PROGRAM

```
import java.util.*;
public class quicksort {
    static int partition(int a[],int low,int high)
    {
        int s=high;
        int b=low;
        int pivotindex=low;
        int temp;
        while(s>b)
        {
            while(a[b]<=a[pivotindex] && b<high)
            {
                b++;
            }
            while(a[s]>a[pivotindex])
            {
                s--;
            }
            if(s>b)
            {
                temp=a[s];
                a[s]=a[b];
                a[b]=temp;
            }
        }
        temp=a[s];
        a[s]=a[pivotindex];
        a[pivotindex]=temp;
        return s;
    }
    static void quicksort(int a[],int low,int high)
    {
        if(low<high)
        {
            int q=partition(a,low,high);
            quicksort(a, low, q-1);
            quicksort(a, q+1, high);
        }
    }
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        int a[]=new int[20];
        int n,i;
```

```
        System.out.print("Enter the number of elements:");
        n=sc.nextInt();
        System.out.print("Enter the elements:");
        for(i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        quicksort(a,0,n-1);
        System.out.print("The sorted elements are:");
        for(i=0;i<n;i++)
        {
            System.out.print(a[i]+" ");
        }

    }
}
```

Date 15/12/23

Expt. No. 15 Page No. 28

QUICK SORT

Aim:-

To write a Java program for implementing quick sort.

Algorithm:-

Step 1: Start

Step 2: Create a class named quicksort and define the methods partition, quicksort and main method inside it.

Step 3: Inside method partition assign s=high, b=low, pivotindex=low.

Step 4: Repeat steps 5 to 9 until s>b.

Step 5: Repeat step 6 until a[b] <= a[pivotindex] and b<high.

Step 6: Assign b=b+1.

Step 7: Repeat step 8 until a[s] > a[pivotindex].

Step 8: Assign s=s-1.

Step 9: If s is greater than b, Swap a[s] and a[b].

Step 10: Swap the elements a[s] and a[pivotindex].

Step 11: Return s.

Step 12: Inside the method quicksort, check if low is less than high.

Step 13: If low is less than high invoke the method partition(a,low,high) and assign the value returned by the function to q.

Step 14: Recursively call the method quicksort (a,low,q-1) and quicksort(a,q+1,high).

Step 15: Inside main method, create an Object of the Scanner class.

Step 16: Declare variables n,i and an array 'a'.

Step 17: Read the number of elements as n

Step 18: Assign i=0.

Teacher's Signature

OUTPUT

```
Enter the number of elements:6
Enter the elements:68 76 45 90 34 21
The sorted elements are:21 34 45 68 76 90
```

Step 19: Repeat step 20 until i<n.

Step 20: Read each element and store it in i^{th} index.

Step 21: Invoke the method quicksort and pass a, 0 and n-1 as it's arguments.

Step 22: Display the sorted array by printing each element using a for loop.

Step 23: Stop.

Result:

The program to implement quicksort was executed successfully and verified the output.

PROGRAM

```
import java.util.Scanner;
class LinkedList {
    private Node head;

    class Node {
        private int data;
        private Node left;
        private Node right;

        public Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }

    public void insert(int data) {
        Node temp = new Node(data);
        if (head == null) {
            head = temp;
        } else {
            Node ptr = head;
            while (ptr.right != null) {
                ptr = ptr.right;
            }
            ptr.right = temp;
            temp.left = ptr;
        }
    }

    public void delete() {
        int x = head.data;
        head = head.right;
        head.left = null;
        System.out.println("Element " + x + " got deleted");
    }

    public void display() {
        if (head == null)
            System.out.println("List is Empty");
        else {
            Node ptr = head;
            System.out.print("Elements:");
            while (ptr != null) {
```

```
                System.out.print(ptr.data + " ");
                ptr = ptr.right;
            }
            System.out.println();
        }
    }
}

class Test {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        Scanner sc = new Scanner(System.in);
        String choice = "";
        while (!choice.equals("4")) {
            System.out.print("\n1. Insert at End \n2. Delete From
            Front \n3.   Display\n4.Exit\n");
            System.out.print("Enter the choice:");
            choice = sc.nextLine();
            switch (choice) {
                case "1":
                    System.out.print("Enter the number to insert:");
                    int data = sc.nextInt();
                    sc.nextLine();
                    list.insert(data);
                    System.out.println("Data inserted Successfully");
                    break;
                case "2":
                    list.delete();
                    break;
                case "3":
                    list.display();
                    break;
                case "4":
                    break;
                default:
                    System.out.println("Invalid Choice");
            }
        }
    }
}
```

