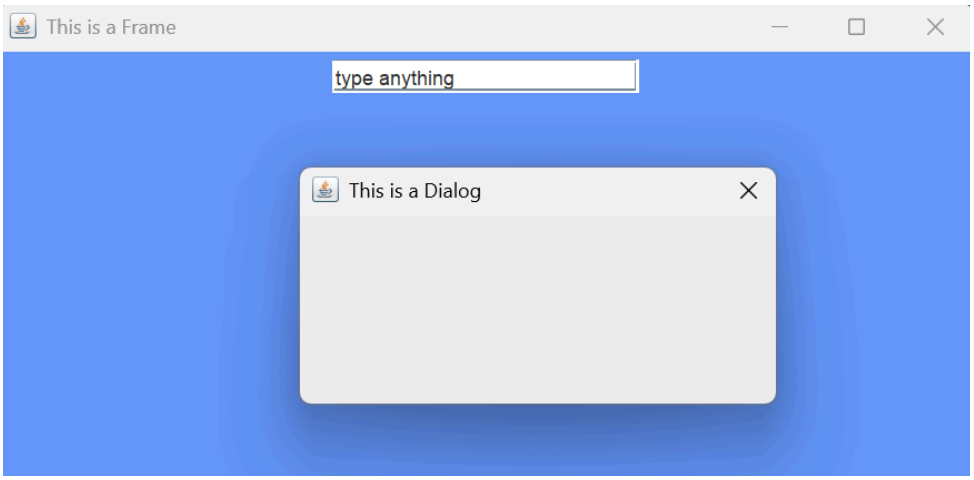
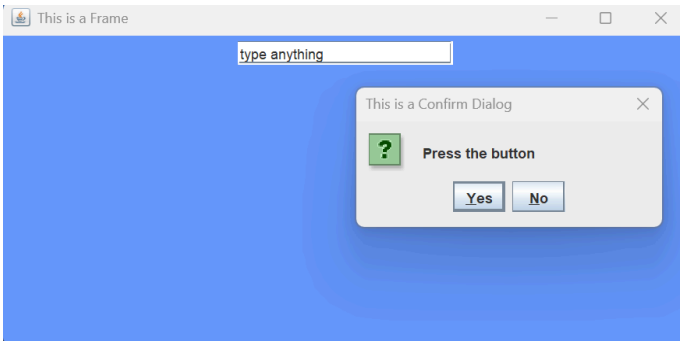
# Paradigm finals cheat sheet

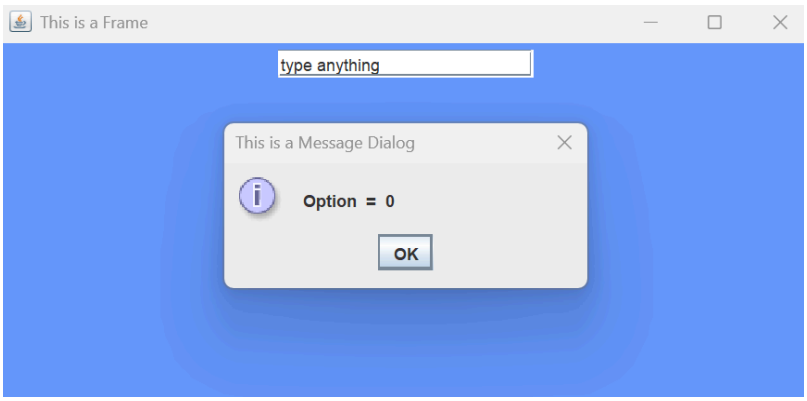
Tough luck. May God us all.

By Pinpin

# w8_1_Container

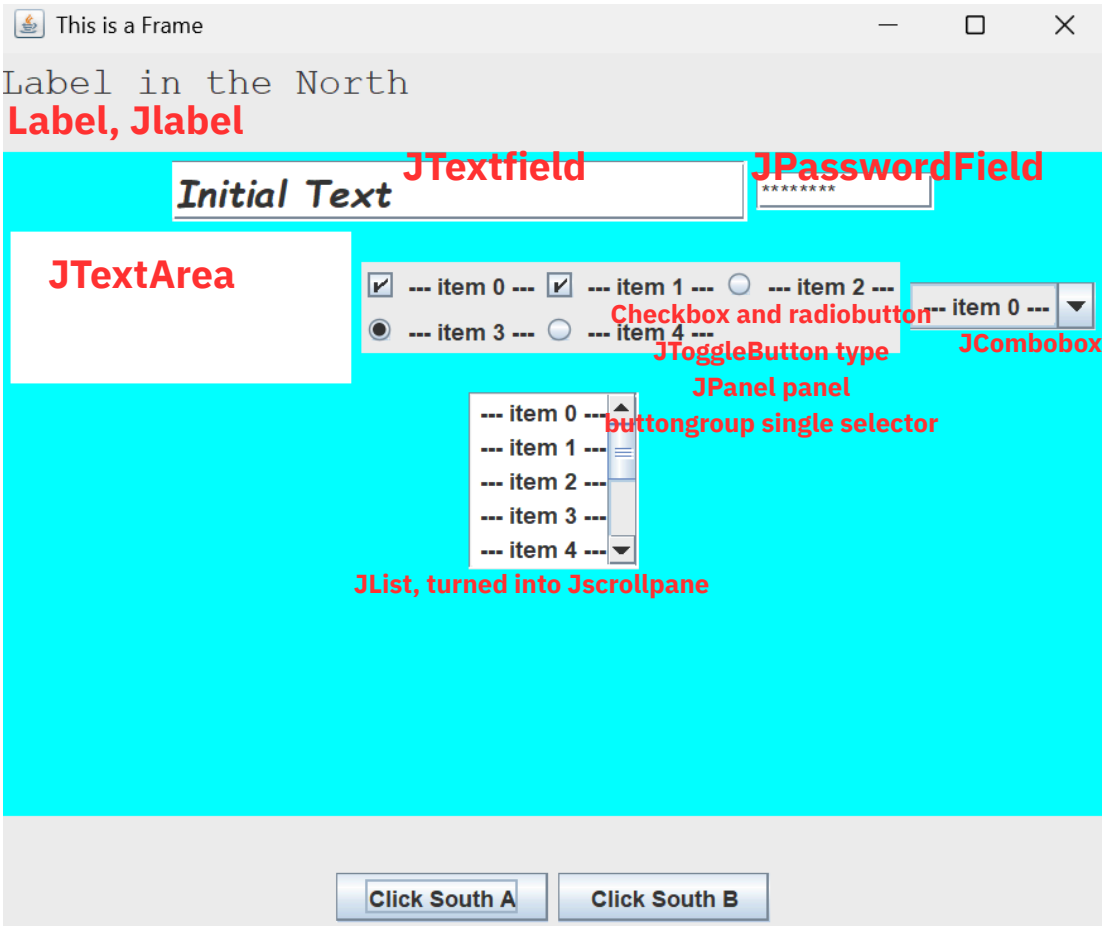


```java
package Lab_Ch8;

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

class w8_1_Container
{
    public static void main(String[] args)
    {
    JFrame frame = new JFrame("This is a Frame");

    // ----- (1) Minimum properties that must be set
    frame.setBounds(200, 200, 600, 300);
    frame.setVisible(true);


    // ----- (2) Other properties
    //           - which one doesn't work ?
    //           - notice the difference between EXIT_ON_CLOSE and DISPOSE_ON_CLOSE

        frame.setMaximizedBounds( new Rectangle(800, 500) );
    frame.setResizable(true);
    frame.setBackground(Color.GREEN);
    frame.getContentPane().setBackground( new Color(100, 150, 250) );
    //frame.setBackground( Color.BLUE );
    frame.setCursor( new Cursor(Cursor.HAND_CURSOR) );
    frame.setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );

        frame.getContentPane().setLayout(new FlowLayout());
        frame.getContentPane().add( new JTextField("type anything", 20) );



        // ----- (1.1) If the frame is visible before components are added,
        //             must call validate or pack to trigger layout manager & update screen
        //frame.validate();
        //frame.pack();

        // ----- (1.2) Or add components first and make frame (with components) visible
        //             Don't need to call validate or pack
        frame.setVisible(true);



    // ----- (3) Dialog
    //           - notice the difference between dialog with and without owner
    //           - notice the difference between modal and non-modal dialogs

    JDialog dialog = new JDialog();
    dialog.setTitle("This is a Dialog");
    dialog.setModal(true);
    dialog.setBounds(400, 400, 300, 150 );
    dialog.setVisible(true);
    dialog.setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE );




    // ----- (4) Quick dialog
        //           - notice the difference between dialog with and without owner

    int op = JOptionPane.showConfirmDialog( null,
         "Press the button", "This is a Confirm Dialog",
             JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE );

    JOptionPane.showMessageDialog( frame,
         ("Option  =  " + op), "This is a Message Dialog",
         JOptionPane.INFORMATION_MESSAGE );

    }
}

