| TEST2c\_2 | Enter the values for a, b, c arbitrarily chosen from equivalenceclass | 2 | 2 | 3 | | Message must be displayed as “the triangle is Isosceles” | Pass |  |
| TEST2c\_3 | Enter the values for a, b, c arbitrarily chosen from equivalenceclass | 3 | 4 | 5 | | Message must be displayed as “the triangle is Scalene” | Pass |  |
| TEST2c\_4 | Enter the values for a, b, c arbitrarily chosen from equivalenceclass | 4 | 1 | 3 | | Message must be displayed as “Not a Triangle” | Pass |  |

**RESULT & DISCUSSION**

**Thus, the above programs are written and executed using JUnit and ECLEMMA, and 100% coverage is achieved.**

**Exp. No. : 6 Date :**

**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**

**public class** nextdate {

**public static** String next(**int** d,**int** m,**int** y,**int** cc){

**if**(d==cc){

d=1;

**if**(m==12){

y++; m=1;

}

**else** {

}

}

**else**{

}

d++;

m++;

**return**(String.*valueOf*(d)+"/"+String.*valueOf*(m)+"/"+String.*valueOf*(y));

}

**public** String nextday(**int** d,**int** m,**int** y){

**if**(d>=1 && d<=31 && m>=1 && m<=12 && y>=1812 && y<=2012){

**switch**(m){ **case** 1:

**case** 3:

**case** 5:

**case** 8:

**case** 10:

**case** 12:**return**(*next*(d,m,y,31));

**case** 4:

**case** 6:

**case** 9:

**case** 11:**return**(*next*(d,m,y,30));

**default**:**return**(*next*(d,m,y,((y%4==0 && y%100!=0) || y%400==0)?29:28));

}

}

**return** "Invalid inputs";

}

}

\***Junit Code**

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

public class test {

//weak and strong normal test cases @Test

public void test1()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(15,3,1912),"16/3/1912");

public void test2()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(15,4,1912),"16/4/1912");

public void test3()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(16,4,1912),"17/4/1912");

public void test4()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(15,3,1912),"16/3/1912");

public void test5()

{

nextdate d1 = new nextdate(); assertEquals(d1.nextday(10,11,1920),"11/11/1920");

}

@Test

public void test6()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(13,15,1912),"Invalid inputs");

public void test7()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(32,1,1813),"Invalid inputs");

public void test8()

{

nextdate d1 = new nextdate(); assertEquals(d1.nextday(7,1,1810),"Invalid inputs");

}

@Test

public void test9()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(7,10,1912),"8/10/1912");

public void test10()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(6,11,2011),"7/11/2011");

public void test11()

{

nextdate d1 = new nextdate(); assertEquals(d1.nextday(18,8,2012),"19/8/2012");

}

@Test

public void test12()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(-1,15,1912),"Invalid inputs");

public void test13()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(6,-1,1810),"Invalid inputs");

public void test14()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(32,10,1811),"Invalid inputs");

public void test15()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(1,2,1912),"2/2/1912");

public void test16()

{

nextdate d1 = new nextdate(); assertEquals(d1.nextday(5,6,2000),"6/6/2000");}

@Test

public void test17()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(21,6,2000),"22/6/2000");

public void test18()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(-1,-1,-1),"Invalid inputs");

public void test19()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(31,1,2001),"1/2/2001");

public void test20()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(31,12,2001),"1/1/2002");

public void test21()

{

nextdate d1 = new nextdate(); assertEquals(d1.nextday(0,0,2013),"Invalid inputs");

}

@Test

public void test22()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(28,2,2011),"1/3/2011");

public void test23()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(28,13,2012),"Invalid inputs");

public void test24()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(28,2,2012),"29/2/2012");

public void test25()

{

nextdate d1 = new nextdate();assertEquals(d1.nextday(28,2,2000),"29/2/2000");

}

@Test

public void test26()

{

}

@Test

nextdate d1 = new nextdate(); assertEquals(d1.nextday(31,1,1812),"1/2/1812");

public void test27()

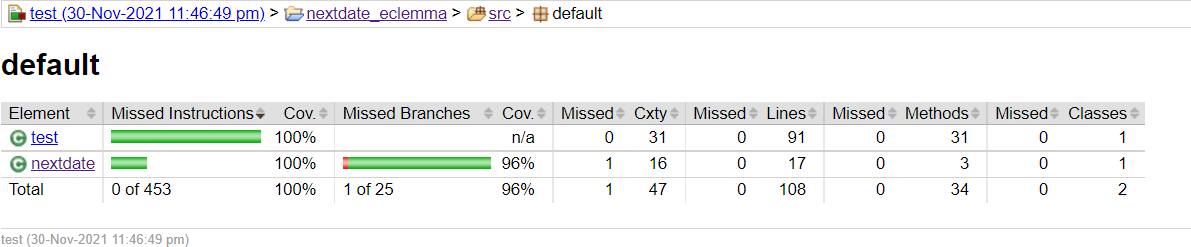
{

nextdate d1 = new nextdate(); assertEquals(d1.nextday(31,12,2012),"1/1/2013");

