Calculator Program

```
import java.awt.*;
import java.awt.event.*;
public class calculator extends java.applet.Applet implements ActionListener {
TextField txtTotal = new TextField("");
Button button[] = new Button[10];
Button divide = new Button("/");
Button mult = new Button("*");
Button plus = new Button ("+");
Button minus = new Button("-");
Button isequalto = new Button("=");
Button clear = new Button("CA");
double num ,numtemp ;
 int counter;
 String strnum = "",strnumtemp = "" ;
 String op = "";
  public void operation() {
    counter ++;
        if (counter == 1) {
         numtemp = num;
         strnum = "";
        num = 0;
         }else{
          if (op == "+") numtemp += num;
          else if (op == "-") numtemp -= num;
         else if (op == "*") numtemp = numtemp * num;
         else if (op == "/") numtemp = numtemp / num;
          strnumtemp = Double.toString(numtemp);
          txtTotal.setText(strnumtemp);
          strnum = "";
          num = 0;
         }
     }
   public void init() {
      setLayout(null);
      plus.setBackground(Color.blue);
      plus.setForeground(Color.white);
      minus.setBackground(Color.blue);
      minus.setForeground(Color.white);
      divide.setBackground(Color.blue);
      divide.setForeground(Color.white);
      isequalto.setBackground(Color.blue);
      isequalto.setForeground(Color.white);
      mult.setBackground(Color.blue);
      mult.setForeground(Color.white);
      clear.setBackground(Color.blue);
      clear.setForeground(Color.red);
   for(int i = 0;i <= 9; i ++) {
         button[i] = new Button(String.valueOf(i));
```

```
        button[i].setBackground(Color.orange);
         button[i].setForeground(Color.blue);
     }
     button[1].setBounds(0,53,67,53);
     button[2].setBounds(67,53,67,53);
     button[3].setBounds(134,53,67,53);
     button[4].setBounds(0,106,67,53);
     button[5].setBounds(67,106,67,53);
     button[6].setBounds(134,106,67,53);
     button[7].setBounds(0,159,67,53);
     button[8].setBounds(67,159,67,53);
     button[9].setBounds(134,159,67,53);
 for (int i = 1;i <= 9; i ++) {
         add(button[i]);
     }
     txtTotal.setBounds(0,0,200,53);
    add(txtTotal);
     plus.setBounds(0,212,67,53);
     add(plus);
     button[0].setBounds(67,212,67,53);
     add(button[0]);
     minus.setBounds(134,212,67,53);
     add(minus);
     divide.setBounds(134,264,67,53);
     add(divide);
     isequalto.setBounds(67,264,67,53);
     add(isequalto);
     mult.setBounds(0,264,67,53);
     add(mult);
     add(clear);
    }
    public void start() {
     for(int i = 0;i <= 9; i ++) {
         button[i].addActionListener(this);
     }
     plus.addActionListener(this);
     minus.addActionListener(this);
    divide.addActionListener(this);
     mult.addActionListener(this);
     isequalto.addActionListener(this);
     clear.addActionListener(this);
   }
    public void stop() {
     for(int i = 0;i <= 9; i ++) {
    button[i].addActionListener(null);
     }
     plus.addActionListener(null);
     minus.addActionListener(null);
     divide.addActionListener(null);
     mult.addActionListener(null);
```

```
 isequalto.addActionListener(null);
 clear.addActionListener(null);
}
public void actionPerformed(ActionEvent e) {
    for(int i = 0;i <= 9; i++) {
        if (e.getSource() == button[i]) {
            play(getCodeBase(),i + ".au");
           strnum += Integer.toString(i);
            txtTotal.setText(strnum);
            num = Double.valueOf(strnum).doubleValue();
        }
    }
    if (e.getSource() == plus) {
     operation();
     op = "+";
    }
   if (e.getSource() == minus) {
     operation();
    op = "-";
  }
    if (e.getSource() == divide) {
     operation();
     op = "/";
    }
    if (e.getSource() == mult) {
     operation();
     op = "*";
    }
   if (e.getSource() == isequalto) {
     if (op == "+") numtemp += num;
     else if (op == "-") numtemp -= num;
    else if (op == "*") numtemp = numtemp * num;
     else if (op == "/") numtemp = numtemp / num;
     strnumtemp = Double.toString(numtemp);
     txtTotal.setText(strnumtemp);
    strnumtemp = "";
     numtemp = 0;
     strnum = "";
     num = 0;
    counter = 0;
    }
    if (e.getSource() == clear) {
     txtTotal.setText("0");
     strnumtemp = "";
     numtemp = 0;
     strnum = "";
    num = 0;
     counter = 0;
    }
} }
```

cal.html

```
<html>
<applet code=”calculator.class” width=400 height=400>
</applet>
</html>
```

Q. What is the purpose of Text Field. List and explain it’s constructors and important methods.

Ans : Text Field: This is also the text container component of Java AWT package. This component contains single line and limited text information. This is declared as follows :

TextField txtfield = new **TextField**(20);

we can fix the number of columns in the text field by specifying the number in the constructor. In the above code we have fixed the number of columns to 20.