```

# w8_2_Component

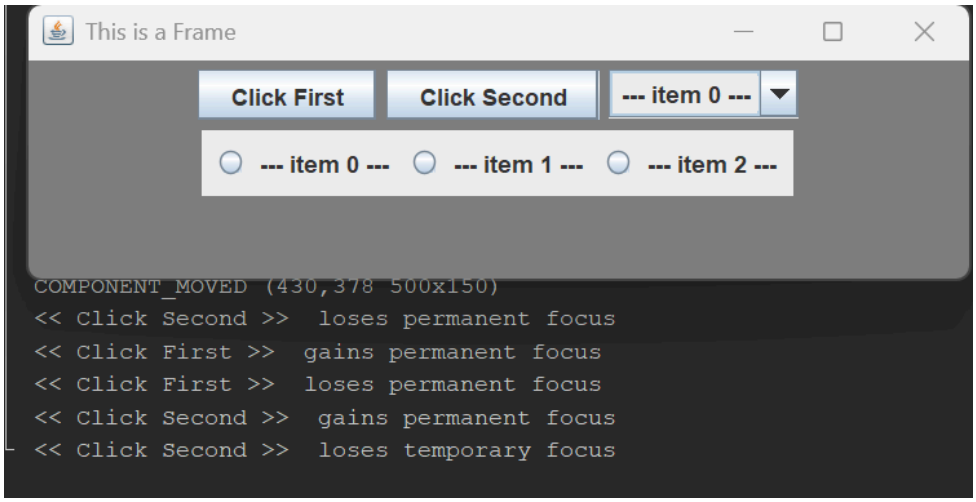


# w8_2_Component

```java
package Lab_Ch8;

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

class w8_2_Component extends JFrame
{
    public w8_2_Component()
    {
    this.setTitle("This is a Frame");
    this.setBounds(200, 100, 600, 500);
        //this.setSize(600, 500); this.setLocationRelativeTo(null);
    this.setVisible(true);
    this.setResizable(true);
    this.setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );


    // ----- (1) setup layout
    JPanel contentpane = (JPanel)getContentPane();
    contentpane.setLayout( new BorderLayout(25, 25) ); //this seems to widen the size of "label in the north"


    // ----- (2) setup components
    String [] items = new String[10];
    for (int i=0; i < 10; i++) items[i] = " --- item " + i + " ---";

    JLabel  label   = new JLabel("Label in the North");
        label.setFont(new Font("Monospaced", Font.PLAIN, 20));

    JButton button1 = new JButton("Click South A");
    JButton button2 = new JButton("Click South B");

    JPanel cregion = new JPanel();
        cregion.setBackground( Color.CYAN );


    // ----- (3) add components to container
    //           notice 2 buttons in the south
    contentpane.add(label, BorderLayout.NORTH);
    contentpane.add(cregion, BorderLayout.CENTER);
    //contentpane.add(button1, BorderLayout.SOUTH);
    //contentpane.add(button2, BorderLayout.SOUTH);



    JPanel southPanel = new JPanel();
    southPanel.add(button1);
    southPanel.add(button2);
    contentpane.add(southPanel, BorderLayout.SOUTH);



    // ----- (5) set cursor to other formats
    button1.setCursor( new Cursor(Cursor.MOVE_CURSOR) );


    // ----- (6) add text field, password field, text area

    JTextField tf = new JTextField("Initial Text", 20);
    tf.setFont(new Font("Comic Sans MS", Font.BOLD+Font.ITALIC, 20));
    cregion.add(tf);

    JPasswordField pf = new JPasswordField("password", 10);
    pf.setEchoChar('*');
    cregion.add(pf);

    JTextArea ta = new JTextArea(5, 20);
    cregion.add(ta);
        //JScrollPane tsc = new JScrollPane(ta);
        //tsc.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    //cregion.add(tsc);

```

```
73
74
75      // ----- (7) add check box, radio button
76          //           try different GridLayout
77
78      JToggleButton [] tb = new JToggleButton[5];
79      tb[0] = new JCheckBox( items[0], true );
80      tb[1] = new JCheckBox( items[1], true );
81      tb[2] = new JRadioButton( items[2] );
82      tb[3] = new JRadioButton( items[3], true );
83      tb[4] = new JRadioButton( items[4], true );
84
85          JPanel buttonPanel = new JPanel();
86          buttonPanel.setLayout( new GridLayout(2, 3) );
87      for (int i=0; i < 5; i++) buttonPanel.add( tb[i] );
88      cregion.add(buttonPanel);
89
90
91
92      // ----- (8) group multiple radio buttons
93
94      ButtonGroup rgroup = new ButtonGroup();
95      for (int i=2; i < 5; i++) rgroup.add( tb[i] );
96
97
98
99       // ----- (9) add list box, combo box
100
101      JComboBox cb = new JComboBox(items);
102      cregion.add(cb);
103
104      JList lb = new JList(items);
105      lb.setVisibleRowCount(5);
106          JScrollPane lsc = new JScrollPane(lb);
107          lsc.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
108      cregion.add(lsc);
109
110
111      //lb.setSelectionMode( ListSelectionModel.SINGLE_SELECTION );
112      //lb.setSelectionMode( ListSelectionModel.SINGLE_INTERVAL_SELECTION );
113          lb.setSelectionMode( ListSelectionModel.MULTIPLE_INTERVAL_SELECTION );
114      //default is multiple interval selection
115
116
117      // ----- (4) validate container once all components are added
118      this.validate();
119      }
120
121      public static void main(String[] args)
122      {
123      new w8_2_Component();
124      }
125  }
126
```

