# ******************Event Handling******************

## ##### Delegation Event Model ######

- The delegation event model defines the standard background mechanism to process on generated event.

- In this model, Source generates events and send that event to the event listener.

- *****Event:**

- Event is an action performed on any source.

- Event is a result's of user action which changes the state of source.

- Source will generates the event.

- **Example:-**

clicking on push button, pressing key from keyboard, clicking mouse button, selecting item from list, check or uncheck checkbox.etc

- *****Source:-**

- Source is a component object on which action is performed.

- Source can generates more than one type of events.

- It is necessary that source must register listener.

- Each event has its own registration method.

- **-Example:-**

Sources :- Button, Choice, List, Checkbox, Menuitem,TextArea,TextField,etc

- We can use following method to register listener on source.

void addActionListener(), void addItemListener(), void addKeyListener(), void addWindowListener()

- We can remove the registration of listener.

- **Example:-**

removeActionListener(), removeItemListener(),removeWindowListener(), etc

- *****Listener:-**

- Listener is an object which gets notified when event occurs.

- Listener listen the event which is generated by source and take action accordingly(called particular method).

- **Example:-**

MouseListener, ActionListener, WindowListener, ItemListener, etc

- We require to import java.awt.event package for event handling.

- **-Diagram:-**

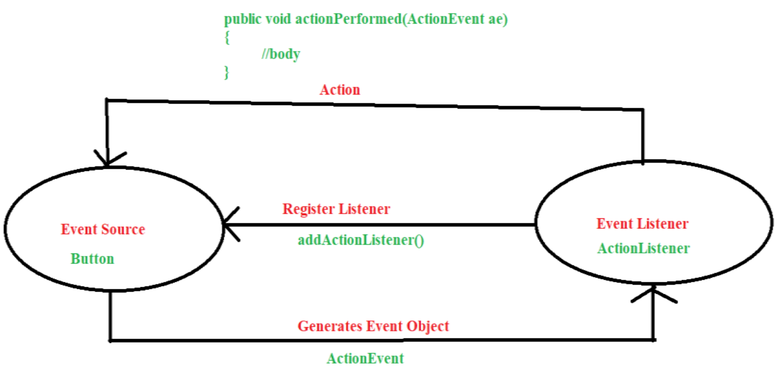


Figure: Delegation Event Model

## #### Class -> ActionEvent & Interface -> ActionListener ####

- **ActionEvent:-**

- An ActionEvent class generates event when when you press button, list item is double clicked, menu-item is selected.

- *****Constructor:-**

1) ActionEvent(Object Source,int type, String cmd)

2) ActionEvent(Object Source, int type, String cmd, int modifier)

3) ActionEvent(Object source, int type, String cmd, long when, int modifier)

modifier - ALT_MASK, CLTR_MASK, SHIFT_MASK, META_MASK, ACTION_PERFORMED

- String getActionCommand() - it will return label string of component.

- int getModifier() - it will return modifier value.

- long getWhen() - it return time of event.

- **ActionListener:-**

- This is interface which defines following method that is invoked when an action event occurs.

```
public void actionPerformed(ActionEvent ae)

{

        //body of method

}
```

## #### Class -> ItemEvent & Interface -> ItemListener ####

- **ItemEvent:-**

- ItemEvent class generates event when we select checkbox, when checkable menu item is selected, etc.

- There are two types of ItemEvent.

1) DESELECTED - When user deselected an item.

2) SELECTED - When user select an item.

