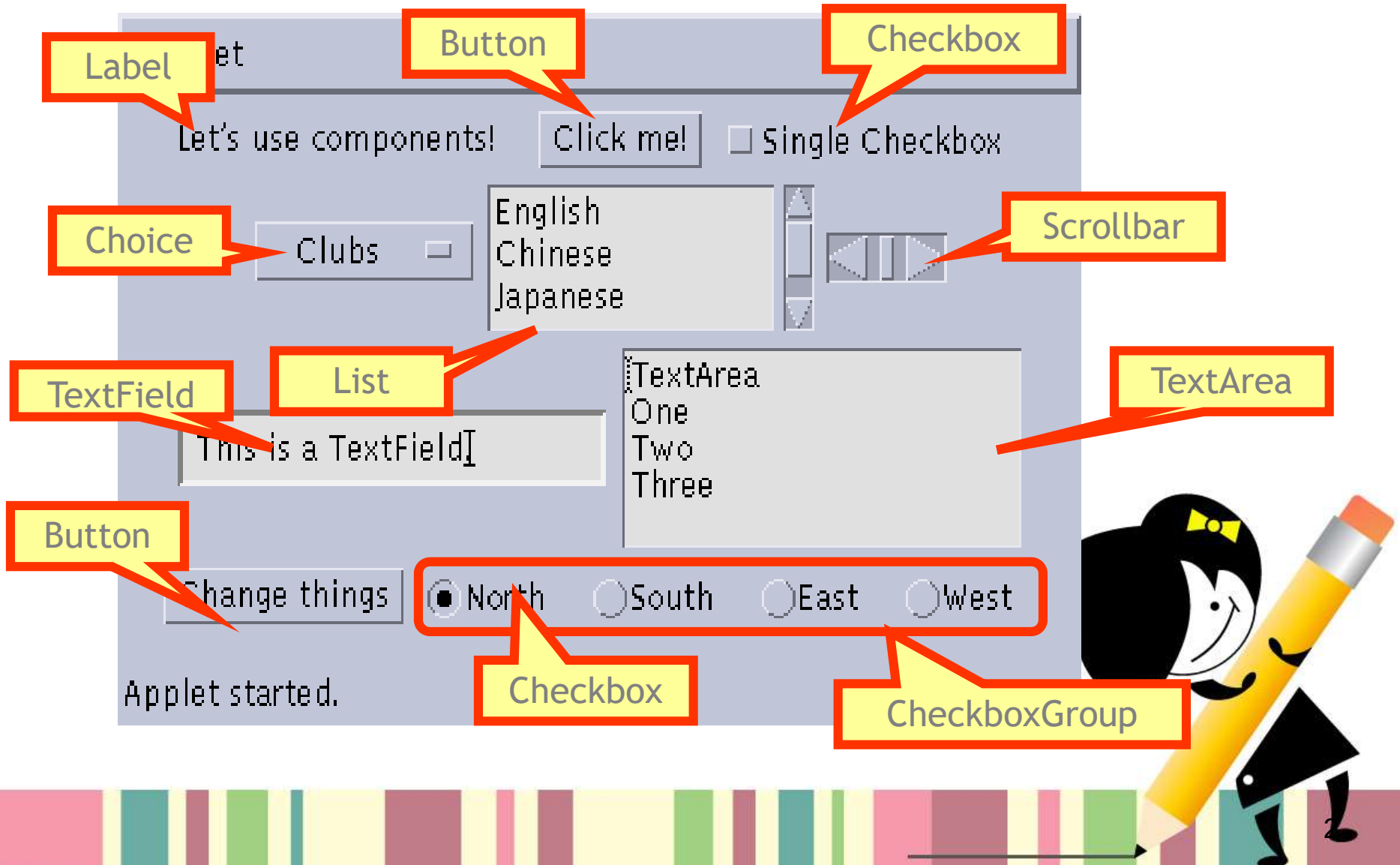
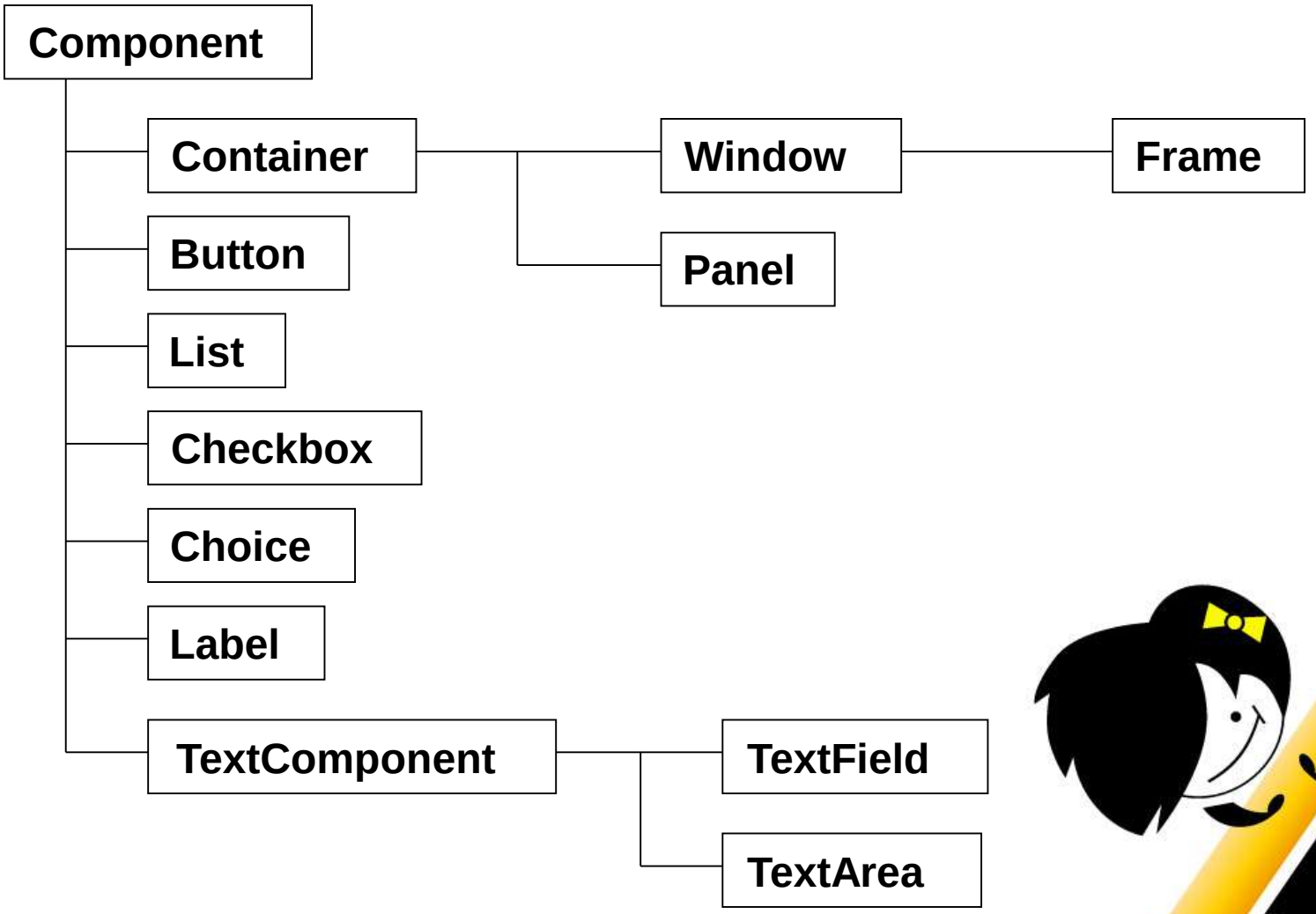
AWT Components
&
Event Handling

