THE NEOTIA UNIVERSITY
LAB MANUAL
ON
OBJECT ORIENTED PROGRAMMING
(JAVA PROGRAMMING)

ExpNo: 1.a

Roots of a Quadratic Equation

Aim: To write a java program that prints all real solutions to quadratic equation $ax^2+bx+c= 0$

Description: Write main method in a class named Quadratic. a, b and c are variables of type double for which we will assign some values. Variable d is used to store discriminant value (i.e., b*b – 4*a*c). r1, r2 are two double variables used to store two roots when discriminant value that is d is greater than zero. When *d* is greater than zero, the two roots are real. When *d* is equal to zero, the two roots are equal. When *d* is less than zero, the two roots are imaginary.

Program:

```
class Quadratic
{       public static void main(String args[])
        {
                double a=5,b=10,c=2,r1,r2,d;
                d=(b*b)-(4*a*c);
                if(d==0)
                {       System.out.println("Roots are real and equal");
                        System.out.println("The roots are : " + (-b/(2*a) ) );
                }
                else if(d<0)
                        System.out.println("The roots are imaginary");
                else if(d>0)
                {       r1=-b+(Math.sqrt(d))/(2*a);
                        r2=-b-(Math.sqrt(d))/(2*a);
                        System.out.println("root1=" + r1);
                        System.out.println("root2=" + r2);
                }
        }
}
```

ExpNo: 1.b **Date:**

Fibnacci element (Recursion)

Aim: To write a java program that uses recursive procedure to print the nth value in the Fibonacci sequence.

Description: The Fibonacci sequence (denoted by f0, f1, f2 …) is as follows: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 … That is, f0 = 1 and f1 = 1 and each succeeding term is the sum of the two preceding terms. Typically, recursion is more elegant and requires fewer variables to make the same calculations. Recursion takes care of its book-keeping by stacking arguments and variables for each method call. This stacking of arguments, while invisible to the user, is still costly in time and space. Fibonacci sequence is recursively defined by f0 = 1, f1 = 1, fi+1 = fi + fi-1 for i = 1, 2 …

Program:

```
class Demo
{
        int fib(int n)
        {       if(n==1)
                        return (1);
                else if(n==2)
                        return (1);
                                        else    return (fib(n-1)+fib(n-2));

                }
        }

class Fib1
{       public static void main(String args[])
        {
                Demo ob=new Demo();
                System.out.println("The 10th fibonacci element is " + ob.fib(10));
        }
}
```

ExpNo: 1.c)

Printing nth Fibonacci element

(Non-Recursion)

Aim: To write a java program that uses non-recursive procedure to print the nth value in the Fibonacci sequence.

Description: The Fibonacci sequence (denoted by f0, f1, f2 …) is as follows: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 … That is, f0 = 1 and f1 = 1 and each succeeding term is the sum of the two preceding terms. Typically, recursion is more elegant and requires fewer variables to make the same calculations but stacking of arguments, while invisible to the user, is still costly in time and space. We can implement this by using iterative procedure also without Recursion.

Program:

```
class Demo
{
        int fib(int n)
        {
                int a=1, b=1, k=3, c=0;
                if( n== 1)
                        return(1);
                else if( n==2 )
                        return(1);
                else
                {                                       while(k <= n)
                                                        {
                                c = a + b;
                                a = b;
                                b = c;
                                k++;
                        }
                        return (c);
                }
        }
}
class Fib2
{
        public static void main(String args[])

         {      Demo ob=new Demo();
                System.out.println("The 10th fibonacci element is " + ob.fib(10));
        }
}
```

ExpNo: 2. a

Prime Numbers

Aim: To write a java program that prompts the user for an integer and then prints out all prime numbers up to that integer.

Description: A prime number is a positive integer that is exactly divisible only by 1 and itself. The first few prime numbers are: 2 3 5 7 11 13 17 23 29 31 37 … To do this; let us consider a number 13. The definition of prime number suggests that to determine whether or not 13 is prime, we need to divide it in turn by the set of numbers 2, 3, 4, 5 …12. If any of these numbers divide into 13 without remainder we will know it cannot be prime number.That is, we can test for mod operation with numbers, from 2 to n-1.

Program:

```
import java.io.*;
class Prime
{       public static void main(String args[])throws IOException
        {       int i,j,a=0,n;
                BufferedReader br= new BufferedReader(new InputStreamReader (
                                                                System.in) );
                System.out.println("Enter range : " );
                n = Integer.parseInt(br.readLine() );
                for(i=2;i<=n;i++)
                {
                        a=0;
                        for(j=2;j<i;j++)
                        {
                                if(i%j==0)
                                        a=1;
                        }
                        if(a==0)
                                System.out.print(i + "\t");
                }
        }
}
```

ExpNo: 2. b)

Multiplication of Two Matrices

Aim: To write a java program to multiply two given matrices.

Description: If the order of matrix *A* is r1 x c1 and of matrix *B* is r2 x c2 (number of columns of *A* = number of rows of *B* = c1=r2), then the order of matrix *C* is r1 x c2, where *C* = *A* x *B* otherwise matrix multiplication is not possible. First accept number of rows and columns of matrix A into r1, c1 then accept number of rows and columns of matrix B into r2, c2. If c1 is not equal to r2 then display a message matrix multiplication is not available otherwise accept two matrices into two arrays and perform the multiplication operation and store the result in another array and display the same as result.

