COURSE OUTCOMES

Course: 8CS4-22: Software Testing and Validation Lab

On completion of this course:	
8CS4-22.1 (CO1)	Understand the process of applying tests to software and the fundamental components of a coverage analysis and unit testing.
8CS4-22.2 (CO2)	Describe & Calculate the mutation score for various programs using jumble testing tool.
8CS4-22.3 (CO3)	Apply the website performance measurement technique using JMeter and Selenium tool to perform Test sequences and validate testing.
8CS4-22.4 (CO4)	Understand the different software testing techniques and strategies and be able to apply specific(automated) unit testing method to the projects.

MAPPING OF CO WITH PO AND PSO

CO's	**PO's**												**PSO's**		
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	-	3	-	-	-	-	-	-	-	-	-	-	3	-	-
CO2	3	-	-	-	-	-	-	-	-	-	-	-	-	2	-
CO3	-	-	-	-	3	-	-	-	-	-	-	-	-	-	3
CO4	-	-	-	-	-	-	-	-	-	-	3	-	-	-	3

Mapping Levels: 1- Low, 2- Moderate, 3-Strong

Experiment 1:

Obj: Find the Coverage & Test Cases of the following program using JaButi Tool:

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

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

DESCRIPTION:

Code Coverage Analysis:

Code Coverage Analysis is the process of discovering code within a program that is not being exercised by test cases. This information can then be used to improve the test suite, either by adding tests or modifying existing tests to increase coverage.

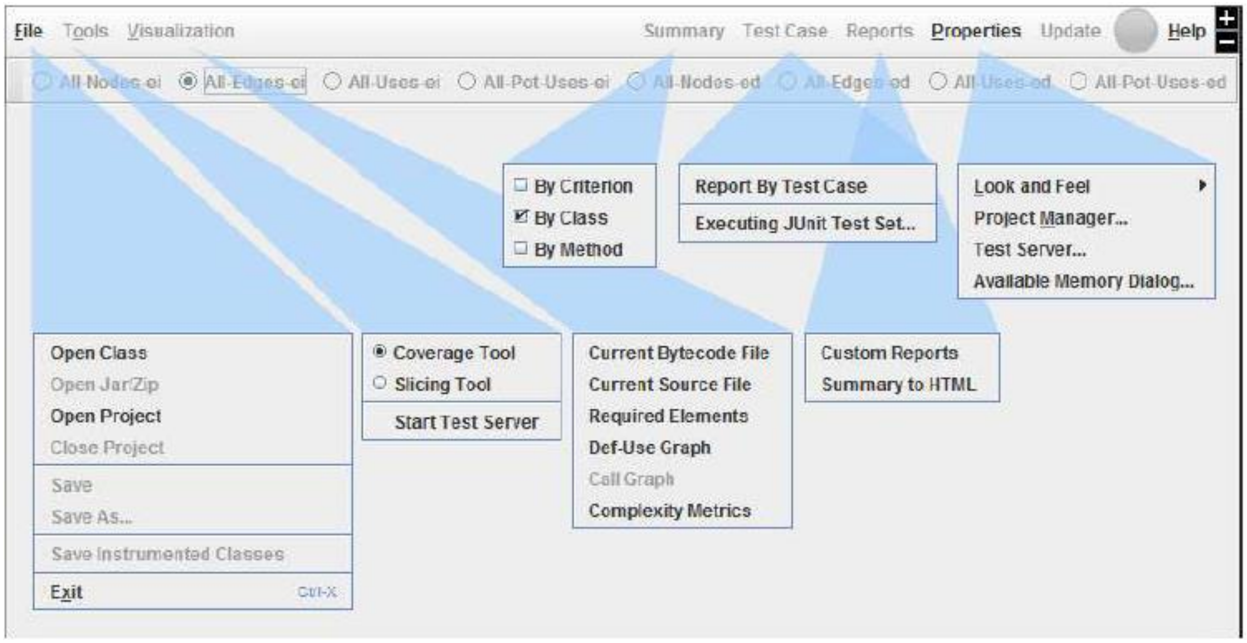
Code Coverage Analysis shines a light on the quality of your unit testing. It enables developers to quickly and easily improve the quality of their unit
tests which ultimately leads to improved quality of the software under development**.**

JaBUTi – Java Bytecode Understanding and Testing

JaBUTi is designed to work with Java bytecode such that no source code is required to perform its activities. It is composed by a coverage analysis tool, by a slicing tool, and by a complexity metric's measure tool. The coverage tool can be used to assess the quality of a given test set or to generate test set based on different control-flow and data-flow testing criteria.

