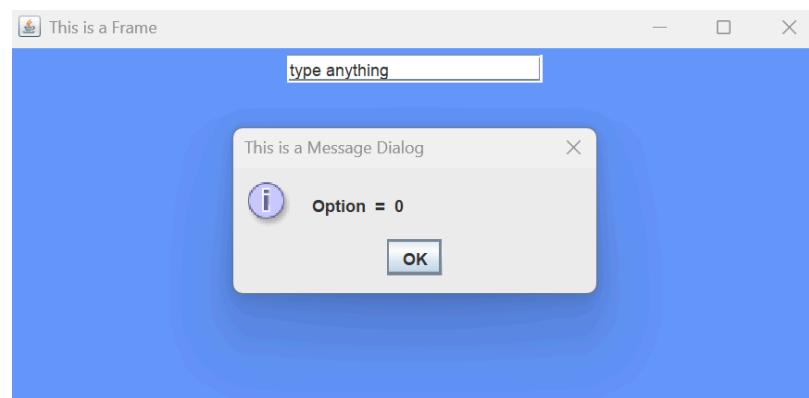




Paradigm finals cheat sheet

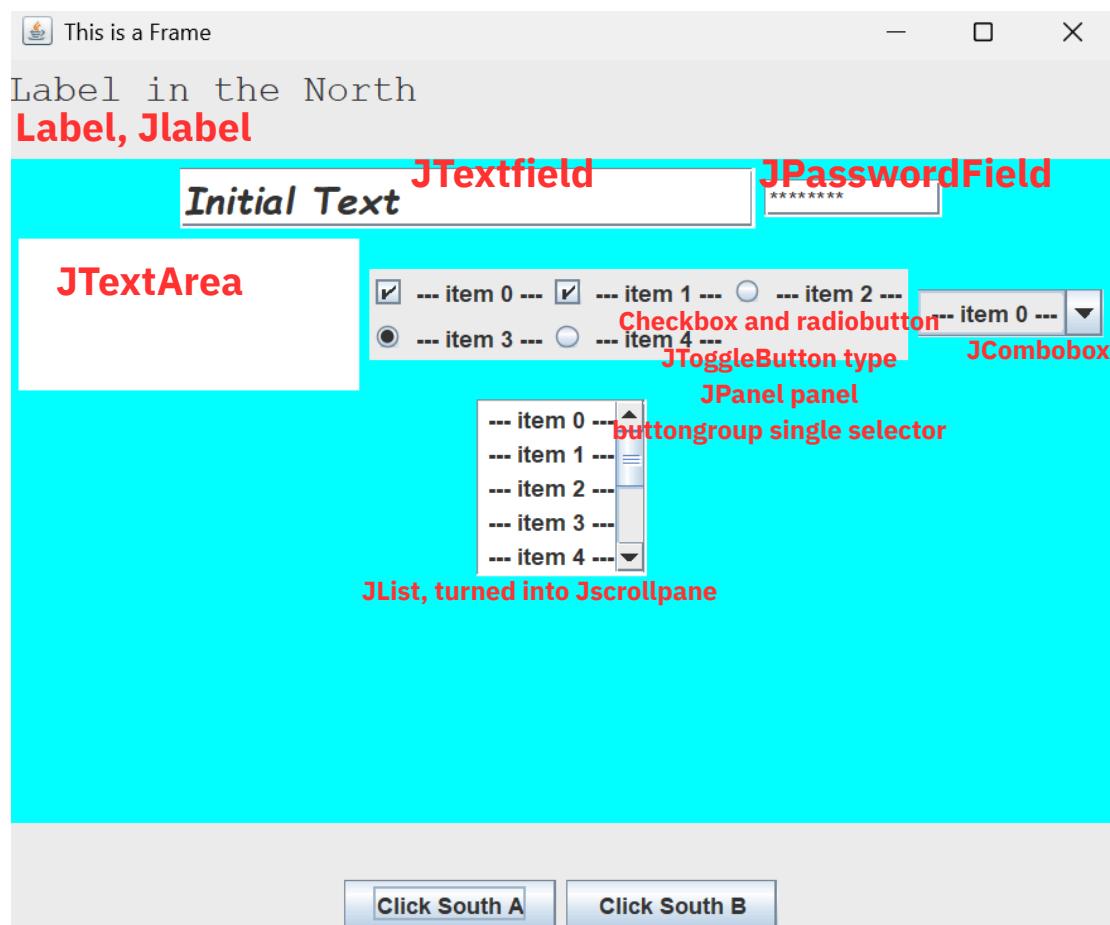
Tough luck. May God us all.

w8_1_Container



```
1 package Lab_Ch8;
2
3 import java.awt.*;
4 import javax.swing.*;
5
6 class w8_1.Container
7 {
8     public static void main(String[] args)
9     {
10        JFrame frame = new JFrame("This is a Frame");
11
12        // ----- (1) Minimum properties that must be set
13        frame.setBounds(200, 200, 600, 300);
14        frame.setVisible(true);
15
16
17        // ----- (2) Other properties
18        //           - which one doesn't work ?
19        //           - notice the difference between EXIT_ON_CLOSE and DISPOSE_ON_CLOSE
20
21        frame.setMaximizedBounds( new Rectangle(800, 500) );
22        frame.setResizable(true);
23        frame.setBackground(Color.GREEN);
24        frame.getContentPane().setBackground( new Color(100, 150, 250) );
25        //frame.setBackground( Color.BLUE );
26        frame.setCursor( new Cursor(Cursor.HAND_CURSOR) );
27        frame.setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
28
29        frame.getContentPane().setLayout(new FlowLayout());
30        frame.getContentPane().add( new JTextField("type anything", 20) );
31
32
33
34        // ----- (1.1) If the frame is visible before components are added,
35        //           must call validate or pack to trigger layout manager & update screen
36        //frame.validate();
37        //frame.pack();
38
39        // ----- (1.2) Or add components first and make frame (with components) visible
40        //           Don't need to call validate or pack
41        frame.setVisible(true);
42
43
44
45        // ----- (3) Dialog
46        //           - notice the difference between dialog with and without owner
47        //           - notice the difference between modal and non-modal dialogs
48
49        JDialog dialog = new JDialog();
50        dialog.setTitle("This is a Dialog");
51        dialog.setModal(true);
52        dialog.setBounds(400, 400, 300, 150 );
53        dialog.setVisible(true);
54        dialog.setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE );
55
56
57
58
59        // ----- (4) Quick dialog
60        //           - notice the difference between dialog with and without owner
61
62        int op = JOptionPane.showConfirmDialog( null,
63                "Press the button", "This is a Confirm Dialog",
64                JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE );
65
66        JOptionPane.showMessageDialog( frame,
67                ("Option = " + op), "This is a Message Dialog",
68                JOptionPane.INFORMATION_MESSAGE );
69
70    }
71 }
```

