

# 210301401 ADVANCED JAVA

UNIT	MODULES	WEIGHTAGE
1	File Handling	20 %
2	Java Collection Framework	20 %
3	Event Handling, Swing and GUI Components	20 %
4	Swing, GUI Components and Layout Manager	20 %
5	Database Connectivity (JDBC)	20 %

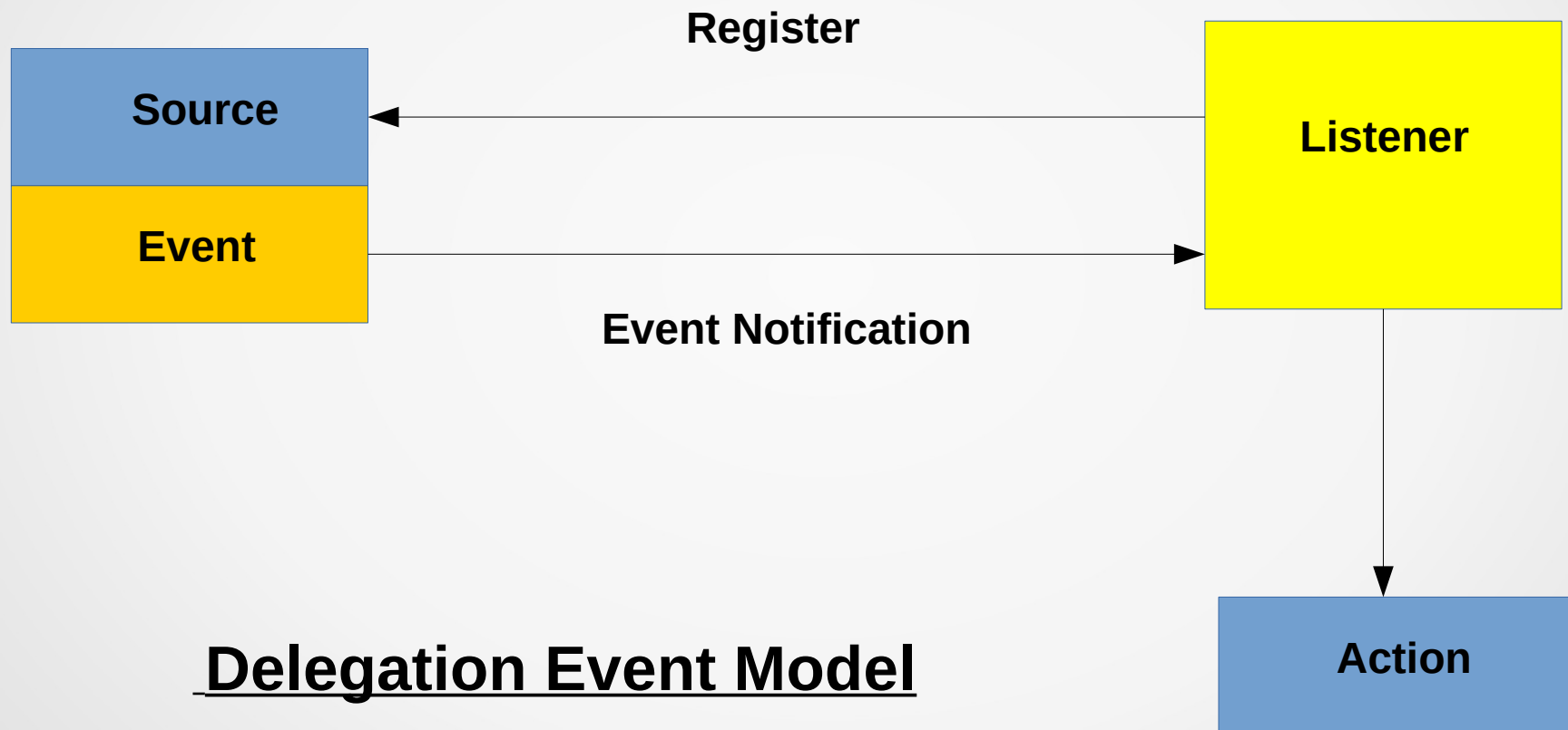
## UNIT -3 Event Handling, Swing and GUI Components

- Event Handling
- Delegation Event Model
- Events
- Events Listeners
- Registering Listeners with sources
- Swing GUI Components

## UNIT - 3 Delegation Event Model

- In Graphical User Interface (GUI) environment, actions are initiated by the press of a button, click of a button, a key press etc.
- There for appropriate mechanisms are needed to capture such events and to react to the events by executing a piece of code.
- Event in Java are handled by **Delegation Event Model**.
- In this model, there is a **source**, which generates events. There is a **listener**, which can listen to the happening of an event and initiate an action. JavaProvide such mechanisms.