- ItemEvent class defines one integer constant ITEM_STATE_CHANGED

- *****Constructor:-**

1) ItemEvent(ItemSelectable source, int type, Object entry, int state)

- *****Methods:-**

1) Object getItem();

2) ItemSelectable getItemSelectable()

3) int getStateChange()

- **ItemListener:-**

- ItemListener interface executes following method when ItemEvent occurs.

```
public void itemStateChanged(ItemEvent ie)
{
        //body
}
```

## Class => ComponentEvent & Interface =>ComponentListener

- **ComponentEvent:-**

- ComponentEvent class generates event when certain parameters value got changes like visibility, size, position, etc

- There are four component related events:

1) COMPONENT_HIDDEN

2) COMPONENT_SHOWN

3) COMPONENT_MOVED

4) COMPONENT_RESIZED

- **ComponentListener:-**

- It contains four different methods:

1) void componentHidden(ComponentEvent ce)

2) void componentShown(ComponentEvent ce)

3) void componentMoved(ComponentEvent ce)

4) void componentResized(ComponentEvent ce)

## Class => ContainerEvent & Interface =>ContainerListener

- **ContainerEvent:-**

- This will generates event when we add or remove components from the container.

- *****Constructor:-**

1)ComponentEvent(Component src, int type)

src - Object which generates event

type - type of event.

- **ContainerListener:-**

- This interface contains two methods

void componentAdded(ContainerEvent ce)

void componentRemoved(ContainerEvent ce)

## ##### Class => KeyEvent & Interface =>KeyListener #####

- **KeyEvent:-**

- When keyboard input is occurred then KeyEvent is generated.

- There are three different types of key event.

1) KEY_PRESSED

2) KEY_RELEASED

3) KEY_TYPED

✓ **There are different integer constant defined by KeyEvent class…**

| | |
|---|---|
| VK_ALT | VK_LEFT |
| VK_CANCEL | VK_PAGE_DOWN |
| VK_CONTRO | VK_PAGE_UP |
| VK_DOWN | VK_RIGHT |
| VK_ENTER | VK_SHIFT |
| VK_ESCAPE | VK_UP |

**VK Means - Virtual Key…**

- *****Constructor:-**

1)KeyEvent(Component src, int type, long when, int modifier, int code, char ch)

- *****Methods:-**

char getKeyChar();

int getKeyCode();

- **KeyListener:-**

-In this interface, following methods are present.

1) void KeyPressed(KeyEvent ke)

2) void KeyReleased(KeyEvent ke)

3) void KeyTyped(KeyEvent ke)

## Class => MouseEvent & Interface =>MouseListener & MouseMotionListener

- **MouseEvent:-**

- MouseEvent class will generates event when mouse buttons are pressed.

- Following are the integer constants related to mouse event.

1) MOUSE_CLICKED - when the user clicked the mouse

2) MOUSE_DRAGGED - When the user dragged the mouse.

3) MOUSE_ENTERED - When the mouse entered a component.

4) MOUSE_EXITED - When the mouse is exited from the component

5) MOUSE_MOVED - when the mouse moved

6) MOUSE_PRESSED - when the mouse was pressed.

7) MOUSE_RELEASED -When the mouse was released

8) MOUSE_WHEEL - When the mouse wheel was moved

- *****Constructor:-**

  MouseEvent(Component src, int type, long when, int modifier, int x, int y, int clicks, boolean triggerpopup)

- *****Methods:-**

  1) int getX()

  2) int getY()

  3) int getClickCount()

4) boolean isPopupTrigger()

## MouseListener & MouseMotionListener:-

- **MouseListener:-**

1) void mouseClicked(MouseEvent me)

2) void mouseEntered(MouseEvent me)

3) void mouseExited(MouseEvent me)

4) void mousePressed(MouseEvent me)

5) void mouseReleased(MouseEvent me)

- **MouseMotionListener:-**

1) void mouseDragged(MouseEvent me)

2) void mouseMoved(MouseEvent me)

## ##Class => WindowEvent & Interface =>WindowListener ##

- **WindowEvent:-**

- WindowEvent class defines 10 types of different window events.