Date 18/12/23

Expt. No. 16 Page No. 30

DOUBLY LINKED LIST

Aim:- To write a Java Program to implement Queue using Double Linked List.

Algorithm:-

Step 1: Start.

Step 2: Import the Scanner class belonging to the package java.util

Step 3: Create a class named 'Linkedlist' and declare a head variable of type Node.

Step 4: Create a class Node within this class and declare variables private data, Node left, Node right.

Step 5: Create a constructor for the class Node with parameters data, left and right and assign their value using this keyword.

Step 6: Within the class Linked List create method insert() defined to insert elements to the node.

Step 7: Create an object 'temp' for the class Node.

Step 8: Set head = temp if head == null.

Step 9: Otherwise set Node ptr = head.

Step 10: Set ptr = ptr.right until ptr.right != null inside a while loop.

Step 11: Assign ptr.right = temp and temp.left = ptr.

Step 12: Create a method delete() defined to delete an element from the node.

Step 13: Check whether the linked list is empty. if not check if head.right = null. then initialize a variable 'x' and assign the data to be deleted into it and set head = null.

Step 14: print the element deleted.

Step 15: Create another method within the class Linkedlist display(),

Teacher's Signature _____

OUTPUT

```
1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:1
Enter the number to insert:4
Data inserted Successfully

1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:1
Enter the number to insert:5
Data inserted Successfully

1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:1
Enter the number to insert:6
Data inserted Successfully

1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:1
Enter the number to insert:7
Data inserted Successfully

1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:3
Elements:4 5 6 7

1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:2
```

```
Element 4 got deleted

1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:3
Elements:5 6 7

1. Insert at End
2. Delete From Front
3. Display
4.Exit
Enter the choice:
4
```

defined to display the elements in the linked list.

Step 17: Check whether the list is empty if not set the ptr to the head and print the elements till the ptr reaches null.

Step 18: Create a class named Test and inside it define main methods.

Step 19: Inside main method create objects of the class Scanner and LinkedList.

Step 20: Read a string 'choice' to choose an option for insertion, deletion, display and exit.

Step 21: Using switch statement the corresponding choice is executed.

case 1: The element to be inserted is input by invoking the insert method() using the object list (list.insert(data)).

case 2: The deletion part is executed by invoking the method delete() using the object list (list.delete()).

case 3: The elements present in the node are displayed by invoking the method display() using the object list (list.display()).

Step 22: Repeat these steps until choice is not equal to 4.

Step 23: Stop.

Result:-

The Java program to implement queue using Double Linked List was executed successfully and verified the output.