# UNIT - 3 Delegation Event Model



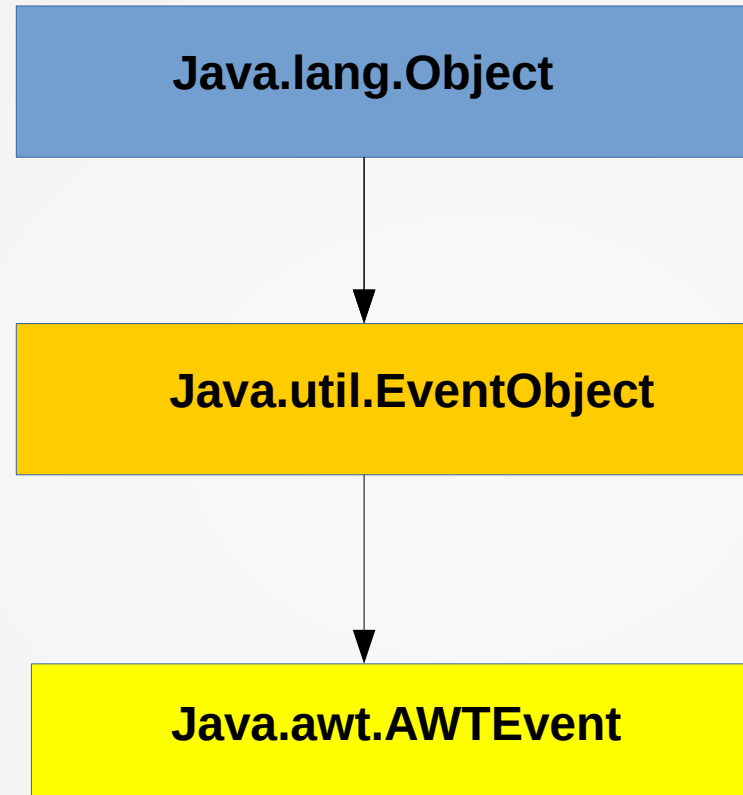
## UNIT - 3 Delegation Event Model

- A **listener has to register with a source**. Any number of listeners can register with a source.
- A listener can register with many event sources.
- When an **event takes place, it is notified to the listeners**, which are registered with the source.
- **The listener then initiates an action.**

# UNIT - 3 Event Handling

- An event is an object that describes the changes of state of a source.
  - i.e – Mouse click is event from the source mouse.
- The superclass of all events is
  - *java.util.EventObject*
- The superclass of all AWT events is
  - *java.util.AWTEvent*
- *AWTEvent* class is an abstract class containing subclasses, which are concrete and are packaged in *java.awt.event*

# UNIT – 3 Event Handling



## UNIT -2 classes of java.awt.event Package

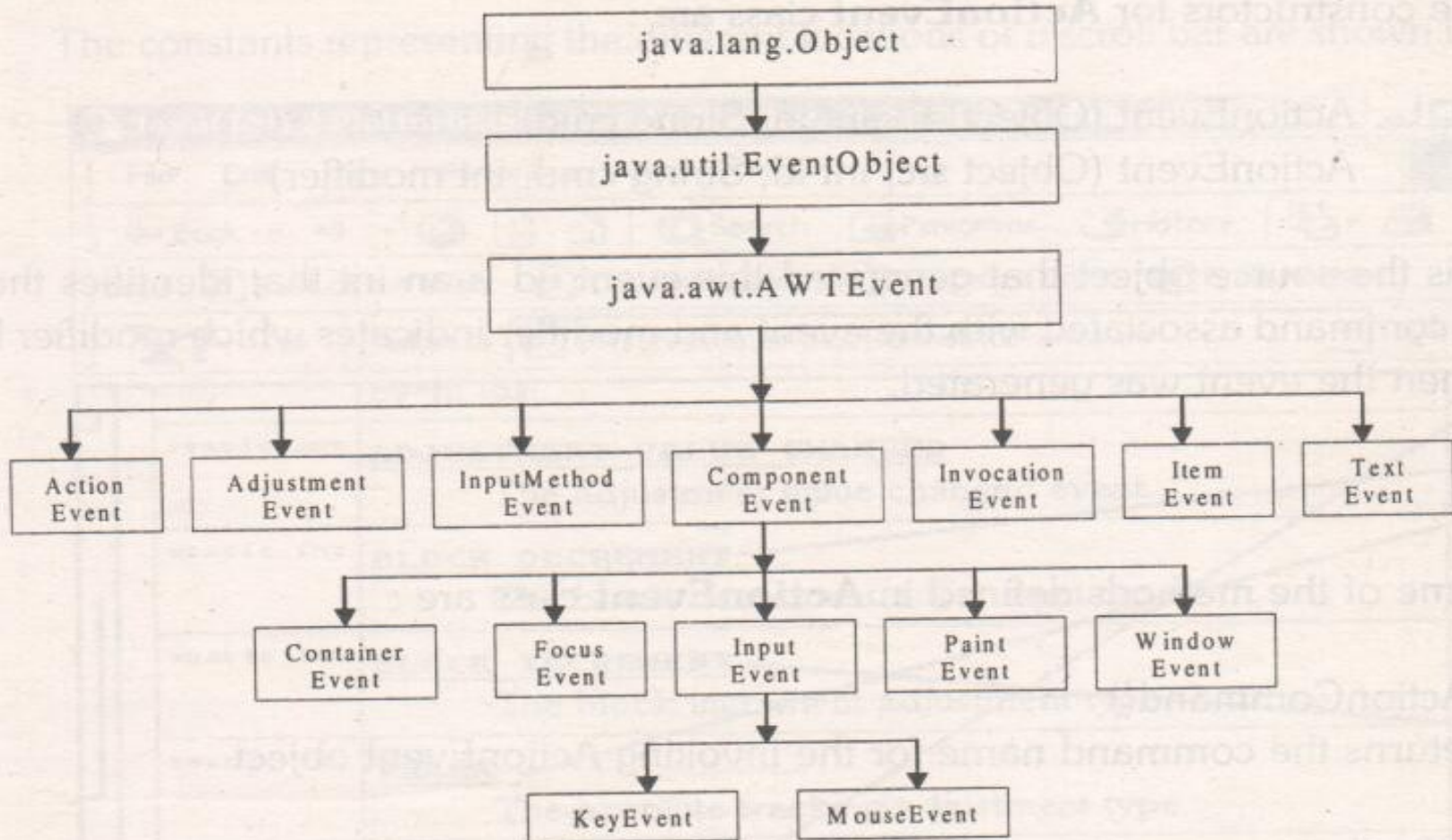
Name of Class	Purpose
<i>Action Event</i>	This class deals with <b>high-level event</b> . The event occurs when the componet-specific action take place. i.e - Button Press, Menu item selection
<i>Adjustment Event</i>	This class deals with events generated by <b>the adjustable objects like Scroll bar change</b>
<i>Component Event</i>	This class deals with the <b>lower – level events</b> . The event occurs when the component is moved, resized or visibility is changed. i.e - Button, Checkbox and Scroll bars display on screen
<i>Item Event</i>	This class deals with the events generated when a <b>Check box or list iteam choice is selcted or deselected</b>