w8_2_Component



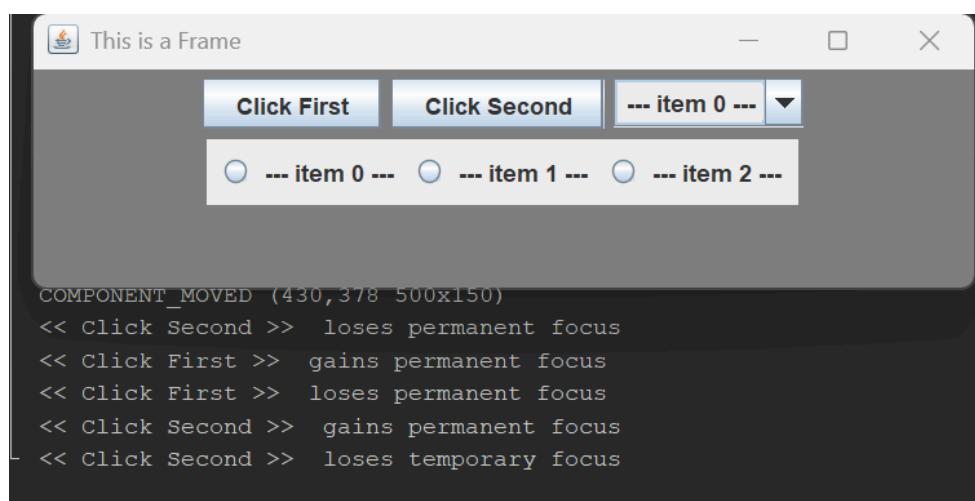
w8_2_Component

```
1 package Lab_Ch8;
2
3 import java.awt.*;
4 import javax.swing.*;
5
6 class w8_2_Component extends JFrame
7 {
8     public w8_2_Component()
9     {
10         this.setTitle("This is a Frame");
11         this.setBounds(200, 100, 600, 500);
12         //this.setSize(600, 500); this.setLocationRelativeTo(null);
13         this.setVisible(true);
14         this.setResizable(true);
15         this setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
16
17
18     // ----- (1) setup layout
19     JPanel contentpane = (JPanel)getContentPane();
20     contentpane.setLayout( new BorderLayout(25, 25) ); //this seems to widen the size of "label in the north"
21
22
23     // ----- (2) setup components
24     String [] items = new String[10];
25     for (int i=0; i < 10; i++) items[i] = " --- item " + i + " ---";
26
27     JLabel label = new JLabel("Label in the North");
28     label.setFont(new Font("Monospaced", Font.PLAIN, 20));
29
30     JButton button1 = new JButton("Click South A");
31     JButton button2 = new JButton("Click South B");
32
33     JPanel cregion = new JPanel();
34     cregion.setBackground( Color.CYAN );
35
36
37     // ----- (3) add components to container
38     //           notice 2 buttons in the south
39     contentpane.add(label, BorderLayout.NORTH);
40     contentpane.add(cregion, BorderLayout.CENTER);
41     //contentpane.add(button1, BorderLayout.SOUTH);
42     //contentpane.add(button2, BorderLayout.SOUTH);
43
44
45
46     JPanel southPanel = new JPanel();
47     southPanel.add(button1);
48     southPanel.add(button2);
49     contentpane.add(southPanel, BorderLayout.SOUTH);
50
51
52
53     // ----- (5) set cursor to other formats
54     button1.setCursor( new Cursor(Cursor.MOVE_CURSOR) );
55
56
57     // ----- (6) add text field, password field, text area
58
59     JTextField tf = new JTextField("Initial Text", 20);
60     tf.setFont(new Font("Comic Sans MS", Font.BOLD+Font.ITALIC, 20));
61     cregion.add(tf);
62
63     JPasswordField pf = new JPasswordField("password", 10);
64     pf.setEchoChar('*');
65     cregion.add(pf);
66
67     JTextArea ta = new JTextArea(5, 20);
68     cregion.add(ta);
69     //JScrollPane tsc = new JScrollPane(ta);
70     //tsc.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
71     //cregion.add(tsc);
72 }
```

w8_2_Component

```
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



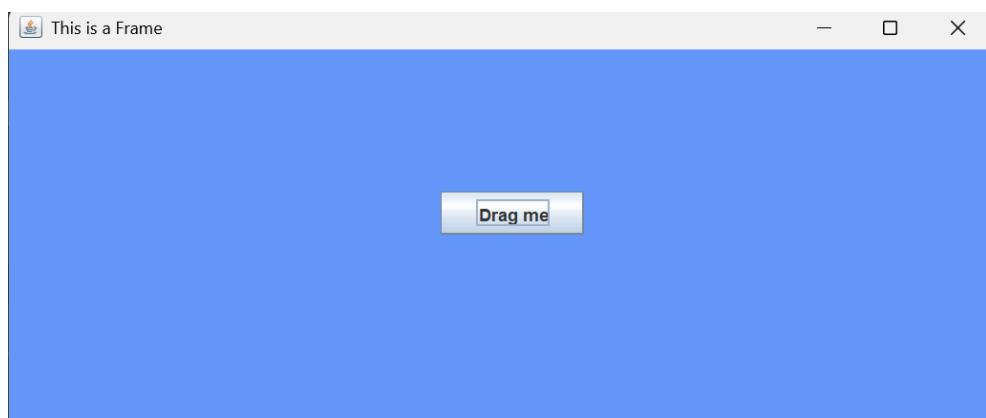
w9_1_ComponentEvent