- Following are the different integer constants:

1) WINDOW_ACTIVATED

2) WINDOW_DEACTIVITED

3) WINDOW_OPENED

4) WINDOW_CLOSED

5) WINDOW_CLOSING

6) WINDOW_ICONIFIED

7) WINDOW_DEINCONFIED

8) WINDOW_GAIN_FOCUS

9) WINDOW_LOST_FOCUS

10) WINDOW_STATE_CHANGED

- *****Constructor:-**

1) WindowEvent(Window src, int type)

2) WindowEvent(Window src, int type, Window other)

3) WindowEvent(Window src, int type,int fromstate, int tostate)

4) WindowEvent(Window src, int type, Window other, int fromstate, int tostate)

- *****Methods:-**

- Window getOppositeWindow()

- int getOldState()

- int getNewState()

- **WindowListener:-**

- WindowListener listen WindowEvent.

- Following methods are called when WindowEvent occurred.

1) void windowActivated(WindowEvent we)

2) void windowDeactivated(WindowEvent we)

3) void windowOpened(WindowEvent we)

4) void windowClosed(WindowEvent we)

5) void windowClosing(WindowEvent we)

6) void windowIconfied(WindowEvent we)

7) void windowDeiconfied(WindowEvent we)

8) void windowLostFocus(WindowEvent we)

## ##### Class => TextEvent & Interface =>TextListener #####

- **TextEvent:-**

- TextEvent class generates event when user or program enter some characters in TextField or TextArea.

- The class TextEvent defines integer constant TEXT_VALUE_CHANGED

- *****Constructor:-**

TextEvent(Object src, int type)

- **TextListener:-**

- void textValueChanged(TextEvent te)