## UNIT -3 classes of java.awt.event Package

Name of Class	Purpose
<i>Key Event</i>	This class deals with the <b>events generated by key strokes.</b> i.e - Key is pressed, typed or released
<i>Mouse Event</i>	This class deals with <b>events generated by mouse clicks and movements.</b> i.e - On mouse action
<i>Text Event</i>	This class deals with <b>events generated by the change of object's text.</b> i.e. - A text of an object is changed
<i>Window Event</i>	This class deals with events generated <b>by the change of window status.</b> i.e - Indicates the changes in the status of the window

# UNIT – 3 Event Handling



**Fig.18.3 Event Class Hierarchy**

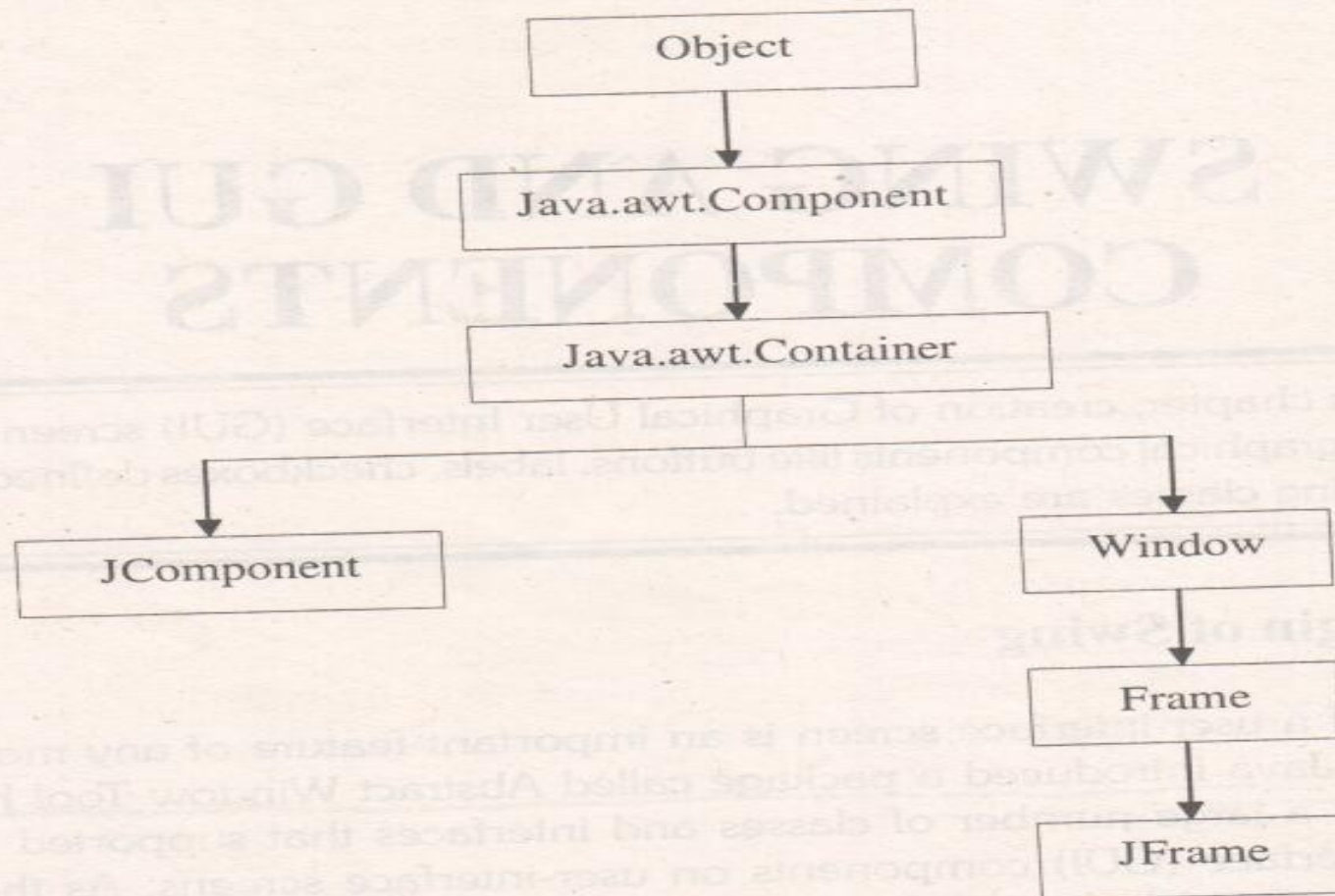
# UNIT - 3 Swing GUI Components

- Initially, Java introduced a package called **Abstract Window Took Kit (AWT)**.
- This package contained a large number of classes and interfaces that **supported the creation of GUI**.
- Newer version of this package in Java2 is called **Swing**.
- The Swing classes are a part of the **Java Foundation Classes (JFC)**.
- The Swing classes are contained in a Java extension package called **javax**.