```
1 package Lab_Ch9;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6
7 // ----- (1) GUI structure & programming concept
8 class w9_1_ComponentEvent extends JFrame
9 {
10     private JPanel      contentpane;
11     private JButton     button1, button2;
12     private JComboBox   combo;
13     private JToggleButton [] tb;
14     private ButtonGroup bgroup;
15
16
17     public w9_1_ComponentEvent()
18     {
19         setTitle("This is a Frame");
20         setSize(500, 150);
21         setLocationRelativeTo(null);
22         setVisible(true);
23         setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
24
25         contentpane = (JPanel)getContentPane();
26         contentpane.setBackground( Color.GRAY );
27         contentpane.setLayout( new FlowLayout() );
28
29         AddComponents();
30         AddListeners();
31     }
32
33
34     public void AddComponents()
35     {
36         String [] items = new String[10];
37         for (int i=0; i < 10; i++) items[i] = " --- item " + i + " ---";
38
39         button1 = new JButton("Click First");
40         button2 = new JButton("Click Second");
41
42         combo = new JComboBox( items );
43
44         bgroup = new ButtonGroup();
45         tb    = new JToggleButton[3];
46         JPanel bpanel = new JPanel();
47         for (int i=0; i < 3; i++)
48         {
49             tb[i] = new JRadioButton( items[i] );
50             bgroup.add( tb[i] );
51             bpanel.add( tb[i] );
52         }
53
54         contentpane.add(button1);
55         contentpane.add(button2);
56         contentpane.add(combo);
57         contentpane.add(bpanel);
58
59         validate();
60     }
61
62
63     public void AddListeners()
64     {
65         // ----- (2) add ComponentListener to frame
66         this.addComponentListener( new MyComponentListener() );
67
68         // ----- (3) add FocusListener to buttons and combo box
69
70         button1.addFocusListener( new MyFocusListener() );
71         button2.addFocusListener( new MyFocusListener() );
72         combo.addFocusListener( new MyFocusListener() );
```

w9_1_ComponentEvent

```
73 }
74
75     public static void main(String[] args)
76     {
77         new w9_1_ComponentEvent();
78     }
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

```
1 package Lab_Ch9;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6
7 class w9_2_MouseEvent extends JFrame
8 {
9     private JPanel contentpane;
10    private MyButton button1;
11
12
13    public w9_2_MouseEvent()
14    {
15        setTitle("This is a Frame");
16        setSize(700, 300);
17        setLocationRelativeTo(null);
18        setVisible(true);
19        setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
20
21        contentpane = (JPanel)getContentPane();
22        contentpane.setBackground( new Color(100, 150, 250) );
23        contentpane.setLayout( null );
24
25        button1 = new MyButton();
26        contentpane.add(button1);
27        repaint();
28    }
29
30
31    public static void main(String[] args)
32    {
33        new w9_2_MouseEvent();
34    }
35 }
36
37 //////////////////////////////////////////////////////////////////
38 class MyButton extends JButton implements MouseListener, MouseMotionListener
39 {
40     private int curX = 100, curY = 20;
41     private int width = 100, height = 30; // for text button
42     //private int width = 100, height = 100; // for icon button
43     private boolean drag;
44
45     // ----- (5) use icon instead of text
46     private ImageIcon clickIcon, dragIcon;
47
48     public MyButton()
49     {
50         String path = "src/main/Java/Lab_Ch9/";
51         clickIcon = new ImageIcon(path + "redbox.png");
52         dragIcon = new ImageIcon(path + "red.png");
53
54         setBounds(curX, curY, width, height);
55         setText("Click me");
56         //setIcon(clickIcon);
57         drag = false;
58
59         // ----- (3) add listeners
60         addMouseListener( this );
61         addMouseMotionListener( this );
62     }
63
64     public void mousePressed( MouseEvent e ) { }
65     public void mouseReleased( MouseEvent e ) { }
66     public void mouseEntered( MouseEvent e ) { }
67     public void mouseExited( MouseEvent e ) { }
68     public void mouseMoved( MouseEvent e ) { }
```