Teacher's Signature _____

PROGRAM

```
import javax.swing.*;
import java.awt.event.*;
class Calculator extends JFrame implements ActionListener {
    private JTextField t1;
    private JButton b1;
    private JButton b2;
    private JButton b3;
    private JButton b4;
    private JButton b5;
    private JButton b6;
    private JButton b7;
    private JButton b8;
    private JButton b9;
    private JButton b10;
    private JButton b11;
    private JButton b12;
    private JButton b13;
    private JButton b14;
    private JButton b15;
    private JButton b16;
    private Integer res;
    private String operation;
    public Calculator() {
        setLayout(null);
        setSize(640, 480);
        t1 = new JTextField();
        t1.setBounds(100, 100, 200, 30);
        b1 = new JButton("1");
        b1.setBounds(100, 140, 50, 30);
        b2 = new JButton("2");
        b2.setBounds(150, 140, 50, 30);
        b3 = new JButton("3");
        b3.setBounds(200, 140, 50, 30);
        b4 = new JButton("+");
        b4.setBounds(250, 140, 50, 30);
        // Third Row
        b5 = new JButton("4");
        b5.setBounds(100, 170, 50, 30);
        b6 = new JButton("5");
        b6.setBounds(150, 170, 50, 30);
        b7 = new JButton("6");
        b7.setBounds(200, 170, 50, 30);
        b8 = new JButton("-");
        b8.setBounds(250, 170, 50, 30);
```

```
// Fourth Row
b9 = new JButton("7");
b9.setBounds(100, 200, 50, 30);
b10 = new JButton("8");
b10.setBounds(150, 200, 50, 30);
b11 = new JButton("9");
b11.setBounds(200, 200, 50, 30);
b12 = new JButton("*");
b12.setBounds(250, 200, 50, 30);
// Fourth Row
b13 = new JButton("/");
b13.setBounds(100, 230, 50, 30);
b14 = new JButton("%");
b14.setBounds(150, 230, 50, 30);
b15 = new JButton("=");
b15.setBounds(200, 230, 50, 30);
b16 = new JButton("C");
b16.setBounds(250, 230, 50, 30);
add(t1);
add(b1);
add(b2);
add(b3);
add(b4);
add(b5);
add(b6);
add(b7);
add(b8);
add(b9);
add(b10);
add(b11);
add(b12);
add(b13);
add(b14);
add(b15);
add(b16);
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);
b10.addActionListener(this);
```

```
        b11.addActionListener(this);
        b12.addActionListener(this);
        b13.addActionListener(this);
        b14.addActionListener(this);
        b15.addActionListener(this);
        b16.addActionListener(this);
    }
    public void doAction(String op) {
        if (operation == null) {
            operation = op;
            res = Integer.parseInt(t1.getText());
            t1.setText("");
        } else{
            switch (operation) {
                case "+":
                    res = res + Integer.parseInt(t1.getText());
                    break;
                case "-":
                    res = res - Integer.parseInt(t1.getText());
                    break;
                case "/":
                    try{
                      if (t1.getText().equals("0")) {
                        throw new ArithmeticException("Divide by Zero");
                      }
                      res = res / Integer.parseInt(t1.getText());
                    } catch (ArithmeticException e) {
                        t1.setText(e.getMessage());
                        operation = null;
                        res = 0;
                    }
                    break;
                case "*":
                    res = res * Integer.parseInt(t1.getText());
                    break;
                case "%":
                    res = res % Integer.parseInt(t1.getText());
                    break;
            }
            if (op.equals("=")) {
                t1.setText(res.toString());
                res = 0;
                operation = null;
            } else {
                operation = op;
```

```
                t1.setText("");
            }
        }
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == b1)
            t1.setText(t1.getText() + "1");
        else if (e.getSource() == b2)
            t1.setText(t1.getText() + "2");
        else if (e.getSource() == b3)
            t1.setText(t1.getText() + "3");
        else if (e.getSource() == b5)
            t1.setText(t1.getText() + "4");
        else if (e.getSource() == b6)
            t1.setText(t1.getText() + "5");
        else if (e.getSource() == b7)
            t1.setText(t1.getText() + "6");
        else if (e.getSource() == b9)
            t1.setText(t1.getText() + "7");
        else if (e.getSource() == b10)
            t1.setText(t1.getText() + "8");
        else if (e.getSource() == b11)
            t1.setText(t1.getText() + "9");
        else if (e.getSource() == b16) {
            t1.setText("");
            res = 0;
            operation = null;
        } else if (e.getSource() == b4) {
            doAction("+");
        } else if (e.getSource() == b8)
            doAction("-");
        else if (e.getSource() == b12)
            doAction("*");
        else if (e.getSource() == b13)
            doAction("/");
        else if (e.getSource() == b14)
            doAction("%");
        else if (e.getSource() == b15)
            doAction("=");
    }
    public static void main(String args[]) {
        new Calculator().setVisible(true);
    }
}
```

Date 18/12/23

Expt. No. 17

Page No. 32

SIMPLE CALCULATOR USING SWING

Aim:- To write a Java program to implement a simple calculator using SWING

Algorithm:-

Step 1: Start.