## UNIT - 3 Swing GUI Components

- The Swing classes are subclasses of **java.awt.Container** & **java.awt.Component**
- The name of the Swing class starts **with the latter J**.
- The **top – level class of swing is Jcomponent** is both a container and a component..
- GUI Components like **button, label, checkbox etc are handled in Jcomponent class**.
- GUI components can be added on a panel window or a frame window.
- The frame in Swing is handled in **Jframe class**.

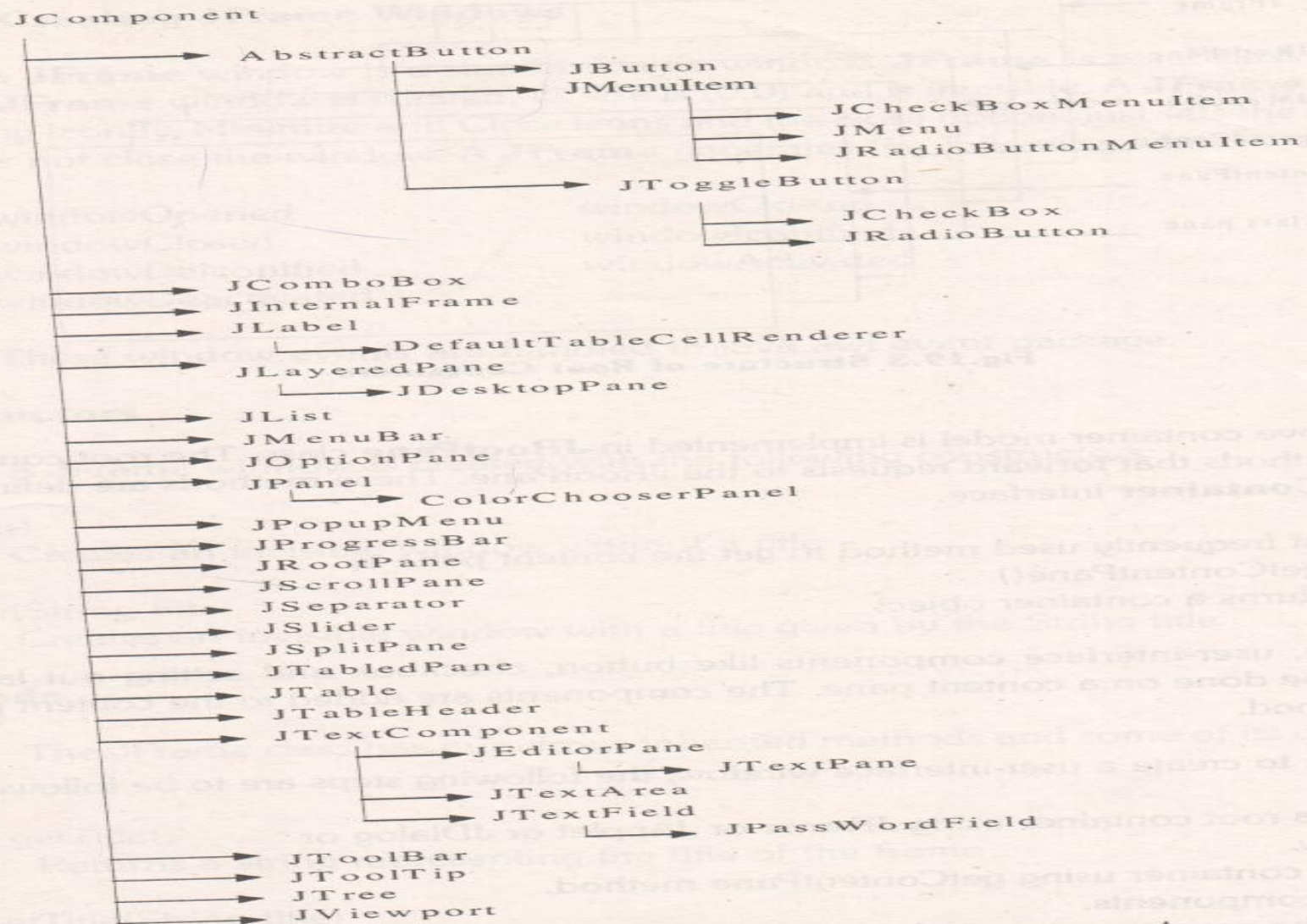
## UNIT – 3 Class Hierarchy of AWT, Jcomponent and JFrame