}

}

**EXECUTION**



**Thus, the above programs are written and executed using JUnit and ECLEMMA, and 100% coverage is achieved.**

**Exp. No. : 7 Date :**

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

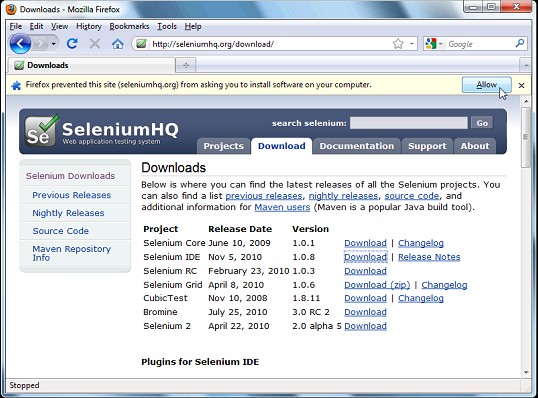
Designing Test Cases using Selenium IDE.

**IMPLEMENTATION**

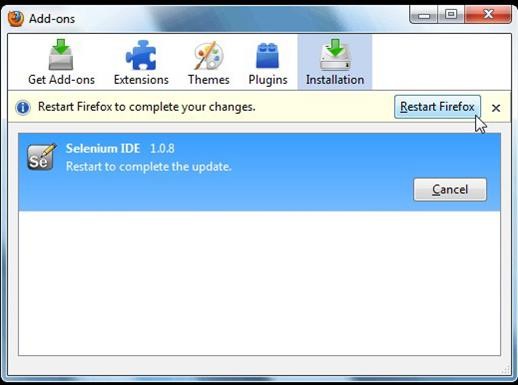
**Installing Selenium IDE**

Step 1: Using Firefox, first, download the IDE from the SeleniumHQ downloads page.

Step 2: Firefox will protect you from installing add-ons from unfamiliar locations, so you will need to click ‘Allow’ to proceed with the installation, as shown in the following screenshot.



Step 3: Select Install Now. The Firefox Add-ons window pops up, first showing a progress bar, and when the download is complete, displays the following.



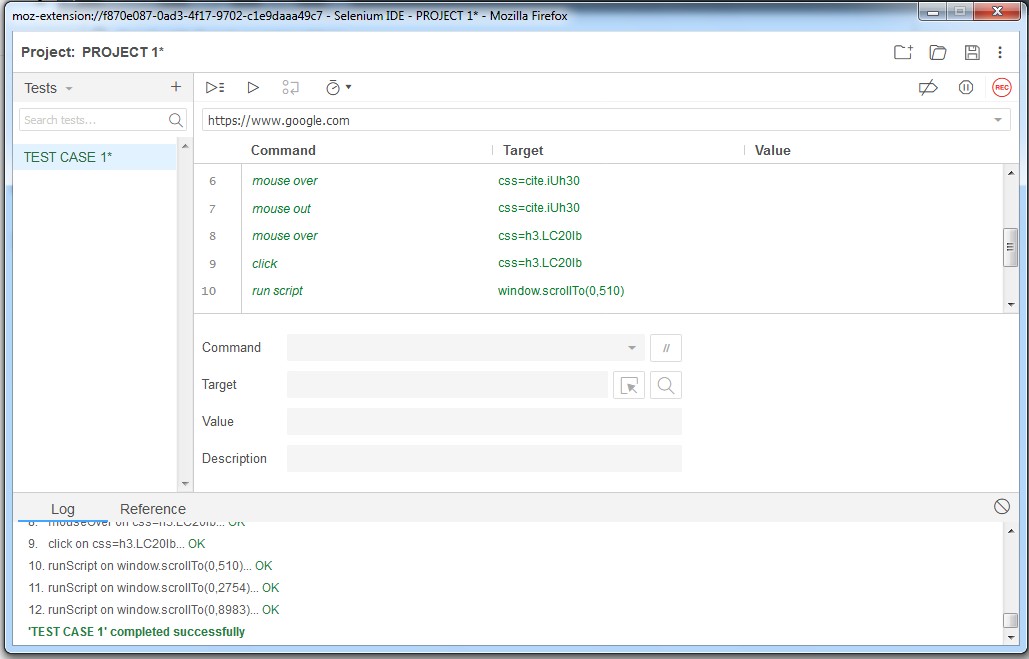
Step 4: Restart Firefox. After Firefox reboots you will find the Selenium-IDE listed under the Firefox Tools menu.

**TEST CASES**

TC’S #1: Manual Steps:

* Open (Example: Type www.google.com)
* Type “Software Te sting” in the Google Search Input Box
* Click outside on an empty spot
* Click Search Button
* Verify the Text Present as “Software Testing”
* Assert the Title as “Software Testing”
* Save the test case with .HTML Extension.

**EXECUTION**



**RESULT:**

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

**Exp. No. : 8 Date :**

**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 Script.

**IMPLEMENTATION**

**INSTALLATION**

Step 1: Download the Selenium Server Standalone as follows: https://[www.seleniumhq.org/download/](http://www.seleniumhq.org/download/)  Latest Release: ChromeDriver 2.43  Selenium Server Standalone.