Program:

```
import java.io.*;
class MatMul
{
        public static void main(String args[])throws IOException
        {
                BufferedReader br=new BufferedReader(new InputStreamReader(
                                                                System.in));
                System.out.println("Enter the row and columns for matrix A:");
                int r1=Integer.parseInt(br.readLine());
                int c1=Integer.parseInt(br.readLine());
                System.out.println("Enter the row and columns for matrix B:");
                int r2=Integer.parseInt(br.readLine());
                int c2=Integer.parseInt(br.readLine());
                int i, j, k;
                int a[ ][ ]=new int[r1][c1];
                int b[ ][ ]=new int[r2][c2];
                int c[ ][ ]=new int[r1][c2];
                if(c1!=r2)
                        System.out.println("Matrix multiplication is not possible");
                else
                {                               System.out.println("Enter the elements for
                                                matrixA:"); for(i=0;i<r1;i++)
                                                {       for(j=0;j<c1;j++)
```

```
                {
                        a[i][j]=Integer.parseInt(br.readLine());
                }
        }
        System.out.println("Enter the elements for matrix B:");
        for(i=0;i<r2;i++)
        {       for(j=0;j<c2;j++)
                {
                        b[i][j]=Integer.parseInt(br.readLine());
                }
        }
        //matrix multiplication
        for(i=0;i<r1;i++)
        {       for(j=0;j<c2;j++)
                {       c[i][j]=0;
                        for(k=0;k<c1;k++)
                        {
                                c[i][j]=c[i][j]+a[i][k]*b[k][j];
                        }
                }
        }
    }
    System.out.println("Resultant matrix is:");
    for(i=0;i<r1;i++)
    {
        System.out.println()
        ; for(j=0;j<c2;j++)
        {
            System.out.print(c[i][j]+"\t");
        }
    }
  }
}
```

ExpNo: 2. c) **StringTokenizer**

Aim: To write a java program that reads a line of integers and then displays each integer and the sum of all integers.

Description: The StringTokenizer class allows an application to break a string into tokens. The discrete parts of a string are called *tokens*. Therefore, given an input string, we can enumerate the individual tokens contained in it using StringTokenizer. The default delimiters are whitespace characters. By parsing individual tokens into integers we will get integers and we can add those integers to find the sum.

Program:

```
import java.util.*;
import java.io.*;
class SumInt
{       public static void main(String args[ ]) throws IOException
        {       BufferedReader br = new BufferedReader( new InputStreamReader(
                                                                System.in) );
                System.out.println("Enter a line of integers : ");
                String str= br.readLine();
                StringTokenizer sc=new StringTokenizer(str," ");
                System.out.println("The integers are:");
                int total=0;
                while(sc.hasMoreTokens())
                {
                        int k=Integer.parseInt(sc.nextToken());
                        System.out.println(k);
                        total=total+k;
                }
                System.out.println("Sum of all integers:"+total);
        }
}
```

ExpNo:3.a

String Palindrome

Aim: To write a java program that checks whether a given string is a palindrome or not.

Description: First accept a string from the keyboard using BufferedReader class of java.io package. Create StringBuffer class object and pass the String as argument to that object. Call reverse() method of StringBuffer object such that the string in that object will get reverse. After converting that StringBuffer object into String type store it in another String variable. Now by making use of equals method of String class compare two strings if both are equal then given string is palindrome otherwise it is not a palindrome.

Program:

```
import java.io.*;
class Palindrome
{
        public static void main(String args[]) throws IOException
        {
                BufferedReader br = new BufferedReader(new InputStreamReader(
                                                                        System.in));
                System.out.print("Enter the string : ");
                String s1=br.readLine();
                StringBuffer sb = new StringBuffer( s1 );
                sb.reverse();
                String s2 = sb.toString();
                System.out.println("The reversed String is : " + s2);
                if( s1.equals(s2) )
                        System.out.println("Given String is palindrome");
                                        else    System.out.println("Given String is not a
                                                palindrome");
                }
        }
```

ExpNo: 3.b

Sorting of Strings

Aim: To write a java program to sort a list of names in ascending order.

Description: First read number of strings. Accept number of strings into an array. Compare first string with remaining strings using String class compareTo () method. compareTo () method compares the given string in dictionary fashion. If the first string is big then by taking a temporary variable swap the two strings, continue this process till the end of the strings. The given strings will be in ascending sorted order.

Program:

```
import java.io.*;
class NameSort
{       public static void main(String args[ ]) throws IOException
        {       BufferedReader br = new BufferedReader(new InputStreamReader(
                                                                System.in));
                System.out.println("Enter how many names you want");
                int n=Integer.parseInt(br.readLine() );
                String a[ ]=new String[n];
                int i,j;
                System.out.println("Enter " + n + " names : ");
                for(i=0;i<n;i++)
                        a[i]=br.readLine();
                for(i=0;i<n;i++)
                {       for(j=i+1;j<n ;j++)
                        {       if(a[i].compareTo(a[j])>0)
                                {       String temp=a[i];
                                        a[i]=a[j];
                                        a[j]=temp;
                                }
                        }
                }
                System.out.println("After sorting the names are");
                for(i=0;i<n;i++)
                        System.out.println(a[i]);
        }
```

ExpNo: 3. c)

Frequency count

Aim: To write a java program that makes frequency count of letters in a given text.

Description: Set is an interface which does not allow any duplication in java.util package. HashSet is an implementation class for Set interface. First of all store the different types of characters in the entered line using charAt() method of String class into HashSet object. By using iterator interface get the individual characters one by one and compare that character with each and every character in that line of text using charAt() method of String class. If it matches then increment character count and display the same.

