

1) A Program asks you to enter a password, and then ask you to enter it again. The program compares the two entries and either accepts the password (if they match) or rejects it (if they don't). You can enter letters or digits. How many valid entries could you left? (Please show and/or explain your calculations).

a) PasswordProgram.java

```
package Program;

public class PasswordVerification {

    public boolean acceptPassword(String password, String RePassword){

        if(password.equals(RePassword)){

            System.out.print("User Authenticated");

            return true;

        }else {

            System.out.print("User unauthorized");

            return false;

        }

    }

}
```

b)TestPassword.java

```
package TestSuite;

import Program.PasswordVerification;

import javafx.scene.layout.Priority;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class TestSuiteOne {

    PasswordVerification pwd;

    @BeforeClass

    public void init(){

        pwd = new PasswordVerification();

    }

    @Test(priority = 0)

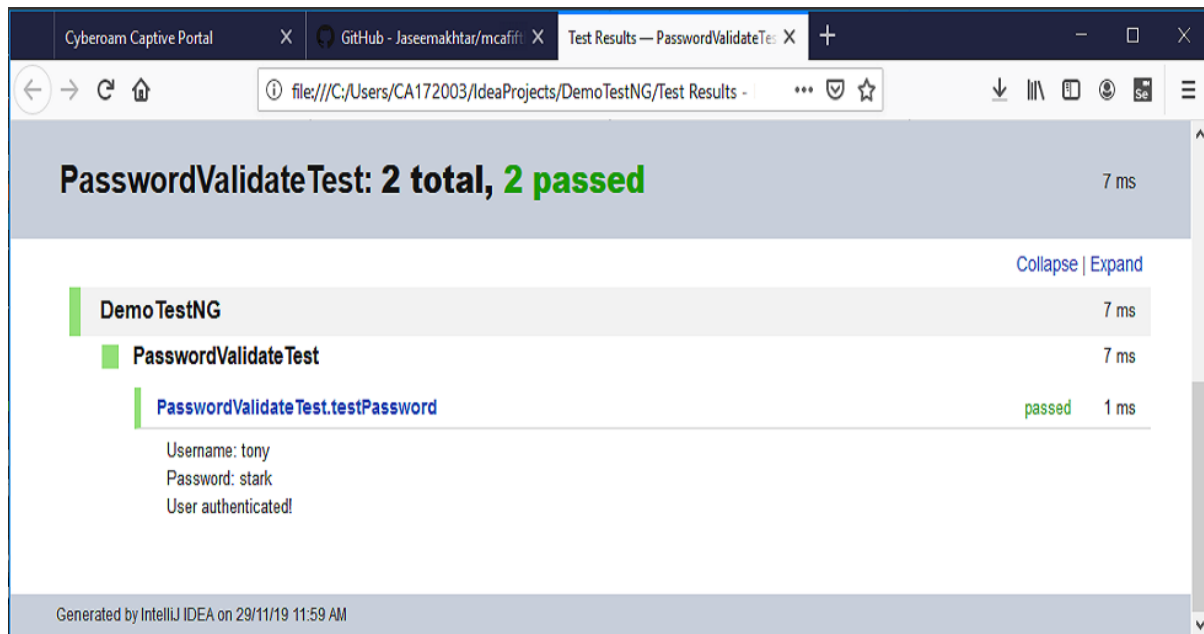
    public void testPassword(){

        Assert.assertTrue(pwd.acceptPassword("Abcd","Abcd"));

    }

}
```

Output



2) Take any system (e.g. ATM system) and study its system specification and report the various bugs.

- a. Machine is accepting ATM card.**
- b. Machine is rejecting expired card.**
- c. Successful entry of PIN Number.**
- d. Unsuccessful operation due to enter wrong PIN number 3 times.**
- e. Successful selection of language.**
- f. Successful selection of account type.**

a)ATM System

package logic;

public class ATMDB {

String cardNo;

int expiryDate;

String password;

public ATMDB(String cardNo, String password, int expiryDate){

 this.cardNo = cardNo;

 this.password = password;

 this.expiryDate = expiryDate;

}

}

public class ATMSystem {

private ArrayList<ATMDB> atmdbs;

private Calendar calendar;

public void start(){

 atmdbs = new ArrayList<>();

 calendar = Calendar.getInstance();

```
ATMDB user1 = new ATMDB("8505 4504 4520 3208", "1234", 2022);

ATMDB user2 = new ATMDB("4522 5604 9087 7541", "5678", 2018);

ATMDB user3 = new ATMDB("5106 8075 6508 8287", "4321", 2021);


atmdbs.add(user1);

atmdbs.add(user2);

atmdbs.add(user3);

println("Welcome to the ATM.");
}

public boolean insertCard(String cardNo, String password){

    for(int i = 0; i < atmdbs.size(); i++){

        ATMDB row = atmdbs.get(i);

        if(row.cardNo.equals(cardNo)){

            println("Card Inserted!");


            if(row.password.equals(password)){

                println("User authenticated.");

                int year = calendar.get(Calendar.YEAR);

                if(row.expiryDate < year){

                    println("Card has expired!");

                    return false;

                }

                return true;

            }else{

                println("Wrong password.");

                return false;

            }

        }

    }

}
```

```
        }  
    }  
  
    println("Card not registered.");  
    return false;  
}  
  
public String selectLanguage(int choice){  
    println("Select Language.");  
    println("1. English");  
    println("2. Hindi");  
    println("Enter your choice.");  
  
    switch (choice){  
        case 1:  
            println("English selected.");  
            return "english";  
        case 2:  
            println("Hindi selected.");  
            return "hindi";  
        default:  
            return "invalid";  
    }  
}  
  
public String selectAccountType(int choice){  
    println("Select Account type.");  
    println("1. Current");  
    println("2. Savings");
```

```
println("Enter your choice.");

switch (choice){

    case 1:

        return "current";

    case 2:

        return "savings";

    default:

        return "invalid";

}

}

public void println(Object o){

    System.out.println(String.valueOf(o));

}

public void end(){

    println("Thank you for using ATM. Bye!");

}

}
```