Step 2: Download Selenium Web Driver from https://[www.seleniumhq.org/download/](http://www.seleniumhq.org/download/) 

Third Party Browser Drivers not developed by seleniumhq  Google Chrome Driver Step 3: Extract the jar file of Selenium Server Standalone and add it to the project (eclipse)

created as follows: Right Click on the Project  Build Path  Configure Build Path 

Library (tab)  Add External Jar  Add the Selenium Server Standalone jar.

**JAVA SCRIPT**

import org.openqa.selenium.By;

import org.openqa.selenium.chrome.ChromeDriver;

public class Demo1 {

public static void main(String[] args)

{

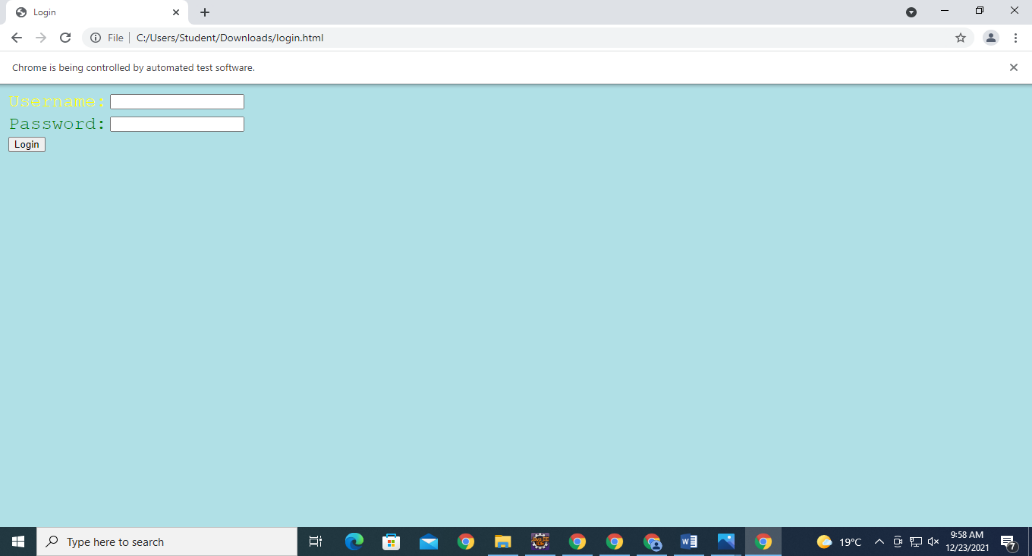
System.setProperty("webdriver.chrome.driver", C:\\Users\\User\\Downloads\\chromedriver.exe"); ChromeDriver driver = new ChromeDriver(); driver.get("[http://www.newtours.demoaut.com](http://www.newtours.demoaut.com/)"); driver.manage().window().maximize();

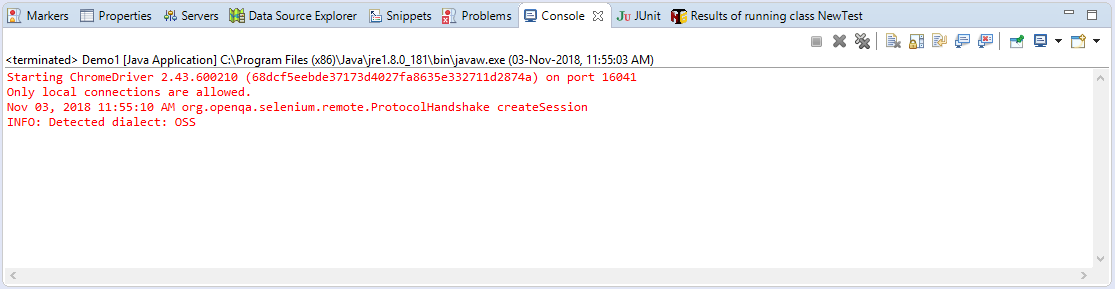
driver.findElement(By.name("userName")).sendKeys("mercury"); driver.findElement(By.name("password")).sendKeys("mercury"); driver.findElement(By.name("login")).click();

}

}

**EXECUTION**

****



**RESULT:**

Thus, the above program is written and executed using selenium web driver.

**Exp. No. : 9 Date :**

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

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

**IMPLEMENTATION**

package prog9;

import org.openqa.selenium.By;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import java.util.List;

public class Demo {

public static void main(String[] args){

System.setProperty("webdriver.chrome.driver","C:\\Users\\Student\\Desktop\\Jar\\chromedriver.exe");

ChromeDriver driver = new ChromeDriver();

driver.get("C:\\Users\\Student\\Desktop\\Jeev.html");

List<WebElement> links=driver.findElements(By.xpath("//img"));

int linkCount = links.size();

System.out.println("Aaradhya Priyedarshni\_1NH18CS004");

System.out.println("Total number of img count on webpage="+ linkCount);

List<WebElement> links1=driver.findElements(By.xpath("//ol"));

int linkCount1 = links1.size();

System.out.println("Total number of orderlist count on webpage="+ linkCount1);

List<WebElement> allElements= driver.findElements(By.xpath("//\*"));

int elementsCount=allElements.size();

System.out.println("Total number of count on webpage="+ elementsCount);