w9_2_MouseEvent

```
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



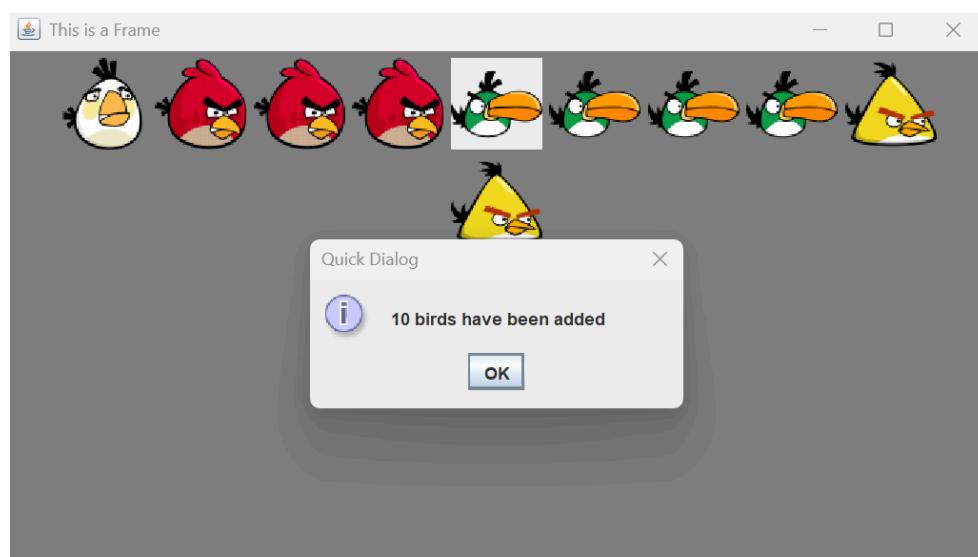
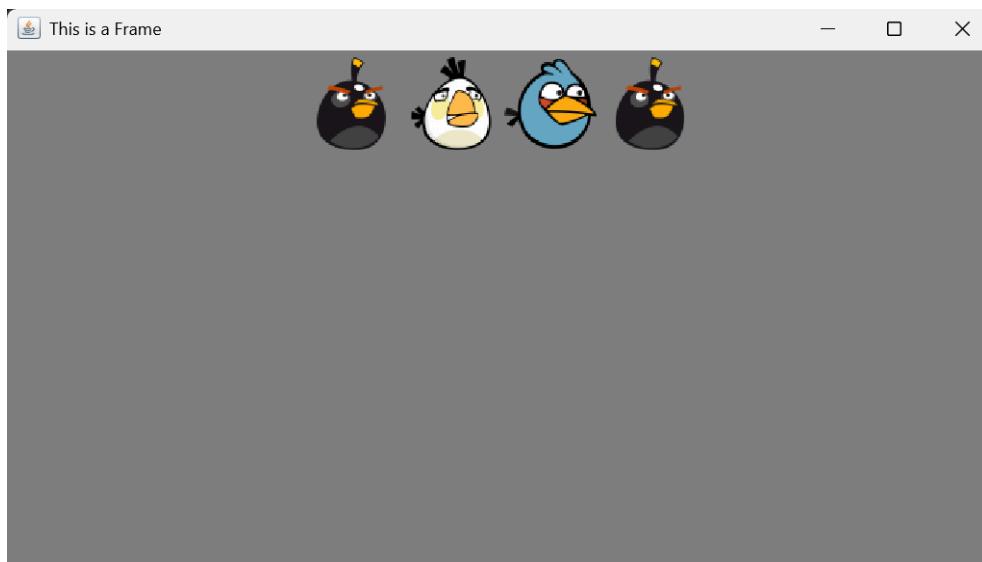
w9_3_KeyEvent

```
1 package Lab_Ch9;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6
7 class w9_3_KeyEvent extends JFrame
8 {
9     private JPanel contentpane;
10    private JPasswordField passfield;
11    private JTextArea textarea;
12    private DoubleText dtext;
13
14    public w9_3_KeyEvent()
15    {
16        setTitle("This is a Frame");
17        setSize(700, 300);
18        setLocationRelativeTo(null);
19        setVisible(true);
20        setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
21
22        contentpane = (JPanel)getContentPane();
23        contentpane.setBackground( new Color(100, 150, 250) );
24        contentpane.setLayout( new FlowLayout() );
25
26        AddComponents();
27
28        // ----- (1) listener is automatically added when dtext is created
29        //           in AddComponents()
30    }
31
32    public void AddComponents()
33    {
34        passfield = new JPasswordField(30);
35        textarea = new JTextArea(8, 30);
36        dtext = new DoubleText( passfield, textarea );
37
38        Font f = new Font("SanSerif", Font.BOLD, 20);
39        passfield.setFont(f);
40        textarea.setFont(f);
41        textarea.setEditable(false);
42
43        contentpane.add(passfield);
44        contentpane.add( new JScrollPane(textarea) );
45        validate();
46    }
47
48    public static void main(String[] args)
49    {
50        new w9_3_KeyEvent();
51    }
52 }
53
54 // -----
55
56 // ----- (1) DoubleText component, with KeyListener
57
58 class DoubleText implements KeyListener
59 {
60     private JPasswordField in;
61     private JTextArea out;
62
63     public DoubleText(JPasswordField p, JTextArea t)
64     {
65         in = p; out = t;
66         in.addKeyListener( this );
67     }
68 }
```

w9_3_KeyEvent

```
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 };
```

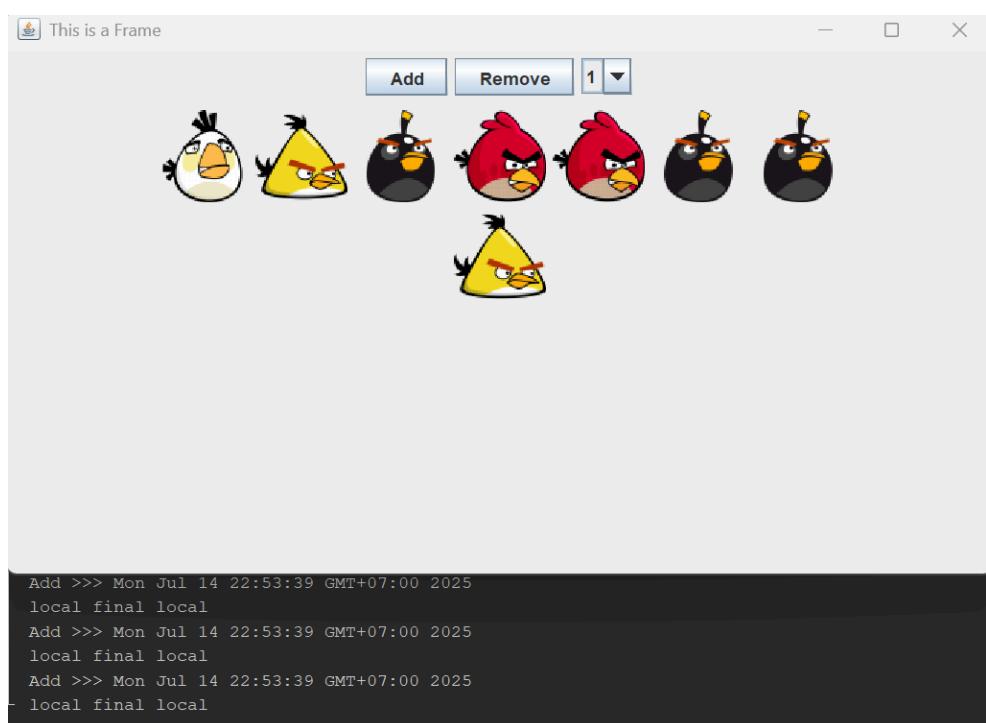
w10_1_WinEvent3



w10_1_WinEvent3