b)TestSuite.java

```
package testSuite;
```

```
import logic.ATMSSystem;
```

```
import org.testng.Assert;
```

```
import org.testng.annotations.*;
```

```
public class SuiteOne {
```

```
    private ATMSSystem atmSystem;
```

```
    @BeforeClass
```

```
    public void start(){
```

```
        atmSystem = new ATMSSystem();
```

```
        atmSystem.start();
```

```
    }
```

```
    @Test(priority = 0)
```

```
    public void acceptCard(){
```

```
        Assert.assertTrue(atmSystem.insertCard("8505 4504 4520 3208", "1234"));
```

```
    }
```

```
    @Test(priority = 1)
```

```
    public void checkExpiry(){
```

```
        Assert.assertTrue(!atmSystem.insertCard("4522 5604 9087 7541", "5678"));
```

```
    }
```



```
@Test(priority = 2)

public void testWrongPin(){

    boolean test = false;

    for(int i = 0; i < 3; i++){

        int randomPassword = (int) Math.floor(Math.random() * 10 + 1000);

        test = atmSystem.insertCard("5106 8075 6508 8287", "" + randomPassword);

    }

    Assert.assertTrue(test);

}
```

```
@Test(priority = 3)

public void selectLanguage(){

    Assert.assertEquals(atmSystem.selectLanguage(1), "english");

}
```

```
@Test(priority = 4)

public void selectAccountType(){

    Assert.assertEquals(atmSystem.selectAccountType(2), "savings");

}
```

```
@AfterClass

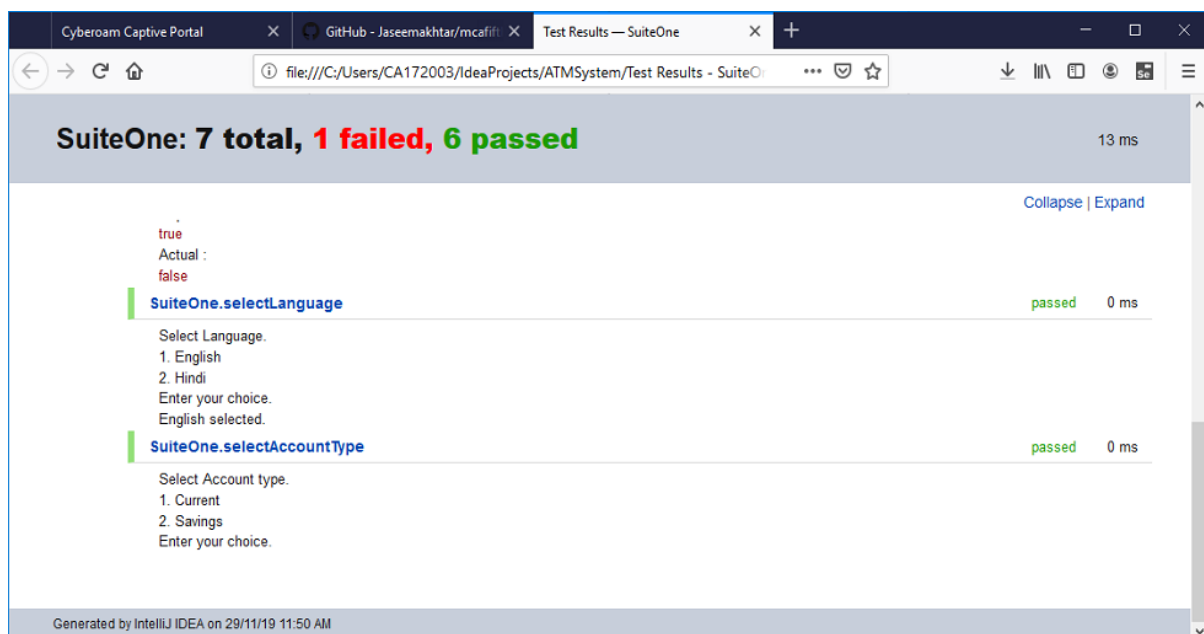
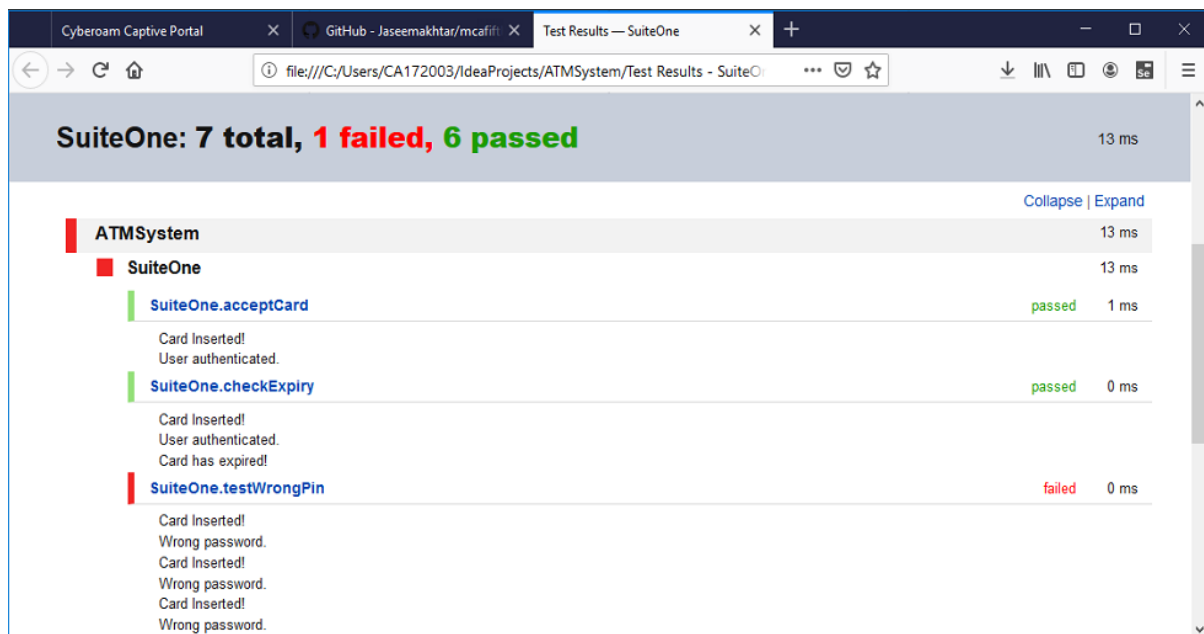
public void end(){

    atmSystem.end();

}

}
```

Output



3) 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 a triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Assume that the upper limit for the size of any side is 10. Derive test cases for your program based on boundary value analysis, execute the test cases and discuss the results.

a)TrangleProblem.java

```
package triangleProblem;
```

```
public class TriangleProblem {
```

```
    int side1, side2, side3;
```

```
    public String getTriangle(int a, int b, int c){
```

```
        side1 = a;
```

```
        side2 = b;
```

```
        side3 = c;
```

```
        String triangle;
```

```
        if(side1 < 10 || side2 < 10 || side3 < 10){
```

```
            triangle = "sides must be greater than 10";
```

```
        }else{
```

```
            if((side1 > side2 + side3) || (side2 > side1 + side3) || (side3 > side1 + side2)){
```

```
                triangle = "not a triangle";
```

```
            }else{
```

```
                if(side1 == side2 && side2 == side3)
```

```
                    triangle = "equilateral";
```

```
        else if(side1 == side2 || side2 == side3 || side3 == side1)
            triangle = "isosceles";
        else
            triangle = "scalene";
    }
}

System.out.println(triangle);

return triangle;
}
}
```

b)TestSuite.java

```
package tests;
```

```
import org.testng.Assert;
```

```
import org.testng.annotations.BeforeClass;
```

```
import org.testng.annotations.Test;
```

```
import triangleProblem.TriangleProblem;
```

```
public class TestSuiteOne {
```

```
    TriangleProblem t;
```

```
    @BeforeClass
```

```
    public void init(){
```

```
        t = new TriangleProblem();
```

```
    }
```

```
    @Test
```

```
    public void testCaseOne(){
```

```
        Assert.assertEquals(t.getTriangle(20, 30, 40), "scalene");
```

```
    }
```

```
}
```