Create a Testing Project:

In JaBUTi the testing activity requires the creation of a testing project. A testing project is characterized by a file storing the necessary information about (i) the base class file, (ii) the complete set of classes required by the base class, (iii) the set of classes to be instrumented (tested), and (iv) the set of classes that are not under testing. Additional information, such as the CLASSPATH environment variable necessary to run the base class is also stored in the project file, whose extension is .jbt. During the execution of any .class file that belongs to the set of classes under testing of a given project, dynamic control-flow information (execution trace) is collected and saved in a separate file that has the same name of the project file but with a different extension (.trc).



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

Program Code:

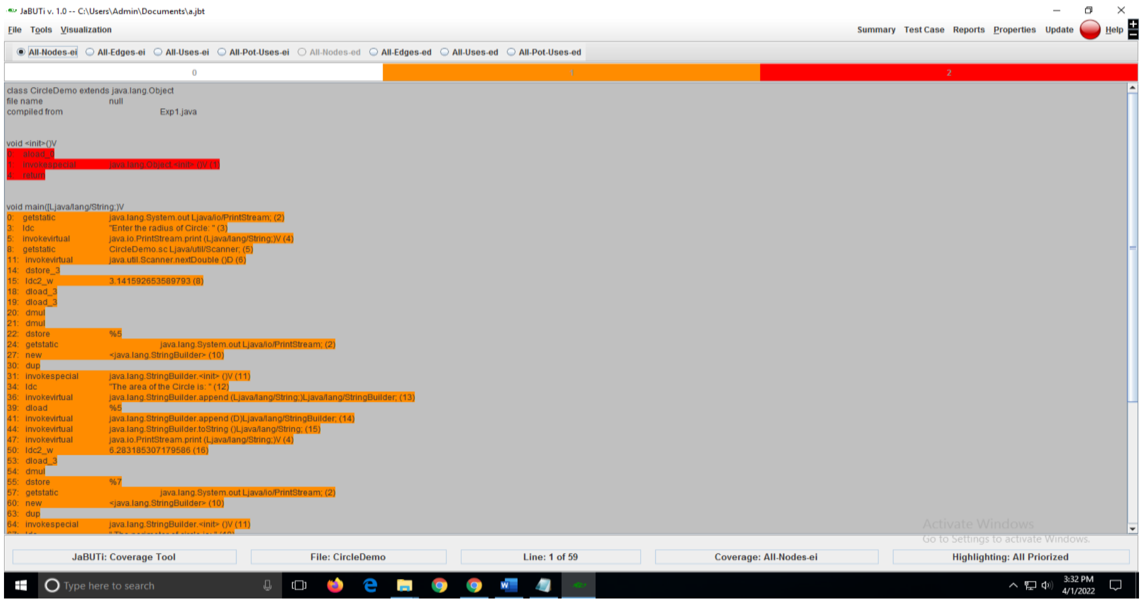
```
import java.util.Scanner;

class CircleDemo

{
 static Scanner sc= new Scanner (System.in);

 public static void main(String args[])
 {
    int a,b;
    System.out.print("Enter the radius of Circle: ");
            double radius = sc.nextDouble();
            double area = Math.PI * (radius * radius);
            System.out.print("The area of the Circle is: "+area);
            double perimeter =Math.PI *2 * radius;
            System.out.print(" The perimeter of circle is: " +perimeter);
 }
}
```

Output:

