LAB 1:

Name of Experiment: Write a program that calculates the area and perimeter of the circle.And find the Coverage & Test Cases of that program using JaButi Tool.

Aim :

Steps to perform Experiment:

Program Code: AreaOfCircle.java

```
import java.util.Scanner;
import java.lang.Math;
public class AreaOfCircle
{
public static void main(String[] args)
{
int option;
double radius, circumference, diameter, area;
//object of the Scanner class
Scanner sc=new Scanner (System.in);
//options available
System.out.println("1. If the radius is known");
System.out.println("2. If the diameter is known");
System.out.println("3. If the circumference is known");
System.out.print("Enter your choice: ");
//taking an option as input from the user
option=sc.nextInt();
switch(option)
{
//Case statements
case 1:
```



```
  System.out.print("Enter the radius of the circle: ");
  radius=sc.nextDouble();
  area=(Math.PI*(radius*radius));
  System.out.print("The area of the circle is: "+area);
  break;
case 2:
  System.out.print("Enter the diameter of the circle: ");
  diameter=sc.nextDouble();
  area=Math.PI*(diameter*diameter)/4;
  System.out.print("The area of the circle is: "+area);
  break;
case 3:
  System.out.print("Enter the circumference of the circle: ");
  circumference=sc.nextDouble();
  area=(circumference*circumference)/(4*Math.PI);
  System.out.print("The area of the circle is: "+area);
  break;
//default case statement executes when an invalid choice is entered
default:System.out.println("invalid choice!");
}
}
}
```

Compile Java Program:>_ javac AreaOfCircle.java

It will create a new file AreaOfCircle.class **// BYTE CODE**

Now assign this ByteCode to JaBUTi Tool for Coverage Testing by following steps

Lab 2:

Name of Experiment: 2. Write a program which read the first name and last name from console and matching with expected result by using JaBuTi.

Steps to perform Experiment:

Program Code: AreaOfCircle.java

```
Import java.util.Scanner;

/**
 * An example program to read a Name from console input in Java
 */
Public class Example {

    Public static void main(String[] args) {
        System.out.print("Enter a Name : ");
        Scanner scanner = new Scanner(System. in);
        String inputString = scanner. nextLine();
        System.out.println("String read from console is : \n"+input
String);
    }
}
```

Note : Repeat all step according to Experiment 1 in JABUTI

Lab: 3

Name of Experiment: Write a program which read the first name and last name from console and matching with expected result by using JaBuTi.

Steps to perform Experiment:

Program Code: qeqvation.java

```
public class q eqvation
{

 public static void main(String[] args) {

  // value a, b, and c
  double a = 2.3, b = 4, c = 5.6;
  double root1, root2;

  // calculate the determinant (b2 - 4ac)
  double determinant = b * b - 4 * a * c;

  // check if determinant is greater than 0
  if (determinant > 0) {

   // two real and distinct roots
   root1 = (-b + Math.sqrt(determinant)) / (2 * a);
   root2 = (-b - Math.sqrt(determinant)) / (2 * a);

   System.out.format("root1 = %.2f and root2 = %.2f", root1, root2);
  }

  // check if determinant is equal to 0
  else if (determinant == 0) {
```

```
    // two real and equal roots
    // determinant is equal to 0
    // so -b + 0 == -b
    root1 = root2 = -b / (2 * a);
    System.out.format("root1 = root2 = %.2f;", root1);
   }

   // if determinant is less than zero
   else {

    // roots are complex number and distinct
    double real = -b / (2 * a);
    double imaginary = Math.sqrt(-determinant) / (2 * a);
    System.out.format("root1 = %.2f+%.2fi", real, imaginary);
    System.out.format("\nroot2 = %.2f-%.2fi", real, imaginary);
   }
  }
}
```

Note: Repeat all step according to Experiment 1 in JABUTI

Lab: 4

Name of Experiment: Write a program that reads commercial website URL from a url from file. You should expect that the URL starts with www and ends with .com. Retrieve the name of the site and output it. For instance, if the user inputs www.yahoo.com, you should output yahoo. After that find the test cases and coverage using JaButi.