Output



4) Design, develop, code and run the program in any suitable language to implement the Next Date function. Analyse it from the perspective of equivalence class value testing, derive different test cases, execute these test cases and discuss the test results.

a)NextData.java

```
package Program;
```

```
public class ProgramEleven {
```

```
    int mDay;
```

```
    int nextDay,nextMonth,nextYear;
```

```
    private boolean CheckYear(long year){
```

```
        if (year%400==0 || year%100==0 || year%4==0){
```

```
            return true;
```

```
        }else {
```

```
            return false;
```

```
        }
```

```
    }
```

```
    public String GenerateNextDate(int d,int m, int y) {
```

```
        System.out.println("Current Date : " + d + "-" + m + "-" + y);
```

```
        switch (m) {
```

```
            case 1:
```

```
            case 3:
```

```
            case 5:
```

```
            case 7:
```

```
            case 8:
```

```
            case 10:
```

```
case 12:
    mDay = 31;
    break;
case 2:
    if (CheckYear(y))
        mDay = 29;
    else
        mDay = 28;

    break;
case 4:
case 6:
case 9:
case 11:
    mDay = 30;
    break;
}
nextDay = d + 1;
nextMonth = m;
nextYear = y;

if (nextDay > mDay) {
    nextDay = 1;
    nextMonth++;
}

if (nextMonth > 12) {
    nextMonth = 1;
```



```
        nextYear++;  
    }  
    String nextDate=nextDay+"-"+nextMonth+"-"+nextYear;  
    return nextDate;  
}  
}
```

b)TestSuite.java

```
package TestSuite;

import Program.ProgramEleven;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class TestSuite {

    ProgramEleven obj;

    @BeforeClass

    public void init(){

        obj = new ProgramEleven();

    }

    @Test(priority = 0)

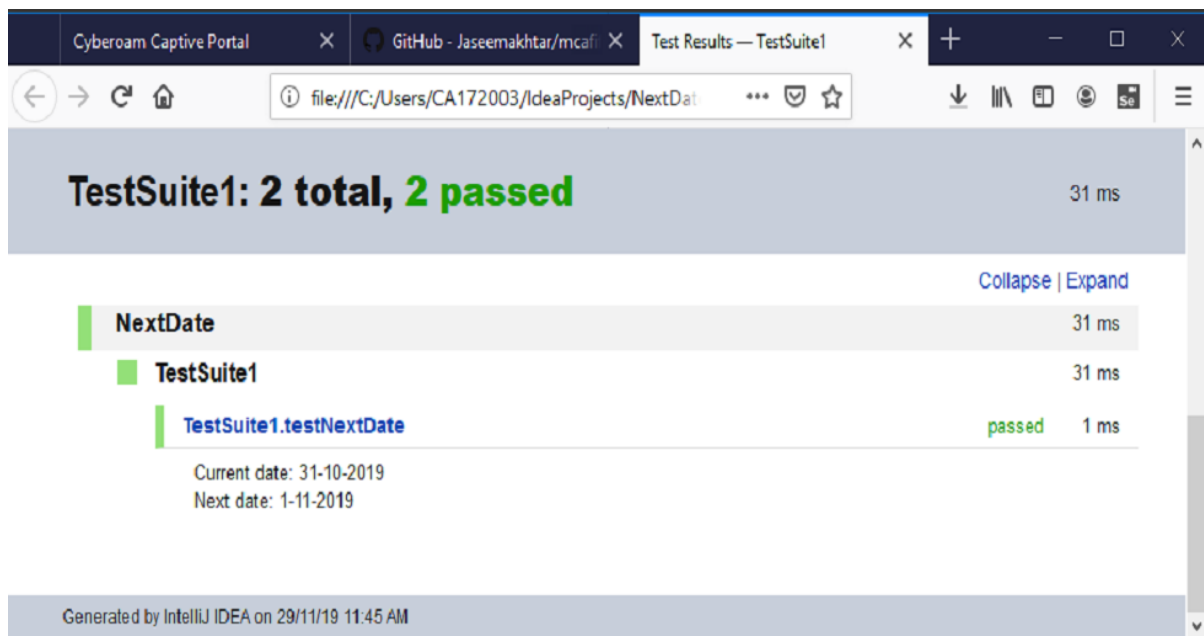
    public void testNextDate(){

        Assert.assertEquals(obj.GenerateNextDate(31,10,2019),"1-11-2019");

    }

}
```

Output



5) Program to test a count number of item present on desktop.**a) DesktopItem.java**

```
package program;

import java.io.File;

public class DesktopItems {
    File desktopLocation;

    public DesktopItems(){
        desktopLocation = new File("C:\\Users\\CA172003\\Desktop");
    }

    public int countItems(){
        int items = 0;

        try {
            File[] files = desktopLocation.listFiles();

            System.out.println("Files present in Desktop: ");

            for (int i = 0; i < files.length; i++) {
                if(!files[i].getName().contains(".ini")){
                    items++;
                    System.out.println(items + ". " + files[i].getName());
                }
            }

            System.out.println("\nTotal items on Desktop: " + items);
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }

        return items;
    }
}
```

b) TestSuiteOne.java

```
package tests;
```

```
import org.testng.Assert;
```

```
import org.testng.annotations.BeforeClass;
```

```
import org.testng.annotations.Test;
```

```
import program.DesktopItems;
```

```
public class TestSuiteOne {
```

```
    DesktopItems desktopItems;
```

```
    @BeforeClass
```

```
    public void init(){
```

```
        desktopItems = new DesktopItems();
```

```
    }
```

```
    @Test
```

```
    public void test(){
```

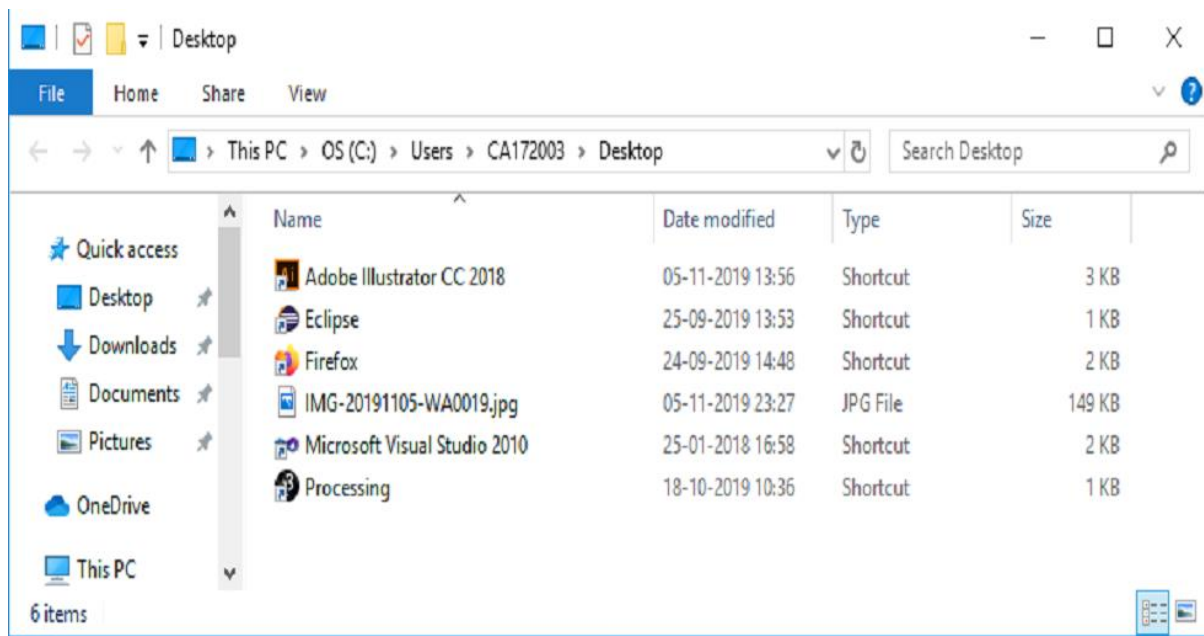
```
        int items = desktopItems.countItems();
```

```
        Assert.assertEquals(items, 3);
```

```
    }
```

```
}
```