# w9_1_ComponentEvent

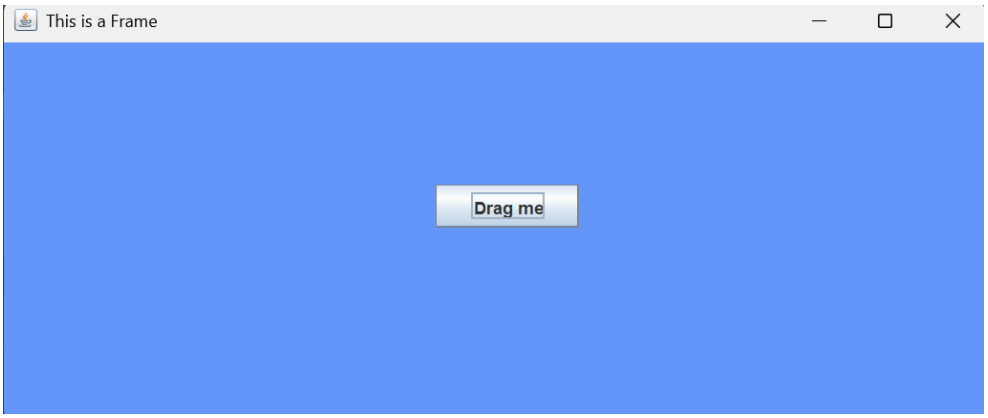


# w9_1_ComponentEvent

```java
package Lab_Ch9;

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

// ----- (1) GUI structure & programming concept
class w9_1_ComponentEvent extends JFrame
{
    private JPanel       contentpane;
    private JButton          button1, button2;
    private JComboBox        combo;
    private JToggleButton [] tb;
    private ButtonGroup      bgroup;


    public w9_1_ComponentEvent()
    {
    setTitle("This is a Frame");
        setSize(500, 150);
        setLocationRelativeTo(null);
    setVisible(true);
    setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );

    contentpane = (JPanel)getContentPane();
    contentpane.setBackground( Color.GRAY );
    contentpane.setLayout( new FlowLayout() );

    AddComponents();
    AddListeners();
    }


    public void AddComponents()
    {
    String [] items = new String[10];
    for (int i=0; i < 10; i++) items[i] = " --- item " + i + " ---";

    button1 = new JButton("Click First");
    button2 = new JButton("Click Second");

    combo = new JComboBox( items );

    bgroup  = new ButtonGroup();
    tb      = new JToggleButton[3];
    JPanel bpanel = new JPanel();
    for (int i=0; i < 3; i++)
    {
            tb[i] = new JRadioButton( items[i] );
            bgroup.add( tb[i] );
            bpanel.add( tb[i] );
    }

    contentpane.add(button1);
    contentpane.add(button2);
    contentpane.add(combo);
    contentpane.add(bpanel);

    validate();
    }


    public void AddListeners()
    {
    // ----- (2) add ComponentListener to frame
    this.addComponentListener( new MyComponentListener() );

    // ----- (3) add FocusListener to buttons and combo box

    button1.addFocusListener( new MyFocusListener() );
    button2.addFocusListener( new MyFocusListener() );
    combo.addFocusListener( new MyFocusListener() );
```

```
73
74      }
75
76      public static void main(String[] args)
77      {
78      new w9_1_ComponentEvent();
79      }
80  }
81
82  ////////////////////////////////////////////////////////////////////////////////
83  class MyComponentListener extends ComponentAdapter
84  {
85      @Override
86      public void componentMoved( ComponentEvent e )
87      {
88      System.out.println( e.paramString() );
89      }
90
91      @Override
92      public void componentResized( ComponentEvent e )
93      {
94      System.out.println( e.paramString() );
95      }
96  };
97
98
99  class MyFocusListener extends FocusAdapter
100 {
101     @Override
102     public void focusLost( FocusEvent e )
103     {
104     String s;
105     if ( e.isTemporary() ) s = "temporary focus";
106     else s = "permanent focus";
107
108     if (e.getComponent() instanceof JButton)
109     {
110             JButton source = (JButton)e.getComponent();
111             System.out.println( "<< " + source.getText() + " >>  loses " + s );
112         }
113     else if (e.getComponent() instanceof JComboBox)
114     {
115             JComboBox source = (JComboBox)e.getComponent();
116             System.out.println( source.getSelectedItem().toString() + "  loses " + s);
117     }
118     }
119
120     @Override
121     public void focusGained( FocusEvent e )
122     {
123     String s;
124     if ( e.isTemporary() ) s = "temporary focus";
125     else s = "permanent focus";
126
127     if (e.getComponent() instanceof JButton)
128     {
129             JButton source = (JButton)e.getComponent();
130             System.out.println( "<< " + source.getText() + " >>  gains " + s );
131     }
132     else if (e.getComponent() instanceof JComboBox)
133     {
134             JComboBox source = (JComboBox)e.getComponent();
135             System.out.println( source.getSelectedItem().toString() + "  gains " + s);
136         }
137     }
138 };
```

# w9_2_MouseEvent

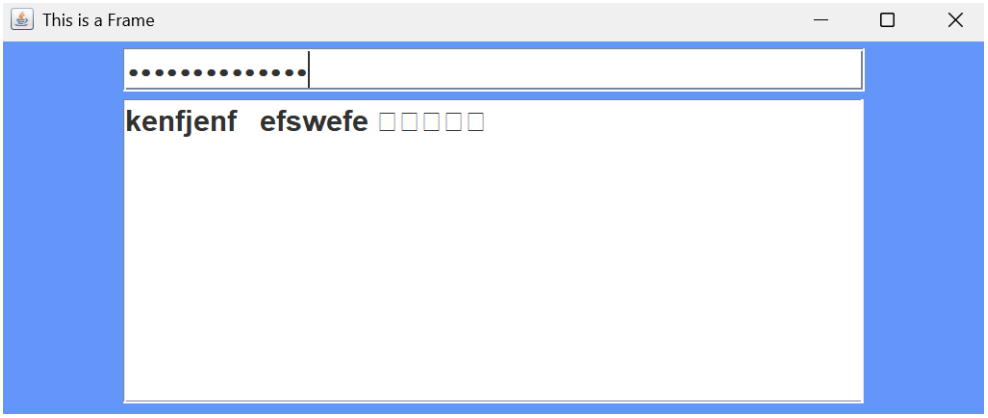


# w9_2_MouseEvent

```java
package Lab_Ch9;

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

class w9_2_MouseEvent extends JFrame
{
    private JPanel  contentpane;
    private MyButton    button1;


    public w9_2_MouseEvent()
    {
    setTitle("This is a Frame");
        setSize(700, 300);
        setLocationRelativeTo(null);
    setVisible(true);
    setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );

    contentpane = (JPanel)getContentPane();
    contentpane.setBackground( new Color(100, 150, 250) );
    contentpane.setLayout( null );

    button1 = new MyButton();
    contentpane.add(button1);
    repaint();
    }


    public static void main(String[] args)
    {
    new w9_2_MouseEvent();
    }
}

////////////////////////////////////////////////////////////////////////////////
class MyButton extends JButton implements MouseListener, MouseMotionListener
{
    private int       curX   = 100, curY   = 20;
    private int       width  = 100, height = 30;     // for text button
    //private int       width  = 100, height = 100;    // for icon button
    private boolean   drag;

    // ----- (5) use icon instead of text
    private ImageIcon  clickIcon, dragIcon;

    public MyButton()
    {
        String path = "src/main/Java/Lab_Ch9/";
    clickIcon   = new ImageIcon(path + "redbox.png");
    dragIcon    = new ImageIcon(path + "red.png");

    setBounds(curX, curY, width, height);
    setText("Click me");
    //setIcon(clickIcon);
    drag = false;

    // ----- (3) add listeners
    addMouseListener( this );
    addMouseMotionListener( this );
    }

    public void mousePressed( MouseEvent e )    { }
    public void mouseReleased( MouseEvent e )   { }
    public void mouseEntered( MouseEvent e )    { }
    public void mouseExited( MouseEvent e ) { }
    public void mouseMoved( MouseEvent e )  { }
```

```
70      // ----- (1) handler for MouseListener
71      @Override
72      public void mouseClicked( MouseEvent e )
73      {
74      if ( !drag )
75      {
76              setCursor( new Cursor(Cursor.MOVE_CURSOR) );
77              setText("Drag me");
78              //setIcon(dragIcon);
79              drag = true;
80      }
81      else
82      {
83              setCursor( new Cursor(Cursor.DEFAULT_CURSOR) );
84              setText("Click me");
85              //setIcon(clickIcon);
86              drag = false;
87      }
88      }
89
90      // ----- (2) handler for MouseMotionListener
91      @Override
92      public void mouseDragged( MouseEvent e )
93      {
94      if ( drag )
95      {
96              curX = curX + e.getX();
97              curY = curY + e.getY();
98
99              // ----- (4) bound for dragging
100             /*
101             Container p = getParent();
102             if (curX < 0)  curX = 0;
103             if (curY < 0)  curY = 0;
104             if (curX + width  > p.getWidth())   curX = p.getWidth() - width;
105             if (curY + height > p.getHeight())  curY = p.getHeight() - height;
106             */
107
108             setLocation(curX, curY);
109         }
110     }
111 };
```