```
1 package Lab_Ch10;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 import javax.swing.event.*;
7
8
9 class w10_1_WinEvent_3 extends JFrame implements MouseListener
10 {
11     private JPanel contentpane;
12     private java.util.Random random;
13
14     private String path = "src/main/Java/Lab_Ch10/";
15     private String[] birds = {"black.png", "blue.png", "green.png",
16                             "red.png", "white.png", "yellow.png"};
17
18     public w10_1_WinEvent_3()
19     {
20         setTitle("This is a Frame");
21         setSize(700, 400);
22         setLocationRelativeTo(null);
23         setVisible(true);
24         setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
25
26         contentpane = (JPanel)getContentPane();
27         contentpane.setLayout( new FlowLayout() );
28         contentpane.setBackground( Color.GRAY );
29
30         random = new java.util.Random();
31
32         // ----- (2) add listeners : notice the difference between (2.1), (2.2), (2.3)
33
34         // ----- (2.1) click anywhere in the frame to add a random bird
35         addMouseListener( this );
36
37         // ----- (2.2) for every 5 birds added, pop up a dialog
38         contentpane.addContainerListener( new MyContainerListener3() );
39
40         // ----- (2.3) count total no. of birds when closing the frame
41         //+++++-----+
42         addWindowListener( new WindowAdapter() {
43             //addWindowListener( new WindowListener() {
44                 @Override
45                 public void windowOpened( WindowEvent e )
46                 {
47                     QuickDialog3.show("Click anywhere to add birds");
48                 }
49                 @Override
50                 public void windowClosing( WindowEvent e )
51                 {
52                     //JFrame frame = (JFrame)e.getWindow();
53                     //JPanel contentpane = (JPanel)frame.getContentPane();
54                     int count = contentpane.getComponentCount();
55                     QuickDialog.show("Total birds = " + count);
56                 }
57
58                 //public void windowClosed( WindowEvent e ) { }
59                 //public void windowActivated( WindowEvent e ) { }
60                 //public void windowDeactivated( WindowEvent e ) { }
61                 //public void windowIconified( WindowEvent e ) { }
62                 //public void windowDeiconified( WindowEvent e ) { }
63             });
64             //+++++-----+
65         }
66
67         // ----- (1) handlers for MouseListener
68         public void mousePressed( MouseEvent e ) { }
69         public void mouseReleased( MouseEvent e ) { }
70         public void mouseEntered( MouseEvent e ) { }
71         public void mouseExited( MouseEvent e ) { }
72
73 }
```

w10 1 WinEvent3



w10_2_SemanticEvent

```
1 package Lab_Ch10;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6
7 class w10_2_SemanticEvent extends JFrame
8 {
9     private JPanel contentpane;
10    private MyPane drawpane;
11    private JButton add_button, rem_button;
12    private JComboBox combo;
13    private JRadioButton [] radio;
14
15    private Integer [] items = {1, 2, 3, 4};
16
17    public w10_2_SemanticEvent()
18    {
19        setTitle("This is a Frame");
20        setBounds(300, 200, 700, 400);
21        setVisible(true);
22        setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE );
23
24        contentpane = (JPanel)getContentPane();
25        contentpane.setLayout( new FlowLayout() );
26        AddComponents();
27    }
28
29    public void AddComponents()
30    {
31        // ----- (1) anonymous listener classes (implicitly implement listeners)
32        String local = "local";
33        final String finlocal = "final local";
34
35        add_button = new JButton("Add");
36        add_button.addActionListener( new ActionListener() {
37            @Override
38            public void actionPerformed( ActionEvent e )
39            {
40                java.util.Date date = new java.util.Date();
41                date.setTime(e.getWhen());
42                System.out.println(e.getActionCommand() + " >> " + date.toString());
43
44                System.out.printf("%s %s \n", local, finlocal);
45
46                drawpane.doAdd();
47            }
48        });
49
50        // local = "new value";
51
52
53        rem_button = new JButton("Remove");
54        rem_button.addActionListener( new ActionListener() {
55            @Override
56            public void actionPerformed( ActionEvent e )
57            {
58                drawpane.doRemove();
59            }
60        });
61
62        // ----- (2) Disable action on remove button --> ActionEvent won't be handled
63        //           But MouseClick is not disabled --> MouseEvent is still handled
64        //rem_button.setEnabled(false);
65        /*
66        rem_button.addMouseListener( new MouseAdapter() {
67            @Override
68            public void mouseClicked( MouseEvent e )
69            {
70                drawpane.doRemove();
71            }
72        });
73        */
74    }
75
76    public static void main( String args[] )
77    {
78        w10_2_SemanticEvent frame = new w10_2_SemanticEvent();
79        frame.setVisible(true);
80    }
81}
```

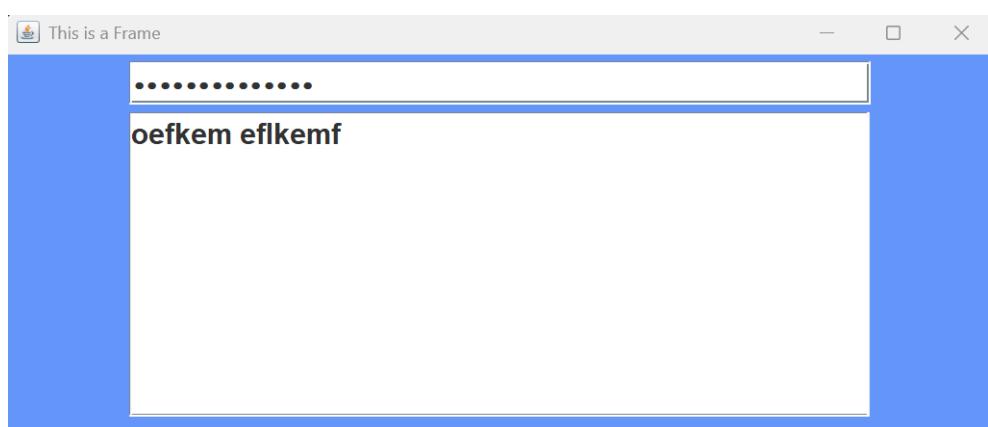
w10_2_SemanticEvent