Output



6) Imagine testing a date field. The field is of the form MM/DD/YYYY (two digit month, two digit day, 4 digit year). Does equivalence class analysis and boundary tests that you would run in order to test the field (Don't bother with non-numeric values for these fields).

a)DateFieldTest.java

```
package program;

public class DateFieldTest {
    int mm, dd, yyyy;    int nDay;

    public boolean checkDate(String date){
        String[] dates = date.split("/");

        System.out.println("Entered Date: " + date);

        mm = Integer.parseInt(dates[0]);
        dd = Integer.parseInt(dates[1]);
        yyyy = Integer.parseInt(dates[2]);

        if(mm > 12 || mm <= 0){
            System.out.println("Invalid Month!");
            return false;
        }

        if(yyyy < 1000){
            System.out.println("Invalid Year!");
            System.out.println("Must be 4 digits!");
            return false;
        }

        switch (mm){
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                nDay = 31;
                break;
```

```
        case 2:
            if(checkYear(yyyy))
                nDay = 29;
            else
                nDay = 28;

            break;

        case 4:
        case 6:
        case 9:
        case 11:
            nDay = 30;
            break;
    }

    if(dd > nDay || dd <= 0){
        System.out.println("Invalid Day!");
        System.out.println("Month " + mm + " has " + nDay + " days");
        return false;
    }

    return true;
}

private boolean checkYear(long year) {
    if (year % 400 == 0)
        return true;
    else if (year % 100 == 0)
        return false;
    else if (year % 4 == 0)
        return true;
    else
        return false;
}
}
```


b) TestSuiteOne

```
package tests;

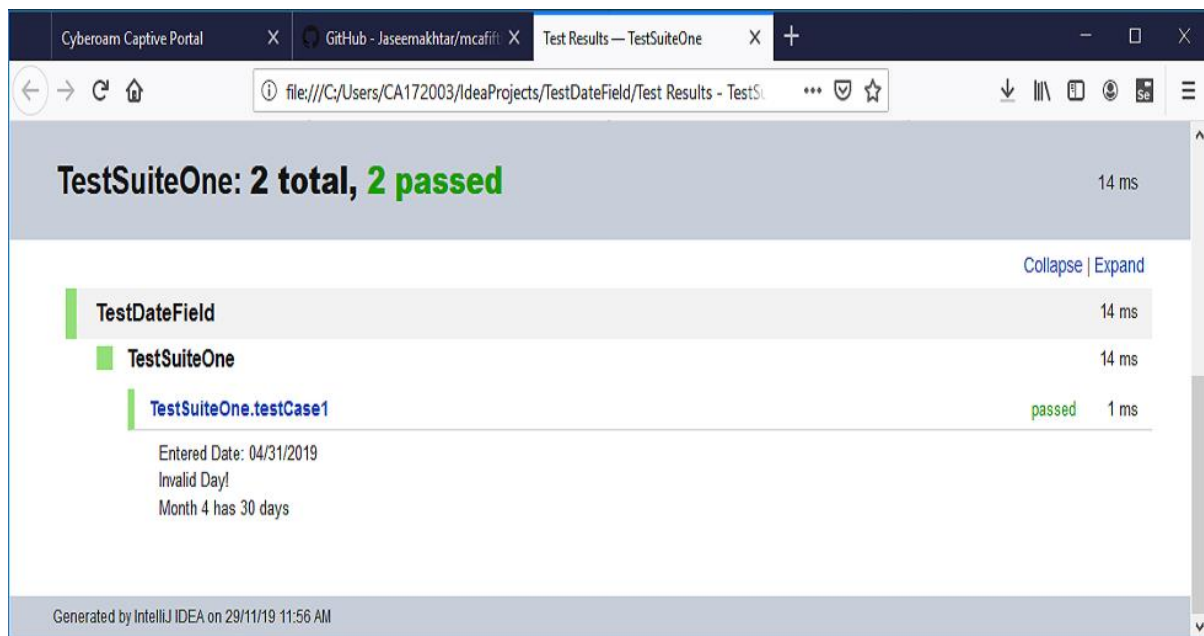
import org.testng.Assert;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
import program.DateFieldTest;

public class TestSuiteOne {
    DateFieldTest dFT;

    @BeforeClass
    public void init(){
        dFT = new DateFieldTest();
    }

    @Test
    public void testCase1(){
        //Date must be in the format MM/DD/YYYY
        Assert.assertTrue(dFT.checkDate("04/31/2019"));
    }
}
```

Output



7) Imagine testing a file name field. For example, go to an open file dialog, you can enter something into the field. Do a domain testing analysis: List a risk, equivalence classes appropriate to the risk, and best representatives of the equivalence classes. For each test case (use a best representative), briefly explain why this is best representative. Keep doing this until you have listed 12 best-representative test cases.

a) FileUploader.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <meta http-equiv="X-UA-Compatible" content="ie=edge">

  <title>File Upload</title>

  <style>

    *{

      font-family: Arial, Helvetica, sans-serif;

    }

  </style>

</head>

<body>

  <form action="">

    <h1>Upload your files to this boring server!</h1>

    <input id="fileUpload" type="file">

  </form>

</body>

</html>
```

b) TestSuiteOne.java

```
package test;

import org.openqa.selenium.By;
import org.openqa.selenium.InvalidArgumentException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
import java.util.concurrent.TimeUnit;

public class TestSuite1 {

    WebDriver driver;

    @BeforeClass
    void init(){

        System.setProperty("webdriver.chrome.driver","C:\\\\testLib\\\\chromedriver.exe");

        driver = new ChromeDriver();

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

    }

    @Test
    void openDialog() throws InterruptedException{

        int failed = 0;
```

```
driver.get("file:///C:/Users/Shubham/IdeaProjects/TestFileUpload/src/FileUploader.html");

System.out.println("Browser opened!");