Some types of components



AWT -Class Hierarchy



Container

- The Container is a component in AWT that can contain another components like buttons, textfields, labels etc.
- The classes that extends Container class are known as container such as **Frame, Dialog, Panel and Applet.**

Creating components

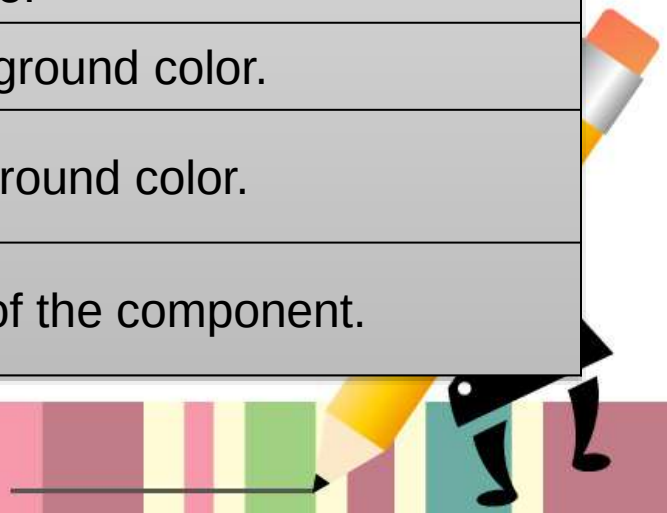
- Label lab = new Label ("Hi, I m Label");
- Button but = new Button ("Click me!");
- Checkbox toggle = new Checkbox ("toggle");
- TextField txt = new TextField ("Initial text.", 20);

