Laboratory Manual

For

Object-Oriented Programming Using Java

(IT 307)

B.Tech (IT)

SEM III

July 2022

Faculty of Technology
Dharmsinh Desai University
Nadiad.
www.ddu.ac.in

TABLE OF CONTENTS

LABWORK BEYOND CURRICULAM

Sample Experiment

1 AIM: Create a Generic Matrix class which implements matrix addition and multiplication common for all types of matrices. Create IntegerMatrix class for integer data types. Create TestMatrix class which tests the IntegerMatrix class.

2 TOOLS/APPARATUS: JDK 1.6.0 or above and notepad

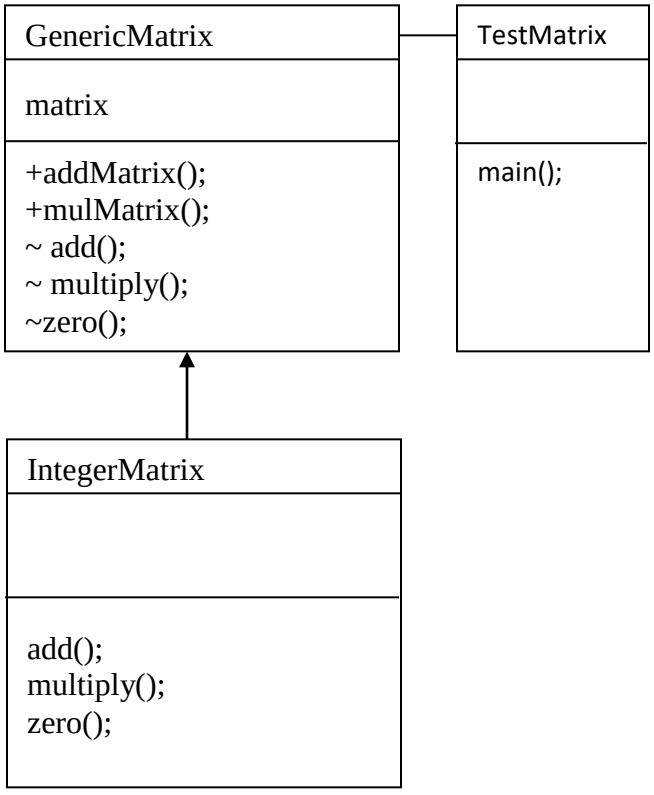
3 STANDARD PROCEDURES:

3.1 Analyzing the Problem:

Identify the noun from given application statement. Here the class name will be GenericMatrix, which is an abstract class. Another class is IntegerMatrix which inherits GenericMatrix for integer data. TestMatrix class is used to tests the IntegerMatrix class.

After declaring the class name identify the variables and methods. There will two methods i.e. addMatrix() and mulMatrix().