**Fig.19.1 Class Hierarchy of AWT, JComponent and JFrame**



# UNIT – 3 Subclasses of JComponent



## UNIT - 3 Creating Window in Swing

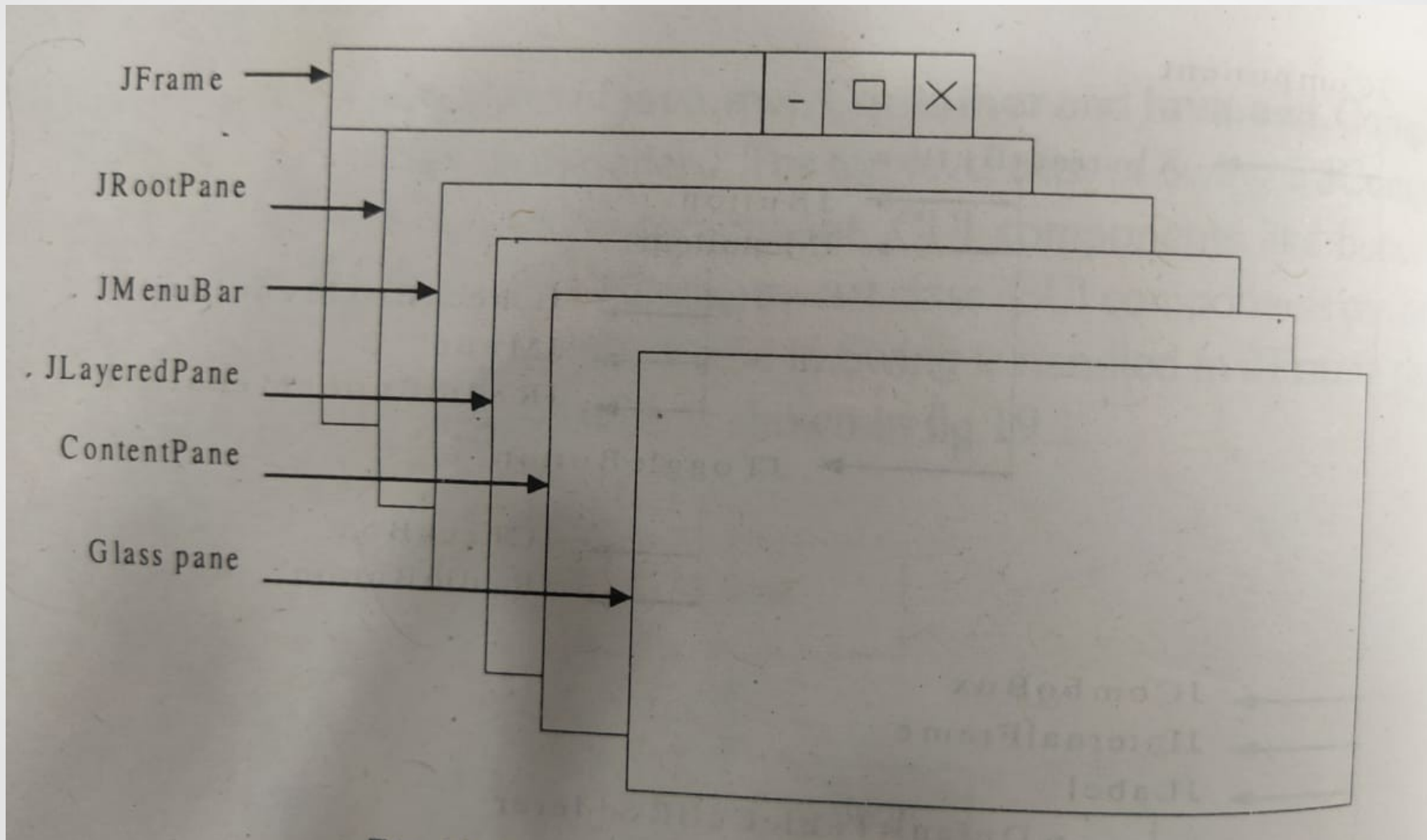
- A Conventional window is created in Swing in a top-level window.
- The top level window is supported by following classes:
  - *JFrame*
  - *JApplet*
  - *JDialog*
  - *JWindow*
- They are called **root container**.

# UNIT - 3 Creating Window in Swing

- These root containers have several panes,
  - *JrootPane*
  - *JmenuBar*
  - *JlayeredPane*
  - *Content pane*
  - *Glass pane*



# UNIT - 3 Creating Window in Swing



# UNIT - 3 Creating Window in Swing

- These root containers have several panes,
  - *JrootPane*
  - *JmenuBar*
  - *JlayeredPane*
  - *Content pane*
  - *Glass pane*
- The above container model **is implemented in JrootPane class**. The root container classes have methods that forward requests to the JrootPane.
- The most frequently used method to get the content pane is :
  - *GetContentPane()*
- The components are added to the content pane using ***add() method***.

## UNIT - 3 Creating Window in Swing

- To create a user-interface window, **the following steps are to be followed:**
  - **Create a root container** using **JFrame** or **JApplet** or **JDialog** or **Jwindow**.
  - **Get the container** using **getContentPane** method.
  - **Create components**
  - **Attach the components to a container** using **add()** method

## UNIT - 3 Swing - JFrame

- A JFrame window is a standard style window.
- **JFrame** is a subclass of **Frame** class.
- When **JFrame** is created, its size is (0,0) and is **invisible**.

# UNIT - 3 Swing - JFrame

- A **JFrame** generates the following window events:
  - windowOpened
  - WindowClosed
  - WindowDeiconified
  - WindowDeactivated
  - WindowClosing
  - WindowIconfied
  - WindowActivated
- **These window events are handled in java.awt.event**

## UNIT -3 Events – WindowEvent Class

Event	Purpose
<b><i>WINDOW_ACTIVATED</i></b>	This event occurs when the <b>window becomes the user's active window</b>
<b><i>WINDOW_CLOSED</i></b>	This event occurs after the <b>window has been closed</b>
<b><i>WINDOW_CLOSING</i></b>	This <b>event occurs when user attempt to close the window</b>
<b><i>WINDOW_DEICONIFIED</i></b>	This event occurs when the <b>window has been changed from a msinimized state to a normal state</b>
<b><i>WINDOW_ICONIFIED</i></b>	This event occurs when the <b>window has been changed from a normal state to a minimized state</b>
<b><i>WINDOW_OPENED</i></b>	This event occurs when the <b>window is made visible</b>

## UNIT -3 Events Listeners – WindowListener

Interface	Interface Methods
<b>WindowListener</b>	<i>Void windowActivated(WindowEvent we)</i>
	<i>Void windowDeactivated(WindowEvent we)</i>
	<i>Void windowClosed(WindowEvent we)</i>
	<i>Void windowClosing(WindowEvent we)</i>
	<i>Void windowDeiconified(WindowEvent we)</i>
	<i>Void windowIconified(WindowEvent we)</i>
	<i>Void windowOpened(WindowEvent we)</i>