```
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 }
```

w10_2_SemanticEvent

```
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 xlabel = getComponent(0);
188                 remove( xlabel );
189
190                 // ----- (4) try with and without validate
191                 validate();
192                 repaint();
193             }
194         }
195     }
196 };
197
```

w10_3_KeyEvent



w10_3_KeyEvent

```

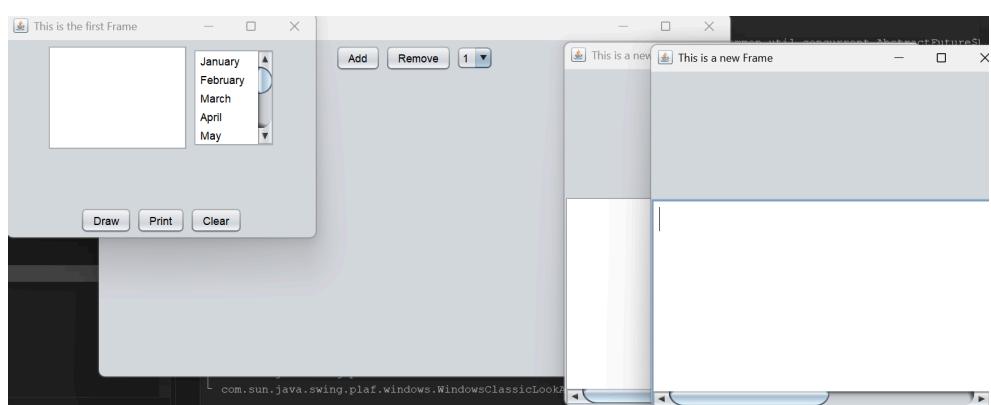
1 package Lab_Ch10;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 import javax.swing.event.*;
7
8 class w10_3_KeyEvent extends JFrame
9 {
10     private JPanel         contentpane;
11     private JPasswordField passfield;
12     private JTextArea       textarea;
13     private DoubleText      dtext;
14
15     public w10_3_KeyEvent()
16     {
17         setTitle("This is a Frame");
18         setBounds(200, 200, 700, 300);
19         setVisible(true);
20         setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
21
22         contentpane = (JPanel)getContentPane();
23         contentpane.setBackground( new Color(100, 150, 250) );
24         contentpane.setLayout( new FlowLayout() );
25
26         AddComponents();
27     }
28
29     public void AddComponents()
30     {
31         passfield = new JPasswordField(30);
32         textarea = new JTextArea(8, 30);
33         dtext = new DoubleText( passfield, textarea );
34
35         Font f = new Font("SanSerif", Font.BOLD, 20);
36         passfield.setFont(f);
37         textarea.setFont(f);
38         textarea.setEditable(false);
39
40         contentpane.add(passfield);
41         contentpane.add( new JScrollPane(textarea) );
42
43         validate();
44     }
45
46     public static void main(String[] args)
47     {
48         new w10_3_KeyEvent();
49     }
50 }
51
52 //----- (1) DoubleText component, with KeyListener
53 /*
54 class DoubleText implements KeyListener
55 {
56     private JPasswordField in;
57     private JTextArea      out;
58
59     public DoubleText(JPasswordField p, JTextArea t)
60     {
61         in = p; out = t;
62         in.addKeyListener( this );
63     }
64
65     @Override
66     public void keyTyped( KeyEvent e )
67     {
68         String current = out.getText();
69
70         if ( e.getKeyChar() != '\b' )
71         {
72             current = current + e.getKeyChar();
73             out.setText(current);
74         }
75         else
76         {
77             int last = current.length() - 1;
78             current = current.substring(0, last);
79             out.setText(current);
80         }
81     }
82
83     public void keyPressed( KeyEvent e )    { }
84     public void keyReleased( KeyEvent e )   { }
85 }
86 */
87
88
89
90

```

w10_3_KeyEvent

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

w10_4_MainFrame

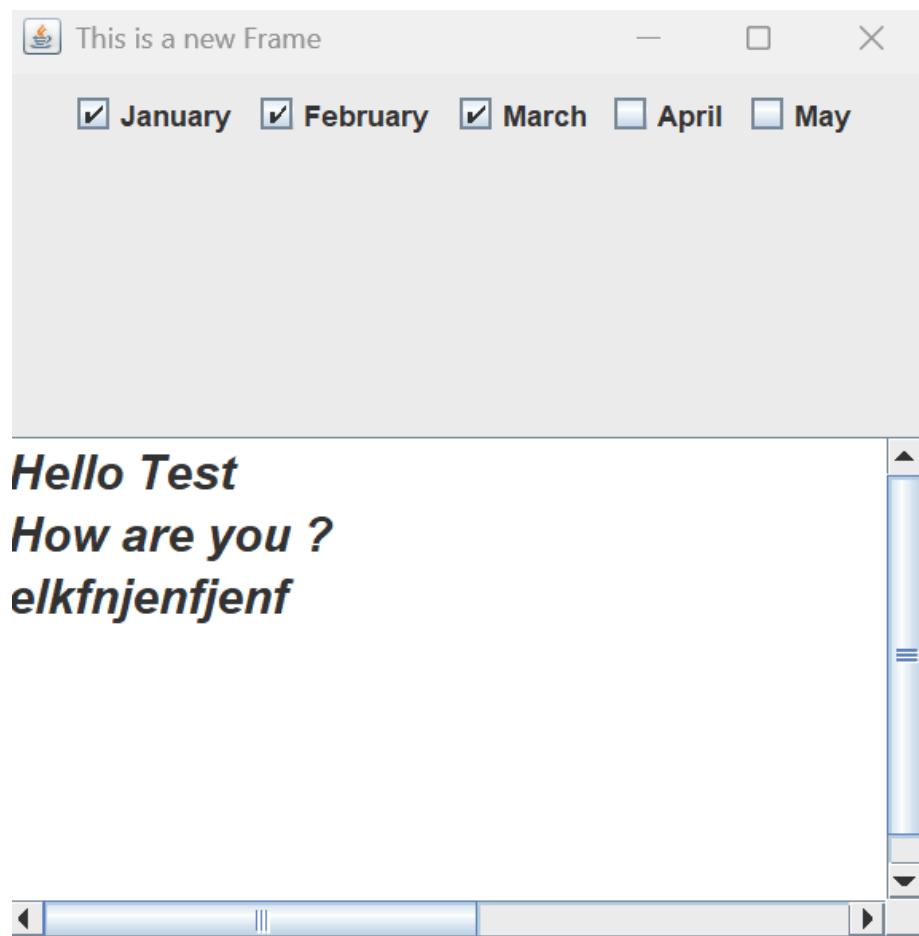


w10_4_MainFrame