Program:

```
import java.util.*;
import java.io.*;
class CountLetters
{       public static void main(String args[]) throws IOException
        {       BufferedReader br = new BufferedReader(new InputStreamReader(
                                                                System.in) );
                HashSet<String> hs = new HashSet<String>();
                System.out.println("Enter a line of text : " );
                String s1 = br.readLine();
                for(int i=0;i<s1.length();i++)
                {       hs.add(s1.charAt(i)+"" );
                }
                Iterator it = hs.iterator();
                while (it.hasNext() )
                {       int count = 0;
                        String a = it.next().toString();
                        for(int i=0;i<s1.length();i++)
                        {       String b = s1.charAt(i) + "" ;
                                if ( a.equals(b) )
                                        count++;
                        }
                        System.out.println(a + " count is : " + count);
                }
        }

}
```

4. a) **Displaying File Properties** **Date:**

Aim: To write a java program that reads a file name from the user then displays information about whether the file exists, whether the file is readable, whether the file is writable, type of file and the length of the file in bytes.

Description:A File object is used to get information or manipulate the information associated with a disk file; such as the permissions, time, date, directory path, is it readable or writable, file size, and so on. File class defines many methods that obtain the standard properties of a File object. For example, getName () returns the file name, and exists() returns true if the file exists, false if it does not.

Program:

```
import java.io.*;
class FileProp
{
        public static void main(String args[]) throws IOException
        {
                BufferedReader br = new BufferedReader(new InputStreamReader (
                                                                        System.in));
                System.out.println("Enter file name :");
                String fname = br.readLine();
                File f = new File (fname);
                System.out.println ("File name: " + f.getName ());
                System.out.println ("Path:"+ f.getPath ());
                System.out.println ("Absolute Path:"+ f.getAbsolutePath ());
                System.out.println ("Parent:"+ f.getParent ());
                System.out.println ("Exists:"+ f.exists ());
                if ( f.exists() )
                {
                        System.out.println ("Is writable: "+ f.canWrite ());
                        System.out.println ("Is readable: "+ f.canRead ());
                        System.out.println ("Is executable: "+ f.canExecute ());
                        System.out.println ("Is directory: "+ f.isDirectory());
                        System.out.println ("File size in bytes: "+ f.length());


                }
        }
}
```

ExpNo: 4. b)

Displaying contents of a file

Aim: To write a java program that reads a file and displays the file on the screen, with a line number before each line.

Description: The FileInputStream class creates an InputStream that we can use to read bytes from a file. When a FileInputStream is created, it is opened for reading. To display a line number for each line at the time of reading, declare a variable of type int and initialize it with 1 and increment the variable by 1 whenever a new line character is encountered.

Program:

```
import java.io.*;
class FileRead
{
        public static void main(String args[]) throws IOException
        {
                int ch,ctr=1;
                String fname;
                BufferedReader br = new BufferedReader(new InputStreamReader (
                                                                        System.in));
                System.out.print("Enter a file name: ");
                fname=br.readLine();
                FileInputStream fin =new FileInputStream(fname);
                System.out.print(ctr+" ");
                while((ch=fin.read())!=-1)
                {
                        System.out.print((char)ch);
                        if(ch=='\n')
                        {
                                ctr++;
                                System.out.print(ctr+" ");
                        }
                }
                fin.close();
        }
```

ExpNo: 4. c)

Displaying number of Characters, Lines & Words in a file

Aim: To write a java program that displays the number of characters, lines and words in a text file.

Description: The FileInputStream class creates an InputStream that we can use to read bytes from a file.When a FileInputStream is created, it is opened for reading. Take three variables of type int and initialize it with zero for counting number of characters, words and lines. Read the contents from file character by character and increment character count by 1 for each character. Whenever a new line character is encountered then increment line count by 1. Whenever a space is encountered then increment word count by 1.

Program:

```
import java.io.*;
class FileStat
{       public static void main(String args[]) throws IOException
        {       int pre=' ' , ch , ctr=0 , L=0 , w=1;
                String fname;
                BufferedReader br=new BufferedReader(new InputStreamReader (
                                                                System.in));
                System.out.print("Enter a file name: ");
                fname=br.readLine();
                FileInputStream fin = new FileInputStream(fname);
                while((ch=fin.read())!=-1)
                {
                        //char count
                        if(ch!=' ' && ch!='\n')
                                ctr++;
                        //line count
                        if(ch=='\n')
                                L++;
                        //word count
                        if(ch==' ' && pre!=' ')
                                w++;
                        pre=ch;
                }
```

```
                System.out.println("Char count="+ctr);
                System.out.println("Word count="+(w+(L-1)));
                System.out.println("Line count="+L);
        }
}
```

ExpNo: 5. a)

Stack ADT using Arrays

Aim: To write a java program that implements stack ADT using an array.

Description: Create an interface StackADT with a constant size and declare methodprototypes push (), pop (), traverse () in that interface. By writing a class implements those method bodies. Declare a variable top which is initialized with -1 for every push operation the top is incremented by 1 and an element of type int is stored into an arry stack (i.e stk[]). For every pop operation the top is decremented by 1. For traverse operation the elements are displayed starting from stk[top] to stk[0].

Program:

```
import java.io.*;
interface StackADT
{
        int SIZE = 10;
        void push(int elem);
        void pop();
        void traverse();
}
class MyClass implements StackADT
{
        int top = -1;
        int stk[];
        MyClass()
        {
                stk = new int[SIZE];
        }
        public void push(int elem)
        {
                if ( top == (SIZE-1) )
                        System.out.println("Stack is full");
                else
                {       top++;
                        stk[top] = elem;
                }
        }
        public void pop()
        {
                if ( top == -1)
```

```
                System.out.println("Stack is empty, no element to delete");
        else
        {       System.out.println("The deleted element is : " +
                stk[top]); top--;
        }
    }
    public void traverse()
    {   if ( top == -1)
                System.out.println("Stack is empty, no elements to traverse");
        else
        {       System.out.println("Elements in the stack are : ");
                for(int i= top; i>= 0 ; i--)
                        System.out.println(stk[i]);
        }
    }
}
class StackDemo
{   public static void main(String args[]) throws IOException
    {
        BufferedReader br = new BufferedReader( new InputStreamReader(
                                                                System.in));
        MyClass ob = new MyClass();
        int ch, elem;
        do
        {
                System.out.println("1.Push");
                System.out.println("2.Pop");
                System.out.println("3.Traverse");
                System.out.println("4.Exit");
                System.out.println("Enter your choice : ");
                ch = Integer.parseInt(br.readLine());
                switch( ch )
                {
                        case 1 : System.out.println("Enter element to insert :");
                                elem = Integer.parseInt( br.readLine() );
                                ob.push(elem);
                                break;
                        case 2: ob.pop();
                                break;
                        case 3: ob.traverse();
                                break;
                        case 4: System.exit(0);
                }
        }while( ch>0 && ch< 5);
```