# UNIT - 3 Swing - JFrame

- A **JFrame** Constructors
  - *JFrame()*
  - *JFrame(String title)*



## UNIT - 3 Swing - JFrame

Event	Purpose
<i>String getTitle()</i>	Returns a <b>string representing the title</b> of the frame
<i>Void setTitle(String title)</i>	<b>Sets the title</b> of the frame to this string
<i>Void setVisible(boolean b)</i>	<b>Shows or hides this frame window</b>
<i>Void setSize(int width, int height)</i>	Sets the <b>size of the window to the pecified</b> width and height in pixels
<i>Void setLayout(LayoutManager mgr)</i>	<b>Sets the layout</b> manager for this container
<i>Int getX()</i>	Returns the <b>X component</b> of the window location

## UNIT - 3 Swing - JFrame

Event	Purpose
<i>Int getY()</i>	Returns the Y component fo the window location
<i>Void setLocation(int x, int y)</i>	<b>Moves the frame window to a new location</b> on the scree, the top left corner of the window is specified by x and y
<i>Int getHeight()</i>	Returns the <b>current height</b> of the window
<i>Int getWidth()</i>	Returns the <b>current width</b> of the window

# UNIT – 3 Swing - JFrame

- Examples :
  - JFrame1.java
  - JFrame2.java

# UNIT – 3 MouseEvent Class

- A Mouse event is generated by **mouse action in a component.**
- **Two types of events**
  - **Mouse Event**
    - This event generated when a mouse button is **pressed, released, clicked, mouse enters a componenet or mouse exits a component.**
  - **Mouse Motion Event**
    - This event generated when the **mouse is moved or dragged.**

## UNIT -3 Events – MouseEvent class

Constants	Purpose
<i><b>MOUSE_CLICKED</b></i>	This represents the mouse clicked event. This MouseEvent occurs when a <b>mouse button is pressed and released.</b>
<i><b>MOUSE_ENTERED</b></i>	This represents the mouse entered event. This MouseEvent occurs when a <b>mouse cursor enters a component's area.</b>
<i><b>MOUSE_EXITED</b></i>	This represents the mouse exited event. This MouseEvent occurs when a <b>mouse cursor exits a component's area.</b>
<i><b>MOUSE_PRESSED</b></i>	This represents the mouse pressed event. This MouseEvent occurs when a <b>mouse button is pushed down.</b>
<i><b>MOUSE_RELEASED</b></i>	This represents the mouse released event. This MouseEvent occurs when a <b>mouse button is released.</b>

## UNIT -3 Events – MouseEvent class

Constants	Purpose
<b>MOUSE_DRAGGED</b>	This represents the mouse dragged event. This <b>MouseEvent</b> occurs when a mouse is dragged.
<b>MOUSE_MOVED</b>	This represents the mouse moved event. This <b>MouseEvent</b> occurs when a mouse is moved.

- **Constructor**
  - *MouseEvent(Component src, int id, long when, int modifiers, int x, int y, int clickcount, boolean puptrig)*

## UNIT -3 MouseEvent Class -Methods

Method Name	Purpose of Method
<i>Int getX()</i>	Returns an <b>integer</b> representing the <b>x position</b> of the event relative to the component.
<i>Int getY()</i>	Returns an <b>integer</b> representing the <b>y position</b> of the event relative to the component.
<i>Void translatePoint(int x, int y)</i>	Translates the event's <b>co-ordinates</b> to a <b>new position</b> by adding <b>x</b> and <b>y</b> to the <b>x</b> and <b>y</b> of the <b>current position</b>
<i>Int clickCount()</i>	Returns the <b>number of clicks</b> associated with this event.
<i>Boolean isPopupTrigger()</i>	Return <b>ture</b> , if this event is the <b>popup menu</b> trigger for this platform
<i>String paramString()</i>	Returns a <b>string</b> identifying this event.

## UNIT -3 Events Listeners – MouseListener

Interface	Interface Methods
<b><i>MouseListener</i></b>	<i>Void mouseClicked(MouseEvent me)</i>
	<i>Void mouseEntered(MouseEvent me)</i>
	<i>Void mouseExited(MouseEvent me)</i>
	<i>Void mousePressed(MouseEvent me)</i>
	<i>Void mouseReleased(MouseEvent me)</i>



## UNIT -3 Events Listeners – MouseMotionListener

Interface	Interface Methods
<b>MouseMotionListener</b>	<i>Void mouseDragged(MouseEvent me)</i>
	<i>Void mouseMoved(MouseEvent me)</i>

- Examples :
  - MouseListenerExample2.java

# UNIT - 3 Swing - JButton

- The JButton is a concrete **subclass of abstract Button** which is a sub class of **Jcomponent**.
- When a button is clicked, an **ActionEvent** is created.
- The JButton class is used to **mouse press** and **mouse release events** can be precessed separately.
- **Constructors:**
  - *JButton()*
  - *JButton(String label)*
  - *JButton(Icon i)*
  - *Jbutton(String label, Icon i)*

## UNIT - 3 Swing - Jbutton class hierarchy

- *Java.lang.Object*
  - *Java.awt.Component*
    - *Java.awt.Container*
      - *Javax.swing.JComponent*
        - *Javax.swing.AbstractButton*
          - *Javax.swing.JButton*

## UNIT - 3 Swing -Jbutton Methods

Method	Description
<b><i>Void addActionListener(ActionListener al)</i></b>	<b>Add the specified action listener to receive action from this button.</b>
<b><i>String getActionCommand()</i></b>	<b>Returns the command name of the action event fired by this button.</b>
<b><i>Void setText(String label)</i></b>	<b>Sets the button's label to the specified string</b>
<b><i>Void getText(String label)</i></b>	<b>Returns the label of the button</b>