# w9_3_KeyEvent

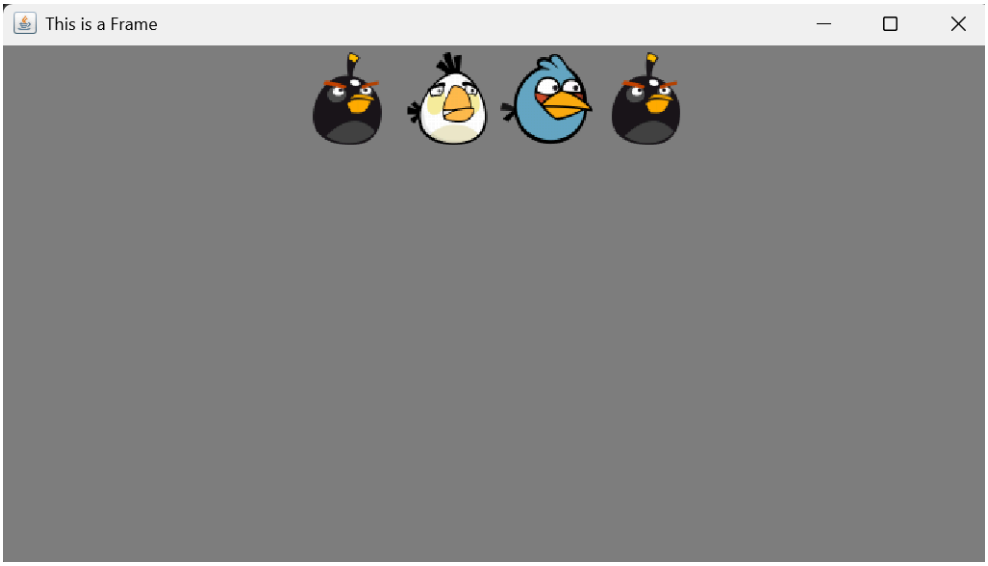


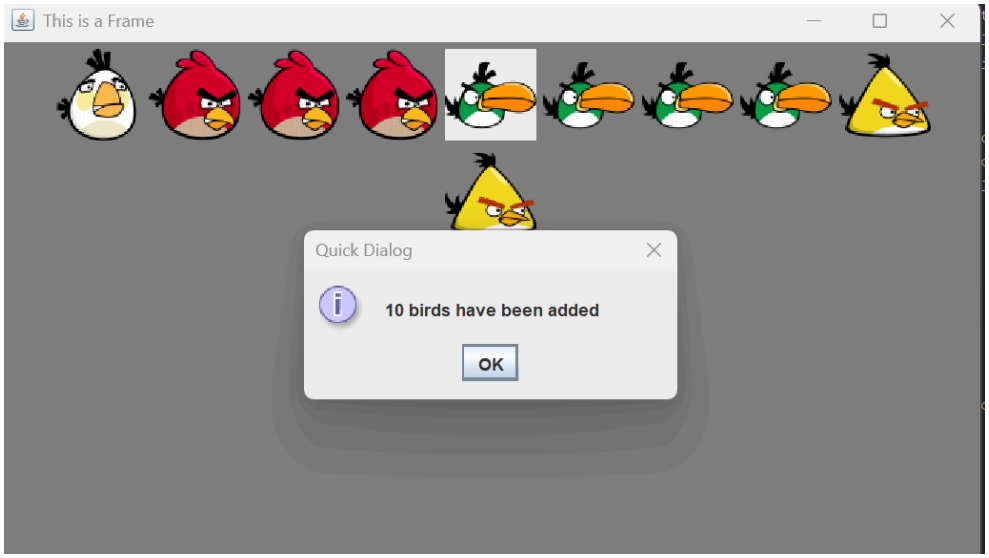
# w9_3_KeyEvent

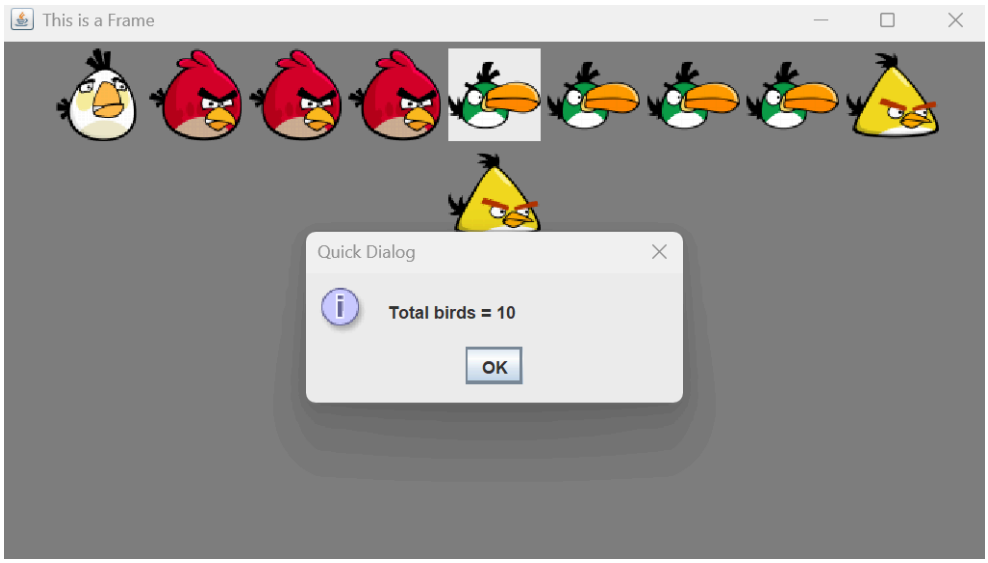
```java
package Lab_Ch9;

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

class w9_3_KeyEvent extends JFrame
{
    private JPanel              contentpane;
    private JPasswordField  passfield;
    private JTextArea       textarea;
    private DoubleText          dtext;

    public w9_3_KeyEvent()
    {
    setTitle("This is a Frame");
        setSize(700, 300);
        setLocationRelativeTo(null);
    setVisible(true);
    setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );

    contentpane = (JPanel)getContentPane();
    contentpane.setBackground( new Color(100, 150, 250) );
    contentpane.setLayout( new FlowLayout() );

    AddComponents();

    // ----- (1) listener is automatically added when dtext is created
    //           in AddComponents()
    }

    public void AddComponents()
    {
    passfield = new JPasswordField(30);
    textarea  = new JTextArea(8, 30);
    dtext     = new DoubleText( passfield, textarea );

    Font f = new Font("SanSerif", Font.BOLD, 20);
    passfield.setFont(f);
    textarea.setFont(f);
    textarea.setEditable(false);

    contentpane.add(passfield);
    contentpane.add( new JScrollPane(textarea) );
    validate();
    }

    public static void main(String[] args)
    {
    new w9_3_KeyEvent();
    }
}

////////////////////////////////////////////////////////////////////////////////

// ----- (1) DoubleText component, with KeyListener

class DoubleText implements KeyListener
{
    private JPasswordField in;
    private JTextArea      out;

    public DoubleText(JPasswordField p, JTextArea t)
    {
    in = p; out = t;
    in.addKeyListener( this );
    }

```

```
69      @Override
70      public void keyTyped( KeyEvent e )
71      {
72      //System.out.printf("t >>  %c  (%s) \n", e.getKeyChar(), e.getKeyText(e.getKeyCode()) );
73      String current = out.getText();
74
75      // ----- (2) get the last char from "in" and add it to "out"
76      current = current + e.getKeyChar();
77      out.setText(current);
78
79      // ----- (3) consume the event
80      //e.consume();
81      }
82
83      @Override
84      public void keyPressed( KeyEvent e )
85      {
86      //System.out.printf("p >>  %c  (%s) \n", e.getKeyChar(), e.getKeyText(e.getKeyCode()) );
87
88      // ----- (4) if DEL is pressed, clear everything
89      //           DEL button has no unicode character -> getKeyCode()
90      //
91      //           what happens if we put move this to keyTyped(...) ?
92      /*
93      if ( e.getKeyCode() == KeyEvent.VK_DELETE )
94      {
95              in.setText("");
96              out.setText("");
97      }
98      */
99      }
100
101     public void keyReleased( KeyEvent e )   { }
102 };
```