```
1 package Lab_Ch10;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 import javax.swing.event.*;
7
8 class w10_4_MainFrame extends JFrame
9 {
10     private JPanel contentpane;
11     private JList list;
12     private JTextArea text;
13     private JButton print_button, draw_button, clear_button;
14
15     private String [] items = {"January", "February", "March", "April", "May", "June",
16         "July", "August", "September", "October", "November", "December"};
17
18     // ----- (1) message to be passed to next frames
19     private Object [] messageFromList;
20     private String messageFromText;
21
22     // ----- (2) next frames
23     w10_2_SemanticEvent sframe;
24     w10_5_PrintFrame pframe;
25
26
27     public w10_4_MainFrame()
28     {
29         setTitle("This is the first Frame");
30         setBounds(200, 200, 350, 250);
31         setVisible(true);
32         setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
33
34         contentpane = (JPanel)getContentPane();
35         contentpane.setLayout( new BorderLayout() );
36
37         AddComponents();
38     }
39
40     public void AddComponents()
41     {
42         // ----- (3) TextArea + CaretEvent
43         text = new JTextArea(6, 15);
44         text.addCaretListener( new CaretListener() {
45             @Override
46             public void caretUpdate( CaretEvent e )
47             {
48                 messageFromText = text.getText();
49             }
50         });
51
52
53         // ----- (4) List + ListSelectionEvent
54         list = new JList( items );
55         list.setVisibleRowCount(5);
56         list.addListSelectionListener( new ListSelectionListener() {
57             @Override
58             public void valueChanged( ListSelectionEvent e )
59             {
60                 if( !e.getValueIsAdjusting() )
61                 {
62                     messageFromList = list.getSelectedValues();
63                 }
64             }
65         });
66
67
68         // ---- (5) Button + ActionEvent : cleanup TextArea and List
69         clear_button = new JButton("Clear");
70         clear_button.addActionListener( new ActionListener() {
71             @Override
72             public void actionPerformed( ActionEvent e )
73             {
74                 text.setText("");
75                 list.clearSelection();
76             }
77         });
78
79
80         // ----- (6) Button + ActionEvent -> open SemanticEvent frame
81         draw_button = new JButton("Draw");
82         draw_button.addActionListener( new ActionListener() {
83             @Override
84             public void actionPerformed( ActionEvent e )
85             {
86                 if (sframe == null) sframe = new w10_2_SemanticEvent();
87                 else sframe.setVisible(true);
88             }
89         });
90     }
```

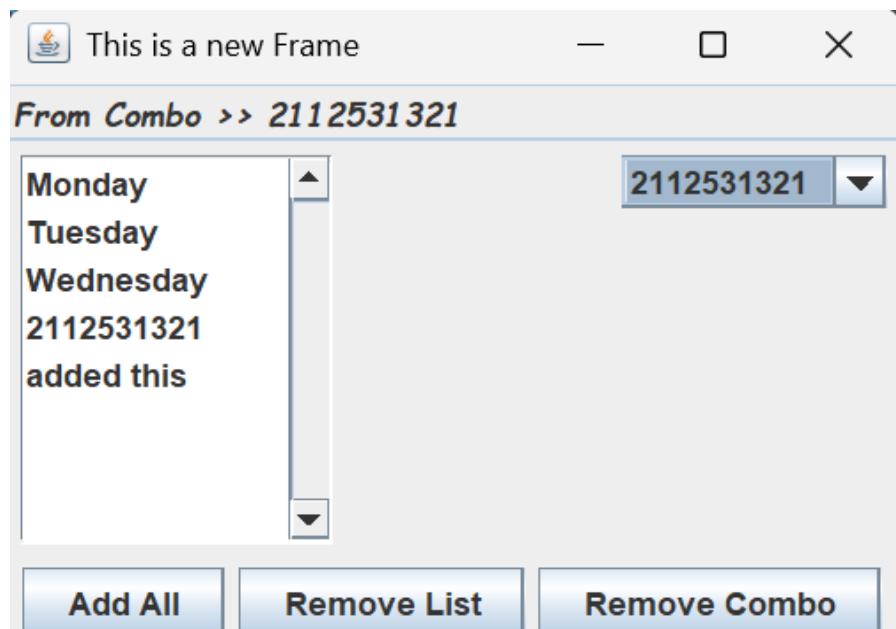
w10_4_MainFrame