****************************************************************

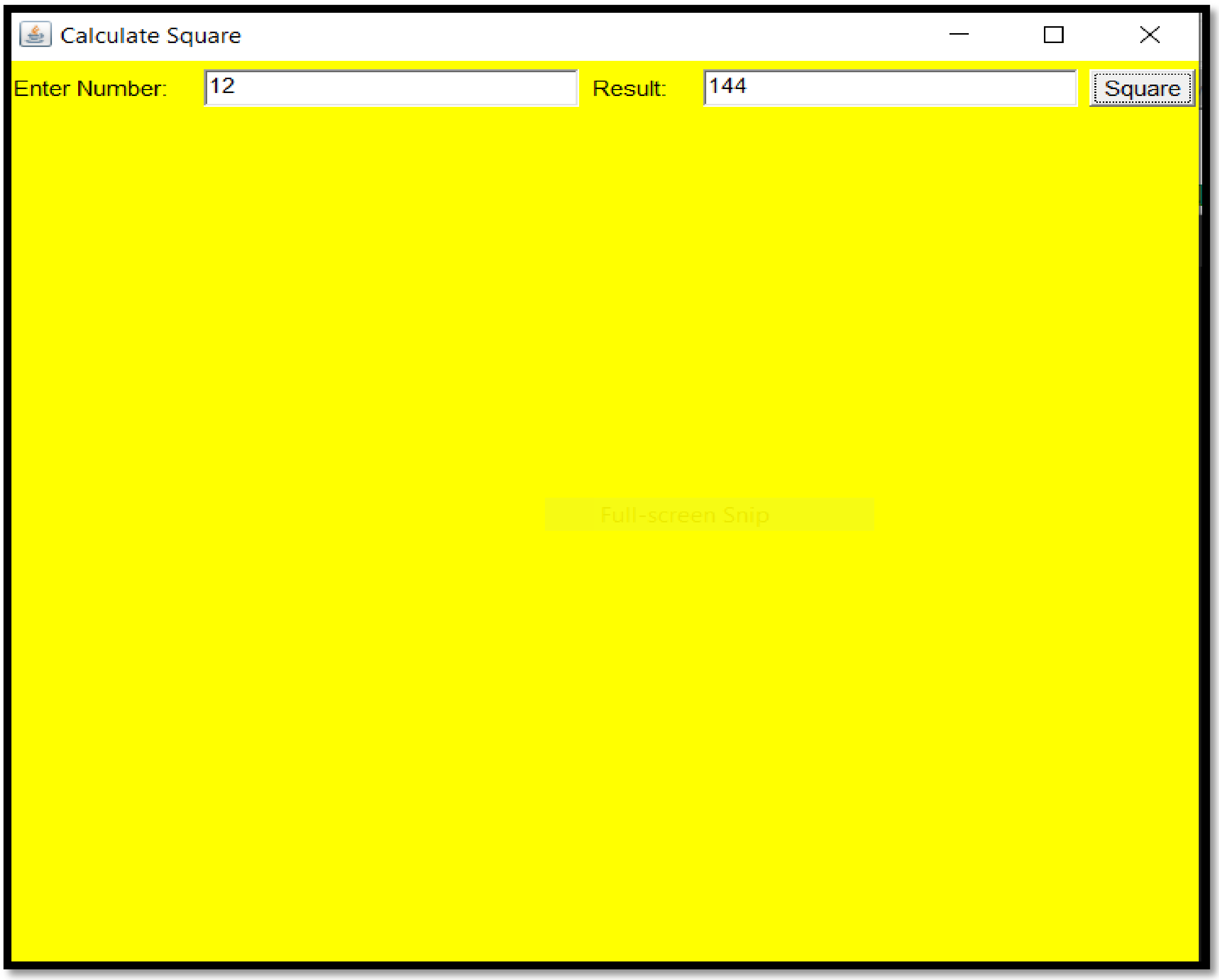
- **Code:-**

```
  /*
        Write a program to find out square of given input number...
        When the mouse cursor enters into textfield it should get cleared.
 */
import java.awt.*;
import java.awt.event.*;
class SquareDemo extends Frame
{
        Button b1;

        TextField tf1,tf2;

        Label L1,L2;

        SquareDemo()

        {

                FlowLayout f1=new FlowLayout();

                setLayout(f1);


                setBackground(Color.yellow);


                L1=new Label("Enter Number:");

                L2=new Label("Result:");


                tf1=new TextField(20);

                tf2=new TextField(20);


                b1=new Button("Square");


                b1.addActionListener(new InnerForSquare());

                tf1.addMouseListener(new InnerForClear());


                add(L1);add(tf1);

                add(L2);add(tf2);
```

```
          add(b1);

      }

      class InnerForSquare implements ActionListener

      {

                  public void actionPerformed(ActionEvent ae)

                  {

                      int  no=Integer.parseInt(tf1.getText());

                      tf2.setText(""+(no*no));

                  }

      }

      class InnerForClear implements MouseListener

      {

              public void mouseEntered(MouseEvent me)

              {

                      tf1.setText(null);

                      tf2.setText(null);

              }

              public void mouseExited(MouseEvent me){}

              public void mouseReleased(MouseEvent me){}

              public void mousePressed(MouseEvent me){}

              public void mouseClicked(MouseEvent me){}

      }

      public static void main(String args[])
```

```
                {
                                SquareDemo s1=new SquareDemo();
                                s1.setVisible(true);
                                s1.setTitle("Calculate Square");
                                s1.setSize(600,600);
                }
}
```