ExpNo: 5. b) **Date:**

Infix to Postfix Conversion

Aim: To write a java program that converts infix expression into postfix expression.

Description: This Program translates an infix expression to a postfix expression. In this program we take "infix" string as an input and displays "postfix" string. In the process, we use a stack as a temporary storage. So, instead of writing a separate program for stack operations, the Infix_Postfix class uses java.util.Stack class.

Program:

```
import java.io.*;
import java.util.*;
class Infix_Postfix
{
	public static void main(String[] args) throws IOException
	{
		BufferedReader br = new BufferedReader( new InputStreamReader (
									System.in));
		System.out.println("Enter Infix Expression : ");
		String s= br.readLine();
		Stack<Character> st=new Stack<Character>();
		String output="";
		int i=0,len=s.length();
		char x;
		st.push('@');
		while(len!=0)
		{	char c=s.charAt(i);
			if(c=='(')
			{	st.push(c);
			}
			else if(c=='+'||c=='-'||c=='*'||c=='/'||c=='^'||c=='$')
			{
				check(st,c,output)
				; st.push(c);
			}
			else if(c==')')
			{	while((x=(Character)st.pop())!='(')
				{	output=output+x;
				}
			}
			else
```

```
                {       output+=s.charAt(i);
                }
                i=i+1;
                len--;
        }
        while((x=(Character)st.pop())!='@')
        {       output+=x;
        }
        System.out.println("postfix Expression is : "+ output);
}
static void check(Stack st,char c,String output)
{
                while(priority(c)<=priority((Character)st.
                peek())) output=output+st.pop();
}
static int priority(char ch)
{
                if(ch=='+'||
                ch=='-')
                return(1);
        else if(ch=='*'||ch=='/')
                return(2);
        else if(ch=='$'||ch=='^')
                return(3);
                                else    return(0);

            }
    }
```

ExpNo: 5 c) **Postfix Evaluation** **Date:**

Aim: To write a java program that evaluates the postfix expression.

Description: This Program illustrates the evaluation of postfix expression. The program works for expressions that contain only *single-digit integers* and the *four arithmetic operators*. The program uses *java.util.Stack*class for creating a stack of integer values. While evaluating a postfix expression if we encounter an operand then push that operand onto stack otherwise if we encounter an operator Pop the top two operands from the stack, apply the operator to them, and evaluate it. Push this result onto the stack. Initially *stack* is empty

Program:

```
import java.io.*;
import java.util.*;
class MyClass
{
        int isoperator(char symbol)
        {
                if(symbol == '+' || symbol == '-' || symbol == '*' || symbol == '/' )
                        return 1;
                                        else     return 0;

                }

        double evaluate(String postfix)
        {
                Stack<Double> stk = new Stack<Double>();
                int i;
                char symbol;
                double oper1,oper2,result;
                for(i=0;i<postfix.length();i++)
                {
                        symbol = postfix.charAt(i);
                        if (isoperator(symbol)==0)
                                stk.push((double)(symbol-48) );
                        else
                        {                               oper2
                                                        =
                                                        stk.po
                                                        p();
                                                        oper1
                                                        =
```

```
stk.po
p();
result = calculate(oper1,symbol,oper2);
```

```
                        stk.push(result);
                    }
                }//end of for.
                result = stk.pop();
                return(result);
        }
        double calculate(double oper1,char symbol,double oper2)
        {
                switch(symbol)
                {
                        case '+' : return(oper1+oper2);
                        case '-' : return(oper1-oper2);
                        case '*' : return(oper1* oper2);
                        case '/' : return(oper1/oper2);
                        default : return(0);
                }
        }
}
class Postfix
{
        public static void main(String args[]) throws IOException
        {       BufferedReader br = new BufferedReader(new InputStreamReader(
                                                                System.in));
                System.out.println("Enter postfix expression : ");
                String postfix = br.readLine();
                MyClass ob = new MyClass();
                System.out.println("The result value is : " + ob.evaluate(postfix));
        }
}
```

ExpNo: 6. a)

Applet to display a Message

Aim: To develop an applet that displays a simple message

Description: An applet is a Java program that runs in a browser. Unlike Java applications applets do not have a main () method. To create applet we can use java.applet.Applet. All applets inherit the super class 'Applet'. An Applet class contains several methods that help to control the execution of an applet. Let the Applet class extends Applet. Provide paint () method in that class. Using drawstring () method of Graphics class display the message on the applet.

Program:

```
import java.io.*;
import java.awt.*;
import java.applet.Applet;
/*<applet code="App1.class" height=100 width=500></applet>*/
public class App1 extends Applet
{
        public void paint(Graphics g)
        {
                g.drawString("MCA III semester Students",50,60);
                setForeground(Color.blue);
        }
}
```

ExpNo: 6. b) **Date:**

Applet to calculate Factorial Value

Aim: To develop an applet that receives an integer in one text field, and computes as factorial value and returns it in another text field, when the button named "Compute" is clicked

Description: Write a class by extending Applet class and implement ActionListener. Attach two textfileds and a button to the applet. Attach actionlistener to the button and provide actionPerformed() method to compute factorial value. First textfield is used for entering a number and after clicking on the button the result will be displayed on the second textfield.