# w10_1_WinEvent3



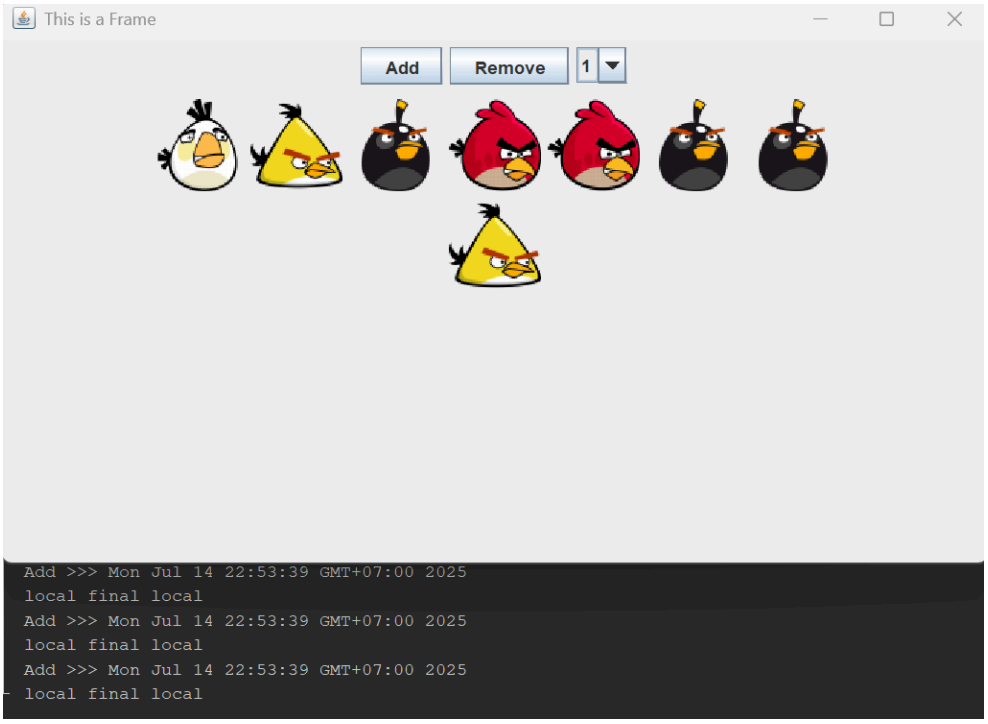



# w10_1_WinEvent3

```java
package Lab_Ch10;

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


class w10_1_WinEvent_3 extends JFrame implements MouseListener
{
    private JPanel      contentpane;
    private java.util.Random    random;

    private String   path  = "src/main/Java/Lab_Ch10/";
    private String[] birds = {"black.png", "blue.png", "green.png",
                      "red.png", "white.png", "yellow.png"};

    public w10_1_WinEvent_3()
    {
    setTitle("This is a Frame");
        setSize(700, 400);
        setLocationRelativeTo(null);
    setVisible(true);
    setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );

    contentpane = (JPanel)getContentPane();
    contentpane.setLayout( new FlowLayout() );
    contentpane.setBackground( Color.GRAY );

    random = new java.util.Random();

    // ----- (2) add listeners : notice the difference between (2.1), (2.2), (2.3)

    // ----- (2.1) click anywhere in the frame to add a random bird
    addMouseListener( this );

    // ----- (2.2) for every 5 birds added, pop up a dialog
    contentpane.addContainerListener( new MyContainerListener3() );

    // ----- (2.3) count total no. of birds when closing the frame
        //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    addWindowListener( new WindowAdapter() {
    //addWindowListener( new WindowListener() {
            @Override
            public void windowOpened( WindowEvent e )
            {
        QuickDialog3.show("Click anywhere to add birds");
            }
            @Override
            public void windowClosing( WindowEvent e )
            {
        //JFrame frame = (JFrame)e.getWindow();
        //JPanel contentpane = (JPanel)frame.getContentPane();
        int count = contentpane.getComponentCount();
        QuickDialog.show("Total birds = " + count);
            }

            //public void windowClosed( WindowEvent e )     { }
            //public void windowActivated( WindowEvent e )  { }
            //public void windowDeactivated( WindowEvent e )    { }
            //public void windowIconified( WindowEvent e )        { }
            //public void windowDeiconified( WindowEvent e )      { }
        });
        //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    }

    // ----- (1) handlers for MouseListener
    public void mousePressed( MouseEvent e )    { }
    public void mouseReleased( MouseEvent e )   { }
    public void mouseEntered( MouseEvent e )    { }
    public void mouseExited( MouseEvent e ) { }

    @Override
```

```
73      @Override
74      public void mouseClicked( MouseEvent e )
75      {
76      int r = random.nextInt(6);
77      JLabel label = new JLabel( new ImageIcon(path + birds[r]) );
78      contentpane.add( label );
79      validate();
80      }
81
82      public static void main(String[] args)
83      {
84      new w10_1_WinEvent_3();
85      }
86  };
87
88  ////////////////////////////////////////////////////////////////////////////////
89  class QuickDialog3
90  {
91      public static void show(String message)
92      {
93          JOptionPane.showMessageDialog(new JFrame(), message, "Quick Dialog",
94                          JOptionPane.INFORMATION_MESSAGE );
95      }
96  };
97
98  ////////////////////////////////////////////////////////////////////////////////
99  class MyContainerListener3 extends ContainerAdapter
100 {
101     @Override
102     public void componentAdded( ContainerEvent e )
103     {
104     int count = e.getContainer().getComponentCount();
105     if (count % 5 == 0)
106     {
107             QuickDialog3.show( count + " birds have been added" );
108
109             JLabel label = (JLabel)e.getChild();
110             label.setOpaque(true);
111
112             // ----- (3) last argument is transparency degree
113             //           0 = transparent, 255 = opaque
114             //label.setBackground( new Color(0, 0, 100, 50) );
115     }
116     }
117 };
118
119
120 ////////////////////////////////////////////////////////////////////////////////
121
```


This is a Frame
Add
Remove
1
Add >>> Mon Jul 14 22:53:39 GMT+07:00 2025
local final local
Add >>> Mon Jul 14 22:53:39 GMT+07:00 2025
local final local
Add >>> Mon Jul 14 22:53:39 GMT+07:00 2025
local final local

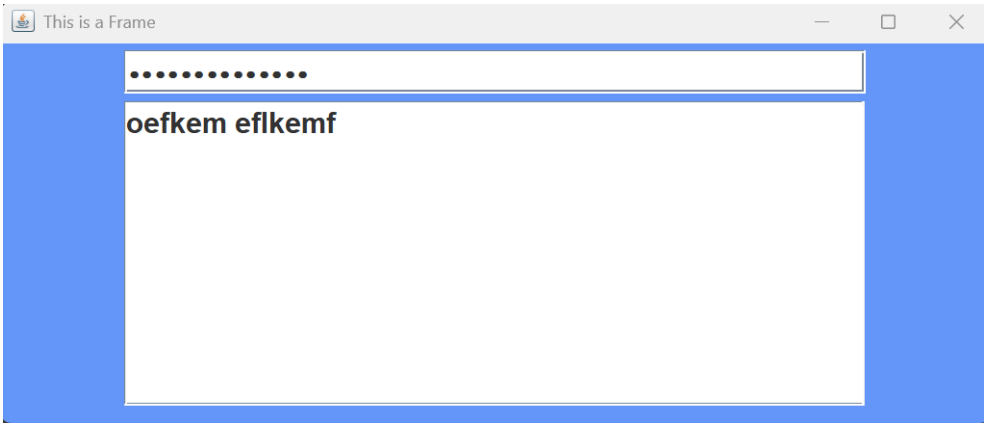
# w10_2_SemanticEvent

```java
package Lab_Ch10;

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

class w10_2_SemanticEvent extends JFrame
{
    private JPanel        contentpane;
    private MyPane    drawpane;
    private JButton   add_button, rem_button;
    private JComboBox     combo;
    private JRadioButton  [] radio;

    private Integer [] items = {1, 2, 3, 4};

    public w10_2_SemanticEvent()
    {
    setTitle("This is a Frame");
    setBounds(300, 200, 700, 400);
    setVisible(true);
    setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE );

    contentpane = (JPanel)getContentPane();
    contentpane.setLayout( new FlowLayout() );
    AddComponents();
    }

    public void AddComponents()
    {
    // ----- (1) anonymous listener classes (implicitly implement listeners)
        String local = "local";
        final String finlocal = "final local";

    add_button = new JButton("Add");
    add_button.addActionListener( new ActionListener() {
            @Override
            public void actionPerformed( ActionEvent e )
            {
                java.util.Date date = new java.util.Date();
                date.setTime(e.getWhen());
                System.out.println(e.getActionCommand() + " >>> " + date.toString());

                System.out.printf("%s %s \n", local, finlocal);

                drawpane.doAdd();
            }
        });

        // local = "new value";


    rem_button = new JButton("Remove");
    rem_button.addActionListener( new ActionListener() {
            @Override
            public void actionPerformed( ActionEvent e )
            {
        drawpane.doRemove();
            }
    });

        // ----- (2) Disable action on remove button --> ActionEvent won't be handled
        //           But MouseClick is not disabled  --> MouseEvent is still handled
        //rem_button.setEnabled(false);
        /*
    rem_button.addMouseListener( new MouseAdapter() {
            @Override
            public void mouseClicked( MouseEvent e )
            {
        drawpane.doRemove();
            }
    });
        */