WebElement fileUpload = driver.findElement(By.id("fileUpload"));

for(int i = 0; i < 11; i++) {

    try {

        //Giving a file name which doesn't exists

        fileUpload.sendKeys("C:\\testLib\\chromedriver" + i + ".exe");

    } catch (InvalidArgumentException e) {

        failed++;

        System.out.println("File not found!");

    }

}

//Giving a file name which exists

fileUpload.sendKeys("C:\\testLib\\chromedriver.exe");

System.out.println("File found! -> chromedriver.exe ");

Assert.assertEquals(failed, 11);

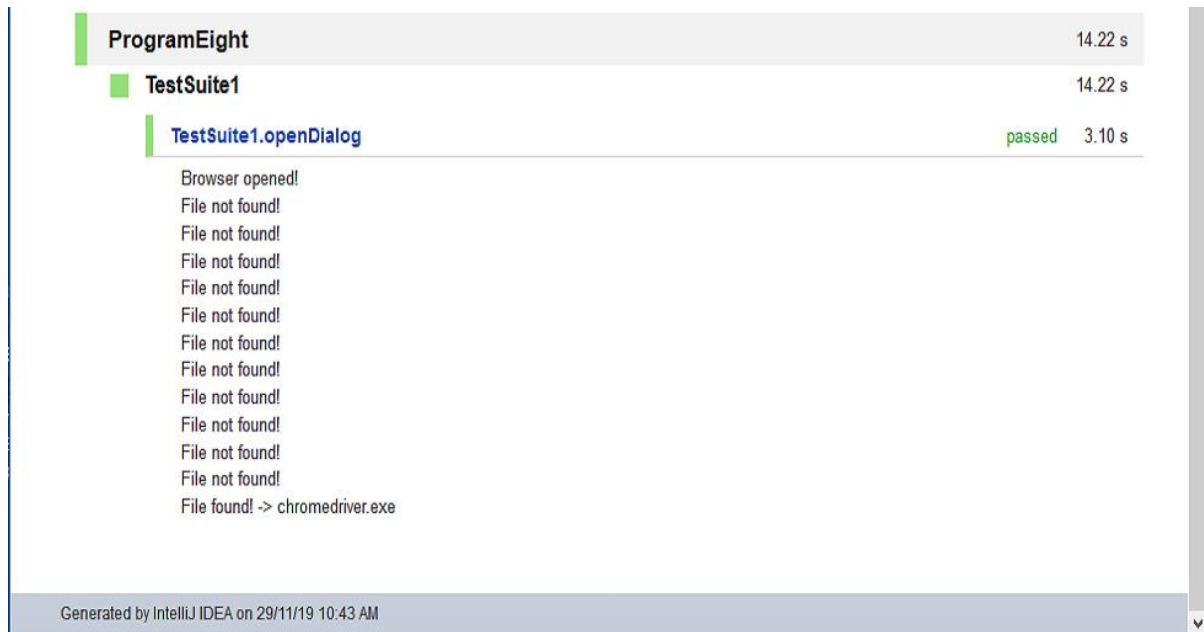
Thread.sleep(700);

driver.close();

}

}
```

Output



The screenshot displays the test results in IntelliJ IDEA. The test suite 'TestSuite1' is shown as passed, with a duration of 3.10 s. The test 'TestSuite1.openDialog' is also shown as passed, with a duration of 3.10 s. The test output shows a sequence of messages: 'Browser opened!', followed by ten 'File not found!' messages, and finally 'File found! -> chromedriver.exe'.

Test Suite	Duration
ProgramEight	14.22 s
TestSuite1	14.22 s
TestSuite1.openDialog	3.10 s

Browser opened!
File not found!
File not found!
File not found!
File not found!
File not found!
File not found!
File not found!
File not found!
File not found!
File not found!
File not found!
File found! -> chromedriver.exe

Generated by IntelliJ IDEA on 29/11/19 10:43 AM

- 8) Write the test cases for any known application(Banking Application)
- a. Checking mandatory input parameters.
 - b. Checking optional input parameters.
 - c. Check whether able to create account entity.
 - d. Check whether you are able to withdraw an amount in the newly created account (and thus updating the balance)
 - e. Check whether you are able to withdraw an amount in the newly created account (after deposit) (and thus updating the balance)

a) Bank.java

```
package program;

import java.util.ArrayList;

public class Bank {

    ArrayList<BankAccount> accounts;

    boolean isLoggedIn;

    String loggedInUser;

    public void init(){

        println("-----");

        println("Welcome to Bank of CS!");

        accounts = new ArrayList<>();

        BankAccount acc1 = new BankAccount();

        acc1.userName = "Tony";

        acc1.password = "pepper";

        acc1.balance = 999;

        accounts.add(acc1);

    }

    public boolean createAccount(String userName, String password){

        for(BankAccount account: accounts){
```

```
        if(account.userName.equals(userName)){
            println("-----");
            println("User already exists!");
            return false;
        }
    }

    BankAccount acc = new BankAccount();
    acc.userName = userName;
    acc.password = password;
    accounts.add(acc);
    login(userName, password);
    return true;
}

public boolean login(String userName, String password){
    for(BankAccount account: accounts){
        if(account.userName.equals(userName) && account.password.equals(password)){
            println("-----");
            println("Login successful!");
            isLoggedIn = true;
            loggedInUser = userName;
        }
    }

    if(!isLoggedIn){
        println("-----");
        println("Login failed!");
    }

    return isLoggedIn;
}
```



```
}

public void logOut(){

    isLoggedIn = false;

}

public boolean deposit(long balance){

    if(isLoggedIn){

        for (BankAccount bankAccount: accounts){

            if (bankAccount.userName.equals(loggedInUser)){

                bankAccount.balance += balance;

                println("-----");

                println("Deposited Amount: " + balance);

                println("Available Amount: " + bankAccount.balance);

                return true;

            }

        }

    }

    return false;

}

public boolean withdraw(long balance){

    if(isLoggedIn){

        for (BankAccount bankAccount: accounts){

            if (bankAccount.userName.equals(loggedInUser)){

                if(bankAccount.balance >= balance){

                    bankAccount.balance -= balance;

                    println("-----");

                    println("Withdrawal Amount: " + balance);

                    println("Available Amount: " + bankAccount.balance);
```

```
        return true;

    }else{

        println("-----");

        println("Not sufficient balance!");

        return false;

    }

}

}

}

return false;

}

public void exit(){

    println("-----");

    println("Thank you for using Bank of CS!");

}

public static void println(Object o){

    System.out.println(o);

}

}
```

b) BankAccount.java

```
package program;

public class BankAccount {

    String userName;

    String password;

    long balance;

}
```

c) TestSuite.java

```
package tests;

import org.testng.Assert;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

import program.Bank;

public class TestSuite1 {

    Bank b;

    @BeforeClass

    public void initTest(){

        b = new Bank();

        b.init();

    }

    @Test(priority = 1)

    public void loggedInTest(){

        Assert.assertTrue(b.login("Tony", "pepper"));

        b.logOut();

    }

    @Test(priority = 2)

    public void accountCreateTest(){

        Assert.assertTrue(b.createAccount("Jasmit", "Jasmit"));