Program:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.applet.Applet;
/*<applet code="App2.class" height=100 width=500></applet>*/
public class App2 extends Applet implements ActionListener
{       JTextField tf1;
        JTextField tf2;
        JButton b;
        JLabel l;
        public void init()
        {
                l =new JLabel("Enter the number & press the button");
                tf1=new JTextField("",5);
                tf2=new JTextField("",10);
                b=new JButton("Compute");
                add(l);
                add(tf1);
                add(tf2);
                add(b);
                b.addActionListener(this);
        }
        public void actionPerformed(ActionEvent ae)
```

```
    {
        String str=ae.getActionCommand();
        String str1;
        int fact=1;
        if(str=="Compute")
        {
            int n=Integer.parseInt(tf1.getText());
            for(int i=1;i<=n;i++)
                fact=fact*i;
            str1=""+fact;
            tf2.setText(str1);
        }
    }
}
```

ExpNo: 7 **Date:**

Simple Calculator

Aim: To write a java program that works as a simple calculator. Use a grid layout to arrangebuttons for the digits +,-,*,/,% operations. Add a text field to display the result.

Description: Write a class by extending JFrame class of javax.swing package and implementing ActionListener interface. Attach the components like JTextField, JButtons, two JPanels to the contentPane() of JFrame by setting grid layout. Attach listeners to the buttons and implement the actionPerformed method to perform operations. One JPanel is used for placing textbox and clear buttons, the second JPanel is used to add remaining components.

Program:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Calculator extends JFrame implements ActionListener
{       Container c;
        JTextField t1;
        String a="",b;
        String oper="",s="",p="";
        int first=0,second=0,result=0;
        JButton
        b0,b1,b2,b3,b4,b5,b6,b7,b8,b9;
        JButton  add,sub,mul,div,mod,res,clear;
        JPanel p1,p2;
        Calculator()
        {       c =getContentPane();
                p1 = new JPanel();
                p2 = new JPanel(new
                GridLayout(4,4)); t1=new
                JTextField(a,10); c.setLayout(new
                GridLayout(2,1)); b0=new
                JButton("0");
                b1=new JButton("1");
                b2=new JButton("2");
                b3=new JButton("3");
                b4=new JButton("4");
                b5=new JButton("5");
                b6=new JButton("6");
                b7=new JButton("7");
```

```
        b8=new JButton("8");
        b9=new JButton("9");
        add=new JButton("+");
        sub=new JButton("-");
        mul=new JButton("*");
        div=new JButton("/");
        mod=new JButton("%");
        res=new JButton("=");
        clear = new
        JButton("CE");
        b0.addActionListener(this)
        ;
        b1.addActionListener(this)
        ;
        b2.addActionListener(this)
        ;
        b3.addActionListener(this)
        ;
        b4.addActionListener(this)
        ;
        b5.addActionListener(this)
        ;
        b6.addActionListener(this)
        ;
        b7.addActionListener(this)
        ;
        b8.addActionListener(this)
        ;
        b9.addActionListener(this)
        ;
        add.addActionListener(this);
        sub.addActionListener(this);
        mul.addActionListener(this);
        div.addActionListener(this);
        mod.addActionListener(this);
        res.addActionListener(this);
        clear.addActionListener(this);
        p1.add(t1);     p1.add(clear);
        p2.add(b0);     p2.add(b1);     p2.add(b2);
        p2.add(b3);     p2.add(b4);     p2.add(b5);
        p2.add(b6);     p2.add(b7);     p2.add(b8);
        p2.add(b9);     p2.add(add);    p2.add(sub);
        p2.add(mul);    p2.add(div);    p2.add(mod);
        p2.add(res);
        c.add(p1);
        c.add(p2);
}
public void actionPerformed(ActionEvent ae)
```

```
{       a=ae.getActionCommand();
        if(a=="0" || a=="1" || a=="2" || a=="3" || a=="4" || a=="5"
                        || a=="6" || a=="7" || a=="8" || a=="9")
        {
                t1.setText(t1.getText()+a);
        }
        if(a=="+" || a=="-" || a=="*" || a=="/" || a=="%")
```

```
            {
                first = Integer.parseInt(t1.getText());
                oper = a;
                t1.setText("");
            }
            if(a=="=")
            {       if(oper=="+")
                        result=first+Integer.parseInt(t1.getText());
                    if(oper=="-")
                        result=first-Integer.parseInt(t1.getText());
                    if(oper=="*")
                        result=first*Integer.parseInt(t1.getText());
                    if(oper=="/")
                        result=first/Integer.parseInt(t1.getText());
                    if(oper=="%")
                        result=first%Integer.parseInt(t1.getText());
                    t1.setText(result+"");
            }
            if(a == "CE")
                    t1.setText("");
        }
        public static void main(String args[])
        {       Calculator ob = new
                Calculator();
                ob.setSize(200,200);
                ob.setVisible(true);
                ob.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
}
```

Conclusion:

ExpNo: 8 **Date:**

Mouse Events

Aim: To write a java program for handling mouse events.

Description: The user may click, release, drag or move a mouse while interacting with the application. If the programmer knows what the user has done, he can write the code according to the mouse event. To trap the mouse events, MouseListener and MouseMotionListener interfaces of jav.awt.event package are used. MouseListener contains five methods mousePressed (), mouseReleased (), mouseClicked (), mouseEntered (), mouseExited () and MouseMotionListener interface contains two methods mouseMoved () and mouseDragged (). Attach these two listeners to applet and implement seven methods.