3.2 Designing the Solution: Class Diagram



3.3 Implementing the Solution

3.3.1 Writing Source Code

```
import java.lang.Integer;
import java.util.Scanner;
abstract class GenericMatrix
{
	private Object matrix[][];
	public GenericMatrix(Object matrix[][])
	{
		this.matrix=matrix;
	}
	public Object[][] addmatrix(Object matrix[][])
	{
		Object result[][]=new Object[matrix.length][matrix[0].length];

	if((this.matrix.length!=matrix.length)||(this.matrix[0].length!=matrix[
0].length))
		{
			System.out.println("Matrixs are not same size.......");
			System.exit(0);
		}

			for(int i=0;i<result.length;i++)
			{
				for(int j=0;j<result[0].length;j++)
				{

	result[i][j]=add(this.matrix[i][j],matrix[i][j]);
				}
			}
			return result;
	}
	public Object[][] multiplymatrix(Object matrix[][])
	{
		Object result[][]=new
Object[this.matrix.length][matrix[0].length];
		if(this.matrix[0].length!=matrix.length)
		{
			System.out.println("Bound Error........");
			System.exit(0);
		}

			for(int i=0;i<result.length;i++)
			{
				for(int j=0;j<result[0].length;j++)
				{
					result[i][j]=zero();
					for(int k=0;k<this.matrix[0].length;k++)
					{
```

```
      result[i][j]=add(result[i][j],multiply(this.matrix[i][k],matrix[k][j]))
;
                                    }
                              }
                        }
                        return result;


      }
      public abstract Object add(Object o1,Object o2);
      public abstract Object multiply(Object o1,Object O2);
      public abstract Object zero();
}
class IntegerMatrix extends GenericMatrix
{
      public IntegerMatrix(Integer[][] m)
      {
            super(m);
      }
      public Object add(Object o1, Object o2)
      {
            Integer i1 = (Integer)o1;
            Integer i2 = (Integer)o2;
            return new Integer(i1.intValue() + i2.intValue());
      }
      public Object multiply(Object o1, Object o2)
      {
            Integer i1 = (Integer)o1;
            Integer i2 = (Integer)o2;
            return new Integer(i1.intValue() * i2.intValue());
      }
      public Object zero()
      {
            return new Integer(0);
      }
}

public class Testmatrix
{

      public static void main(String args[])
      {
            Integer m1[][]=new Integer[4][4];
            Integer m2[][]=new Integer[4][4];
            Scanner s=new Scanner(System.in);
            System.out.println("m1 : ");
            System.out.println();
            for(int i=0;i<m1.length;i++)
            {
                  for(int j=0;j<m1[0].length;j++)
                  {
```

```
                        m1[i][j]=s.nextInt();
                }
        }
        System.out.println("m2 : ");
          System.out.println();
        for(int i=0;i<m1.length;i++)
        {
                for(int j=0;j<m1[0].length;j++)
                {
                        m2[i][j]=s.nextInt();
                }
        }
        IntegerMatrix im1=new IntegerMatrix(m1);
        Object m3[][]=im1.addmatrix(m2);
        Object m4[][]=im1.multiplymatrix(m2);
        System.out.println("Addition  :   ");
        for(int i=0;i<m3.length;i++)
        {
                for(int j=0;j<m3[0].length;j++)
                {
                        System.out.print((Integer)m3[i][j]+" ");

                        //m1[i][j]=new Integer(i);
                        //m2[i][j]=new Integer(i+j);
                }
                System.out.println();
        }
        System.out.println("Mutiplication :  ");
        System.out.println();
        for(int i=0;i<m4.length;i++)
        {
                for(int j=0;j<m4[0].length;j++)
                {
                        System.out.print((Integer)m4[i][j]+" ");

                        //m1[i][j]=new Integer(i);
                        //m2[i][j]=new Integer(i+j);
                }
                System.out.println();
        }
    }
}
```

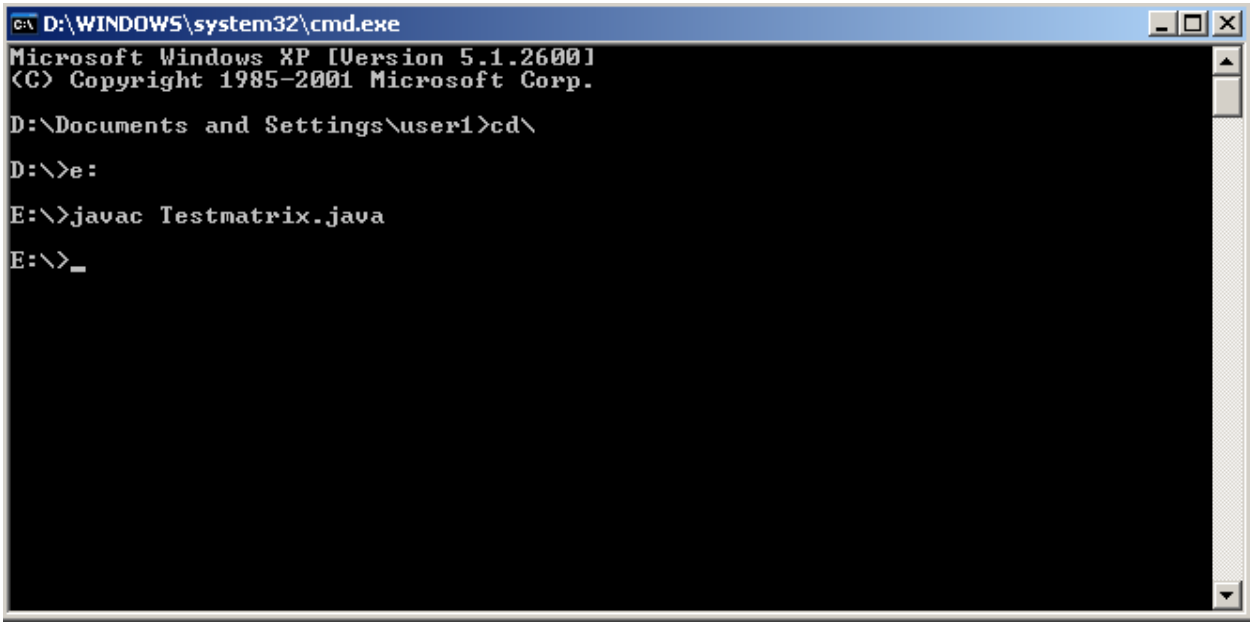
Environment Variable Setup:

Set the environment variables (path and classpath)

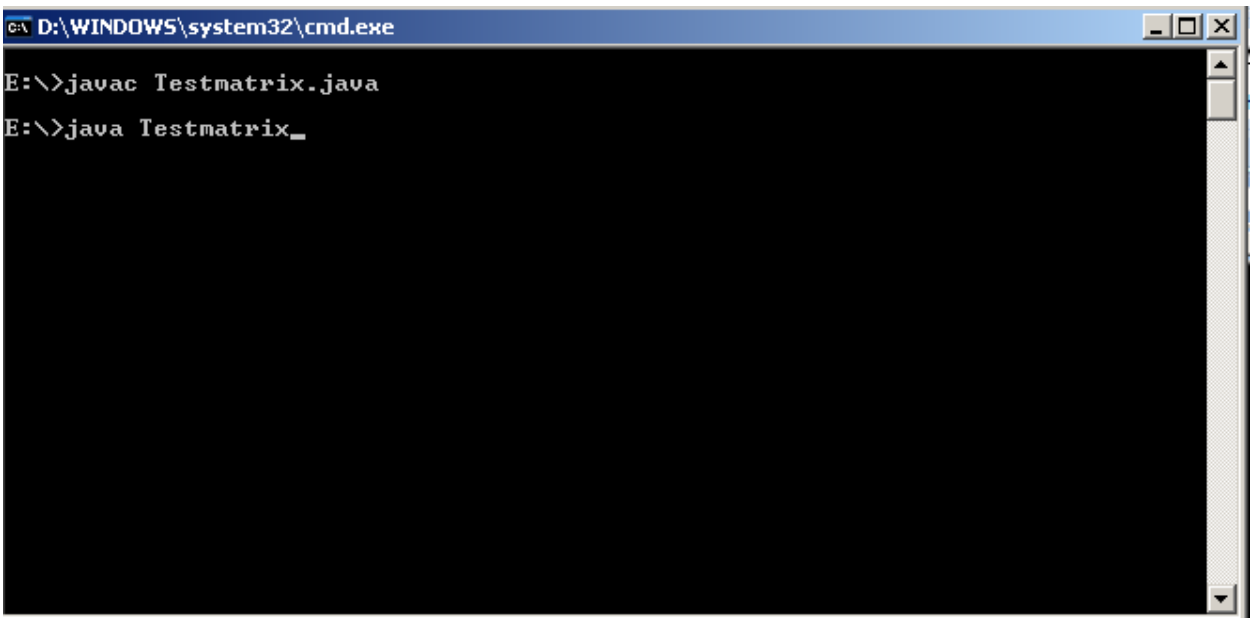
Set path=.;c:\jdk1.6.0\java\bin;
Set classpath=.;c:\jdk1.6.0\java\lib;

3.3.2 Compilation /Running and Debugging the Solution

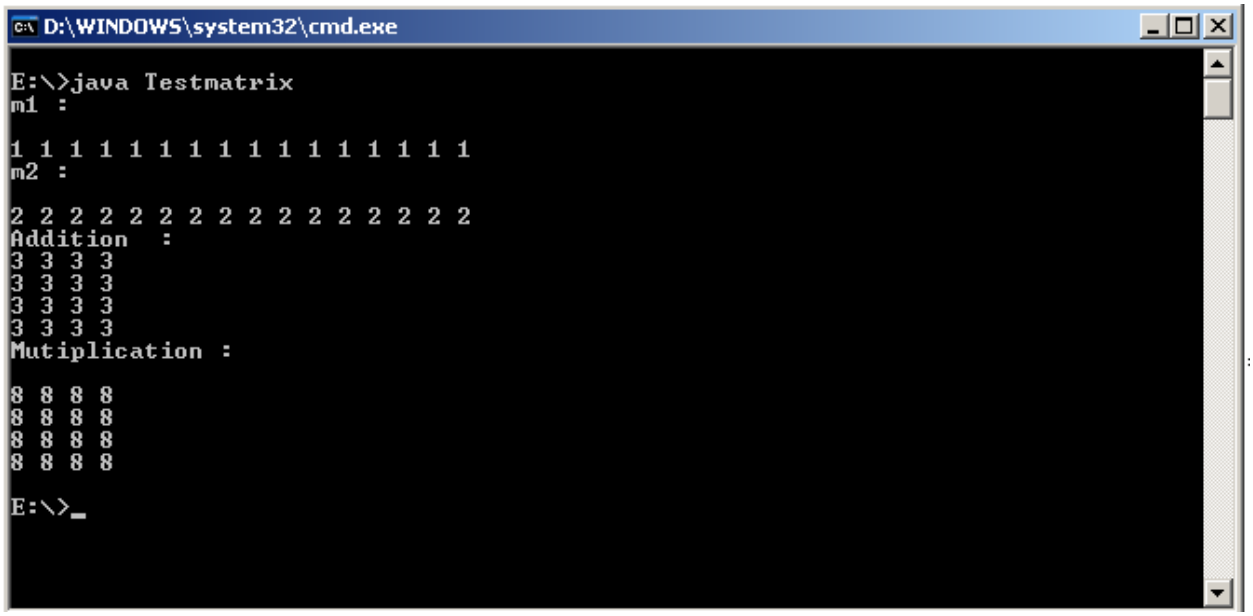
To compile a java program use javac command at the command prompt.



