PROGRAMMING USING JAVA
WEEK 14 ASSIGNMENT

1.Illustrate the working mechanism of Checkbox and Radio button by considering all possible constructors.

A checkbox may be a control that's used to turn an option on or off. It consists of a little box that will either contain a check or not. There's a label related to each checkbox that describes what option the box represents. You modify the state of a checkbox by clicking on. Checkboxes are often used individually or as a part of a gaggle. Checkboxes are objects of the Checkbox class.
Creating Checkbox : Checkbox cb = new Checkbox(Label);

Checkbox Constructor

1. Checkbox() throws HeadlessException: Creates a checkbox whose label is initially blank. The state of the checkbox is unchecked.
2. Checkbox(String str) throws HeadlessException: Creates a checkbox whose label is specified by str. The state of the checkbox is unchecked.
3. Checkbox(String str, Boolean on) throws HeadlessException: It allows you to line the initial state of the checkbox. If one is true, the checkbox is initially checked; otherwise, it's cleared.
4. Checkbox(String str, Boolean on, CheckboxGroup cbGroup) throws HeadlessException or Checkbox(String str, CheckboxGroup cbGroup, Boolean on) throws HeadlessException: It creates a checkbox whose label is specified by str and whose group is specified by cbGroup. If this checkbox isn't a part of a gaggle, then cbGroup must be null. the worth of on determines the initial state of the checkbox.

Methods of Checkbox

1. boolean getState(): To retrieve the present state of a checkbox.
2. void setState(boolean on): To line its state, call setState(). Here, if one is true, the box is checked. If it's false, the box is cleared.
3. String getLabel(): you'll obtain the present label related to a checkbox by calling getLabel().
4. void setLabel(String str): To line the label, call setLabel(). The string passed in str becomes the new label related to the invoking checkbox.

AWT Checkbox Class Declaration

public class Checkbox extends Component implements ItemSelectable, Accessible

AWT Checkbox Example import
java.awt.*;

```
public class CheckboxExample
{
    CheckboxExample(){
     Frame f= new Frame("Checkbox Example");
```

```
        Checkbox checkbox1 = new Checkbox("C++");
        checkbox1.setBounds(100,100, 50,50);
        Checkbox checkbox2 = new Checkbox("Java", true);
        checkbox2.setBounds(100,150, 50,50);

        f.add(checkbox1);
        f.add(checkbox2);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
public static void main(String args[])
{
   new CheckboxExample(); }
}
```

Radio Buttons

It is possible to make a group of mutually exclusive checkboxes during which one and just one checkbox up the group are often checked at anybody time. These checkboxes are often called radio buttons because they act just like the station selector on a car radio, only one station is often selected at anybody time. To create a group of mutually exclusive checkboxes, you want to first define the group to which they're going to belong then specify that group once you construct the checkboxes. Checkbox groups are objects of type CheckboxGroup. Only the default constructor is defined, which creates an empty group.

Creating Radiobutton :

```
CheckboxGroup cbg = new CheckboxGroup(); Checkbox
rb = new Checkbox(Label, cbg, boolean);
```

AWT CheckboxGroup Class Declaration

```
public class CheckboxGroup extends Object implements Serializable
```

Java AWT CheckboxGroup Example import
java.awt.*;

```
public class CheckboxGroupExample
{
     CheckboxGroupExample(){
     Frame f= new Frame("CheckboxGroup Example");
```

```
        CheckboxGroup cbg = new CheckboxGroup();
        Checkbox checkBox1 = new Checkbox("C++", cbg, false);
        checkBox1.setBounds(100,100, 50,50);
        Checkbox checkBox2 = new Checkbox("Java", cbg, true);
        checkBox2.setBounds(100,150, 50,50); f.add(checkBox1);
        f.add(checkBox2);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
public static void main(String args[]) {
   new CheckboxGroupExample();
} }
```

2.Develop a swing application to implement simple calculator as shown in the following figure.