Program:

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

/*<applet code= "Mouse.class" height=300 width=400></applet>*/

public class Mouse extends Applet implementsMouseListener,MouseMotionListener
{
        String txt="";
        int x=10,y=30;
        public void init()
        {       addMouseListener(this);
                addMouseMotionListener(this)
                ;
        }
        public void mouseClicked(MouseEvent me)
        {       txt="Mouse Clicked";
                setForeground(Color.pink);
                repaint();
        }
        public void mouseEntered(MouseEvent me)
        {       txt="Mouse Entered";
                repaint();
        }
        public void mouseExited(MouseEvent me)
        {       txt="Mouse Exited";
                setForeground(Color.blue);
```

```
        repaint();
    }
    public void mousePressed(MouseEvent me)
    {       txt="Mouse Pressed";
            setForeground(Color.blue);
            repaint();
    }
    public void mouseMoved(MouseEvent me)
    {       txt="Mouse Moved";
            setForeground(Color.red);
            repaint();
    }
    public void mouseDragged(MouseEvent me)
    {       txt="Mouse Dragged";
            setForeground(Color.green);
            repaint();
    }
    public void mouseReleased(MouseEvent me)
    {       txt="Mouse Released";
            setForeground(Color.yellow);
            repaint();
    }
    public void paint(Graphics g)
    {       g.drawString(txt,30,40);
            showStatus("Mouse events
            Handling");
    }
}
```

ExpNo: 9. a)

Creating Threads

Aim: To write a java program that creates three threads. First thread displays "Good Morning" every one second, the second thread displays "Hello" every two seconds and the third thread displays "Welcome" every three seconds.

Description: Write a class by extending Thread Class or implementing Runnable interface. Provide run() method in that class. Create three objects to that class. Create three threads using Thread class of lang package and attach single thread to single object for three objects. Call start() method of run() threads.

Program:

```
class MyThread implements Runnable
{
        String str;
        int time;
        MyThread(String s, int t)
        {       str = s;
                time = t;
        }
        public void run()
        {
                for(int i=1;i<=10;i++)
                {
                        System.out.println(str + "\t" + i);
                        try
                        {       Thread.sleep(time);
                        }
                        catch(Exception e)
                        {       System.out.println(e);
                        }
                }
        }
}
class ThDemo
{
        public static void main(String args[])
        {
                MyThread ob1 = new MyThread("Good Morning", 1000 );
                MyThread ob2 = new MyThread("Hello", 2000 );
                MyThread ob3 = new MyThread("Welcome", 3000 );
```

```
            Thread t1 = new Thread(ob1);
            Thread t2 = new Thread(ob2);
            Thread t3 = new Thread(ob3);
            t1.start();
            t2.start();
            t3.start();
        }
}
```

ExpNo: 9. b)

Inter Thread Communication

Aim: To write a java program that correctly implements Producer consumer problem using the concept of inter thread communication.

Description: In some cases two or more threads should communicate with each other. One thread output may be send as input to other thread. For example, a consumer thread is waiting for a Producer to produce the data (or some goods). When the Producer thread completes production of data, then the Consumer thread should take that data and use it. In producer class we take a StringBuffer object to store data, in this case; we take some numbers from 1 to 5. These numbers are added to StringBuffer object. Until producer completes placing the data into StringBuffer the consumer has to wait. Producer sends a notification immediately after the data production is over.

Program:

```
class Producer implements Runnable
{       StringBuffer sb;
        Producer ()
        {       sb = new StringBuffer();
        }
        public void run ()
        {
                synchronized (sb)
                {
                        for (int i=1;i<=5;i++)
                        {       try
                                {
                                        sb.append (i + " : ");
                                        Thread.sleep (500);
                                        System.out.println (i + " appended");
                                }
                                catch (InterruptedException ie){}
                        }
                        sb.notify ();
                }
        }
}
class Consumer implements Runnable
{       Producer prod;
        Consumer (Producer prod)
        {       this.prod = prod;
```

```
        }
        public void run()
        {       synchronized (prod.sb)
                {       try
                        {
                                prod.sb.wait ();
                        }
                        catch (Exception e) { }
                        System.out.println (prod.sb);
                }
        }
}
class Communicate
{
        public static void main(String args[])
        {
                Producer obj1 = new Producer ();
                Consumer obj2 = new Consumer (obj1);
                Thread t1 = new Thread (obj1);
                Thread t2 = new Thread (obj2);
                t2.start ();
                t1.start ();
        }
}
```

Input / Output:

Conclusion:

ExpNo: 10

User Interface that allows user to perform Integer Division

Aim: To write a program that creates a user interface to perform integer divisions. The user enters two numbers in the textfields,Num1 and Num2.The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException Display the exception in a message dialog box.

Description: Create an user interface using swing components. First create a class by extending JFrame class of javax.swing package and implement ActionListener interface to perform an action when the button is clicked on the frame. Create three textfields, a button and a label and attach those components onto the getContentPane() of JFrame.
Attach actionListener to the button and provide actionPerformed() method to perform the operation division. When an exception is raised then display that information in a JOptionPane.

Program:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class ARDemo extends JFrame implements ActionListener
{       JTextField tf1, tf2, tf3;
        JButton b;
        JLabel l;
        ARDemo()
        {  Container  c  =  getContentPane();
              c.setLayout(new FlowLayout());
              l=new  JLabel("Enter  Numbers  and  press  divide
              button"); tf1=new JTextField("",5);
              tf2=new  JTextField("",5);
              tf3=new  JTextField("",5);
              b=new
              JButton("Divide");
              c.add(l);
              c.add(tf1);
              c.add(tf2);
              c.add(b);
              c.add(tf3);
              b.addActionListener(this);
        }
        public void actionPerformed(ActionEvent ae)
```

```
{
        if(ae.getActionCommand()=="Divide")
        {       try
                {
                        int n1=Integer.parseInt(tf1.getText());
                        int n2=Integer.parseInt(tf2.getText());
                        int n=n1/n2;
                        tf3.setText(""+n);
                }
                catch(ArithmeticException e1)
                {
                JOptionPane.showMessageDialog (null,"Arthimetic Exception");
                }
                catch(NumberFormatException e2)
                {
        JOptionPane.showMessageDialog(null,"NumberFormatException");
                }
        }
}
public static void main(String args[])
{       ARDemo ob = new ARDemo();
        ob.setSize(800,600);
        ob.setVisible(true);
        ob.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
```