}

}

**.**

**.**

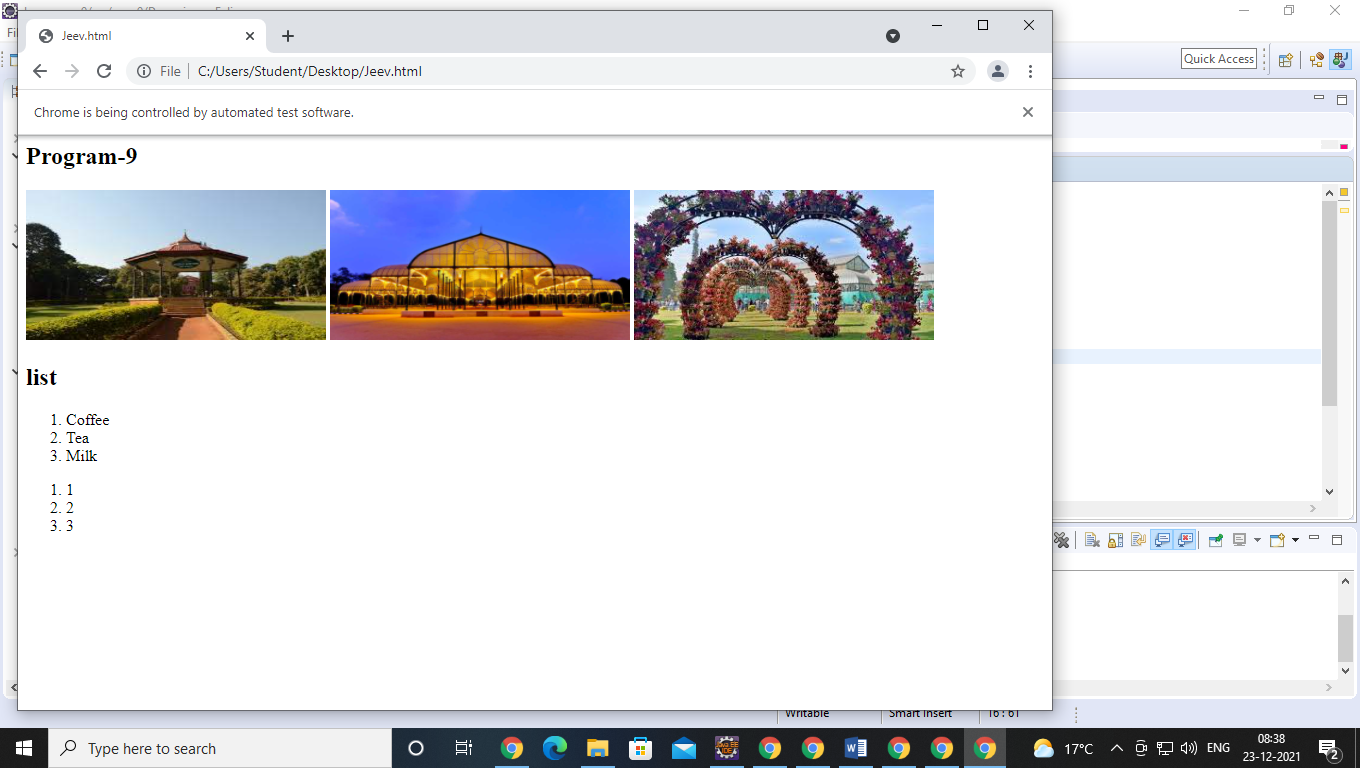
**.**

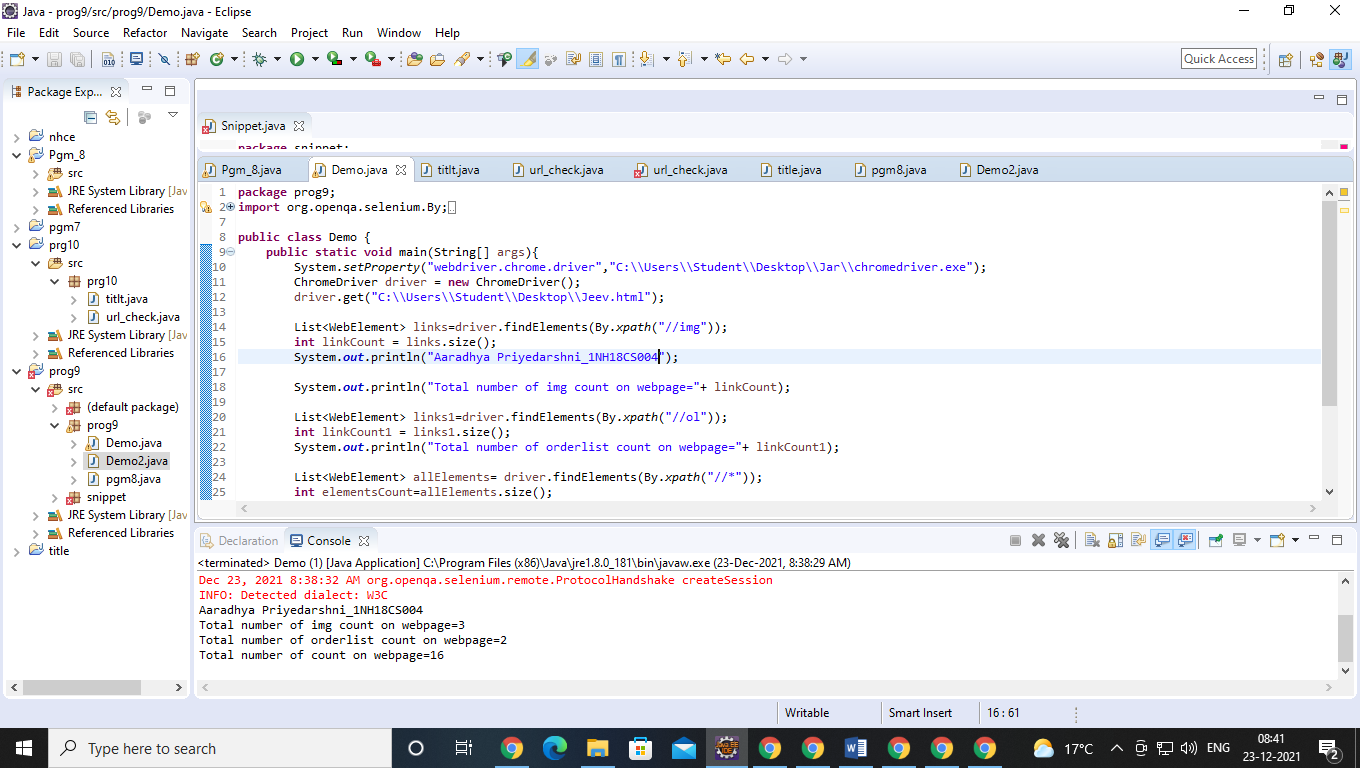
**.**

**.**

**.**

**RESULT**





**Exp. No. : 10 Date :**

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

Write a test program to demonstrate URL and title check point

**IMPLEMENTATION**

package prog9;

import org.openqa.selenium.chrome.ChromeDriver;

public class Demo2 {

public static void main(String[] args){

System.setProperty("webdriver.chrome.driver","C:\\Users\\Student\\Desktop\\Jar\\chromedriver.exe");

ChromeDriver driver = new ChromeDriver();

driver.get("http:\\google.com");

driver.manage().window().maximize();

String str=driver.getTitle();

String str1=driver.getCurrentUrl();

System.out.println("The Title of webpage is "+ str+" ");

if(str.equals("Google"))

System.out.println(true);

else

System.out.println(false);

System.out.println("The link of webpage is "+ str1+" ");

if(str1.equals("https://www.google.com/?gws\_rd=ssl"))

System.out.println(true);

else

System.out.println(false);