TextField Example

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class TextFieldDemo extends Applet implements ActionListener
  {
  TextField name,pass;
  public void init()
   {
   Label namep=new Label("Name: ",Label.RIGHT);
   Label passp=new Label("Password:",Label.RIGHT);
   name=new TextField(12);
   pass=new TextField(8);
   add(namep);
   add(name);
   add(passp);
   add(pass);
   name.addActionListener(this);
   pass.addActionListener(this);
   }
 public void actionPerformed(ActionEvent ae)
   {
   repaint();
   }
 public void paint(Graphics g)
   {
   g.drawString("Name: "+name.getText(),6,60);
   g.drawString("Selected text in name :"+name.getSelectedText(),6,80);
   g.drawString("Password:" +pass.getText(),6,100);
}}
```

Event Handling

Events :

In the delegation event model, an event is an object that describes a state change in a source. It can be generated as a consequence of a person interacting with the elements in a graphical user interface. Some of the activities that cause events to be generated are pressing a button, entering a character via the keyboard, selecting an item in a list, and clicking the mouse.

Event listeners

A listener is an object that is notified when an event occurs. It has two major requirements. First, it must have been registered with one or more sources to receive notifications about specific types of events. Second, it must implement methods to receive and process these notifications.

Note : The methods that receive and process events are defined in a set of interfaces found in java.awt.event.

Event Listener Interface

Interface	**Description**
ActionListener	Defines one method to receive action events.
AdjustmentListener	Defines one method to receive adjustment events.
ItemListener	Defines one method to recognize when the state of an item changes.
KeyListener	Defines three methods to recognize when a key is pressed, released, or typed.
MouseListener	Define five methods to recognize when the mouse is clicked, enters a component, exits a component, is pressed, or is released.
MouseMotionListener	Defines two methods to recognize when the mouse is dragged or moved.
TextListener	Defines one method to recognize when a text value changes.
WindowListener	Defines seven methods to recognize when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

The ActionListener interface

This interface defines the actionPerformed() method that is invoked when an action
Event occours.

Syntax : void actionPerformed(ActionEvent ae)

The ItemListener interface

This listener defines the itemStateChanged() method that is invoked when the state
Of an item changes.

Syntax : void itemStateChanged(ItemEvent ie)

The keyListener interface

The interface defines three methods :

void keyPressed(KeyEvent ke)
void keyReleased(KeyEvent ke)
void keyTyped(KeyEvent ke)

The MouseListener interface

This interface defines five methods.

void mouseClicked(MouseEvent me)
void mouseEntered(MouseEvent me)
void mouseExited(MouseEvent me)
void mousePressed(MouseEvent me)
void mouseReleased(MouseEvent me)

The MouseMotionListener interface

This interface defines two methods.

void mouseDragged(MouseEvent me)
void mouseMoved(MouseEvent me)

The TextListener interface

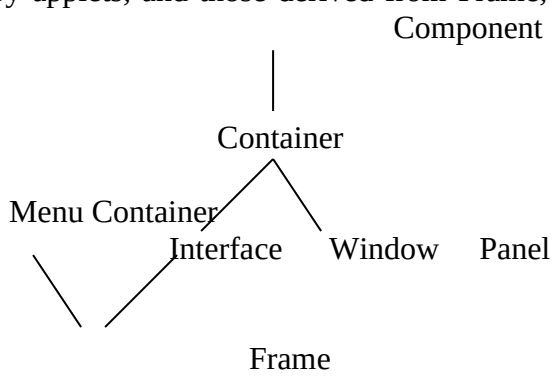
Syntax : void textChanged(TextEvent te)

AWT

AWT stands for **Abstract Window ToolKit.** The Abstract Windowing Toolkit (AWT) is Java's platform-independent windowing, graphics, and user-interface widget toolkit. The AWT is part of the Java Foundation Classes (JFC) - the standard API for providing a graphical user interface (GUI) for a Java program.

Window Fundamentals

The AWT defines windows according to a class hierarchy that adds functionality and specify with each level. The two most common windows are those derived from panel, which is used by applets, and those derived from Frame, which creates a standard window.



Component

At the top the AWT hierarchy is the Component class. Component is an abstract class that encapsulates all of the attributes of a visual component. All user interface elements that are displayed on the screen and that interact with the user are subclasses managing events, such as mouse and keyboard input, positioning and sizing the window, and repainting. A component object is responsible for