To run a java program use java command at the command prompt.



3.4 Testing the Solution



The above program will work for integer data types only.

Here two dimensional array is used of size 4*4. If user enters beyond the size of the array then exception will be generated.

Required Software/ Software Tool

Java Development Kit (JDK)
Install available current version of JDK.
Editor (Notepad)

Procedure to establish the environment to develop java application by setting two environment variables i.e. PATH and CLASSPATH.

Step 1: To use all executable commands (javac, java and appletviewer) from any directory on the command prompt. Setting of PATH environment variable is required. To do this edit C:\autoexec.bat file. Write following code after last path setting of path variable if it already exists.

; C:\jdkx\bin;

If no setting of path variable is exists then write following code.

PATH=%PATH%; C:\jdkx\bin;

➔ Where x represents the version number. Eg. 1.6.0 or 1.6.2

Step 2: Set CLASSPATH environment variable to access java library classes. To set it write the following code in C:\autoexec.bat file.

CLASSPATH=%CLASSPATH%; C:\jdkx\lib;

COMMON PROCEDURE:

Step 1: For the given problem statement design
Flowchart/Algorithm/Logic

Step 2: Define class name(s), find out appropriate attributes and create the methods which will show the flow of the program.

Step 3: Write Java code in a file having extension .java. While saving code in a file consider rules for giving name to file.

To write a code or to edit a code any Text Editor will work. Use editor that is available to you (Notepad).

Step 4: Compile java source code using JAVA compiler **javac.** For each class one class file (byte code) is generated, which is stored in a file having named that of class with extension .class.

Step 5: Using JAVA interpreter run/interpret the byte code. To run the byte code the interpreter **java** is used.

OR

Step 5: To design an Applet, java source code requires of applet tag (HTML coding) as comment or separate .htm or .html file should be created to view an Applet.

Step 5a: Use **appletviewer** or any java compatible web browser (Internet explorer) to view and to test an applet.

Step 6: Test your program using sample input and write down output.

EXPERIMENT – 1

Aim: Write the programs using the concept of nested for loops and recursion.

- Generate following patterns:

- Write a program that prints Fibonacci series.
- Write a program that finds out sum of digits of a number.
- Write a program that reverses a number.

Procedure:

- To generate patterns use nested for loops and apply your logic. Provide your logic then compile your program and run your program.
- Fibonacci series is 0 1 1 3 5……so on. Take the number from user using Integer. parseInt(args[0]). Provide your logic and print this series.
- To reverse any number, first take the number from user the function specified in above description. Write the logic and display the reversed number.

EXPERIMENT– 2

Aim: Write the programs using the concept of command line argument.

- Write a program that implements a command line calculator.
- Write a program to implement user interactive calculator.
- Write a program to print all combinations of four digit number.
  A four digit number is generated using only four digits {1, 2, 3, 4} and the number has second digit greater than first digit and fourth digit is less than third digit.
  Case 1: Duplication of digit is allowed.
  Case 2: Duplication of digit is not allowed.
- Write a program that prints multiplication table in a matrix format from the number 1 to 10.
- Write a program that finds out factorial of a number.
- Write a program that finds out n^{th} Fibonacci number.