ExpNo: 11

Client / Server Application

Aim: To write a java program that implements a simple client/server application. The client sends data to a server. The server receives the data, uses it to produce a result and then sends the result back to the client. The client displays the result on the console.

Description: Socket programming is a low level way to establish a connection between client and server. Socket is a point of connection where client and server exchange information using a port number. Read input from keyboard and send that information from client to server using DataOutputStream. In turn server read data from BufferedReader and performs computation on the data and sends the output to client using PrintStream class. The client read the information from BufferedReader and displays it on the monitor. The streams are available in java.io package.

Program:

//server

```
import java.io.*;
import java.net.*;
class Server
{
        public static void main(String ar[])throws Exception
        {       ServerSocket ss=new ServerSocket(7777);
                Socket s=ss.accept();
                System.out.println("Connection Established...");
                BufferedReader br = new BufferedReader(new InputStreamReader
                                                        (s.getInputStream()));
                int n=Integer.parseInt(br.readLine());
                PrintStream ps=new PrintStream(s.getOutputStream());
                ps.println(3.14*n*n);
                s.close();
                ss.close();
        }
}
```

//client

```
import java.io.*;
import java.net.*;
class Client
{
        public static void main(String ar[])throws Exception
        {
```

```
            Socket s=new Socket("localhost",7777);
            System.out.println("Enter the radius of the circle ");
            BufferedReader kb =new BufferedReader(new InputStreamReader (
                                                        System.in));
            int n = Integer.parseInt(kb.readLine());
            DataOutputStream ds=new DataOutputStream (s.getOutputStream());
            ds.writeBytes( n + "\n");
            BufferedReader br = new BufferedReader(new InputStreamReader
                                                        (s.getInputStream()));
            System.out.println("Area of the circle from server:"+ br.readLine());
        }
}
```

ExpNo: 12. a)

Program that simulates Traffic Light

Aim: To write a java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow or green. When a radio button is selected, the light is turned on and only one light can be on at a time. No light is on when the program starts.

Description: First create a class by extending Frame class of java.awt package. Override paint () method of Frame class. We can use the method of Graphics class fillOval() to display three lights by setting color. Radio buttons are created such that only one button can be selected at once, based on this selection corresponding fillOval() with color will be displayed.

Program:

```
import java.awt.*;
import java.awt.event.*;
class MyFrame extends Frame implements ItemListener
{
        Checkbox b1, b2, b3;
        CheckboxGroup  cbg;
        MyFrame()
        {
                setLayout( new FlowLayout() );
                cbg = new CheckboxGroup();
                b1 = new Checkbox("RED",cbg,false);
                b2 = new Checkbox("GREEN",cbg,false);
                b3 = new Checkbox("YELLOW",cbg,false);
                add(b1);        add(b2);        add(b3);
                b1.addItemListener(this);
                b2.addItemListener(this);
                b3.addItemListener(this);
                addWindowListener( new WindowAdapter()
                {
                        public void windowClosing(WindowEvent we)
                        {
                                System.exit(0);
                        }
                });
        }
        public void itemStateChanged(ItemEvent ie)
```

```
        {       repaint();
        }
        public void paint(Graphics g)
        {
                if(b1.getState() )
                {       g.setColor(Color.red);
                        g.fillOval(100,100,100,100);
                }
                if(b2.getState())
                {       g.setColor(Color.green);
                        g.fillOval(100,100,100,100);
                }
                if(b3.getState())
                {       g.setColor(Color.yellow);
                        g.fillOval(100,100,100,100);
                }
        }
}
class TLight
{
        public static void main(String args[])
        {
                MyFrame ob = new MyFrame();
                ob.setTitle("Traffic Light");
                ob.setSize(600, 300);
                ob.setVisible(true);
        }
}
```

ExpNo: 12. b)

Program allows user to draw lines, ovals and rectangles

Aim: To write a java program that allows the user to draw lines, rectangles and ovals.

Description: First create a class by extending Frame class and implement MouseListener interface. Override paint() method of Frame class. We can use methods of Graphics class to draw lines, rectangles and ovals.Provide the methods of wanted MouseListener interface and provide empty implementations to other methods of MouseListener interface.