```
91 // ----- (7) Button + ActionEvent -> open PrintFrame
92 print_button = new JButton("Print");
93 print_button.addActionListener( new ActionListener() {
94     @Override
95     public void actionPerformed( ActionEvent e )
96     {
97         pframe = new w10_5_PrintFrame();
98
99         // ----- (7.2) get messages from components
100        //           no need to catch events in (3) and (4)
101        //messageFromText = text.getText();
102        //messageFromList = list.getSelectedValues();
103
104        // ----- (7.1) pass messages to the next frame
105        pframe.setTextMessage( messageFromText );
106        pframe.setCheckMessage( messageFromList );
107        pframe.AddComponents();
108
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 }
```



w10_5_PrintFrame

```
1 package Lab_Ch10;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 import javax.swing.event.*;
7
8 class w10_5_PrintFrame extends JFrame
9 {
10     private JPanel      contentpane;
11     private JCheckBox []    check;
12     private JTextArea     text;
13
14     // ----- (1) message from previous frame
15     private Object [] items   = null;
16     private String      message = "";
17
18     public w10_5_PrintFrame()
19     {
20         setTitle("This is a new Frame");
21         setBounds(200, 200, 400, 400);
22         setVisible(true);
23         setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE );
24
25         contentpane = (JPanel)getContentPane();
26         contentpane.setLayout( new BorderLayout() );
27     }
28
29     public void setCheckMessage( Object [] m ) { items = m; }
30     public void setTextMessage( String m )       { message = m; }
31
32
33     public void AddComponents()
34     {
35         JPanel cpanel = new JPanel();
36         if (items != null)
37         {
38             check = new JCheckBox [ items.length ];
39             for (int i=0; i < items.length; i++)
40             {
41                 check[i] = new JCheckBox( items[i].toString() );
42                 cpanel.add( check[i] );
43             }
44         }
45
46         text = new JTextArea(message, 8, 40);
47         text.setFont( new Font("SanSerif", Font.BOLD | Font.ITALIC, 20) );
48
49         contentpane.add(cpanel, BorderLayout.NORTH);
50         contentpane.add(new JScrollPane(text), BorderLayout.SOUTH);
51
52         validate();
53     }
54
55
56     public static void main(String[] args)
57     {
58         // ----- (2) test this frame independently
59         w10_5_PrintFrame frame = new w10_5_PrintFrame();
60
61         // ----- (3) suppose that messages are passed from previous frame
62         frame.setTextMessage( "Hello Test \nHow are you ?" );
63         String [] si = {"January", "February", "March", "April", "May", "June"};
64         frame.setCheckMessage( si );
65
66         frame.AddComponents();
67     }
68 };
69 }
```



w10_6_DataModel

```
1 package Lab_Ch10;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 import javax.swing.event.*;
7
8 class w10_6_DataModel extends JFrame
9 {
10     private JPanel contentpane;
11     private JList list;
12     private DefaultListModel listmodel;
13     private JComboBox combo;
14     private DefaultComboBoxModel combomodel;
15     private JButton add_button, remove_button1, remove_button2;
16     private JTextField text;
17
18     public w10_6_DataModel()
19     {
20         setTitle("This is a new Frame");
21         setBounds(200, 200, 350, 250);
22         setVisible(true);
23         setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
24
25         contentpane = (JPanel)getContentPane();
26         contentpane.setLayout( new BorderLayout() );
27
28         AddComponents();
29     }
30
31     public void AddComponents()
32     {
33         text = new JTextField("Selection", 15);
34         text.setFont(new Font("Comic Sans MS", Font.BOLD | Font.ITALIC, 12));
35         text.setEditable(false);
36
37         String []days = {"Monday", "Tuesday", "Wednesday"};
38
39         // ---- (1) Readonly list
40         //list = new JList(days);
41
42         // ---- (2) Updatable list
43         listmodel = new DefaultListModel();
44         for(String s : days) listmodel.addElement(s);
45         list = new JList(listmodel);
46
47         list.setPreferredSize( new Dimension(100, 80) );
48         list.setSelectionMode(ListSelectionModel.SINGLE_INTERVAL_SELECTION);
49         list.addListSelectionListener( new ListSelectionListener() {
50             @Override
51             public void valueChanged( ListSelectionEvent e )
52             {
53                 // Try with & without condition
54                 if ( !e.getValueIsAdjusting() )
55                 {
56                     // Checking selection from JList (for both readonly & updatable lists)
57                     Object[] items = list.getSelectedValues();
58                     if ( items.length > 0 ) text.setText("From List >> " + items[0]);
59                     else                  text.setText("");
60                 }
61             }
62         });
63
64
65         // ---- (3) Can update JComboBox directly (automatically linked to combomodel)
66         combo      = new JComboBox(days);
67         combomodel = (DefaultComboBoxModel) combo.getModel();
68
69         combo.setPreferredSize( new Dimension(100, 20) );
70         combo.addItemListener( new ItemListener() {
71             @Override
72             public void itemStateChanged( ItemEvent e )
73             {
74                 Object item = combo.getSelectedItem();
75                 text.setText("From Combo >> " + item);
76             }
77         });
78 }
```

w10_6_DataModel

```
79     add_button = new JButton("Add All");
80     add_button.addActionListener( new ActionListener() {
81         @Override
82         public void actionPerformed( ActionEvent e )
83         {
84             // ----- (4) Add new item to listmodel & JComboBox
85             //           No method to add item to JList
86             String item = JOptionPane.showInputDialog("Enter input");
87             if (listmodel != null)    listmodel.addElement(item);
88
89             combo.addItem(item);
90             //if (combomodel != null)  combomodel.addElement(item);
91         }
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 }
```