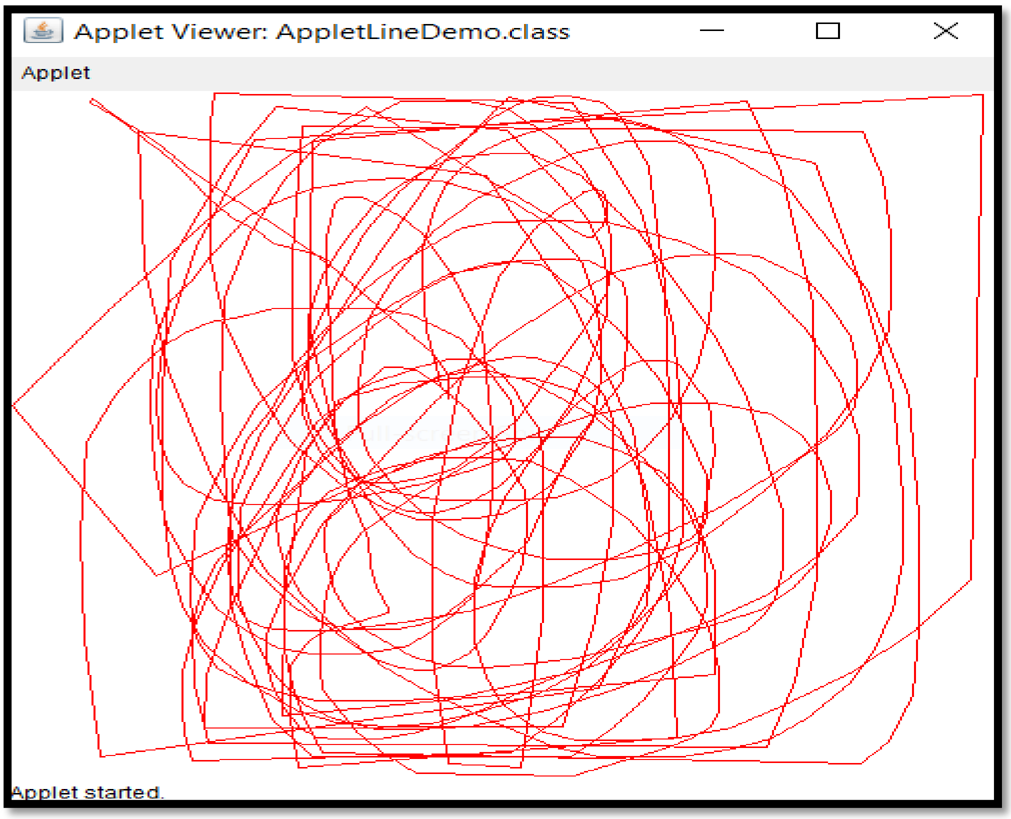
- **Output:-**



- **Code:-**

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class AppletLineDemo extends Applet
{
    int x1=0,y1=0;
    int flag=1;
    public void init()
    {
        addMouseListener(new InnerClicked());
        addMouseMotionListener(new InnerMoved());
    }
    class InnerClicked extends MouseAdapter
    {
        public void mouseClicked(MouseEvent me)
        {
            if(flag==1)
            {
                x1=me.getX();
                y1=me.getY();
                flag=2;
            }
        }
    }
    class InnerMoved extends MouseMotionAdapter
    {
        public void mouseMoved(MouseEvent me)
        {
                        int x=me.getX();
                        int y=me.getY();
                        Graphics g=getGraphics();
                        if(flag==2)
                        {
                                g.setColor(Color.red);
                                g.drawLine(x,y,x1,y1);
                                x1=x;
                                y1=y;
                        }
        }
    }
}