Procedure:

- To perform any program using command line argument, use the argument provided into the main(). Compile your program using javac and then at the runtime you have to provide the arguments i.e. java filename argument1 argument2 ….
- For user interactive program, ask user to provide the value of the variable using parseInt() function for integer variables. Provide your logic then compile your program and run your program.
- To find out factorial of a number either you can use recursive function or by using normal function. Provide your logic then compile your program and run your program.

EXPERIMENT– 3

Aim: Write the programs using the concept of arrays and StringBuffer class.

- Write a program to merge two arrays in third array. Also sort the third array in ascending order.
- Write a program to add and to multiply two int matrices.
- Write a program that reads email address from user and check whether email address is valid or not and separate out email id from email server name.
  If input is abc@xyz.com
  Output: It is valid address
  Email id: abc
  Email server address: xyz.com
- Write a program that converts all characters of a string in capital letters. (Use StringBuffer to store a string). Don't use inbuilt function.

Procedure:

- To merge two arrays, create three arrays, store the data in two separate arrays. Merge the two arrays into third array by providing your logic, also sort that array. Compile your program and run your program.
- To represent matrics you have to create the arrays. Provide your logic then compile your program and run your program.
- To separate out the email id, use StringBuffer class. You can use the constructor of this class. Provide your logic then compile your program and run your program.

EXPERIMENT – 4

Aim: Write the programs using the concept of Generic class, Inheritance, Interface and Package.

- Write a program to add and to multiply two matrices using Generic class concept.
- Create an abstract class Shape and derived classes Rectangle and Circle from Shape class. Implement abstract method of shape class in Rectangle and Circle class.
  Shape class contains: origin (x,y) as data member.
  display() and area() as abstract methods.
  Circle class contains: radius as data member.
  Rectangle class contains: length and width
  (Use Inheritance, overloading and overriding concept)
- Write a program to show an implementation of Package.
- Write a program to show an implementation of Interface.

Procedure:

- To perform the program using generic class concept, you have to create the variable of type Object class in separate class. To perform the operation on matrices of type integer then you have to create a separate class to convert the Object type variable into Integer type variable. In main class create main() which will create the objects of the above class. Provide your logic then compile your program and run your program.
- To create an abstract class use the concept of inheritance.

EXPERIMENT – 5

Aim: Write a program that uses the concept of unit testing using junit.
Tools/Apparatus: Eclipse

- Write a program that will check four test cases of adding student, check number of students currently available (i.e. size of student), remove one student, and remove all student at a time.

Procedure:

- We have created three java classes, i.e., **JunitTestCaseExample.java, TestJunitTestCaseExample.java**,and **TestRunner**.java.
- In the **JunitTestCaseExample.java** class, we created the code which we want to test. In this class, we create a list of names and four methods to add an element, remove an element, get the list's size, and remove all elements from the list.
- In the **TestJunitTestCaseExample.java**, we write the test cases for the JunitTestCaseEample.java class. We create an object of the **JunitTestCaseExample.java** class, and by using its object, we will test all its methods. We use the **assertEquals()** method to check the actual result with the expected output.
- We create the TestRunner.java class to execute the test cases. It contains the main() method in which we run the TestJunitTestCaseExample.java class using the runClasses() method of the JUnitCore. In this method, we pass the class file of the TestJunitTestCaseExample.java. The result returned by the runClasses() method will store into the result variable of type Result.
- Now, we will run the TestRunner.java class to execute the test cases. We will get the output true when the test cases are passed and false when the test cases are not passed.

EXPERIMENT – 6

Aim: Write the program which creates the Frame and implements MouseListener.

- Write a program to display mouse position when the mouse is pressed.
- Write a program to display multiplication table in a Frame.

Procedure:

- For mouse position you have to implement MouseListener interface and use different methods of that. First get the x and y coordinates when mouse is pressed and then display it.

EXPERIMENT – 7

Aim: Implementing a GUI based calculator application and drawing different figures on a Canvas.

- Implement a GUI based calculator application. It has two TextFields for two input numbers, one TextField for result and four Buttons named Add, Sub, Mul and Div for addition, subtraction, multiplication and division respectively.
- Write a program to draw various figures on a Canvas. The user selects figure from a CheckboxGroup, the selected figure is then displayed in the Canvas.

Procedure:

- For calculator, first create objects of two textfields for input and add them into the frame. Then create one object of textfield for result and four buttons for four arithmetic operations: addition, subtraction, multiplication and division. Add them into the frame. Perform particular operation add that code in the click event of that particular button.
- Create checkboxes for different figures and them into the CheckboxGroup. Create one canvas to draw the figures and canvas into the frame.