Adding components to the Applet

```
class MyApplet extends Applet
 {
    public void init ()
    {     ….
        add (lab); // same as this.add(lab);
        add (but);
        add (toggle);
        add (txt);
          ...
     }
     …
}
```

Common methods of Component class

Method	Description
void add(Component c)	inserts a component on this component.
void setSize(int width,int height)	sets the size (width and height) of the component.
Dimension getSize()	Returns the current size of this component.
void setLayout(LayoutManager m)	defines the layout manager for the component.
void setVisible(boolean status)	changes the visibility of the component, by default false.
void setBackground(Color c)	Sets the background color.
void setForeground(Color c)	Sets the foreground color.
void setFont(Font f)	Sets the font of the component.



Label class

- This class is a Component which displays a single line of text.
- Labels are read-only. That is, the user cannot click on a label to edit the text it displays.
- Text can be aligned within the label
- Constructors are:
  - Label()
  - Label(String label)
  - Label(String label, int align) Label.LEFT,Label.RIGHT, Label.CENTER (Default align is left)

Ex: Label welcome = new Label("Hi");
this.add(welcome);

- Methods:
  - String getLabel()
  - void setLabel()

Button class

- This class represents a push-button with label.
- Constructors:
  - Button( )
  - Button(String caption)
- For ex:
  - Button okButton = new Button("Ok");
  - Button cancelButton = new Button("Cancel");
  - this.add(okButton));
  - this.add(cancelButton));
- **Example**

Checkbox class

- This class represents a GUI checkbox with a label.
- The Checkbox maintains a boolean state indicating whether it is checked or not.
- If a Checkbox is added to a **CheckBoxGroup**, it will behave like a **radio button**.
- Constructors are:
  - Checkbox()
  - Checkbox(String label)
  - Checkbox(String label,boolean on)
  - Checkbox(String label,boolean on, CheckboxGroup gr)
  - Checkbox(String label, CheckboxGroup gr ,boolea
- Methods:
  - String getLabel() , void setLabel()
  - boolean getState(), void setState(boolean state)

Ex: Checkbox cbOk = new Checkbox("OK",true)
this.add(cbOk)

Choice Class

- This class represents a dropdown list of Strings.
- Only one item from the list can be selected at one time and the currently selected element is displayed.
- Constructors: Choice()
- Methods:
  - void add(String item)
  - void insert(String item, int index)
  - void remove(String item)
  - void remove(int position)
  - void removeAll()
  - String getSelectedItem()
  - int getSelectedIndex()

Choice class…

- For ex:

```
Choice aChoice = new Choice();
aChoice.add(“Java");
aChoice.add(“Oracle");
aChoice.add(“OS");
aChoice.add(“CG");
this.add(aChoice);
```

List class

- This class is a Component which displays a list of Strings.
- The list is scrollable, if necessary.
- Sometimes called Listbox in other languages.
- Lists can be set up to allow single or multiple selections.
- The list will return an array indicating which Strings are selected
- Constructors:
  - List()
  - List(int num)
  - List(int num, boolean multi)
- Methods: String[] getSelectedItems(),
  int[] getSelected Indexes()
  - **Others are Similar as Choice class**

List class…

- For ex:

  List aList = new List(4,true);

  aList.add(“JAVA");

  aList.add(“Oracle");

  aList.add(“CG");

  aList.add(“OS");

  this.add(aList);

Example:

TextField class

- This class displays a single line of optionally editable text.
- This is one of the most commonly used Components in the AWT
- Constructors are:
  - TextField()
  - TextField(String str)
  - TextField(int numofchars)
  - TextField(String str, int numofchars)

Ex:TextField username= new TextField(30);
this.add(username);

- Methods: getText, setText,
- getSelectedText, isEditable, setEditable, setEchoChar,

  getEchochar, echoCharIsSet

TextArea class

- This class displays multiple lines of optionally editable text.
- TextArea has same methods as TextField class and also provides the methods:
  - append(), insert()
- // 5 rows, 80 columns
- TextArea address = new TextArea(5, 80);
- this.add(address);
- Example

Scrollbar

Scrollbar class is used to add horizontal and vertical scrollbar.