    }

    @Test(priority = 3)

    public void depositedTest(){

        Assert.assertTrue(b.deposit(1000));

    }

}
```

```
@Test(priority = 4)

public void withdrawalTest(){

    Assert.assertTrue(b.withdraw(1000));

}


@AfterClass

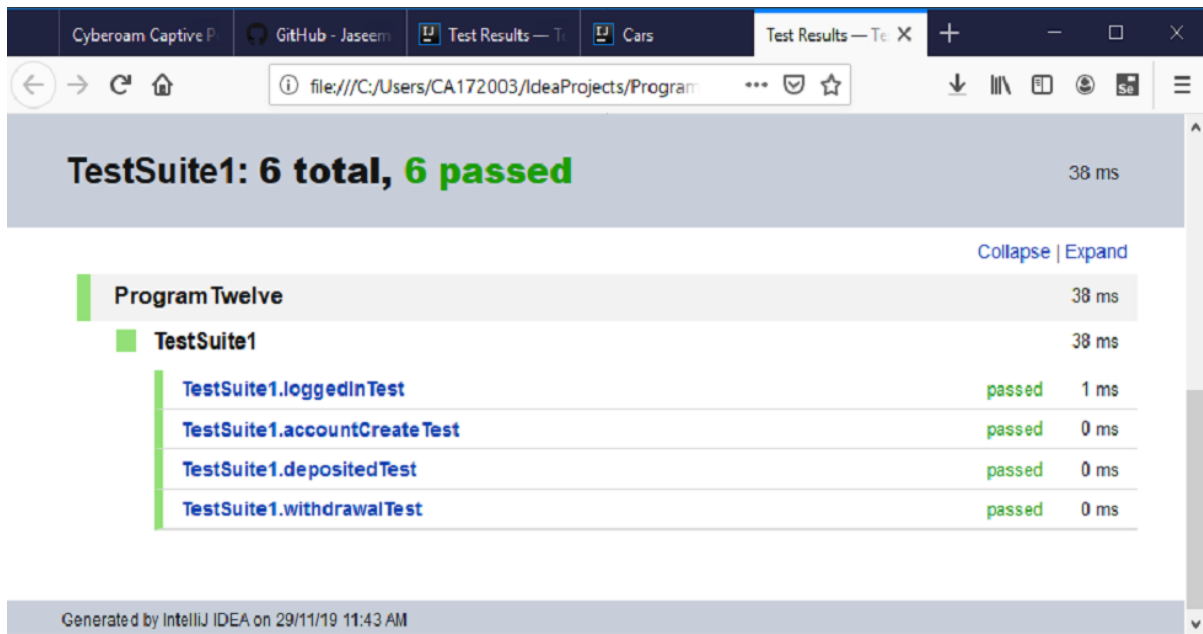
public void exitTest(){

    b.exit();

}

}
```

Output



The screenshot shows the IntelliJ IDEA Test Results window. The top bar indicates the current file is `file:///C:/Users/CA172003/IdeaProjects/Program`. The main content area displays the test results for `TestSuite1`, which has 6 tests in total, all of which passed. The total execution time for the suite is 38 ms. Below the suite summary, the individual tests are listed with their execution times.

Test Name	Status	Execution Time
TestSuite1	6 total, 6 passed	38 ms
Program Twelve		38 ms
TestSuite1		38 ms
TestSuite1.loggedInTest	passed	1 ms
TestSuite1.accountCreateTest	passed	0 ms
TestSuite1.depositedTest	passed	0 ms
TestSuite1.withdrawalTest	passed	0 ms

Generated by IntelliJ IDEA on 29/11/19 11:43 AM

9) Program to test an update 10 student records into table into Excel file.**a) UpdateExcelFile.java**

package program;

```
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
```

```
public class UpdateExcelFile {
```

```
    public int updateRecords(int colToUpdate, String newValue){
```

```
        int updated = 0;
```

```
        try {
```

```
            File file = new
```

```
File("/home/shubham/IdeaProjects/ReadExcelFile/src/program/mcafive.xls");
```

```
        String fileName = file.getName();
```

```
        String fileExtension = fileName.substring(fileName.indexOf("."));
```

```
        String sheetName = "Sheet1";
```

```
        FileInputStream fis = new FileInputStream(file);
```

```
        Workbook excelFile = null;
```

```
        if(fileExtension.equals(".xlsx")){
```

```
            //If it is xlsx file then create XSSFWorkbook object
```

```
            excelFile = new XSSFWorkbook(fis);
```

```
        }else if(fileExtension.equals(".xls")){
```

```
            //If it is xls file then create HSSFWorkbook object
```

```
            excelFile = new HSSFWorkbook(fis);
```

```
        }
```

```
        Sheet sheet = excelFile.getSheet(sheetName);
```

```
        System.out.println("Sno RollNum Name Sem");
```

```
for(int row = 1; row < sheet.getLastRowNum() + 1; row++){
    Row sheetRow = sheet.getRow(row);
    for(int col = 0; col < sheetRow.getLastCellNum(); col++){
        Cell cell = sheetRow.getCell(col);
        System.out.print(cell + " ");
    }
    System.out.println();
}

System.out.println();
System.out.println("Updating...");
System.out.println();

for(int row = 1; row < sheet.getLastRowNum() + 1; row++){
    Row sheetRow = sheet.getRow(row);
    for(int col = 0; col < sheetRow.getLastCellNum(); col++){
        Cell cell = sheetRow.getCell(col);
        if(col == colToUpdate - 1){
            cell.setCellValue(new Value);
            updated++;
        }
        System.out.print(cell + " ");
    }
    System.out.println();
}

fis.close();

FileOutputStream fos = new FileOutputStream(file);
excelFile.write(fos);

System.out.println();
System.out.println("Updated: " + updated + " records.");

fos.close();

} catch (Exception e) {
    e.printStackTrace();
}
return updated;
}
}
```

b) TestSuite.java