EXPERIMENT – 8

Aim: Write an application to simulate traffic lights and calculator using GridbagLayout.

- Write an application to simulate traffic lights. The program lets the user select one of the three lights red, yellow and green. Upon selecting a menu item, the light is turned on and there is only one light on at a time.
- Using GridBagLayout to lay out, implement Windows' 98 calculator like application. Implement four basic arithmetic operations.

Procedure:

- Take three checkboxes for three lights red, yellow and green. Add them into the CheckboxGroup. For turning the light on, make that particular light visible by making it true.
- Take four buttons for four different arithmetic operations and one textfield for the result. Use the GridBagLayout.

EXPERIMENT – 9

Aim: Write a program that uses the concept of Exception Handling.

- Write a program to meet following requirements:
  - Create an array with 100 elements that are randomly chosen.
  - Create a TextField to enter an array index and another TextField to display the array element at the specified index.
  - Create a show Button to cause the array element to be displayed. If the specified index is out bound, display message "Out of Bound".

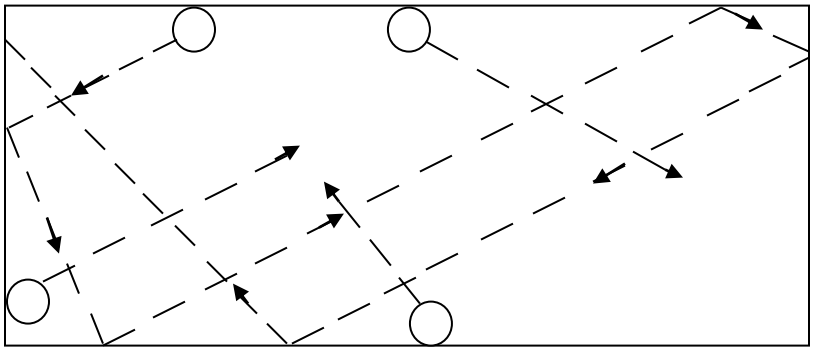
Procedure:

- First create array and store random values in that. Create two textfields. From one textfield get the value of array index. Using that index value find out the element stored at that index and display it in the another textfield. To display the message "out of bound" use the exception class.

EXPERIMENT – 10

Aim: Write the programs that uses the concept of Threads.

- Write an application to display moving banner. Enter message to be moved, color and size through three different TextFields. Select direction (left or right) using Choice. On pressing start Button message should start moving.
- Write an applet that shows four moving objects (balls) as shown below:



Four directions (↗ ↘ ↖ ↙) are used. On touching boundary, the object (ball) changes its direction and start moving in that changed direction.

Procedure:

- For moving banner you have to use the concept of thread. You have to change the positions of x and y coordinates accordingly. Put that code into the click event of the Start button.
- For moving balls you have to use four threads. In that you have to change the directions according to the size of the frame.

EXPERIMENT – 11

Aim: Write a program that uses the concept of File I/O. Write a program that uses the concept of socket programming.

- Write a program to implement Notepad like application.
- Write a program that will count number of characters, words and lines in a file. The name should be passes as command line argument
- Implement multiple client chat application using socket programming

Procedure:

- For Notepad like application create one menubar which contains different menus for opening a file, creating a new file, deleting a file, to copy file etc. Create one file and one frame. When user selects particular menu item like opening a file then file contents should be displayed in the frame.
- For socket programming, you have to create two programs each for client and server. To implement multiple client chat application you have use multi threading concept. For each client individual thread is created. First compile and run your server. Server will be on and then open another command prompt to compile and run the client.

EXPERIMENT – 12

Aim: Write an application to play the tic-tac-toe game.

- The game lets two players alternately enter X and O in the cells. A winner is declared if a player occupies three cells in a line vertically, horizontally or diagonally. A tie is declared if all cells are occupied without a winner.

Procedure:

- To create an application, you have to create Frame. Create the GUI to play the tic-tac-toe game. Put different buttons labeling 0 and *. You have to implement ActionListener interface and override actionPerformed() method and appropriate action should be performed.

EXPERIMENT – 13

Aim: Write a program to display student exam scores.

- The program prompts the user to enter the user's first name and the password in a dialog box. Upon receiving the correct user name and password, the program displays the student's full name and the exam score.

Procedure:

- Create user defined dialog box by inheriting the Dialog class. Take two types of information i.e. user name and password. Verify the login and if it is correct then display student's full name and exam score either in different text field or in text area.