Steps to perform Experiment:

Program Code:

```
package mypkg;

 import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;


public class HelloServlet extends HttpServlet {

  @Override

  public void doGet(HttpServletRequest request, HttpServletResponse response)

          throws IOException, ServletException {

    // Set the response message's MIME type

    response.setContentType("text/html;charset=UTF-8");

    // Allocate a output writer to write the response message into the network socket

    PrintWriter out = response.getWriter();


    // Write the response message, in an HTML page

    try {

      out.println("<!DOCTYPE html>");
```

```
        out.println("<html><head>");
        out.println("<meta http-equiv='Content-Type' content='text/html; charset=UTF-8'>");
        out.println("<title>Hello, World</title></head>");
        out.println("<body>");
        out.println("<h1>Hello, world!</h1>");  // says Hello
        // Echo client's request information
        out.println("<p>Request URI: " + request.getRequestURI() + "</p>");
        out.println("<p>Protocol: " + request.getProtocol() + "</p>");
        out.println("<p>PathInfo: " + request.getPathInfo() + "</p>");
        out.println("<p>Remote Address: " + request.getRemoteAddr() + "</p>");
        // Generate a random number upon each request
        out.println("<p>A Random Number: <strong>" + Math.random() + "</strong></p>");
        out.println("</body>");
        out.println("</html>");
      } finally {
        out.close();  // Always close the output writer
      }
    }
}
```

Note: Repeat all step according to Experiment 1 in JABUTI

Lab: 5

Name of Experiment: Write a program for a calculator and find the test case and coverage and Def-use-graph.

Steps to perform Experiment:

Program Code:

```
import java.util.Scanner;

class Main {

  public static void main(String[] args) {


    char operator;
    Double number1, number2, result;


    // create an object of Scanner class
    Scanner input = new Scanner(System.in);


    // ask users to enter operator
    System.out.println("Choose an operator: +, -, *, or /");
    operator = input.next().charAt(0);


    // ask users to enter numbers
    System.out.println("Enter first number");
    number1 = input.nextDouble();


    System.out.println("Enter second number");
```

```
    number2 = input.nextDouble();

    switch (operator) {

      // performs addition between numbers
      case '+':
        result = number1 + number2;
        System.out.println(number1 + " + " + number2 + " = " + result);
        break;

      // performs subtraction between numbers
      case '-':
        result = number1 - number2;
        System.out.println(number1 + " - " + number2 + " = " + result);
        break;

      // performs multiplication between numbers
      case '*':
        result = number1 * number2;
        System.out.println(number1 + " * " + number2 + " = " + result);
        break;

      // performs division between numbers
      case '/':
```



```
29
```



```
        result = number1 / number2;
        System.out.println(number1 + " / " + number2 + " = " + result);
        break;

      default:
        System.out.println("Invalid operator!");
        break;
    }

    input.close();
  }
}
```

Note : Repeat all step according to Experiment 1 in JABUTI

Lab: 6

Name of Experiment: Write a program that reads two words representing passwords from the java console and outputs the number of character in the smaller of the two. For example, if the words are open and sesame, then the output should be 4, the length of the shorter word, open. And test this program using JaButi.

Steps to perform Experiment:

Program Code:

```
import java.util.Scanner;

public class Exercise11 {

public static final int PASSWORD_LENGTH = 8;

public static void main(String[] args) {

    Scanner input = new Scanner(System.in);
    System.out.print(
        "1. A password must have at least eight characters.\n" +
        "2. A password consists of only letters and digits.\n" +
        "3. A password must contain at least two digits \n" +
        "Input a password (You are agreeing to the above Terms and Conditions.): ");
    String s = input.nextLine();

    if (is_Valid_Password(s)) {
      System.out.println("Password is valid: " + s);
    } else {
      System.out.println("Not a valid password: " + s);
    }

  }

  public static boolean is_Valid_Password(String password) {

    if (password.length() < PASSWORD_LENGTH) return false;

    int charCount = 0;
    int numCount = 0;
    for (int i = 0; i < password.length(); i++) {

      char ch = password.charAt(i);
```



```
      if (is_Numeric(ch)) numCount++;
      else if (is_Letter(ch)) charCount++;
      else return false;
    }

    return (charCount >= 2 && numCount >= 2);
  }

  public static boolean is_Letter(char ch) {
    ch = Character.toUpperCase(ch);
    return (ch >= 'A' && ch <= 'Z');
  }

  public static boolean is_Numeric(char ch) {

    return (ch >= '0' && ch <= '9');
  }

}
```

Note: Repeat all step according to Experiment 1 in JABUTI

Lab: 7

Name of Experiment: Analyze the performance of following website using JMeter.