Enter the radius of Circle: 8

The area of the Circle is: 201.06192982974676

The perimeter of circle is: 50.26548245743669



JaBUTi v. 1.0 -- C:\Users\Admin\Documents\a.jbt

File Tools Visualization — Summary Test Case Reports Properties Update Help

(● All-Nodes-ei) ○ All-Edges-ei ○ All-Uses-ei ○ All-Pot-Uses-ei ○ All-Nodes-ed ○ All-Edges-ed ○ All-Uses-ed ○ All-Pot-Uses-ed

```
class CircleDemo extends java.lang.Object
file name                     null
compiled from                              Exp1.java


void <init>()V
0:   aload_0
1:   invokespecial            java.lang.Object.<init> ()V (1)
4:   return


void main([Ljava/lang/String;)V
0:   getstatic                java.lang.System.out Ljava/io/PrintStream; (2)
3:   ldc                      "Enter the radius of Circle: " (3)
5:   invokevirtual            java.io.PrintStream.print (Ljava/lang/String;)V (4)
8:   getstatic                CircleDemo.sc Ljava/util/Scanner; (5)
11:  invokevirtual            java.util.Scanner.nextDouble ()D (6)
14:  dstore_3
15:  ldc2_w                   3.141592653589793 (8)
18:  dload_3
19:  dload_3
20:  dmul
21:  dmul
22:  dstore                   %5
24:  getstatic                          java.lang.System.out Ljava/io/PrintStream; (2)
27:  new                      <java.lang.StringBuilder> (10)
30:  dup
31:  invokespecial            java.lang.StringBuilder.<init> ()V (11)
34:  ldc                      "The area of the Circle is: " (12)
36:  invokevirtual            java.lang.StringBuilder.append (Ljava/lang/String;)Ljava/lang/StringBuilder; (13)
39:  dload                    %5
41:  invokevirtual            java.lang.StringBuilder.append (D)Ljava/lang/StringBuilder; (14)
44:  invokevirtual            java.lang.StringBuilder.toString ()Ljava/lang/String; (15)
47:  invokevirtual            java.io.PrintStream.print (Ljava/lang/String;)V (4)
50:  ldc2_w                   6.283185307179586 (16)
53:  dload_3
54:  dmul
55:  dstore                   %7
57:  getstatic                          java.lang.System.out Ljava/io/PrintStream; (2)
60:  new                      <java.lang.StringBuilder> (10)
63:  dup
64:  invokespecial            java.lang.StringBuilder.<init> ()V (11)
```

JaBUTi: Coverage Tool | File: CircleDemo | Line: 1 of 59 | Coverage: All-Nodes-ei | Highlighting: All Priorized

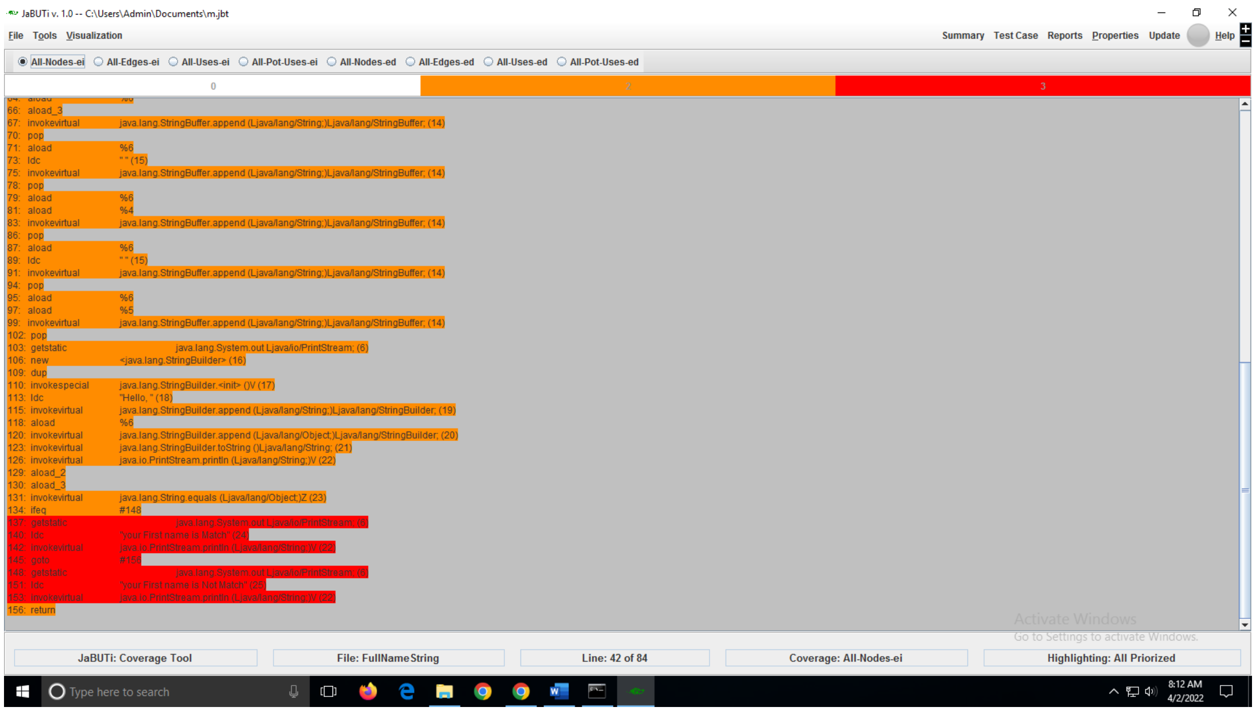
1.2: Write a program which read the first name and last name from console and matching with expected result by using JaBuTi.