Step 2: Import all classes from javax.swing.package.

Step 3: Import all classes from java.awt.event package.

Step 4: Create a class Calculator that extends JFrame which implements ActionListener.

Step 5: Declare private instance variable JTextField t1 to display input and result.

Step 6: Declare private JButton variables from b_1 to b_{16} for number buttons, operation buttons and clear buttons.

Step 7: Declare a private integer variable to store the result.

Step 8: Declare a private String operation variable to store the arithmetic operation.

Step 9: Create a constructor for the class calculator.

Step 10: Set the layout to null and size of frame to 640x480 pixels.

Step 11: Initialize the textfield t_1 and set its Bounds.

Step 12: Initialize the buttons b_1 to b_{16} and set bounds.

Step 13: Add buttons and TextField to the frame using add() method.

Step 14: Invoke the method addActionListener for all the buttons.

Step 15: Create a method doAction() that takes operation as parameters.

Step 16: If operation = null, store operation and display the

Teacher's Signature _____

entered number in res.

Step 17: Using switch statement corresponding operation is performed.

case "+": Set res = res + value in the textbox t1.

case "-": Set res = res - value in the textbox t1.

case "/": If value in t_1=0, throw Exception for Division by zero. else set res = res/value in text box t1.

case "*": Set res = res * value in textbox t1.

case "%": Set res = res % value in textbox t1.

Step 18: if operation is "=", display result and reset the operation to null and res=0.

Step 19: Create and override actionPerformed(ActionEvente)method.

Step 20: if b1 is pressed, append 1, else if $b_2, b_3, b_5, b_6, b_7, b_9, b_{10}, b_{11}$ is pressed, append 2,3,4,5,6,7,8,9 to the text box respectively.

Step 21: if b_4 is pressed, invoke doAction("+").

Step 22: if b_8 is pressed, invoke doAction("-").

Step 23: if b_{12} is pressed, invoke doAction("+").

Step 24: if b_{13} is pressed, invoke doAction("/").

Step 25: if b_{14} is pressed, invoke doAction("%").

Step 26: if b_{15} is pressed, invoke doAction("=").