Constructors :

Scrollbar	Constructs a new vertical scroll bar.
Scrollbar(int orientation)	Constructs a new scroll bar with the specified orientation.
Scrollbar(int orientation, int value, int visible, int minimum, int maximum)	Constructs a new scroll bar with the specified orientation, initial value, visible amount, and minimum and maximum values.

Frame

- The Window class defines a top-level Window with no Borders or Menu bar.
- Frame defines a top-level Window with Borders and a Menu Bar
- Once defined, a Frame is a Container which can contain Components
  - Frame aFrame = new Frame("Hello World");
  - aFrame.setSize(100,100);
  - aFrame.setLocation(10,10);
  - aFrame.setVisible(true);
- Example with Application
- Example with Applet

Panel

- When writing a GUI application, the GUI portion can become quite complex.
- To manage the complexity, GUIs are broken down into groups of components. Each group generally provides a unit of functionality.
- A Panel is a rectangular Container whose sole purpose is to hold and manage components within a GUI.

Example

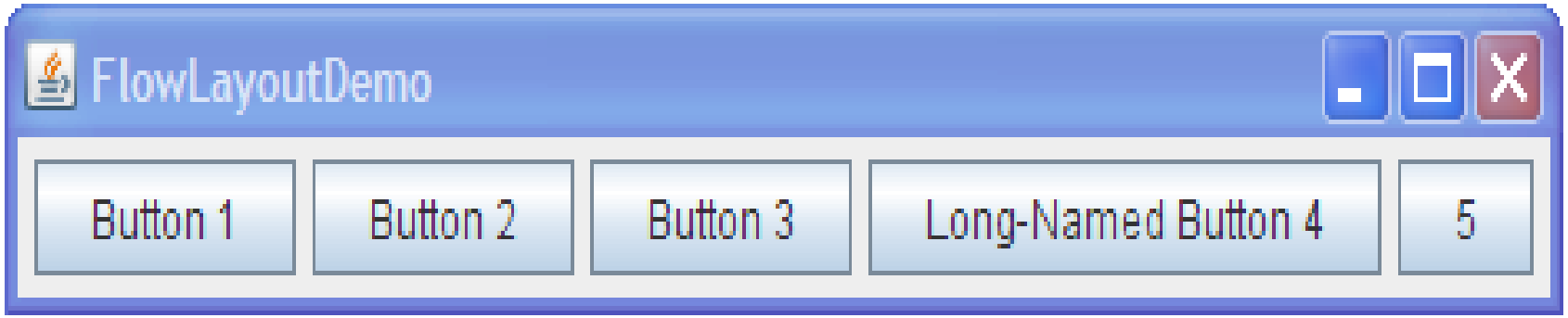
- Panel aPanel = new Panel();
- aPanel.add(new Button("Ok"));
- aPanel.add(new Button("Cancel"));
- this.add(aPanel);

What is Layout????

- A layout manager determines how Components will be arranged when they are added to a container.
- Default Layout of the Container is **Flowlayout.**
- Java includes Bunch of General-purpose layout manager :-
  - FlowLayout
  - BorderLayout
  - GridLayout
  - GridBagLayout
  - CardLayout
  - GroupLayout
  - SpringLayout
  - BoxLayout

FlowLayout

- The FlowLayout Class in the java.awt Package.
- This class lets component flow from left to right in the order that they are added to container.
- By default, the component on each row will be centered when you use the FlowLayout( ) Constructor no argument.
- Parameters of the Constructor will be:-
  - FlowLayout.LEFT
  - FlowLayouut.RIGHT
  - FlowLayout.CENTER
- Ex. FlowLayout fl=new FlowLayout (FlowLayout.Right);


FlowLayoutDemo
Button 1
Button 2
Button 3
Long-Named Button 4
5

BorderLayout:-

- The BorderLayout Class in the java.awt Package.
- It Divides the container into Five sections: north, south, east, west and center.
- A BorderLayout is created with either the
  - BorderLayout( ) {without gap}
  - BorderLayout(int , int) {with horizontal and vertical gap}
- After creating a border layout and set it up as a container's layout manager, components are added using a call to the add( ) method.
- Syntax:- add(component, String)