}

}

**.**

**.**

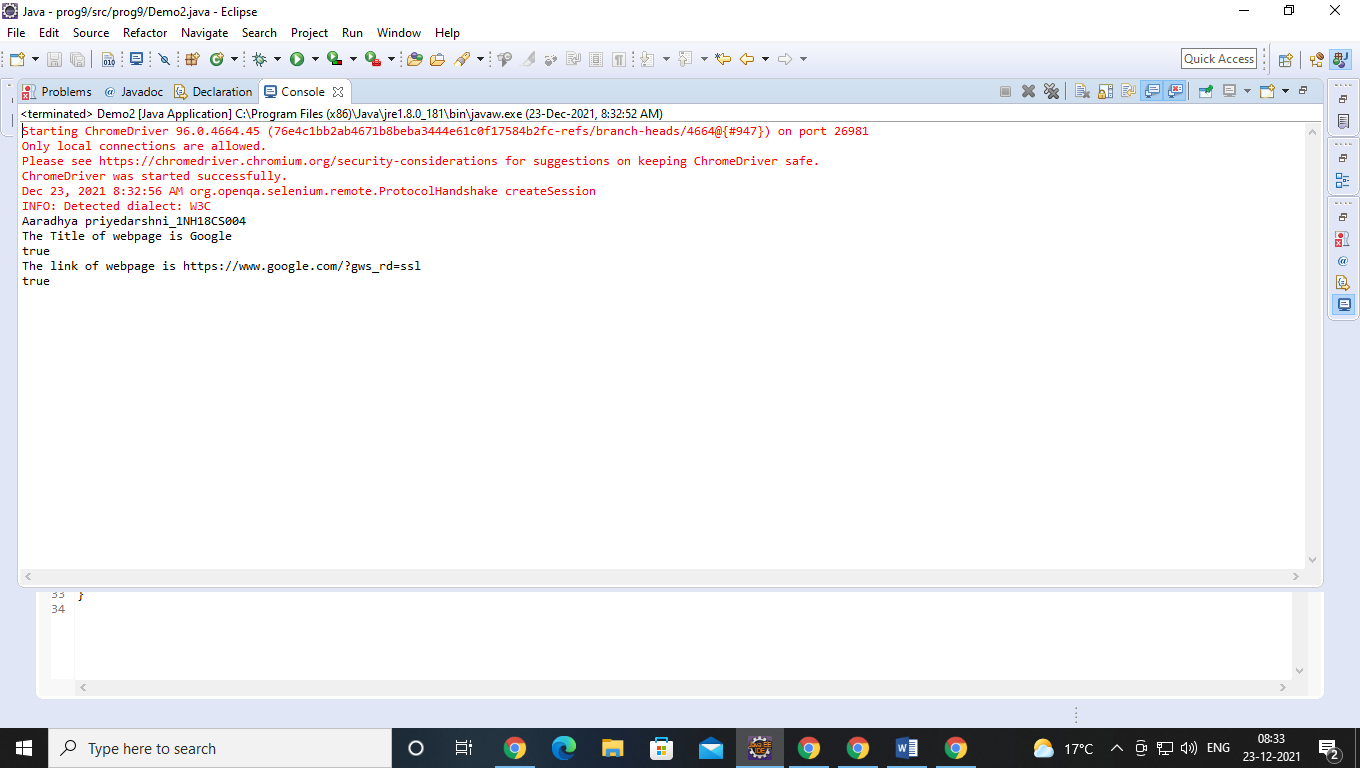
**.**

**.**

**.**

**.**

**RESULT**



**Exp. No. : 11 Date :**

**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**

package seledriver;

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 dropdown3 {

public static void main(String[] args) throws InterruptedException {

//pgm 11\_Write a test program to demonstrate selecting and deselecting option from multi select dropdown

//Creating instance of Chrome driver

System.setProperty("webdriver.chrome.driver", "D:\\Software\\Eclipse&JAR\\Jar\\chromedriver\_win32\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

// Navigate to the URL

driver.get("https://demoqa.com/select-menu");

//Maximizing window

driver.manage().window().maximize();

//Selecting the multi-select element by locating its id

Select select = new Select(driver.findElement(By.id("cars")));

//Get the list of all the options

System.out.println("The dropdown options are -");

List<WebElement> options = select.getOptions();

for(WebElement option: options)

System.out.println(option.getText());

//Using isMultiple() method to verify if the element is multi-select, if yes go onto next steps else eit

if(select.isMultiple()){

//Selecting option as 'Opel'-- ByIndex

System.out.println("Select option Opel by Index");

select.selectByIndex(2);

Thread.sleep(10000);

//Selecting the option as 'Saab'-- ByValue

System.out.println("Select option saab by Value");

select.selectByValue("saab");

Thread.sleep(10000);

// Selecting the option by text

System.out.println("Select option Audi by Text");

select.selectByVisibleText("Audi");

Thread.sleep(10000);

//Get the list of selected options

System.out.println("The selected values in the dropdown options are -");

List<WebElement> selectedOptions = select.getAllSelectedOptions();

for(WebElement selectedOption: selectedOptions)

System.out.println(selectedOption.getText());

// Deselect the value "Audi" by Index

System.out.println("DeSelect option Audi by Index");

select.deselectByIndex(3);

Thread.sleep(10000);

//Deselect the value "Opel" by visible text

System.out.println("Select option Opel by Text");

select.deselectByVisibleText("Opel");

Thread.sleep(10000);

//Validate that both the values are deselected

System.out.println("The selected values after deselect in the dropdown options are -");

List<WebElement> selectedOptionsAfterDeselect = select.getAllSelectedOptions();

for(WebElement selectedOptionAfterDeselect: selectedOptionsAfterDeselect)

System.out.println(selectedOptionAfterDeselect.getText());

//Step#8- Deselect all values

select.deselectAll();