/*<applet code="AppletLineDemo.class" width=500 height=500>
</applet>*/
```

- **Output:-**



******************************************************

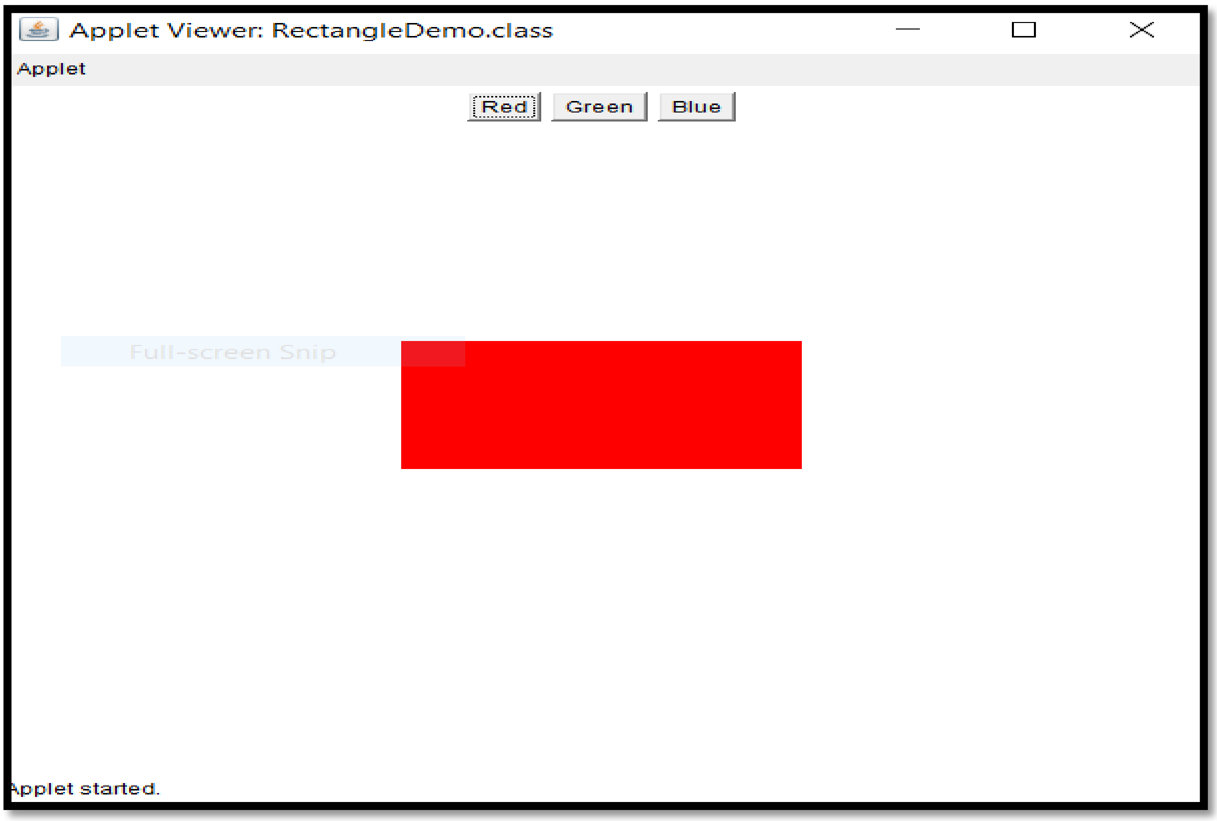
- **Code:-**

```
import java.applet.*;

import java.awt.*;

import java.awt.event.*;

public class RectangleDemo extends Applet implements
ActionListener

{

    int flag=0;

    public void init()

    {
```

```
                    Button b1=new Button("Red");
                    Button b2=new Button("Green");
                    Button b3=new Button("Blue");
                    add(b1);add(b2);add(b3);
                    b1.addActionListener(this);
                    b2.addActionListener(this);
                    b3.addActionListener(this);

          }
          public void actionPerformed(ActionEvent ae)
          {
                         String str=ae.getActionCommand();
                         if(str.equals("Red"))
                         {
                                   flag=1;
                                   repaint();
                         }
                         else if(str.equals("Green"))
                         {
                              flag=2;
                              repaint();
                         }
                         else if(str.equals("Blue"))
                    {
                        flag=3;
                        repaint();
                    }


        }
        public void paint(Graphics g)
        {
                if(flag==1)
                {
                    g.setColor(Color.red);
                }
                else if(flag==2)
                {
                    g.setColor(Color.green);
                }
                else if(flag==3)
                {
                    g.setColor(Color.blue);
                }
                g.fillRect(200,200,200,100);
        }
      }
```