```

```
75
76      // ----- (3) get no. of birds from combo (use ItemEvent or ActionEvent)
77      combo = new JComboBox(items);
78      combo.addItemListener( new ItemListener() {
79              @Override
80              public void itemStateChanged( ItemEvent e )
81              {
82          int index = combo.getSelectedIndex();
83          drawpane.setCount( items[index] );
84              }
85      });
86
87
88      // ----- (4) change from combo to radio buttons (use ItemEvent or ChangeEvent)
89      //           compare how we check user's selection in combo & radio buttons
90          radio = new JRadioButton[4];
91          JPanel rpanel      = new JPanel();
92          ButtonGroup rgroup = new ButtonGroup();
93          int outsideloop    = 0;
94          for (int i=0; i < 4; i++)
95          {
96              // treated as different (final) variables in different iterations
97              // new value can be assigned but only in declaration statement
98              int insideloop = i;
99
100             radio[i] = new JRadioButton( items[i].toString() );
101             if (i == 0) radio[i].setSelected(true);
102             rgroup.add( radio[i] );
103             rpanel.add( radio[i] );
104
105             radio[i].addItemListener( new ItemListener() {
106                 @Override
107         public void itemStateChanged( ItemEvent e )
108         {
109                     JRadioButton temp = (JRadioButton)e.getItem();
110                     if (temp.isSelected())
111                     //if (e.getStateChange() == ItemEvent.SELECTED)
112                     {
113                         System.out.printf("outside = %d, inside = %d \n", outsideloop, insideloop);
114
115             int count = Integer.parseInt( temp.getText() );
116             drawpane.setCount( count );
117                     }
118                     //outsideloop++;        // updated inside inner class
119                     //insideloop++;         // updated inside inner class
120         }
121             });
122
123
124             //outsideloop++;        // updated inside for-loop (enclosing method)
125             //insideloop++;         // updated inside for-loop (enclosing method)
126     }
127
128         //outsideloop++;            // updated inside enclosing method
129
130
131     drawpane = new MyPane();
132
133     contentpane.add(add_button);
134     contentpane.add(rem_button);
135     contentpane.add(combo);             // using combo (3)
136     //contentpane.add(rpanel);          // using radio buttons (4)
137     contentpane.add(drawpane);
138
139     validate();
140     }
141
142
143     public static void main(String[] args)
144     {
145     new w10_2_SemanticEvent();
146     }
147 };
148
```

```
149 ////////////////////////////////////////////////////////////////////////////////
150
151 // ----- (1) a pane for drawing birds
152
153 class MyPane extends JPanel
154 {
155     private java.util.Random  random;
156     private int           count = 1;
157
158     private String   path  = "src/main/Java/Lab_Ch10/";
159     private String[] birds = {"black.png", "blue.png", "green.png",
160                       "red.png", "white.png", "yellow.png"};
161
162     public MyPane()
163     {
164     setPreferredSize( new Dimension(550, 310) );
165     random = new java.util.Random();
166     }
167
168     public void setCount( int c )   { count = c; }
169
170     public void doAdd()
171     {
172     for (int i = 0; i < count; i++)
173     {
174             int r = random.nextInt(6);
175             JLabel label = new JLabel( new ImageIcon(path + birds[r]) );
176             add(label);
177             validate();
178     }
179     }
180
181     public void doRemove()
182     {
183         for (int i = 0; i < count; i++)
184         {
185             if (getComponentCount() > 0)
186             {
187                 Component clabel = getComponent(0);
188                 remove( clabel );
189
190                 // ----- (4) try with and without validate
191                 validate();
192                 repaint();
193             }
194         }
195     }
196 };
197
```

# w10_3_KeyEvent

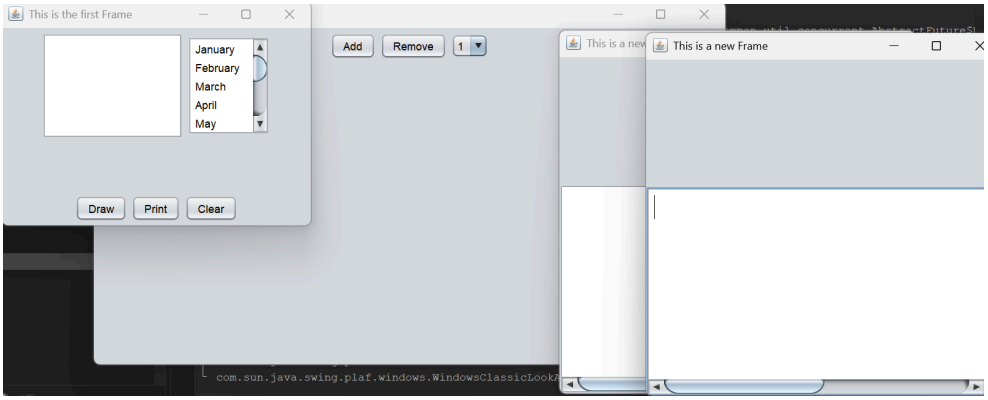


# w10_3_KeyEvent

```java
package Lab_Ch10;

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

class w10_3_KeyEvent extends JFrame
{
    private JPanel          contentpane;
    private JPasswordField  passfield;
    private JTextArea       textarea;
    private DoubleText      dtext;

    public w10_3_KeyEvent()
    {
    setTitle("This is a Frame");
    setBounds(200, 200, 700, 300);
    setVisible(true);
    setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );

    contentpane = (JPanel)getContentPane();
    contentpane.setBackground( new Color(100, 150, 250) );
    contentpane.setLayout( new FlowLayout() );

    AddComponents();
    }

    public void AddComponents()
    {
    passfield = new JPasswordField(30);
    textarea  = new JTextArea(8, 30);
    dtext     = new DoubleText( passfield, textarea );

    Font f = new Font("SanSerif", Font.BOLD, 20);
    passfield.setFont(f);
    textarea.setFont(f);
    textarea.setEditable(false);

    contentpane.add(passfield);
    contentpane.add( new JScrollPane(textarea) );

    validate();
    }

    public static void main(String[] args)
    {
    new w10_3_KeyEvent();
    }
}

////////////////////////////////////////////////////////////////////////////////

// ----- (1) DoubleText component, with KeyListener
/*
class DoubleText implements KeyListener
{
    private JPasswordField in;
    private JTextArea      out;

    public DoubleText(JPasswordField p, JTextArea t)
    {
    in = p; out = t;
    in.addKeyListener( this );
    }

    @Override
    public void keyTyped( KeyEvent e )
    {
    String current = out.getText();

    if ( e.getKeyChar() != '\b' )
    {
            current = current + e.getKeyChar();
            out.setText(current);
    }
    else
    {
            int last = current.length() - 1;
            current  = current.substring(0, last);
            out.setText(current);
        }
    }

    public void keyPressed( KeyEvent e )    { }
    public void keyReleased( KeyEvent e )   { }
};
*/

////////////////////////////////////////////////////////////////////////////////
```

```
91
92  // ----- (2) DoubleText component, with CaretListener
93
94  class DoubleText implements CaretListener
95  {
96      private JPasswordField in;
97      private JTextArea      out;
98
99      public DoubleText(JPasswordField p, JTextArea t)
100     {
101     in = p; out = t;
102     in.addCaretListener( this );
103     }
104
105     @Override
106     public void caretUpdate( CaretEvent e )
107     {
108     out.setText( in.getText() );
109         //out.setText( new String(in.getPassword()) );
110     }
111 };
112
```