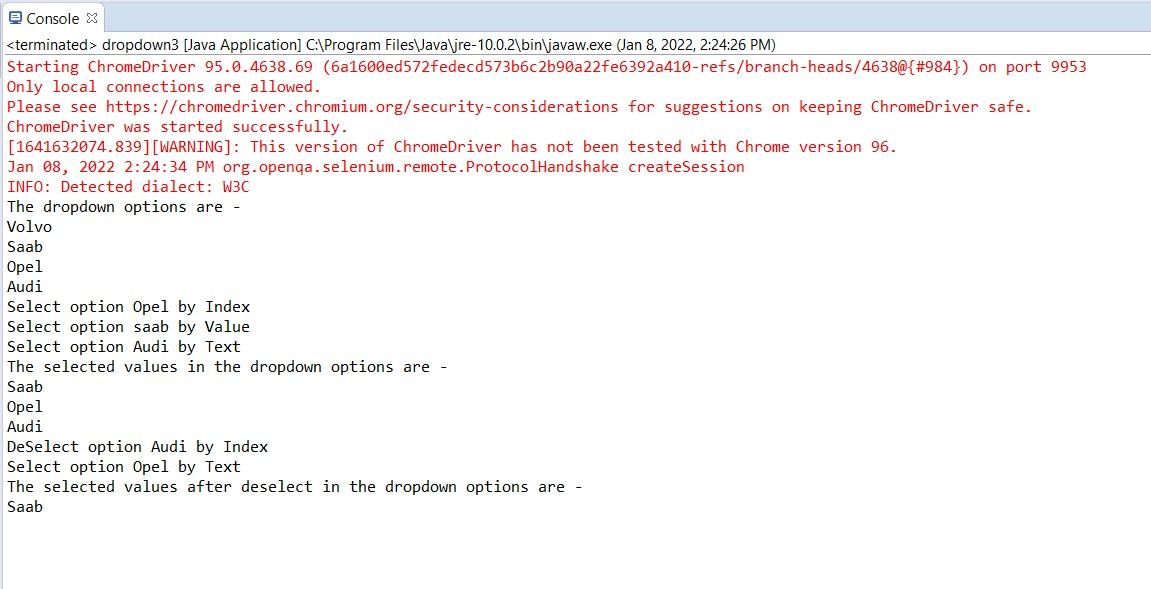
}

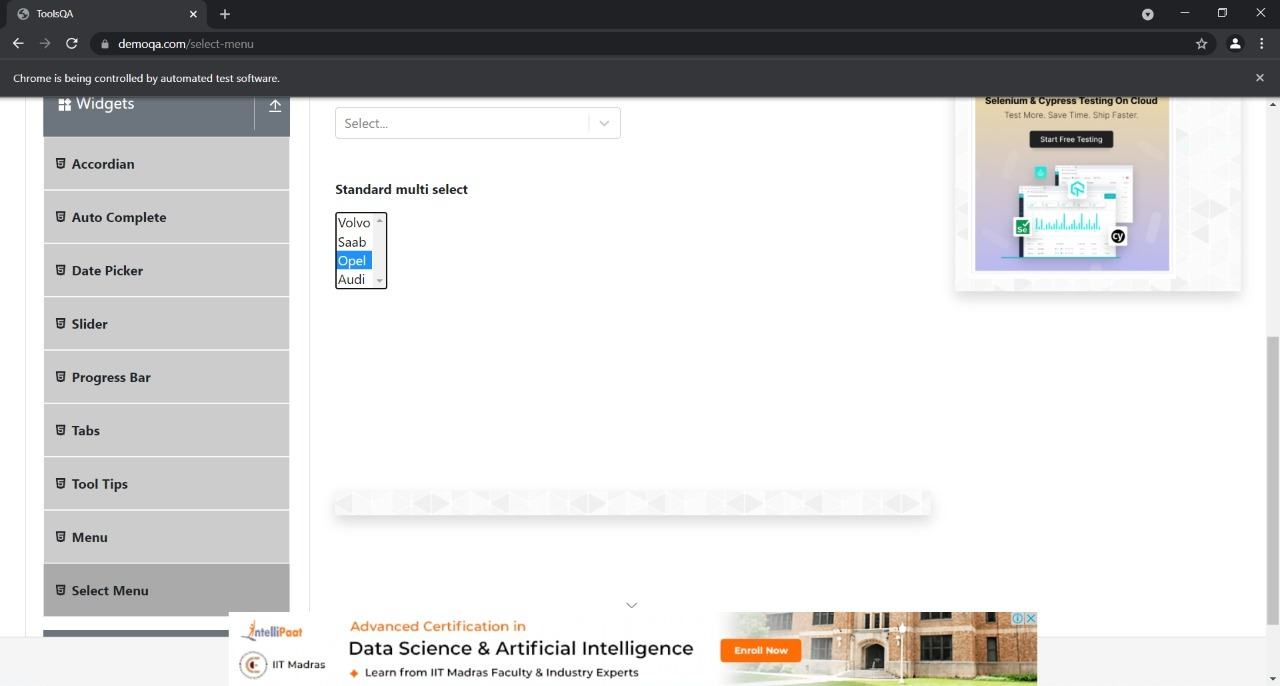
driver.quit();