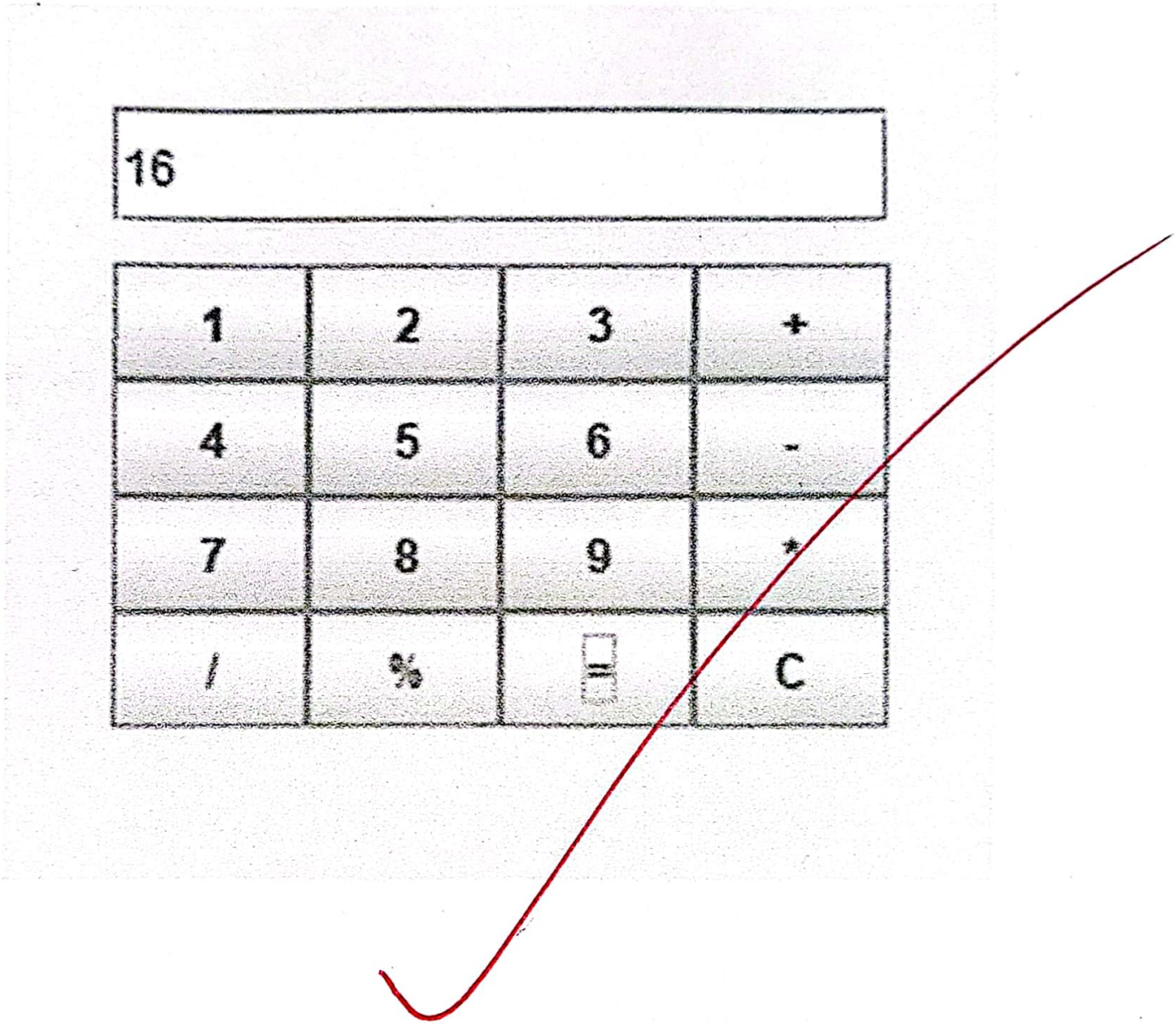
Step 27: Create a main method.

Step 28: Instantiate an object for the class Calculator and Set visibility of the frame using setVisible(true)

Step 29: Stop.

Teacher's Signature _____

OUTPUT



Date _____

Expt. No. _____ Page No. 34

Result:

The Java Program to implement a simple calculator using swing was implemented successfully and verified the output.

Teacher's Signature _____

PROGRAM

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class TrafficLight extends JPanel implements ActionListener{
    private JRadioButton r1;
    private JRadioButton r2;
    private JRadioButton r3;
    private Color red_c;
    private Color green_c;
    private Color orange_c;

    public TrafficLight() {
        setBounds(0, 0, 600, 480);
        r1 = new JRadioButton("Red");
        r2 = new JRadioButton("Green");
        r3 = new JRadioButton("Orange");
        ButtonGroup group = new ButtonGroup();
        r1.setSelected(true);
        group.add(r1);
        group.add(r2);
        group.add(r3);
        add(r1);
        add(r2);
        add(r3);
        red_c = Color.red;
        green_c = getBackground();
        orange_c = getBackground();
        r1.addActionListener(this);
        r2.addActionListener(this);
        r3.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (r1.isSelected() == true) {
            red_c = Color.red;
            green_c = getBackground();
            orange_c = getBackground();
        } else if (r2.isSelected() == true) {
            red_c = getBackground();
            green_c = Color.green;
            orange_c = getBackground();
        } else if (r3.isSelected() == true) {
            red_c = getBackground();
```