## UNIT - 3 Swing -Jbutton Methods

Method	Description
<i>Icon getIcon()</i>	<b>Returns the icon</b> of the button
<i>Void setIcon(Icon i)</i>	<b>Sets the icon</b> for this button
<i>Void removeActionEvent(ActionEvent ae)</i>	<b>Remove the actionlistener</b>
<i>Void processActionEvent(ActionEvent ae)</i>	<b>Processes the action events</b> occurring on this button
<i>Void setRolloverIcon(Icon i)</i>	<b>Sets the icon i as the rollover icon</b> for the button

## UNIT - 3 Action Event class

- An ActionEvent is **generated** when a button is pressed or menu item is selected.
- **Constructors:**
  - *ActionEvent(Object src, int id, String cmd)*
  - *ActionEvent(Object src, int id, String cmd, int modifier)*

## UNIT -3 Events – ActionEvent Class

Constants	Purpose
<i>ALT_MASK</i>	The alt modifier. An indicator that the <b>alt key was held down</b> during the event.
<i>CTRL_MASK</i>	The control modifier. An indicator that <b>the control key was held</b> down during the event.
<i>META_MASK</i>	The meta modifier. An indicator that the <b>meta key was held down</b> during the event.
<i>SHIFT_MASK</i>	The shift modifier. An indicator that the <b>shift key was held down</b> during the event.

## UNIT -3 Action Event Class -Methods

Method Name	Purpose of Method
<b><i>String getActionCommand()</i></b>	<i>Return the command name for the invoking ActionEvent object.</i>
<b><i>Int getModifier()</i></b>	Return an int value that indicates which modifier key was pressed when the event was generated.
<b><i>String paramString()</i></b>	Returns a string identifying the event.



## UNIT -3 Action Listeners

Interface	Interface Methods
<b>ActionListener</b>	<i>Void actionPerformed(ActionEvent ae)</i>

# UNIT – 3 Swing -Jbutton

- Examples :
  - Jfrmbut\_1.java
  - Jfrmbut\_img.java

# UNIT - 3 Swing - JLabel

- JLabel is a built in Java Swing class that holds text you can display within an applet.
- JLabel class is concrete subclass of **Jcomponent**.
- A JLabel display a **single line of read only text** in a container.
- Java.lang.Object
  - *Java.awt.Component*
    - *Java.awt.Container*
      - *Javax.swing.Jcomponent*
        - *Javax.swing.JLabel*

# UNIT - 3 JLabel Constructors

- ***JLabel ()***
- ***JLabel (Icon image)***
- ***JLabel (Icon image, int horizontalAlignment)***
- ***JLabel (String text)***
- ***JLabel (String text, Icon icon, int horizontalAlignment)***
- ***JLabel (String text, int horizontalAlignment)***

# UNIT - 3 Jlabel Align

- The JLabel has the following int type constants that indicate the alignment of the label content:
  - ***JLabel.CENTER***
  - ***JLabel.LEFT***
  - ***JLabel.RIGHT***
  - ***JLabel.TOP***
  - ***JLabel.BOTTOM***

## UNIT - 3 Swing -JLabel Methods

Method	Description
<i>Int getHorizontalAlignment()</i>	Returns the <b>horizontal alignment</b> for the label's content.
<i>Int getVertical Alignment()</i>	Retruns the <b>vertical alignment</b> for the label's content
<i>Icon getIcon()</i>	<b>Returns the icon</b> of the label
<i>String getText()</i>	<b>Returns the text of the label</b>
<i>void setFont(Font f)</i>	<b>Sets the font</b> for the label's text
<i>void setText(String str)</i>	<b>Sets the specified string</b> str as the label's content

## UNIT - 3 Swing -JLabel Methods

Method	Description
<i>Void setHorizontalAlignment(int alignment)</i>	Sets the horizontal alignment for the label's content
<i>Void setIcon(Icon i)</i>	Sets the specified icon as the label's content
<i>Void setVerticalAlignment(int alignment)</i>	Sets the vertical alignment for the label's content

# UNIT – 3 Swing -JLabel

- JFrmlbl\_1.java



## UNIT - 3 Adjustment Event class

- The Adjustment event is **generated by a scroll bar**.
- **Five types of adjustment events** are defined for the adjustment of a scroll bar.
- **Constructors:**
  - *AdjustmentEvent(Adjustable src, int id, int type, int value)*

## UNIT -3 Events – AdjustmentEvent class

constants	Purpose
<b><i>BLOCK_DECREMENT</i></b>	The mouse is clicked <b>inside the scroll bar to decrease</b> its value
<b><i>BLOCK_INCREMENT</i></b>	The mouse is clicked <b>inside the scroll bar to increase</b> its value
<b><i>TRACK</i></b>	The <b>slider is dragged</b>
<b><i>UNIT_DECREMENT</i></b>	The <b>button at the end of the scroll bar is clicked to decrease</b> its value
<b><i>UNIT_INCREMENT</i></b>	The <b>button at the end of the scroll bar is clicked to increase</b> its value

## UNIT -3 AdjustmentEvent Class -Methods

- The listener interfaces are defined in **java.awt.event** package

class	Methods
<i>Adjustable</i> <b>getAdjustable()</b>	Returns the <b>adjustable object</b> where this event originated
<i>Int</i> <b>getAdjustableType()</b>	Returns <b>the type of adjustment</b> which caused the value changed event
<i>Int</i> <b>getValue()</b>	Returns the <b>current value</b> in the adjustment event
<i>String</i> <b> paramString()</b>	Returns a <b>string</b> representing the state of this event.