```
package test;
```

```
import org.testng.Assert;
```

```
import org.testng.annotations.BeforeClass;
```

```
import org.testng.annotations.Test;
```

```
import program.UpdateExcelFile;
```

```
public class TestSuite1 {
```

```
    private UpdateExcelFile instance;
```

```
    @BeforeClass
```

```
    public void initTestSuite1(){
```

```
        instance = new UpdateExcelFile();
```

```
    }
```

```
    @Test
```

```
    public void testCaseUpdateFile(){
```

```
        Assert.assertEquals(instance.updateRecords(4, "v"), 8);
```

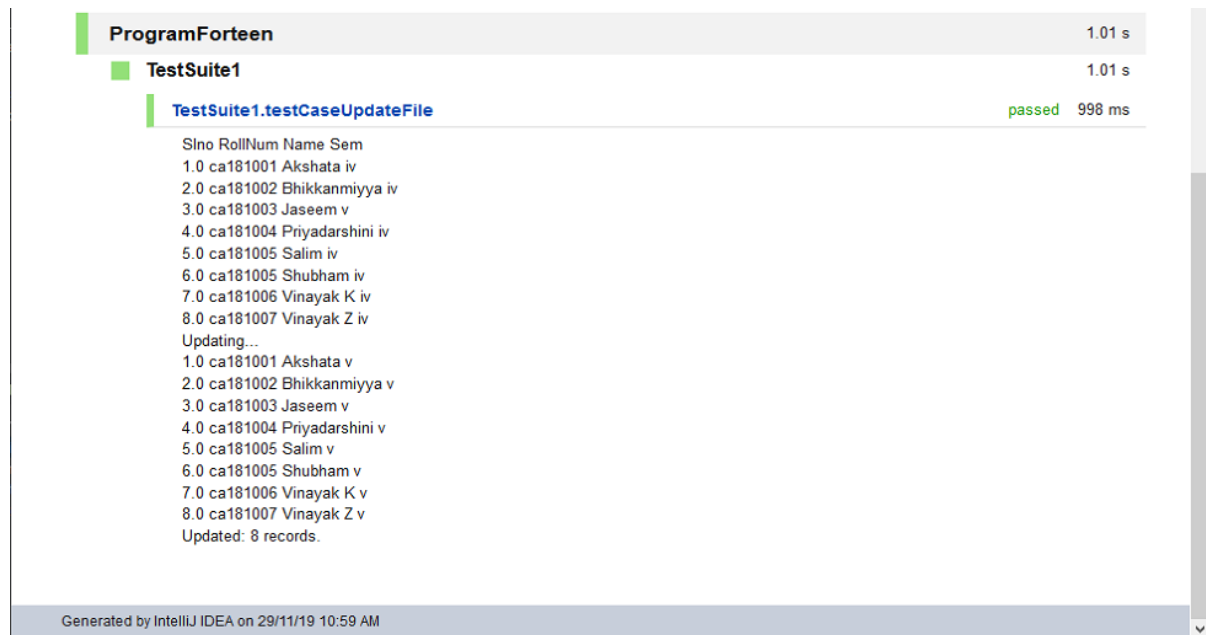
```
    }
```

```
}
```


Output

slno	rollNo	name	semester
1	ca181001	Akshata	iv
2	ca181002	Bhikkanmiyya	iv
3	ca181003	Jaseem	v
4	ca181004	Priyadarshini	iv
5	ca181005	Salim	iv
6	ca181005	Shubham	iv
7	ca181006	Vinayak K	iv
8	ca181007	Vinayak Z	iv

slno	rollNo	name	semester
1	ca181001	Akshata	v
2	ca181002	Bhikkanmiyya	v
3	ca181003	Jaseem	v
4	ca181004	Priyadarshini	v
5	ca181005	Salim	v
6	ca181005	Shubham	v
7	ca181006	Vinayak K	v
8	ca181007	Vinayak Z	v



ProgramForteen 1.01 s

TestSuite1 1.01 s

TestSuite1.testCaseUpdateFile passed 998 ms

```
S/no RollNum Name Sem
1.0 ca181001 Akshata iv
2.0 ca181002 Bhikkanmiyya iv
3.0 ca181003 Jaseem v
4.0 ca181004 Priyadarshini iv
5.0 ca181005 Salim iv
6.0 ca181005 Shubham iv
7.0 ca181006 Vinayak K iv
8.0 ca181007 Vinayak Z iv
Updating...
1.0 ca181001 Akshata v
2.0 ca181002 Bhikkanmiyya v
3.0 ca181003 Jaseem v
4.0 ca181004 Priyadarshini v
5.0 ca181005 Salim v
6.0 ca181005 Shubham v
7.0 ca181006 Vinayak K v
8.0 ca181007 Vinayak Z v
Updated: 8 records.
```

Generated by IntelliJ IDEA on 29/11/19 10:59 AM

10) Program to test to provide total number of object present/available on the page.**a) CountObjects.java**

package program;

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

```
import java.util.List;
import java.util.concurrent.TimeUnit;
```

```
public class CountObjects {
    WebDriver webDriver;
```

```
    public CountObjects(){
```

```
        System.setProperty("webdriver.chrome.driver", "/home/CA172007/seleniumLibs/chromedriver");
```

```
        webDriver = new ChromeDriver();
        webDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    }
```

```
    public int openPageAndCount(){
        int items = 0;
```

```
        try {
            webDriver.get("https://github.com/Shub2495/");
            WebElement rootElement = webDriver.findElement(By.cssSelector("body"));
            List<WebElement> childs = rootElement.findElements(By.xpath("./*"));
            for (WebElement webElement: childs){
                if(webElement.isDisplayed()){
                    items++;
                    System.out.println(items + " " + webElement.getTagName());
                }
            }
        }
```

```
        System.out.println("Total visible items: " + items);
        Thread.sleep(3000);
        webDriver.close();
```

```
        }catch (Exception e){  
            System.out.println(e.getMessage());  
        }  
  
        return items;  
    }  
}
```

b) TestSuite.java

```
package test;
```

```
import org.testng.Assert;
```

```
import org.testng.annotations.BeforeClass;
```

```
import org.testng.annotations.Test;
```

```
import program.CountObjects;
```

```
public class TestSuite1 {
```

```
    CountObjects countObjects;
```

```
    @BeforeClass
```

```
    public void init(){
```

```
        countObjects = new CountObjects();
```

```
    }
```

```
    @Test
```

```
    public void testCaseCount(){
```

```
        Assert.assertEquals(countObjects.openPageAndCount(), 705);
```

```
    }
```

```
}
```

Output

ProgramFifteen 42.99 s

TestSuite1 42.99 s

TestSuite1.testCaseCount passed 38.77 s

684 a
685 li
686 a
687 li
688 a
689 li
690 a
691 li
692 a
693 li
694 a
695 div
696 div
Total visible items: 696

Generated by IntelliJ IDEA on 29/11/19 11:16 AM

11) Program to test to get the number of list items in a list / combo box.**a) Demo.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Cars</title>

  <style>
    *{
      font-family: Arial;

    }
  </style>
</head>
<body>
  <h1>Cars </h1>
  <select id="car-list">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="opel">Opel</option>
    <option value="audi">Audi</option>
    <option value="maruti">Maruti</option>
  </select>

</body>
</html>
```

b) CountListItems.java

```
package program;

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

import java.util.List;
import java.util.concurrent.TimeUnit;

public class CountListItems {

    WebDriver webDriver;

    public CountListItems(){

        System.setProperty("webdriver.chrome.driver","/home/user/seleniumLibs/chromedriver")

        webDriver = new ChromeDriver();

        webDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

    }

    public int openWebPage(){

        int items = 0;

        try{

            webDriver.get("file:///home/user/IdeaProjects/SeleniumListItems/src/program/Demo.html");

            System.out.println("Browser Opened!");
```



```
List<WebElement> webElements =  
webDriver.findElements(By.cssSelector("#car-list > option"));  
  
for (WebElement w : webElements) {  
    System.out.println("Item: " +w.getText());  
}  
  
System.out.println("Found List Items: " + webElements.size());  
  
items = webElements.size();  
  
Thread.sleep(5000);  
  
webDriver.close();  
  
} catch (Exception e){  
    System.out.println(e.getMessage());  
}  
  
return items;  
}  
}
```