```
                green_c = getBackground();
                orange_c = Color.orange;
            }
            repaint();
        }

        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.drawOval(50, 50, 50, 50);
            g.drawOval(50, 110, 50, 50);
            g.drawOval(50, 170, 50, 50);
            g.setColor(red_c);
            g.fillOval(50, 50, 50, 50);
            g.setColor(orange_c);
            g.fillOval(50, 110, 50, 50);
            g.setColor(green_c);
            g.fillOval(50, 170, 50, 50);
        }
    }

    class Test1 {
        public static void main(String args[]) {
            JFrame f1 = new JFrame();
            f1.setVisible(true);
            f1.setSize(600, 480);
            f1.setLayout(null);
            TrafficLight t = new TrafficLight();
            f1.add(t);
        }
    }
```

Date 22/12/23.

Expt. No. 18 Page No. 35

TRAFFIC SIGNAL SIMULATION.

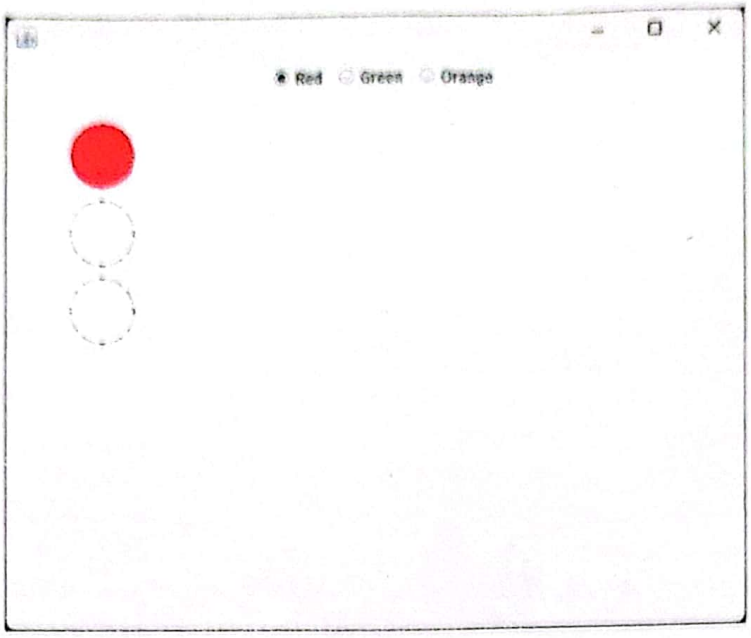
Aim:-

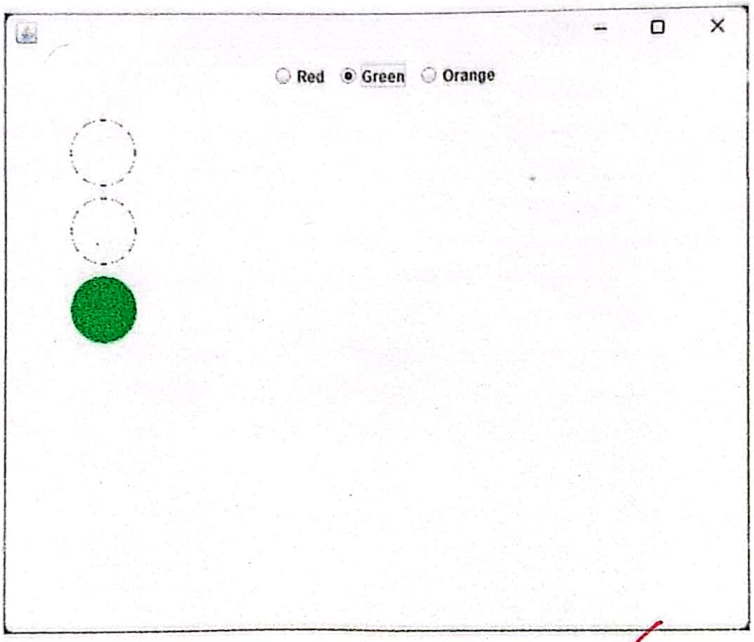
To write a Java Program to implement Traffic light using java swing.

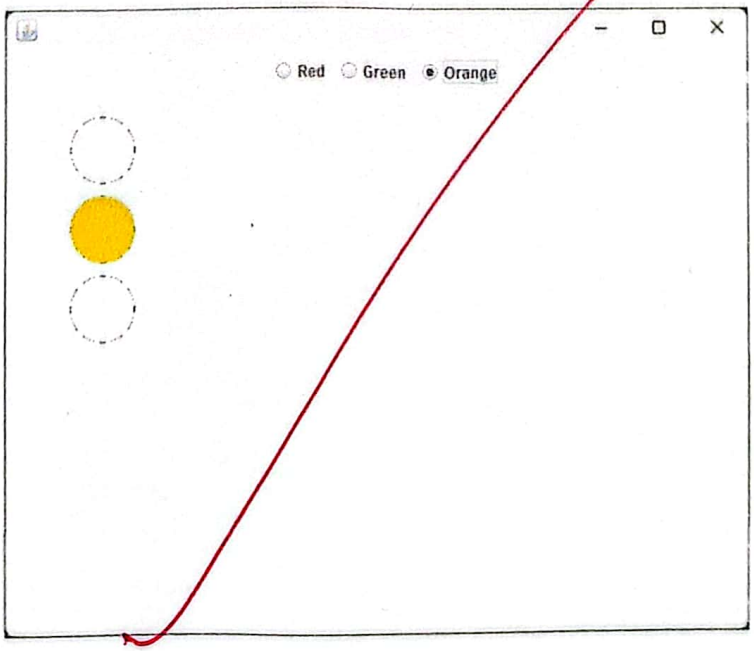
Algorithm:-

Step 1: Start.
Step 2: Import all the classes belonging to javax.swing package.
Step 3: Import all the classes belonging to java.awt and java.awt.event packages.
Step 4: Create a class Trafficlight which extends JPanel and implements ActionListener.
Step 5: Declare JRadioButton objects r_1, r_2 and r_3.
Step 6: Declare Color objects red_c, green_c and yellow_c.