## UNIT -3 Events Listeners – AdjustmentListener

- The listener interfaces are defined in **java.awt.event** package

Interface	Interface Methods
<b>AdjustmentListener</b>	<i>Void adjustmentValueChanged(AjustmentEvent ae)</i>

## UNIT - 3 ComponentEvent class

- The Component **is an object having a graphical representation** that can be displayed on the screen and that can interact with user.
- Buttons, Checkboxes and scroll bars are the example of components.
- Component event **is generated when a component is moved, changed in size or changed in visibility.**
- **Constructors:**
  - *ComponentEvent(Component src, int id)*

## UNIT -3 Events – ComponentEvent class

constants	Purpose
<b><i>COMPONENT_MOVED</i></b>	This event indicates that the <b>component position has changed.</b>
<b><i>COMPONENT_RESIZED</i></b>	This event indicates that the <b>component size has changed</b>
<b><i>COMPONENT_SHOWN</i></b>	This event indicates that the <b>component was made visible</b>
<b><i>COMPONENT_HIDDEN</i></b>	This event indicates that the <b>component was made invisible</b>

## UNIT -3 Component Event Class -Methods

- The listener interfaces are defined in **java.awt.event** package

class	Methods
<i>Component</i> <i>getComponent()</i>	Returns the <b>originator of the event</b>
<i>String</i> <i> paramString()</i>	Returns the <b>String identifying the event</b>

## UNIT -3 Events Listeners – ComponentListener

- The listener interfaces are defined in **java.awt.event** package

Interface	Interface Methods
<b>ComponentListener</b>	<i>Void componentHidden(ComponentEvent ce)</i>
	<i>Void componentMoved(ComponentEvent ce)</i>
	<i>Void componentResized(ComponentEvent ce)</i>
	<i>Void componentShown(ComponentEvent ce)</i>



## UNIT - 3 ItemEvent class

- A semantic event indicates that an item, like check box or choice is selected or deselected.
- **Constructors:**
  - *ItemEvent(ItemSelectable src, int id, Object item, int stateChange)*

## UNIT -3 Events – ItemEvent class

constants	Purpose
<b><i>DESELECTED</i></b>	This state change value <b>indicates</b> that an item is <b>deselected</b>
<b><i>ITEM_STATE_CHANGED</i></b>	This event <b>indicates</b> that an item's state has <b>hanged.</b>
<b><i>SELECTED</i></b>	This state change value <b>indicates</b> that an item is <b>selected.</b>

## UNIT -3 Item Event Class -Methods

class	Methods
<i>ItemSelectable getItemSelectable()</i>	<i>Returns the ItemSelectable object that originated the event.</i>
<i>Object getItem()</i>	<b>Returns the item object</b> that was affected by the event.
<i>Int getStateChange()</i>	Returns an <b>integer</b> that indicates <b>whether the item was selected or deselected.</b>
<i>String paramString()</i>	Returns a string identifying the event.

## UNIT -3 Events Listeners – ItemListener

Interface	Interface Methods
<b>ItemListener</b>	<i>Void itemStateChanged(ItemEvent ie)</i>

## UNIT - 3 KeyEvent class

- The key event is generated when key is pressed, typed or released.
- **Constructors:**
  - *KeyEvent(Component src, int id, long when, int modifier, int keycode, char keyChar)*
  - *KeyEvent(Component src, int id, long when, int modifier, int keycode)*

## UNIT -3 Events – KeyEvent class

constants	Purpose
<b><i>KEY_TYPED</i></b>	This event is <b>generated when a character is entered</b>
<b><i>KEY_PRESSED</i></b>	This event is <b>generated when a key is pushed down</b>
<b><i>KEY_RELEASED</i></b>	This event is <b>generated when a key is released</b>
<b><i>VK_0 to VK_9</i></b>	Represents the keys <b>ASCII 0 to ASCII 9</b>
<b><i>VK_A to VK_Z</i></b>	Represents the keys <b>ASCII A to ASCII Z</b>

## UNIT -3 KeyEvent Class -Methods

<i>class</i>	<i>Methods</i>
<b><i>Int getKeyCode()</i></b>	<b><i>Returns the integer code for an actual key on the keyboard.</i></b>
<b><i>Void setKeyCode()</i></b>	<b><i>Sets the keyCode value to represent a physical key</i></b>
<b><i>Void setKeyChar(char keyChar)</i></b>	<b><i>Sets the keychar value to represent a logical character</i></b>
<b><i>Char getKeyChar()</i></b>	<b><i>Returns the Unicode character defined for this key event.</i></b>

## UNIT -3 KeyEvent Class -Methods

<i>class</i>	<i>Methods</i>
<i>String getKeyText(int keyCode)</i>	<i>Returns a string describing the keyCode such as “Home”, F1”...</i>
<i>String getKeyModifierText(int modifiers)</i>	<i>Returns a String describing the modifier keys such as “Shift” or “Shif” + “ctrl” that were held down during the event.</i>
<i>Boolean isActionKey()</i>	<i>Returns true if the key is an action key</i>
<i>String paramString()</i>	<i>Returns a parameter string identifying this event.</i>



## UNIT -3 Events Listeners – KeyListener

Interface	Interface Methods
<b>KeyListener</b>	<i>Void keyPressed(KeyEvent ke)</i>
	<i>Void keyReleased(KeyEvent ke)</i>
	<i>Void keyTyped(KeyEvent ke)</i>

## UNIT - 3 TextEvent class

- A text event is generated when the text of an object is changed.
- **Constructors:**
  - *TextEvent(Component src, int id)*
- **Constant:**
  - *TEXT\_VALUE\_CHANGED*
    - which indicates that the object's text is changed.

## UNIT - 3 TextEvent class

- **Method:**
  - *String paramString()*
    - Returns a string identifying this text event.
- **TextListener:**
  - *Void textValueChanged(TextEvent te)*



# **UNIT 3 COMPLETED**