Site	Website	Type
Amazon	Amazon.com	shopping
Flip kart	Flipkart.com	shopping
Railway reservation	Irctc.co.in	Ticket booking site
Train searching	Erail.in	Train searching

Steps to perform Experiment:

JMeter Introduction

JMeter is a software that can perform load test, performance-oriented business (functional) test, regression test, etc., on different protocols or technologies.

Stefano Mazzocchi of the Apache Software Foundation was the original developer of JMeter. He wrote it primarily to test the performance of Apache JServ (now called as Apache Tomcat project). Apache later redesigned JMeter to enhance the GUI and to add functional testing capabilities.

JMeter is a Java desktop application with a graphical interface that uses the Swing graphical API. It can therefore run on any environment / workstation that accepts a Java virtual machine, for example - Windows, Linux, Mac, etc.

The protocols supported by JMeter are -

- Web - HTTP, HTTPS sites 'web 1.0' web 2.0 (ajax, flex and flex-ws-amf)
- Web Services - SOAP / XML-RPC
- Database via JDBC drivers
- Directory - LDAP
- Messaging Oriented service via JMS
- Service - POP3, IMAP, SMTP
- FTP Service

JMeter Features

Following are some of the features of JMeter -

- Being an open source software, it is freely available.
- It has a simple and intuitive GUI.

Step 1: Verify Java Installation

First of all, verify whether you have Java installed in your system. Open your console and execute one of the following **java** commands based on the operating system you are working on.

OS	Task	Command
Windows	Open Command Console	c:\> java -version
Linux	Open Command Terminal	$ java -version
Mac	Open Terminal	machine: ~ joseph$ java -version

If you have Java installed in your system, you would get an appropriate output based on the OS you are working on.

OS	Output
Windows	java version "1.7.0_25" Java(TM) SE Runtime Environment (build 1.7.0_25-b15) Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
Linux	java version "1.7.0_25" Java(TM) SE Runtime Environment (build 1.7.0_25-b15) Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
Mac	java version "1.7.0_25" Java(TM) SE Runtime Environment (build 1.7.0_25-b15) Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)

If you do not have Java installed, install the Java Software Development Kit (SDK) from www.oracle.com/technetwork/java/javase/downloads/index.html. We are assuming Java 1.7.0_25 as the installed version for this tutorial.

Step 2: Set Java Environment

Set the **JAVA_HOME** environment variable to point to the base directory location, where Java is installed on your machine. For example –

OS	Output
Windows	Set the environment variable JAVA_HOME to C:\Program Files\Java\jdk1.7.0_25
Linux	export JAVA_HOME=/usr/local/java-current

Mac	export JAVA_HOME=/Library/Java/Home

Append Java compiler location to System Path.

OS	Output
Windows	Append the string; C:\Program Files\Java\jdk1.7.0_25\bin to the end of the system variable, Path.
Linux	export PATH=$PATH:$JAVA_HOME/bin/
Mac	not required

Verify Java Installation using **java -version** command as explained above.

Step 3: Download JMeter

Download the latest version of JMeter from https://jmeter.apache.org/download_jmeter.cgi. For this tutorial, we downloaded *apache-jmeter-2.9* and copied it into C:\>JMeter folder.

The directory structure should look like as shown below −

- apache-jmeter-2.9
- apache-jmeter-2.9\bin
- apache-jmeter-2.9\docs
- apache-jmeter-2.9\extras
- apache-jmeter-2.9\lib\
- apache-jmeter-2.9\lib\ext
- apache-jmeter-2.9\lib\junit
- apache-jmeter-2.9\printable_docs

You can rename the parent directory (i.e. apache-jmeter-2.9) if you want, but do not change any of the sub-directory names.

Step 4: Run JMeter

After downloading JMeter, go to the *bin* directory. In this case, it is **/home/manisha/apache-jmeter-2.9/bin**. Now click on the following −

OS	Output
Windows	jmeter.bat

POORNIMA

EXPERIMENT-6

Objective:- Calculate the mutation score of the following programs using Jumble Tool.