Program:

```
import java.awt.*;
import java.awt.event.*;
public class DrawFigures extends Frame implements ActionListener, MouseListener
{
        Button line, rect, oval;
        int startX, startY, endX, endY, drawCode = 1;
        public DrawFigures()
        {       line = new Button("DRAW LINE");
                line.addActionListener(this);
                add(line);
                rect = new Button("DRAW RECTANGLE");
                rect.addActionListener(this);
                add(rect);
                oval = new Button("DRAW OVAL");
                oval.addActionListener(this);
                add(oval);
                addWindowListener( new WindowAdapter()
                {       public void windowClosing(WindowEvent we)
                        {
                                System.exit(0);
                        }
                });
                addMouseListener(this);
        }
        public void mouseClicked(MouseEvent e)
        {
        }
        public void mousePressed(MouseEvent e)
        {       startX =
                e.getX(); startY
                = e.getY();
        }
```

```
public void mouseReleased(MouseEvent e)
{       endX = e.getX();
        endY = e.getY();
        Graphics g = getGraphics();
        if (drawCode == 1)
        {       drawLine(g);
        }
        else if (drawCode == 2)
        {       drawRect(g);
        }
        else
        {       drawOval(g);
        }
}
void drawLine(Graphics g)
{       g.drawLine(startX, startY, endX, endY);
}
void drawRect(Graphics g)
{
        if (startX > endX)
        {       g.drawRect(endX, endY, -(endX - startX), -(endY - startY));
        }
        else
        {       g.drawRect(startX, startY, endX - startX, endY - startY);
        }
}
void drawOval(Graphics g)
{
        if (startX > endX)
        {       g.drawOval(endX, endY, -(endX - startX), -(endY - startY));
        }
        else
        {       g.drawOval(startX, startY, endX - startX, endY - startY);
        }
}
public void mouseEntered(MouseEvent e)
{
}
public void mouseExited(MouseEvent e)
{
}
public void actionPerformed(ActionEvent e)
{       if (e.getActionCommand().equals("DRAW LINE"))
        {
```

```
                drawCode = 1;
            }
            else if (e.getActionCommand().equals("DRAW RECTANGLE"))
            {       drawCode = 2;
            }
            else
            {       drawCode = 3;
            }
        }
        public static void main(String[] args)
        {
            DrawFigures ob = new DrawFigures();
            ob.setTitle("Draw Lines, Rectangles and Ovals");
            ob.setLayout(new FlowLayout());
            ob.setSize(500, 500);
            ob.setVisible(true);
        }
}
```

ExpNo: 13 a)

Abstract Class and its Implementation

Aim: To Write a Java program to create an abstract class named Shape that contains an empty method named numberOfSides ( ).Provide three classes named Trapezoid, Triangle and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides ( ) that shows the number of sides in the given geometrical figures.

Description:A method without method body is called abstract method. A class thatcontains abstract method is called abstract class. It is possible to implement the abstract methods differently in the subclasses of an abstract class. These different implementations will help the programmer to perform different tasks depending on the need of the sub classes. Moreover, the common members of the abstract class are also shared by the sub classes.

Program:

```
import java.lang.*;
abstract class Shape
{
        abstract void numberOfSides();
}
class Traingle extends Shape
{
        public void numberOfSides()
        {       System.out.println("three");
        }
}
class Trapezoid extends Shape
{       public void numberOfSides()
        {       System.out.println("four");
        }
}
class Hexagon extends Shape
{       public void numberOfSides()
        {       System.out.println("six");
        }
}
public class Sides
{
```

```
        public static void main(String arg[])
        {       Traingle T=new Traingle();
                Trapezoid Ta=new Trapezoid();
                Hexagon H=new Hexagon();
                T.numberOfSides();
                Ta.numberOfSides();
                H.numberOfSides();
        }
}
```

ExpNo: 13. b)

Displaying Contents of a file in a Table

Aim: To write a java program to display the table using JTable component. Suppose that a table named Table.txt is stored in a text file. The first line in the file is the header, and the remaining lines correspond to rows in the table. The elements are separated by commas.

Description: JTable represents data in the form of a table. The table can have rows of data, and column headings. Read the first line in the file and by using StringTokenizer get the column headings and store it in a single dimensional array. Read the next lines in the file and by using StringTokenizer get the row data and store that data in two dimensional array. Attach scrollpane to JTable component and attach it to JFrame.

Program:

```
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.util.*;
class MyTable extends JFrame
{       JTable tab;
        JScrollPane scrollPane;
        Vector<String> initData = new Vector<String>();;
        String data[][] ;
        String cols[];
        File file = null;
        FileInputStream fis = null;
        BufferedInputStream bis = null;
        Scanner sc = null;
        public MyTable()
        {       Container c = getContentPane ();
                c.setLayout (new FlowLayout ());
                try
                {
                        file = new File("D:\\jqr\\Table.txt");
                        fis = new FileInputStream(file);
                        bis = new BufferedInputStream(fis);
                        sc = new Scanner( new InputStreamReader (bis) );
                        while (sc.hasNextLine())
```

```
                {
                        initData.add( sc.nextLine());
                }
                StringTokenizer st = new StringTokenizer( initData.get(0), ",");
                cols = new String[ st.countTokens() ];
                data = new String[ initData.size()-1][st.countTokens()];
                for( int i = 0; i< initData.size(); i++)
                {       if( i== 0)
                        {       int j= 0;
                                while( st.hasMoreTokens() )
                                {       cols[j] = st.nextToken();
                                        j++;
                                }

                        }
                        else
                        {
                                int k=0;
                StringTokenizer str = new StringTokenizer( initData.get(i),",");
                                while( str.hasMoreTokens() )
                                {
                                        data[i-1][k] = str.nextToken();
                                        k++;
                                }
                        }
                }
                fis.close();
                bis.close();
        }
        catch (FileNotFoundException e)
        {       e.printStackTrace();
        }
        catch (IOException e)
        {       e.printStackTrace();
        }
        JTable tab = new JTable (data, cols);
        scrollPane = new JScrollPane(tab);
        c.add(scrollPane);

    }
}
class TableDemo
{
    public static void main(String args[])
```

```
    {
        MyTable ob = new MyTable();
        ob.setTitle("Table Demo");
        ob.setSize(600,400);
        ob.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
        ob.setVisible( true );
    }
}
```