# w10_4_MainFrame

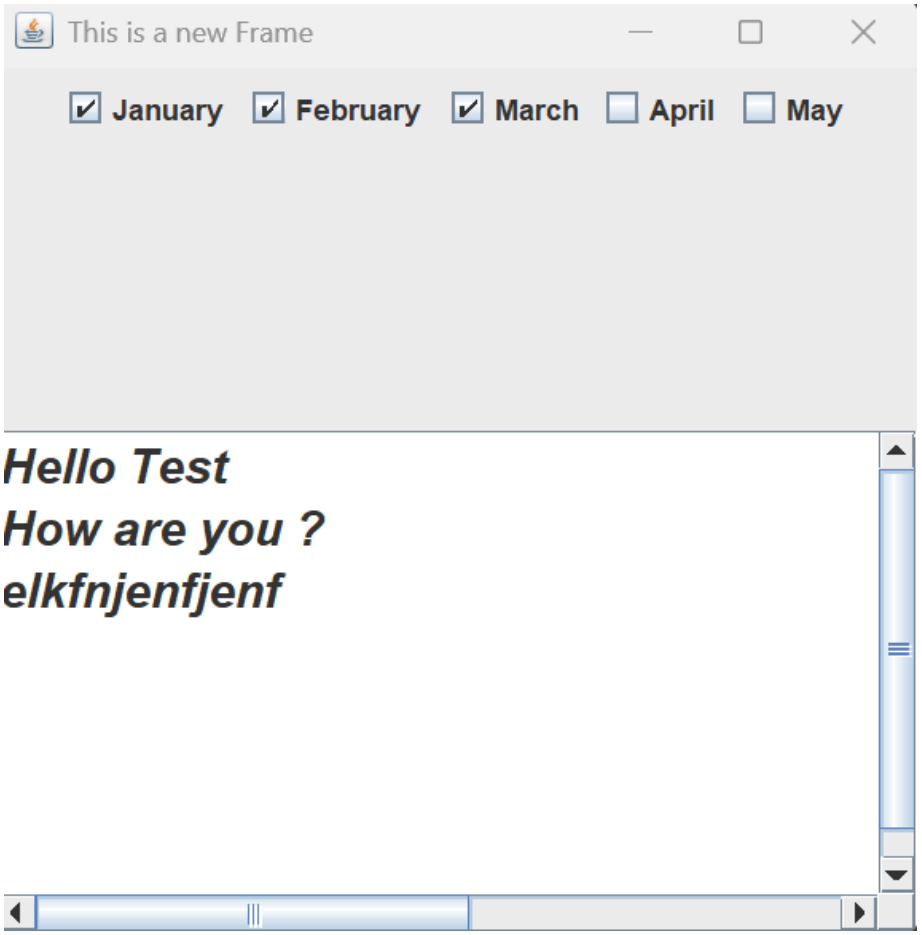


# w10_4_MainFrame

```java
package Lab_Ch10;

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

class w10_4_MainFrame extends JFrame
{
    private JPanel  contentpane;
    private JList   list;
    private JTextArea   text;
    private JButton print_button, draw_button, clear_button;

    private String [] items = {"January", "February", "March", "April", "May", "June",
                       "July", "August", "September", "October", "November", "December"};

    // ----- (1) message to be passed to next frames
    private Object [] messageFromList;
    private String    messageFromText;

    // ----- (2) next frames
    w10_2_SemanticEvent sframe;
    w10_5_PrintFrame pframe;


    public w10_4_MainFrame()
    {
    setTitle("This is the first Frame");
    setBounds(200, 200, 350, 250);
    setVisible(true);
    setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );

    contentpane = (JPanel)getContentPane();
    contentpane.setLayout( new BorderLayout() );

    AddComponents();
    }

    public void AddComponents()
    {
    // ----- (3) TextArea + CaretEvent
    text = new JTextArea(6, 15);
    text.addCaretListener( new CaretListener() {
            @Override
            public void caretUpdate( CaretEvent e )
            {
        messageFromText = text.getText();
            }
    });


    // ----- (4) List + ListSelectionEvent
    list = new JList( items );
    list.setVisibleRowCount(5);
    list.addListSelectionListener( new ListSelectionListener() {
            @Override
            public void valueChanged( ListSelectionEvent e )
            {
        if( !e.getValueIsAdjusting() )
        {
                    messageFromList = list.getSelectedValues();
        }
            }
    });


    // ---- (5) Button + ActionEvent : clearup TextArea and List
    clear_button = new JButton("Clear");
    clear_button.addActionListener( new ActionListener() {
            @Override
            public void actionPerformed( ActionEvent e )
            {
        text.setText("");
        list.clearSelection();
            }
    });


    // ----- (6) Button + ActionEvent -> open SemanticEvent frame
    draw_button  = new JButton("Draw");
    draw_button.addActionListener( new ActionListener() {
            @Override
            public void actionPerformed( ActionEvent e )
            {
        if (sframe == null) sframe = new w10_2_SemanticEvent();
        else                sframe.setVisible(true);
            }
        });

```

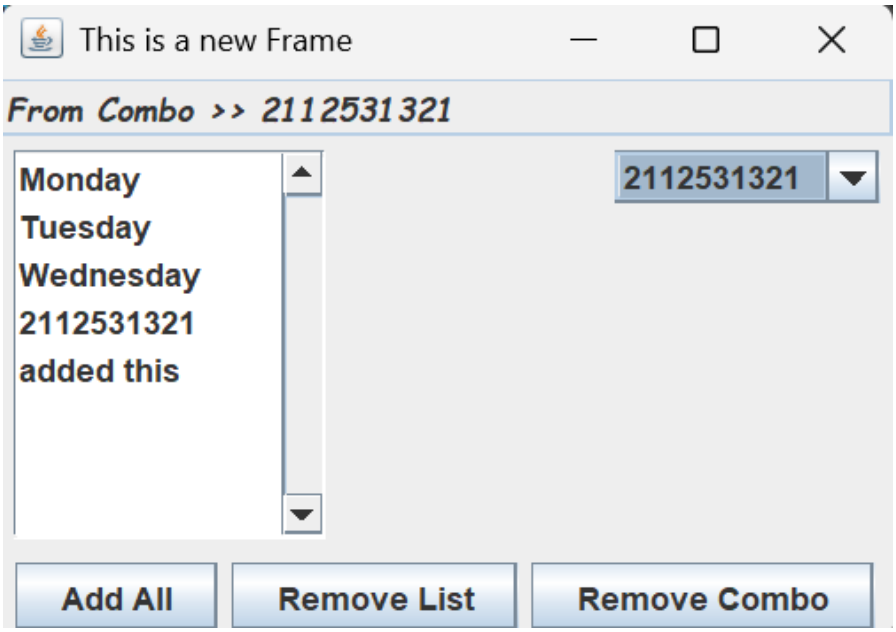
w10_4_MainFrame

```
91
92      // ----- (7) Button + ActionEvent -> open PrintFrame
93      print_button = new JButton("Print");
94      print_button.addActionListener( new ActionListener() {
95              @Override
96              public void actionPerformed( ActionEvent e )
97              {
98          pframe = new w10_5_PrintFrame();
99
100         // ----- (7.2) get messages from components
101         //             no need to catch events in (3) and (4)
102         //messageFromText = text.getText();
103         //messageFromList = list.getSelectedValues();
104
105         // ----- (7.1) pass messages to the next frame
106         pframe.setTextMessage( messageFromText );
107         pframe.setCheckMessage( messageFromList );
108         pframe.AddComponents();
109             }
110     });
111
112
113     JPanel mpanel = new JPanel();
114     mpanel.add( new JScrollPane(text) );
115     mpanel.add( new JScrollPane(list) );
116
117     JPanel spanel = new JPanel();
118     spanel.add(draw_button);
119     spanel.add(print_button);
120     spanel.add(clear_button);
121
122     contentpane.add( mpanel, BorderLayout.CENTER );
123     contentpane.add( spanel, BorderLayout.SOUTH );
124
125     validate();
126     }
127
128
129     public static void main(String[] args)
130     {
131     // ----- (8) check look & feel
132     UIManager.LookAndFeelInfo [] laf = UIManager.getInstalledLookAndFeels();
133     System.out.println("===== available look and feel =====");
134     for (int i=0; i < laf.length; i++)
135     {
136             System.out.println( laf[i].getClassName() );
137     }
138
139     // ----- (9) set look and feel
140     try
141     {
142             String look1 = "javax.swing.plaf.metal.MetalLookAndFeel";
143             String look2 = "com.sun.java.swing.plaf.motif.MotifLookAndFeel";
144             String look3 = "com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
145             String look4 = "com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel";
146             String look5 = "javax.swing.plaf.nimbus.NimbusLookAndFeel";
147             UIManager.setLookAndFeel(look5);
148     }
149     catch (Exception e) { System.out.println(e); }
150
151     new w10_4_MainFrame();
152     }
153 };
154
```