Program :

```
import java.awt.event.*;
import javax.swing.*;
import java.awt.*;
class calculator extends JFrame implements ActionListener {
        // create a frame
        static JFrame f;

        // create a textfield static
        JTextField l;

        // store operator and operands String
        s0, s1, s2;

        // default constructor
        calculator()
        {
                s0 = s1 = s2 = "";
        }

        // main function
        public static void main(String args[]) {
                // create a frame
                f = new JFrame("calculator");
```

```
		try {
			// set look and feel

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
		}
		catch (Exception e) {
			System.err.println(e.getMessage());
```

```
// create a object of class calculator c =
new calculator();

// create a textfield
l = new JTextField(16);

// set the textfield to non editable
l.setEditable(false);

// create number buttons and some operators
JButton b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, ba, bs, bd, bm, be, beq, beq1;

// create number buttons 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");

// equals button
beq1 = new JButton("=");

// create operator buttons ba =
new JButton("+"); bs = new
JButton("-"); bd = new
JButton("/"); bm = new
JButton("*"); beq = new
JButton("C");

// create . button
be = new JButton(".");

// create a panel
JPanel p = new JPanel();

// add action listeners
bm.addActionListener(c);
bd.addActionListener(c);
bs.addActionListener(c);
ba.addActionListener(c);
b9.addActionListener(c);
b8.addActionListener(c);
```

```
        b7.addActionListener(c);
        b6.addActionListener(c);
        b5.addActionListener(c);
        b4.addActionListener(c);
        b3.addActionListener(c);
        b2.addActionListener(c);
        b1.addActionListener(c);
        b0.addActionListener(c);
        be.addActionListener(c);
        beq.addActionListener(c);
        beq1.addActionListener(c);

        // add elements to panel p.add(l);
        p.add(ba);
        p.add(b1);
        p.add(b2);
        p.add(b3);
        p.add(bs);
        p.add(b4);
        p.add(b5);
        p.add(b6);
        p.add(bm);
        p.add(b7);
        p.add(b8);
        p.add(b9);
        p.add(bd);
        p.add(be);
        p.add(b0);
        p.add(beq);
        p.add(beq1);

        // set Background of panel
        p.setBackground(Color.blue);

        // add panel to frame f.add(p);

        f.setSize(200, 220);
        f.show();
}
public void actionPerformed(ActionEvent e) {
        String s = e.getActionCommand();

        // if the value is a number
        if ((s.charAt(0) >= '0' && s.charAt(0) <= '9') || s.charAt(0) == '.') {
                // if operand is present then add to second no
                if (!s1.equals(""))
                        s2 = s2 + s;
                else
                        s0 = s0 + s;
```

```
            // set the value of text
            l.setText(s0 + s1 + s2);
else if (s.charAt(0) == 'C') {
            // clear the one letter s0 = s1
            = s2 = "";

            // set the value of text
            l.setText(s0 + s1 + s2); }
else if (s.charAt(0) == '=') {

            double te;

            // store the value in 1st
            if (s1.equals("+"))
                        te = (Double.parseDouble(s0) + Double.parseDouble(s2)); else if
            (s1.equals("-"))
                        te = (Double.parseDouble(s0) - Double.parseDouble(s2)); else if
            (s1.equals("/"))
                        te = (Double.parseDouble(s0) / Double.parseDouble(s2)); else
                        te = (Double.parseDouble(s0) * Double.parseDouble(s2));

            // set the value of text l.setText(s0 + s1 +
            s2 + "=" + te);

            // convert it to string
            s0 = Double.toString(te);

            s1 = s2 = "";
} else {
            // if there was no operand
            if (s1.equals("") || s2.equals(""))
                        s1 = s;
            // else evaluate
            else {
                        double te;

                        // store the value in 1st
                        if (s1.equals("+"))
                                    te = (Double.parseDouble(s0) + Double.parseDouble(s2)); else if
                        (s1.equals("-"))
                                    te = (Double.parseDouble(s0) - Double.parseDouble(s2)); else if
                        (s1.equals("/"))
                                    te = (Double.parseDouble(s0) / Double.parseDouble(s2)); else
                                    te = (Double.parseDouble(s0) * Double.parseDouble(s2));

                        // convert it to string
                        s0 = Double.toString(te);
```

```
                    // place the operator
                    s1 = s;

                    // make the operand blank s2 = "";
                }

                // set the value of text
                l.setText(s0 + s1 + s2);
            }
        }
```

} **Output :**