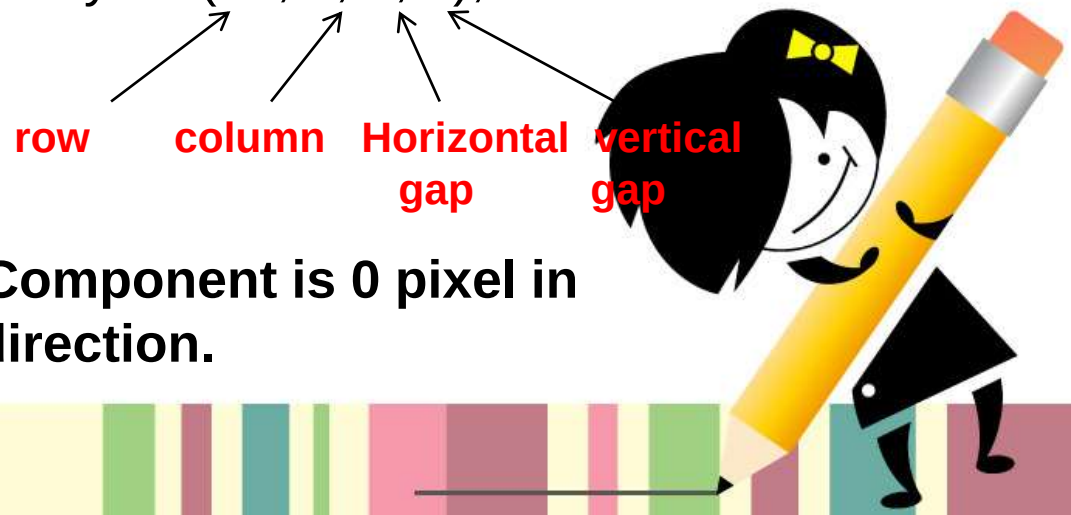
Ex. JPanel, Jbutton etc.

BorderLayout.NORTH

BorderLayout….

GridLayout

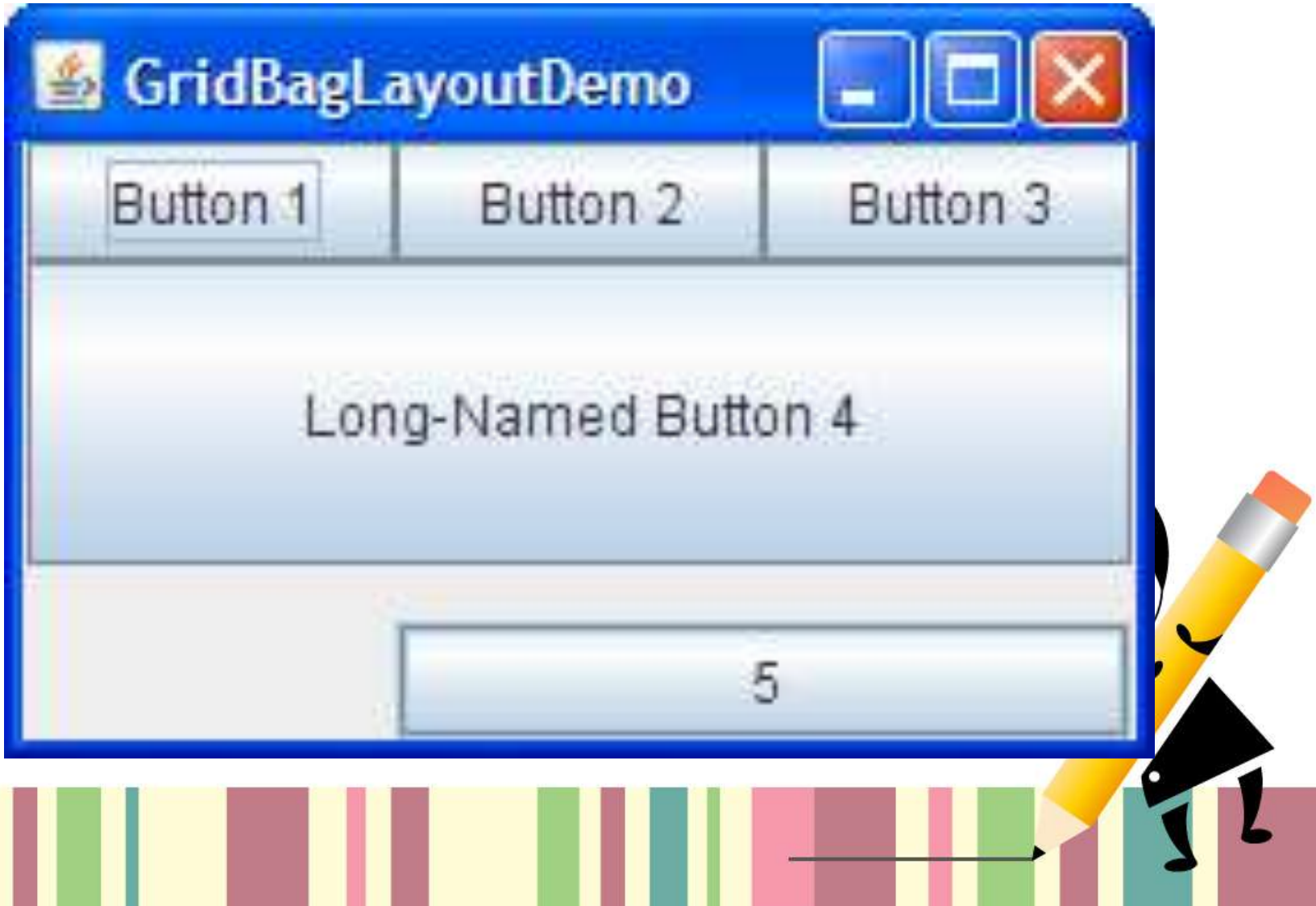
- The GridLayout Class in the java.awt Package.
- This layout manager can be used to stack component from top to bottom or left to right.
- The GridLayout manager must be created with two arguments to its constructor:
  - The number of rows in grid and
  - The number of columns
- Ex. GridLayout gr1=new GridLayout (10, 3);
- Ex. GridLayout gr2=new GridLayout (10, 3, 5, 8);