```
/*<applet code="RectangleDemo.class" width=600 height=600>

</applet>*/
```

- **Output:-**



************************************************************

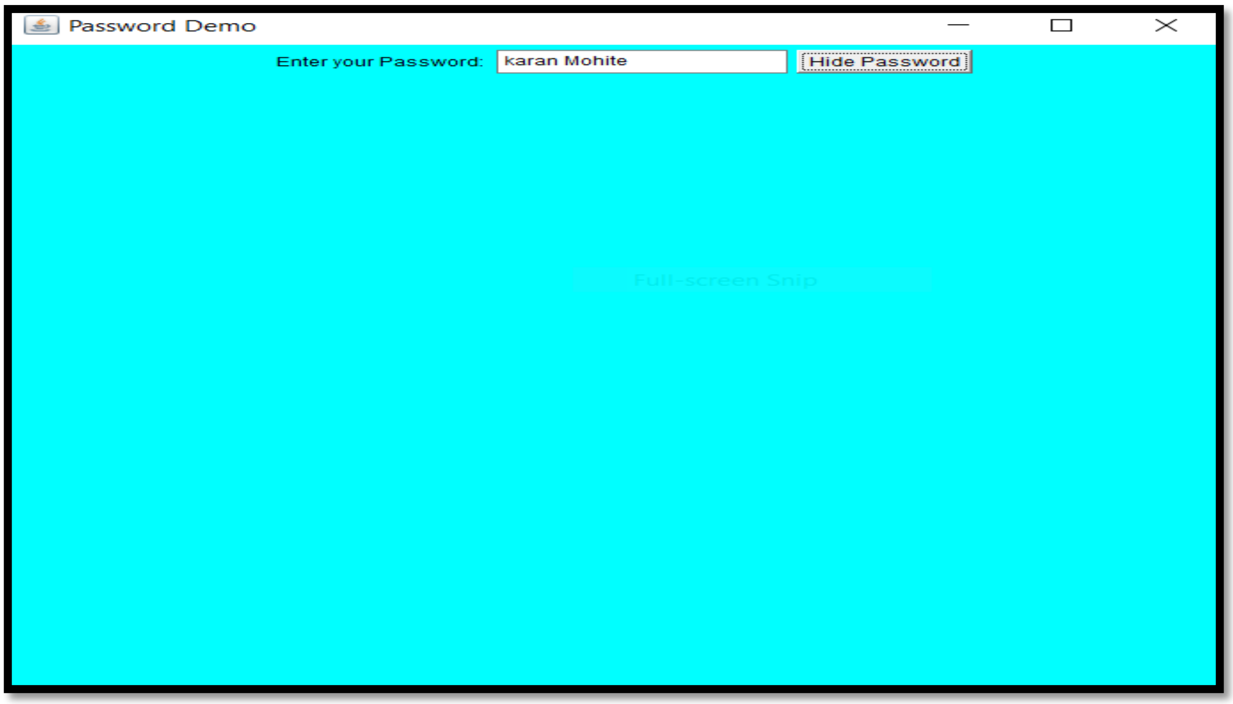
- **Code:-**

```
import java.awt.*;

import java.awt.event.*;

class PasswordDemo extends Frame implements ActionListener

{

        Button b1;

        TextField tf1;

        PasswordDemo()
```

```
{
        FlowLayout f1=new FlowLayout();
        setLayout(f1);
        setBackground(Color.cyan);
        Label L1=new Label("Enter your Password:",Label.RIGHT);
        b1=new Button("See Password");
        tf1=new TextField(20);
        tf1.setEchoChar('*');

        b1.addActionListener(this);

        add(L1); add(tf1); add(b1);
}
public void actionPerformed(ActionEvent ae)
{
        String str=ae.getActionCommand();
        if(str.equals("See Password"))
        {
                tf1.setEchoChar('\0');
                b1.setLabel("Hide Password");
        }
        else if(str.equals("Hide Password"))
        {
```

```
                        tf1.setEchoChar('*');

                        b1.setLabel("See Password");

                }


            }

            public static void main(String args[])

            {

                PasswordDemo p1=new PasswordDemo();

                p1.setTitle("Password Demo");

                p1.setSize(700,700);

                p1.setVisible(true);

            }

    }
```

- **Output:-**



*Inspiring Your Success*

VJTech Academy…