c) TestSuite.java

```
package testSuite;

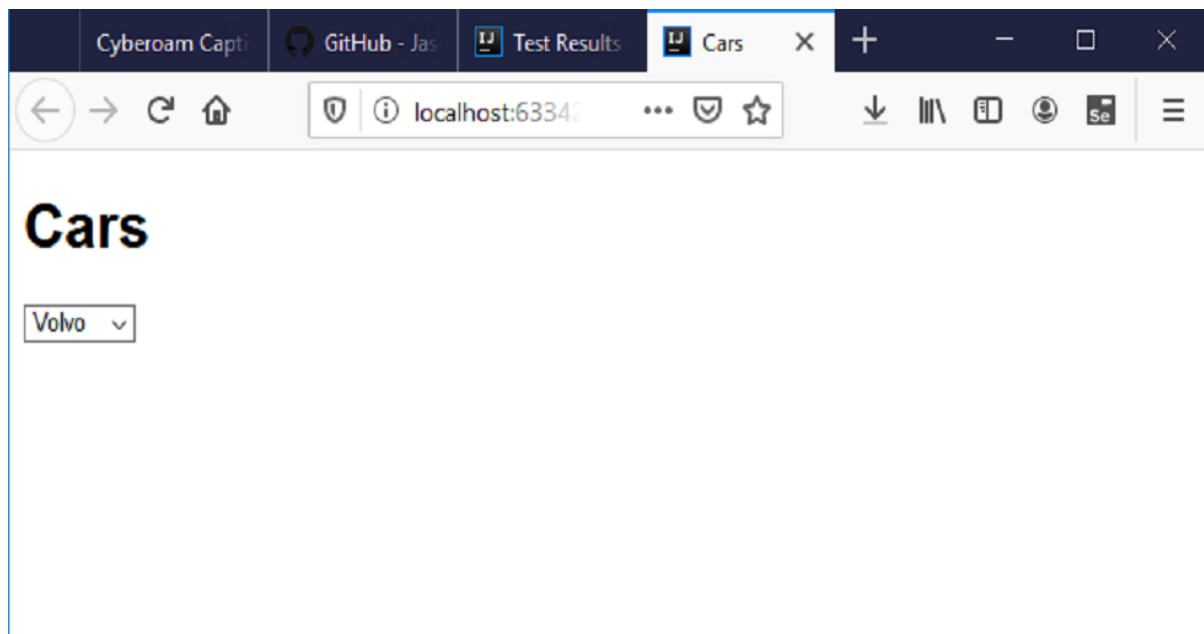
import org.testng.Assert;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
import program.CountListItems;

public class TestSuite1 {
    CountListItems countListItems;

    @BeforeClass
    public void initTest(){
        countListItems = new CountListItems();
    }

    @Test
    public void testItems(){
        Assert.assertEquals(5, countListItems.openWebPage());
    }
}
```

Output



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

a) ProgramThirteen.java

```
package program;
```

```
import java.util.ArrayList;
```

```
public class ProgramThirteen {
```

```
    ArrayList<StudentRecord> studentRecords;
```

```
    public ProgramThirteen(){
```

```
        studentRecords = new ArrayList<>();
```

```
        StudentRecord akshata = new StudentRecord();
```

```
        akshata.name = "Akshata";
```

```
        akshata.avg = 90;
```

```
        StudentRecord bhikkanmiya = new StudentRecord();
```

```
        bhikkanmiya.name = "Bhikkanmiya";
```

```
        bhikkanmiya.avg = 70;
```

```
        StudentRecord jaseem = new StudentRecord();
```

```
        jaseem.name = "Jaseem";
```

```
        jaseem.avg = 74;
```

```
        StudentRecord salim = new StudentRecord();
```

```
        salim.name = "Salim";
```

```
        salim.avg = 92;
```

```
        StudentRecord shubham = new StudentRecord();
```

```
        shubham.name = "Shubham";
```

```
        shubham.avg = 80;
```

```
        StudentRecord vinayak_k = new StudentRecord();
```

```
        vinayak_k.name = "Vinayak K";
```

```
        vinayak_k.avg = 79;
```

```
        StudentRecord vinayak_z = new StudentRecord();
```

```
        vinayak_z.name = "Vinayak Z";
```

```
        vinayak_z.avg = 78;
```

```
        StudentRecord sachin = new StudentRecord();
```

```
sachin.name = "Sachin";
sachin.avg = 20;

studentRecords.add(akshata);
studentRecords.add(bhikkanmiya);
studentRecords.add(jaseem);
studentRecords.add(salim);
studentRecords.add(shubham);
studentRecords.add(vinayak_k);
studentRecords.add(vinayak_z);
studentRecords.add(sachin);
}

public int displayPassed(){
    int totalPassed = 0;

    for (StudentRecord studentRecord: studentRecords){
        if(studentRecord.avg > 60){
            System.out.println("Name: " + studentRecord.name);
            System.out.println("Your score: " + studentRecord.avg + "%");
            System.out.println("***** Passed *****");
            totalPassed += 1;
        }else{
            System.out.println("Name: " + studentRecord.name);
            System.out.println("Your score: " + studentRecord.avg + "%");
            System.out.println("***** Failed *****");
            System.out.println("_____");
        }
    }
    return totalPassed;
}
}
```

b) StudentRecord.java

package program;

```
public class StudentRecord {
    String regno;
    String name;
    String sem;
    int avg;
}
```

c) TestSuite.java

```
package tests;
```

```
import org.testng.Assert;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;
import program.ProgramThirteen;
```

```
public class TestSuite1 {
```

```
    ProgramThirteen programThirteen;
```

```
    @BeforeClass
```

```
    public void init(){
        programThirteen = new ProgramThirteen();
    }
```

```
    @Test
```

```
    public void scoreCheck(){
        Assert.assertEquals(programThirteen.displayPassed(), 8);
    }
}
```

Output

The screenshot displays a test runner interface. At the top, a summary bar for 'TestSuite1' shows '2 total, 2 passed' in green text and a duration of '28 ms'. Below this, a tree view shows the test structure: 'ProgramThirteen' (28 ms), 'TestSuite1' (28 ms), and 'TestSuite1.scoreCheck' (1 ms, passed). The 'TestSuite1.scoreCheck' test is expanded, showing three sub-tests: 'Name: Akshata' with 'Your score: 90%' and '**** Passed ****'; 'Name: Bhikkanmiya' with 'Your score: 70%' and '**** Passed ****'; and 'Name: Jaseem' with 'Your score: 74%' and '**** Passed ****'. The interface includes 'Collapse' and 'Expand' links for the suite, and a vertical scrollbar on the right.

```
TestSuite1: 2 total, 2 passed 28 ms
  ProgramThirteen 28 ms
    TestSuite1 28 ms
      TestSuite1.scoreCheck passed 1 ms
        Name: Akshata
        Your score: 90%
        **** Passed ****
        Name: Bhikkanmiya
        Your score: 70%
        **** Passed ****
        Name: Jaseem
        Your score: 74%
        **** Passed ****
```