Note:- Default gap between Component is 0 pixel in both vertical and horizontal direction.


GridLayoutDemo
Button 1
Button 2
Button 3
Long-Named Button 4
5
Horizontal gap:
Vertical gap:
0
0
Apply gaps

GridBagLayout

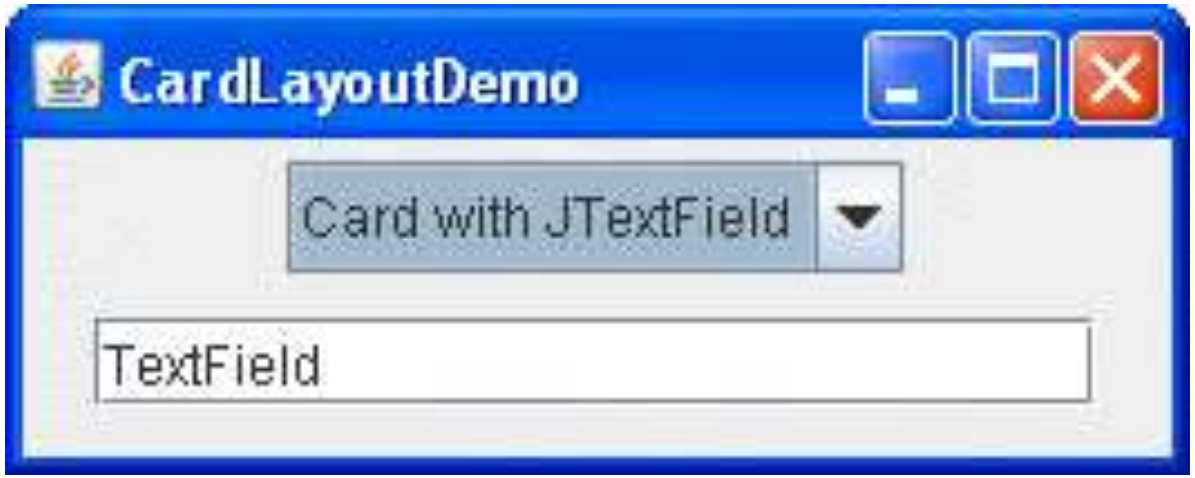


CardLayout

- The CardLayout Class in the javax.swing Package.
- This layout differ from the other Layout because they hide some component from the view.
- The most common method to use card layout is to use a panel for each card. Components are added to the panel first, and then the panel are added to the container that is set to use card layout.
- setLayout(new CardLayout());
- After set a container to use the card layout manager ,we use add() method call to add cards to the layo
- Syntax:- add(component, String)

Ex. JPanel, Jbutton etc.