Code:

```
import java.util.Scanner;

public class FullNameString {

 public static void main(String[] args) {

  Scanner input = new Scanner(System.in);
  String savedname= "Manish";
  System.out.print("Enter first name :: ");
  String firstName = input.nextLine();

  System.out.print("Enter middle name :: ");
  String middleName = input.nextLine();

  System.out.print("Enter surname :: ");
  String lastName = input.nextLine();

  // Here StringBuffer is used to combine 3 strings into single
  StringBuffer fullName = new StringBuffer();
  fullName.append(firstName);
  fullName.append(" ");   // For space between names
  fullName.append(middleName);
  fullName.append(" ");   // For space between names
  fullName.append(lastName);
  System.out.println("Hello, " + fullName);
  if (savedname.equals(firstName))
  {
  System.out.println("your First name is Match");
 }
 else {
    System.out.println("your First name is Not Match");
 }
}
}
```

Output:

```
Enter first name :: manish
Enter middle name :: ss
Enter surname :: choubisa
Hello, manish ss choubisa
your First name is Not Match
```


JaBUTi v. 1.0 -- C:\Users\Admin\Documents\m.jbt
File Tools Visualization
Summary Test Case Reports Properties Update Help
(● All-Nodes-ei) ○ All-Edges-ei ○ All-Uses-ei ○ All-Pot-Uses-ei ○ All-Nodes-ed ○ All-Edges-ed ○ All-Uses-ed ○ All-Pot-Uses-ed
0
2
3

```
66:  aload_3
67:  invokevirtual          java.lang.StringBuffer.append (Ljava/lang/String;)Ljava/lang/StringBuffer; (14)
70:  pop
71:  aload                  %6
73:  ldc                    " " (15)
75:  invokevirtual          java.lang.StringBuffer.append (Ljava/lang/String;)Ljava/lang/StringBuffer; (14)
78:  pop
79:  aload                  %6
81:  aload                  %4
83:  invokevirtual          java.lang.StringBuffer.append (Ljava/lang/String;)Ljava/lang/StringBuffer; (14)
86:  pop
87:  aload                  %6
89:  ldc                    " " (15)
91:  invokevirtual          java.lang.StringBuffer.append (Ljava/lang/String;)Ljava/lang/StringBuffer; (14)
94:  pop
95:  aload                  %6
97:  aload                  %5
99:  invokevirtual          java.lang.StringBuffer.append (Ljava/lang/String;)Ljava/lang/StringBuffer; (14)
102:  pop
103:  getstatic                      java.lang.System.out Ljava/io/PrintStream; (6)
106:  new                   <java.lang.StringBuilder> (16)
109:  dup
110:  invokespecial         java.lang.StringBuilder.<init> ()V (17)
113:  ldc                   "Hello, " (18)
115:  invokevirtual         java.lang.StringBuilder.append (Ljava/lang/String;)Ljava/lang/StringBuilder; (19)
118:  aload                 %6
120:  invokevirtual         java.lang.StringBuilder.append (Ljava/lang/Object;)Ljava/lang/StringBuilder; (20)
123:  invokevirtual         java.lang.StringBuilder.toString ()Ljava/lang/String; (21)
126:  invokevirtual         java.io.PrintStream.println (Ljava/lang/String;)V (22)
129:  aload_2
130:  aload_3
131:  invokevirtual         java.lang.String.equals (Ljava/lang/Object;)Z (23)
134:  ifeq                  #148
137:  getstatic                      java.lang.System.out Ljava/io/PrintStream; (6)
140:  ldc                   "your First name is Match" (24)
142:  invokevirtual         java.io.PrintStream.println (Ljava/lang/String;)V (22)
145:  goto                  #156
148:  getstatic                      java.lang.System.out Ljava/io/PrintStream; (6)
151:  ldc                   "your First name is Not Match" (25)
153:  invokevirtual         java.io.PrintStream.println (Ljava/lang/String;)V (22)
156:  return
```

JaBUTi: Coverage Tool | File: FullNameString | Line: 42 of 84 | Coverage: All-Nodes-ei | Highlighting: All Priorized