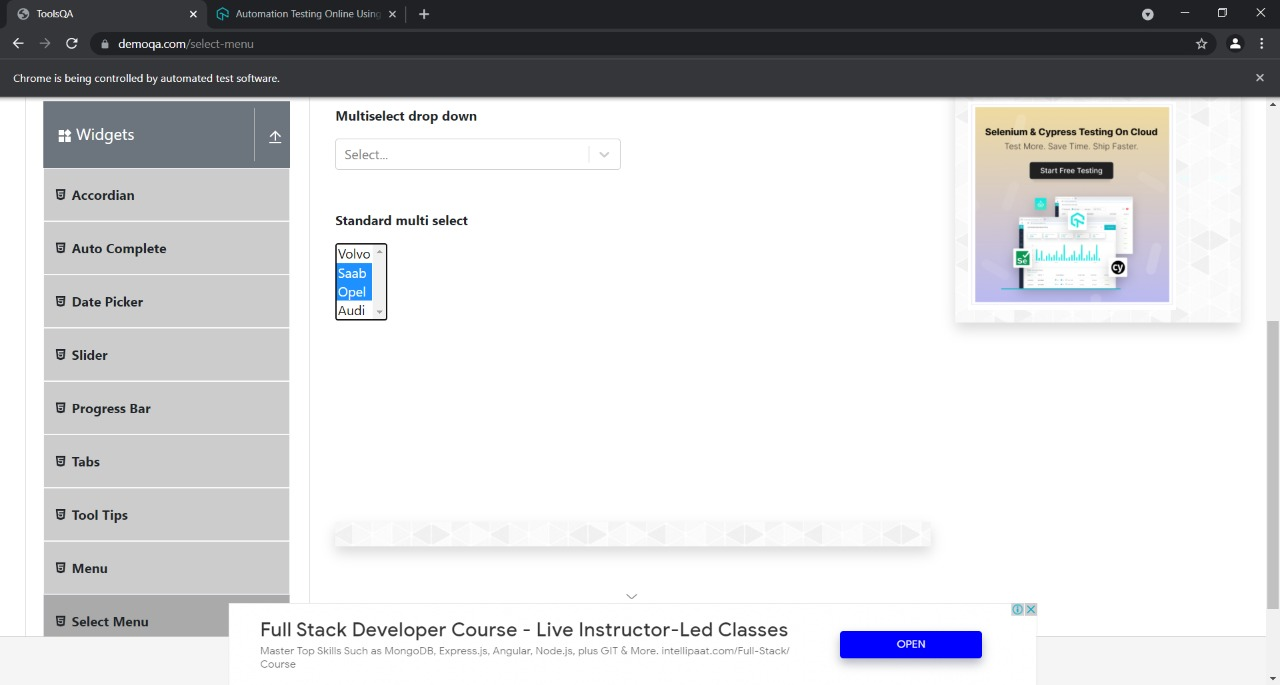
}

}

**RESULT**





****

**Exp. No. : 12 Date :**

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

Write a test program to demonstrate Synchronization

**IMPLEMENTATION (12A)**

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

//import org.testng.annotations.Test;

public class Prog12imp {

public static void main(String[] args) throws InterruptedException {

System.setProperty ("webdriver.chrome.driver","C:\\Users\\Jeevitha R\\Desktop\\Jar\\chromedriver.exe" );

ChromeDriver driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(10,TimeUnit.MINUTES) ;

String eTitle = "Demo Guru99 Page";

String aTitle = "" ;

// launch Chrome and redirect it to the Base URL

driver.get("http://demo.guru99.com/test/guru99home/" );

//Maximizes the browser window

driver.manage().window().maximize() ;

//get the actual value of the title

aTitle = driver.getTitle();

//compare the actual title with the expected title

if (aTitle.equals(eTitle))

{

System.out.println( "Test Passed") ;

}

else {

System.out.println( "Test Failed" );

}

//close browser

driver.close();

}

}

this is Implicit function-12

**12 B. explicit Function**

import java.util.List;

import java.util.concurrent.TimeUnit;

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.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

public class prog12 {

public static void main(String[] args) throws InterruptedException {

System.setProperty("webdriver.chrome.driver","C:\\Users\\Jeevitha R\\Desktop\\Jar\\chromedriver.exe" );

ChromeDriver driver = new ChromeDriver();

WebDriverWait wait=new WebDriverWait(driver, 10);

String eTitle = "Demo Guru99 Page";

String aTitle = "" ;

// launch Chrome and redirect it to the Base URL

driver.get("http://demo.guru99.com/test/guru99home/" );

//Maximizes the browser window

driver.manage().window().maximize() ;

//get the actual value of the title

aTitle = driver.getTitle();

//compare the actual title with the expected title

if (aTitle.contentEquals(eTitle))

{

System.out.println( "Test Passed") ;

}

else {

System.out.println( "Test Failed" );

}

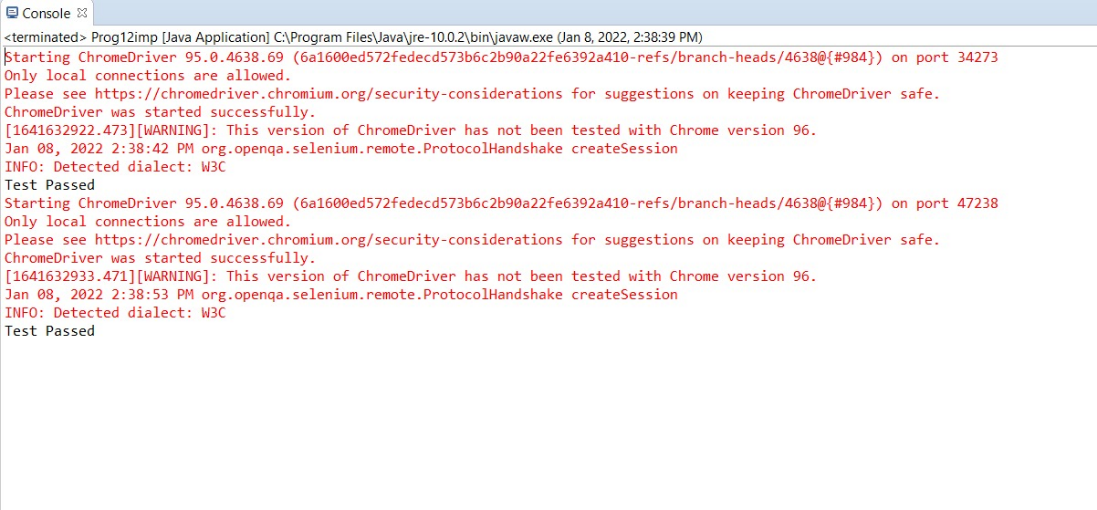
//driver.close();

WebElement guru99=wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath( "//a")));

guru99.click();

}

}

**RESULT**