Program to detects the first occurrence of a duplicate and returns the value to the calling function

Program:-

```
package testPackage;
import java.util.Arrays;
import java.util.List;
public class SampProg {
protected int repeatedNumber (final List a)
 {
  int len = a.size(), i, dup = -1;
  int[] arr = new int[len];
  for (i=0; i<len; i++) {
     arr[i] = a.get(i);
  }
  Arrays.sort(arr);
  try {
    for (i=1; i<len; i++)
     {
      if (arr[i] == arr[i-1])
       {
       dup = arr[i];
       break;
    }}}
catch (Exception e) {
  System.out.println (e.getMessage());
}
  return dup;
}
```

Test Cases:-

Mutating testPackage.SampProg.
Tests:testPackage.SampProgTest
Mutation points = 11, unit test time limit 2.94s.
M FAIL: (testPackage.SampProg.java:8): -1 → 1
M FAIL: (testPackage.SampProg.java:10): 0 → 1
M FAIL: (testPackage.SampProg.java:10): negated conditional
M FAIL: (testPackage.SampProg.java:16): 1 → 0
M FAIL: (testPackage.SampProg.java:18): 1 → 0
M FAIL: (testPackage.SampProg.java:18): - → +
M FAIL: (testPackage.SampProg.java:18): negated conditional
M FAIL: (testPackage.SampProg.java:16): += → -=
M FAIL: (testPackage.SampProg.java:16): negated conditional

Jumbling took 7.545s
Score: 18%

POORNIMA

EXPERIMENT-7

Objectives- Calculate the coverage analysis of the program given in experiment-5 using Eclemma free open-source tool.

Eclemma is a free Java code coverage tool for Eclipse, available under the Eclipse Public License. It brings code coverage analysis directly into the Eclipse workbench.

i) Fast develop/test cycle- Launches from within the workbench like JUnit test runs can directly be analyzed for code coverage.

ii) Rich coverage analysis- Coverage results are immediately summarized and highlighted in the Java source code editors.

iii) Non-invasive- EclEmma does not require your projects or performing any other setup.

EclEmma adds a so called launch mode to the ~~Eli~~ Eclipse workbench. It is called Coverage mode and work exactly like the existing Run and Debug modes.

Simply launch your applications or unit tests in the Coverage mode to collect coverage information. Currently the following launch types are supported-

i) Local Java application
ii) Eclipse/RCP ~~algorithm~~ application.
iii) JUnit plug-in test
iv) JUnit RAP test
v) SWTBot test
vi) Scala application

⇒ Analysis-

i) Coverage Overview- Coverage view lists coverage summaries for your Java projects, allowing drill-down to method level.

ii) Source highlighting- The result of a coverage session is also directly visible in the Java source editors.

Additional features support analysis for your test coverage-

i) Different counters- Select whether instruction, branches, lines

methods, types on cyclomatic complexity should be summarized

ii) Multiple coverage sessions :- Switching between coverage data from multiple sessions is possible.

iii) Merge sessions :- If multiple different test runs should be considered for analysis coverage sessions can easily be merged.

Import / Export :-

i) Execution data import :- A wizard allows to import JaCoCo exec execution data files from external launches.

ii) Coverage report export :- Coverage data can be exported in HTML, XML or CSV format or as JaCoCo execution data files (*.exec).

Program :-

```
public boolean addAll (int index, Collection c) {
    if (c.isEmpty()) {
        return false;
    } else if (size == index || size == 0) {
        return addAll (c);
    } else {
        Listable succ = getListableAt (index);
    Listable prud = (null == succ) ? null : succ.prev();
    Iterator it = c.Iterator ();
    while (it.hasNext()) {
        prud = insertListable (prud, succ, it.next());
    }
    return true;
}
}
```

POORNIMA

EXPERIMENT - 8

Objective:- Generate Test sequences and validate using selenium tool for given below:-

Site	Website	Type
Amazon	Amazon.com	Shopping
Flipkart	Flipkart.com	Shopping
Railway reservation	Irctc.co.in	Ticket booking site
Train searching	Erail.in	Train Searching

Selenium is one of the most widely used open-source web UI (User Interface) automation testing frameworks. It enables testers to execute tests across different browsers, platforms and languages.

Components of Selenium:- As already stated, Selenium is a framework (i.e., it is not a single software but a suite composed of different components).