Name of the CARD (this can be any thing)


CardLayoutDemo
Card with JTextField
TextField


CardLayoutDemo
Card with JButtons
Button 1
Button 2
Button 3

BoxLayout:-

- The BoxLayout Class in the javax.swing Package.
- This layout manager can be used to stack component from top to bottom or left to right.
- The BoxLayout manager must be created with two arguments to its constructor:
- Container it will manage.
- Class variables
  - X_AXIS (for left-to-right)
  - Y_AXIS (for top-to-bottom)

Ex:BoxLayout bl=new BoxLayout (new JPanel, BoxLayout.X_AXIS);


BoxLayoutDemo
Button 1
Button 2
Button 3
Long-Named Button 4
5

Example of Border Layout

Example of Grid Layout

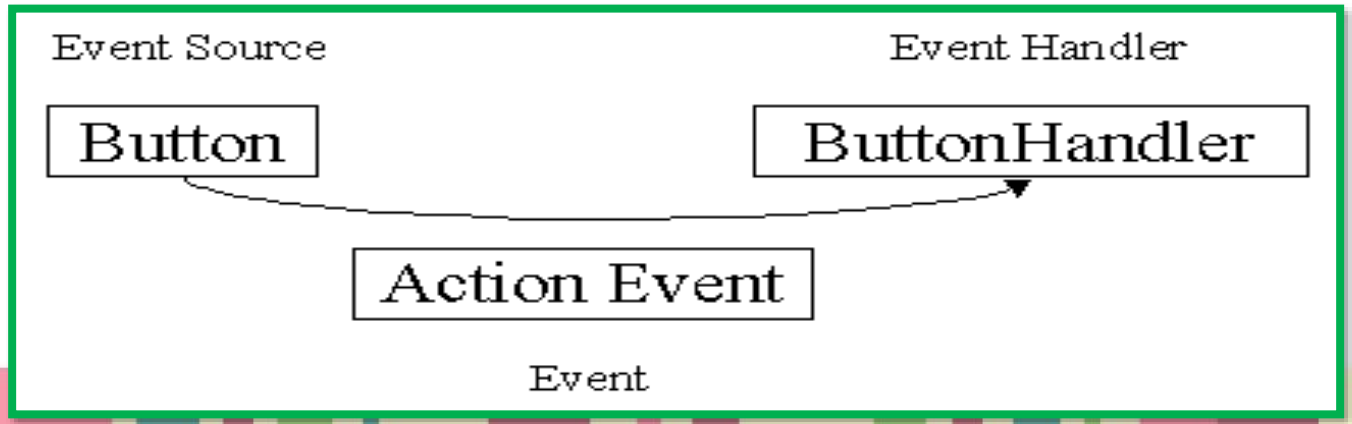
EVENT HANDLING

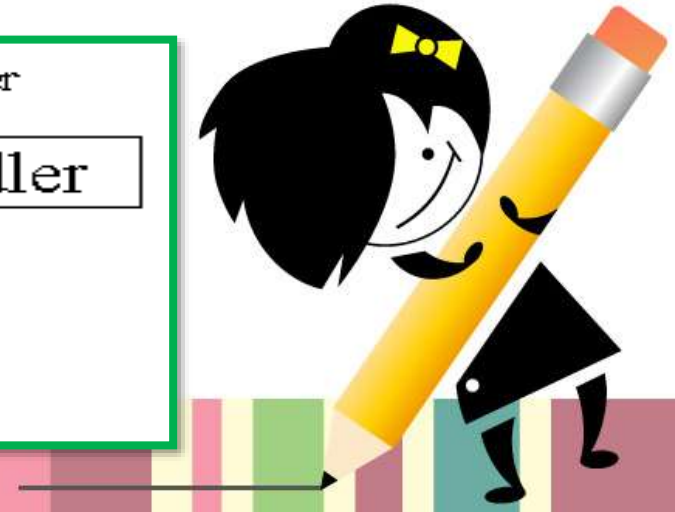
What is Event?

- GUI components like button, List, Checkbox etc. communicate with the rest of the applications through events.
- The source of an event is the component that causes that event to occur.
- The listener of an event is an object that receives the event and processes it appropriately.