This is a new Frame
January
February
March
April
May
Hello Test
How are you ?
elkfnjenfjenf

# w10_5_PrintFrame

```java
package Lab_Ch10;

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

class w10_5_PrintFrame extends JFrame
{
    private JPanel          contentpane;
    private JCheckBox []    check;
    private JTextArea       text;

    // ----- (1) message from previous frame
    private Object [] items   = null;
    private String    message = "";

    public w10_5_PrintFrame()
    {
    setTitle("This is a new Frame");
    setBounds(200, 200, 400, 400);
    setVisible(true);
    setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE );

    contentpane = (JPanel)getContentPane();
    contentpane.setLayout( new BorderLayout() );
    }

    public void setCheckMessage( Object [] m )  { items = m; }
    public void setTextMessage( String m )      { message = m; }


    public void AddComponents()
    {
    JPanel cpanel = new JPanel();
    if (items != null)
    {
            check = new JCheckBox [ items.length ];
            for (int i=0; i < items.length; i++)
            {
        check[i] = new JCheckBox( items[i].toString() );
        cpanel.add( check[i] );
            }
    }

    text = new JTextArea(message, 8, 40);
    text.setFont( new Font("SanSerif", Font.BOLD | Font.ITALIC, 20) );

    contentpane.add(cpanel, BorderLayout.NORTH);
    contentpane.add(new JScrollPane(text), BorderLayout.SOUTH);

    validate();
    }


    public static void main(String[] args)
    {
    // ----- (2) test this frame independently
    w10_5_PrintFrame frame = new w10_5_PrintFrame();

    // ----- (3) suppose that messages are passed from previous frame
    frame.setTextMessage( "Hello Test \nHow are you ?" );
    String [] si = {"January", "February", "March", "April", "May", "June"};
    frame.setCheckMessage( si );

    frame.AddComponents();
    }
};

```


This is a new Frame
From Combo >> 2112531321
Monday
Tuesday
Wednesday
2112531321
added this
2112531321
Add All
Remove List
Remove Combo

# w10_6_DataModel

```java
package Lab_Ch10;

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

class w10_6_DataModel extends JFrame
{
    private JPanel                contentpane;
    private JList          list;
    private DefaultListModel       listmodel;
    private JComboBox              combo;
    private DefaultComboBoxModel   combomodel;
    private JButton        add_button, remove_button1, remove_button2;
    private JTextField         text;

    public w10_6_DataModel()
    {
    setTitle("This is a new Frame");
    setBounds(200, 200, 350, 250);
    setVisible(true);
    setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );

    contentpane = (JPanel)getContentPane();
    contentpane.setLayout( new BorderLayout() );

    AddComponents();
    }

    public void AddComponents()
    {
    text = new JTextField("Selection", 15);
    text.setFont(new Font("Comic Sans MS", Font.BOLD | Font.ITALIC, 12));
    text.setEditable(false);

        String []days = {"Monday", "Tuesday", "Wednesday"};

    // ----- (1) Readonly list
    //list = new JList(days);

        // ----- (2) Updatable list
        listmodel = new DefaultListModel();
        for(String s : days) listmodel.addElement(s);
        list = new JList(listmodel);

        list.setPreferredSize( new Dimension(100, 80) );
        list.setSelectionMode(ListSelectionModel.SINGLE_INTERVAL_SELECTION);
        list.addListSelectionListener( new ListSelectionListener() {
            @Override
            public void valueChanged( ListSelectionEvent e )
            {
                // Try with & without condition
                if ( !e.getValueIsAdjusting() )
                {
                    // Checking selection from JList (for both readonly & updatable lists)
                    Object[] items = list.getSelectedValues();
                    if (items.length > 0) text.setText("From List >> " + items[0]);
                    else                  text.setText("");
                }
            }
        });


        // ----- (3) Can update JComboBox directly (automatically linked to combomodel)
        combo      = new JComboBox(days);
        combomodel = (DefaultComboBoxModel) combo.getModel();

        combo.setPreferredSize( new Dimension(100, 20) );
        combo.addItemListener( new ItemListener() {
            @Override
            public void itemStateChanged( ItemEvent e )
            {
                Object item = combo.getSelectedItem();
                text.setText("From Combo >> " + item);
            }
        });
```

```
79
80      add_button  = new JButton("Add All");
81      add_button.addActionListener( new ActionListener() {
82              @Override
83              public void actionPerformed( ActionEvent e )
84              {
85          // ----- (4) Add new item to listmodel & JComboBox
86                  //           No method to add item to JList
87          String item = JOptionPane.showInputDialog("Enter input");
88          if (listmodel != null)   listmodel.addElement(item);
89
90                  combo.addItem(item);
91                  //if (combomodel != null)  combomodel.addElement(item);
92              }
93      });
94
95
96      remove_button1  = new JButton("Remove List");
97      remove_button1.addActionListener( new ActionListener() {
98              @Override
99              public void actionPerformed( ActionEvent e )
100             {
101         // ----- (5) get selected items & remove them from listmodel
102                 //           No method to remove item from JList
103         Object x[] = list.getSelectedValues();
104         for (int i=0; i < x.length; i++)
105         {
106                     if (listmodel != null) listmodel.removeElement(x[i]);
107         }
108             }
109     });
110
111
112     remove_button2  = new JButton("Remove Combo");
113     remove_button2.addActionListener( new ActionListener() {
114             @Override
115             public void actionPerformed( ActionEvent e )
116             {
117         // ----- (6) get selected items & remove them from JComboBox
118                 Object x = combo.getSelectedItem();
119                 //combo.removeItem(x);
120                 if (combomodel != null) combomodel.removeElement(x);
121             }
122     });
123
124
125         JPanel listPanel = new JPanel();
126         JScrollPane scroll = new JScrollPane(list);
127         scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
128         listPanel.add(scroll);
129
130         JPanel comboPanel = new JPanel();
131         comboPanel.add(combo);
132
133     JPanel buttonPanel = new JPanel();
134     buttonPanel.add(add_button);
135     buttonPanel.add(remove_button1);
136         buttonPanel.add(remove_button2);
137
138         contentpane.add( text,        BorderLayout.NORTH);
139         contentpane.add( listPanel,   BorderLayout.WEST);
140         contentpane.add( comboPanel,  BorderLayout.EAST);
141         contentpane.add( buttonPanel, BorderLayout.SOUTH);
142     validate();
143     }
144
145     public static void main(String[] args)
146     {
147     new w10_6_DataModel();
148     }
149 };
150
```