Step 7: Create a constructor of the class Trafficlight.
Step 8: Set the panel bounds.
Step 9: Initialize radio buttons for each light colour.
Step 10: Create a ButtonGroup object group.
Step 11: Set initial selection to redlight using r_1.setselected(true) method.
Step 12: Add all the radio buttons to the group using group.add().
Step 13: Invoke add() method for all Radiobuttons.
Step 14: Add ActionListeners to each radio button using addActionListener(this) method.
Step 15: Create and implement actionPerformed(ActionEvent e) method.
Step 16: Inside it check which button is selected using isselected()

Teacher's Signature _____


Red
Green
Orange


Red
Green
Orange


Red
Green
Orange

method and update colours accordingly.

Step 17: Invoke repaint() method to update the UI.

Step 18: Override the paintComponent method passing Graphics objects g.

Step 19: Invoke super.paintComponent(g) method.

Step 20: Draw circles for each traffic light.

Step 21: Fill each circle with appropriate colours.

Step 22: Define a class Test1 and create main method inside it.

Step 23: Inside main method, Create a JFrame object f1.

Step 24: Set frame visibility.

Step 25: Set frame size and layout to null.

Step 26: Create an instance of Traffichight class as t.

Step 27: Add to the frame using add() method.

Step 28: Stop.

Result:-

The Java Program to implement traffic light using Java swing was executed successfully and verified the output.

Reena mg