Event Delegation Model

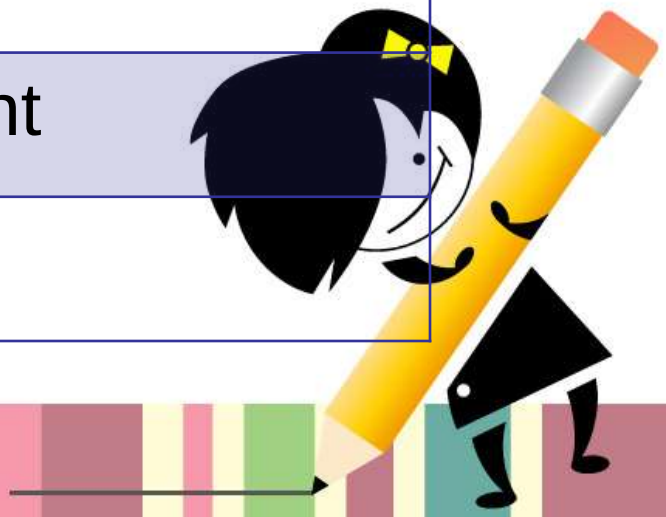
- Event
  - Actions like clicking button with a mouse, pushing down a key on keyboard
- Event Source
  - Sources that cause events such as a button, a key
- Event Handler
  - Methods that get an occurred event, then handle proper actions





Event classes

EventObjcet	FocusEvent
AWTEvent	ItemEvent
ActionEvent	KeyEvent
AdjustmentEvent	MouseEvent
ComponentEvent	MouseWheelEvent
ContainerEvent	TextEvent
WindowEvent	



AWTEvent class

- Subclass of EventObjcet class and member of java.awt package
- Super class of all AWT events handled by event delegation model
- Methods:
  - int getID(): returns type of event in integer, for ex 1001 for button
  - String toString()

Event Class	Description	Listener Interface
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released also when the enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener

Event	Description	Listener
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEvent Listener
ContainerEvent	generated when component is added or removed from container	ContainerListener
AdjustmentEvent	generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

ActionEvent & ActionListener

- Action event is probably the easiest and most common event handlers to implement.
- Constructors of ActionEvent:
  - ActionEvent(Object Source, int type, String command)
  - ActionEvent(Object Source, int type, String command, int modifiers)
- Methods of ActionEvent: Example
  - String getActionCommand()
  - int getModifiers()
- Method of ActionListener
  - void actionPerformed(ActionEvent obj)

ItemEvent & ItemListener

- Generated when a checkbox or list item is clicked
- Constructor:
  - ItemEvent(Item source, int type, int state)
- Constants:
  - SELECTED
  - DESELECTED
  - ITEM_STATE_CHANGED
- This class implements method:
  - void itemStateChanged(ItemEvent obj)

Example

MouseEvent & MouseListener

- Constants:
  - MOUSE_CLICKED
  - MOUSE_PRESSED
  - MOUSE_RELEASED
  - MOUSE_DRAGGED
  - MOUSE_ENTERED
  - MOUSE_EXITED
  - MOUSE_MOVED
- Methods:
  - getX( ),getY( ), getClickCount( )

Example

MouseListener & MouseMotionListener

- Methods **of MouseListener**
  - void mouseClicked(MouseEvent obj)
  - void mousePressed(MouseEvent obj)
  - void mouseReleased(MouseEvent obj)
  - void mouseEntered(MouseEvent obj)
  - void mouseExited(MouseEvent obj)

- Methods **of MouseMotionListene**
  - void mouseMoved(MouseEvent obj)
  - void mouseDragged(MouseEvent obj)

KeyEvent & KeyListener

- Event is generated when key is pressed of the keyboard
- Constant:
  - KEY_PRESSED
  - KEY_RELEASED
  - KEY_TYPED

Example

- Methods of KeyListener:
  - void keyPressed(KeyEvent obj)
  - void keyReleased(KeyEvent obj)
  - void keyTyped(KeyEvent obj)

FocusEvent & FocusListener

- Generated component gets or lost focus.
- Constants:
  - FOCUS_GAINED
  - FOCUS_LOST

- Listener methods:
  - void focusGained(FocusEvent e)
  - void focusLost(FocusEvent e)

Example

TextEvent & TextListener

- Generated when user inputs text into TextField or TextArea
- Constants:
  - TEXT_VALUE_CHANGED

- Listener method:
  - void textValueChanged(TextEvent e)

Example

WindowEvent & WindowListener

- Handles events related to window
- Constants:
  - WINDOW_ACTIVATED
  - WINDOW_DEACTIVATED
  - WINDOW_CLOSING
  - WINDOW_CLOSED
  - WINDOW_OPENED
  - WINDOW_GAINED_FOCUS
  - WINDOW_LOST_FOCUS
  - WINDOW_ICONIFIED (Minimize)
  - WINDOW_DEICONIFIED
  - WINDOW_STATE_CHANGED

Example

- **Methods of listener is according to above constants**