Selenium is composed of the following components:-

i) Selenium IDE
ii) Selenium WebDriver
iii) Selenium Grid.

i) Selenium IDE:- is a basically a Record / Run tool that is available as a browser plugin for Mozilla firefox and Google Chrome. You need not have any programming language experience to work with the IDE.

Selenium IDE in Selenium 4 is more than a playback and recording tool, the features of which can be leveraged to make the most out of Selenium automation testing.

ii) Selenium WebDriver:- is an enhanced version of Selenium RC and overcomes the limitations faced in Selenium RC. It controls the browser by direct communication. It has a faster execution time

as compared to IDE and RC.

Selenium Grid:- enables the users to perform parallel test execution. It is used along with Selenium RC to run parallel tests across different browsers and machines.

Users can run simultaneous tests in multiple environments, thereby saving a lot of time and expediting the time to market.

Steps to perform Experiment:- Below is a step by step process on how to download and install Selenium IDE for firefox:-

i) Open firefox Add-ons and Download selenium IDE
ii) Click on Add
iii) Click on OK
iv) Click and ~~it~~ Open Selenium IDE

POORNIMA

EXPERIMENT-9

Objective:- Perform a case study on JUnit testing and write a program for understanding how testing is actually performed in Java.

JUnit testing:- Junit is a unit testing framework for java programming language. It plays a crucial role test-driven development, and is a family of unit testing frameworks collectively known as xUnit.

JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented.

Unit Test Case:- A Unit Test case is a part of code, which ensures that another part of code (method) works as expected. To achieve the desired results quickly, a test framework is required. JUnit is a perfect unit test framework for java programming language.

Java Program.

```
package JavaTpoint.JunitExamples;
import java.util.ArrayList;
import java.util.List;
public class JunitTestCaseExample {
  private List<String> students = new ArrayList<String>();
  public void remove (String name){
    students.remove (name);
  }
  public void add (String name) {
    students.add (name);
  }
  public void removeAll () {
    students.clear ();
  }
```

```
    public int sizeOfStudent(){
        return students.size();
    }
}
```

TestJunitTestCase.java

```
package JavaTpoint.JunitExamples;
import static org.junit.Assert.assertEquals;
import org.junit.Test;
public class TestJunitTestCaseExample {
    JunitTestCaseExample obj = new JunitTestCaseExample();
    public void testAdd(){
        obj.add("Emma");
        obj.add("Ronan");
        obj.add("Antonio");
        obj.add("Paul");
        assertEquals("Adding 4 student to list", 4, obj.sizeOfStudent());
    }
    public void testSize(){
        obj.add("Emma");
        obj.add("Ronan");
        obj.add("Antonio");
        obj.add assertEquals("Checking size of List", 3, obj.sizeOfStudent());
    }
    public void testRemove(){
        obj.add("Antonio");
        obj.add("Paul");
        obj.remove("Paul");
```

POORNIMA

```
    assertEquals("Removing 1 student from list", 1, obj.sizeOfStudent());
}
public void removeAll() {
    obj.removeAll();
}
}
```

TestRunner.java.

```
package JavaTpoint.JunitExamples;
import org.junit.runner.Result;
import org.junit.runner.JUnitCore;
import org.junit.runner.notification.Failure;
public class TestRunner {
  public static void main(String[] args) {
  Result result = JUnitCore.runClasses(TestJunitTestCaseExample.class);

  for(Failure fail : result.getFailures()) {
    System.out.println(fail.toString());
  }
  System.out.println(result.wasSuccessful());
  }
}
```

Output :-

True.

POORNIMA

EXPERIMENT-10

Objective:- Write JUnit test case of a java program that used to find the maximum number for an array.

Program Code:-

```
package com.javatpoint.logic;
public class Calculation {
    public static int findMax(int arr[]){
        int max=0;
    for(int i=1; i<arr.length; i++){
        if(max<arr[i])
            max=arr[i];
        }
    return max;
} }
```

Test Code:-

```
package com.javatpoint.testcase;
import static org.junit.Assert.*;
import com.javatpoint.logic.*;
import org.junit.Test;
public class TestLogic {
    public void testFindMax(){
assertEquals(4, Calculation.findMax(new int[]{1,3,4,2}));
assertEquals(-1, Calculation.findMax(new int[]{-12,-1,-3,-4,-2}));
    }
}
```

Output:-

To run this right click on Testlogic class -> Run As -> JUnit Test

Output:- Assertion Error