Unit 4

Contents

- Event Handling
- Delegation Event Model
- Event Classes
- Event Sources
- Event Listeners
- AWT: Frame Class
- AWT Controls

Contents

- Swing – Architecture
- Components of swing
  - JLabel
  - Jbutton
  - JCheckBox
  - JRadioButton
  - JList
  - JComboBox
  - JTextField
  - JTextArea
  - JPanel
  - JFrame

Contents

- Layout Managers
  - Flow Layout
  - Grid Layout
  - Card Layout
  - Border Layout
  - Box Layout
  - Null Layout

Event Handling

- Event is an object that describes a state change in a source.
- Generated as a consequence of a person interacting with the elements in a GUI.
- Example – Entering a character via keyboard, selecting an item from a list, clicking the mouse etc.

Event Handling

- Events may also occur indirectly like

1. When a timer expires
2. When a counter exceeds a value
3. A software or hardware failure occurs
4. When an operation is completed.

Delegation Event Model

- Defines standard and consistent mechanisms to generate and process events.
- Source generates an event and sends to one or more listeners.
- Listener simply waits until it receives an event.
- Once received, the listener processes the event and then returns.

Delegation Event Model

- Listeners must register with a source in order to receive an event notification.
- The benefit is that notifications are sent only to listeners that want to receive them.

Event Sources

- A source is an object that generates an event.
- Occurs when the internal state of that object changes in some way.
- A source must register listeners in order for the listeners to receive notifications about a specific type of event.

Event Sources

- When an event occurs, all registered listeners are notified and receive a copy of the event object.- Multicasting
- Some sources allow only one listener to register. When an event occurs, the registered listener is notified. - Unicasting

Event Listeners

- A listener is an object that is notified when an event occurs. It has 2 requirements.

1. It must have been registered with one or more sources to receive notifications about specific type of events.
2. It must implement methods to receive and process these notifications.

- Methods that receive and process events are defined in a set of interfaces in java.awt.event

Event Class

1. ActionEvent - Generated when a button is pressed, a list item is double-clicked or a menu item is selected.
2. AdjustmentEvent – Generated when a scroll bar is manipulated.
3. ComponentEvent – Generated when a component is hidden, moved, resized or becomes visible.
4. ContainerEvent – Generated when a component is added or removed.

Event Class

5. FocusEvent – Generated when a component gains or loses keyboard focus.
6. InputEvent – Abstract super class for all component input event classes.
7. ItemEvent - Generated when a checkbox or list item is clicked. Also occurs when a choice selection is made or a checkable menu item is selected or deselected.

Event Class

8. KeyEvent – Generated when input is received from the keyboard.
9. MouseEvent – Generated when the mouse is dragged, moved, clicked, pressed or released. Also generated when the mouse enters or exits a component.

Event Class

10. TextEvent – Generated when the value of a text area or text field is changed.
11. WindowEvent – Generated when a window is activated, closed, deactivated, deiconified, iconified, opened or quit.

Event Listener Interface

1. ActionListener – Defines one method to receive action events.

   void actionPerformed(ActionEvent ae)

2. AdjustmentListener – Defines one method to receive adjustment events.

void adjustmentValueChanged(AdjustmentEvent ae)

Event Listener Interface

3. ComponentListener – Defines 4 methods to recognize when a component is hidden, moved, resized or shown.

void componentResized(ComponentEvent ce)

void componentMoved(ComponentEvent ce)

void componentShown(ComponentEvent ce)

void componentHidden(ComponentEvent ce)

Event Listener Interface

4. ContainerListener – Defines 2 methods to recognize when a component is added to or removed from a container.

void componentAdded(ContainerEvent ce)

void componentRemoved(ContainerEvent ce)

Event Listener Interface

5. FocusListener – Defines 2 methods to recognize when a component loses or gains keyboard focus.

void focusGained(FocusEvent fe)

void focusLost(FocusEvent fe)

Event Listener Interface

6. ItemListener – Defines 1 method to recognize when the state of an item changes.

void itemStateChanged(ItemEvent ie)

Event Listener Interface

7. KeyListener – Defines 3 methods to recognize when a key is pressed, released or typed.

void keyPressed(KeyEvent ke)

void keyReleased(KeyEvent ke)

void keyTyped(KeyEvent ke)

Event Listener Interface

8. MouseListener – Defines 5 methods to recognize when a mouse is clicked, enters a component, exits a component, is pressed or released.

   void mouseClicked(MouseEvent me)

   void mouseEntered(MouseEvent me)

   void mouseExited(MouseEvent me)

   void mousePressed(MouseEvent me)

   void mouseReleased(MouseEvent me)

Event Listener Interface

9. MouseMotionListener – Defines 2 methods to recognize when a mouse is dragged or moved.

   void mouseDragged(MouseEvent me)

   void mouseMoved(MouseEvent me)

Event Listener Interface

10. TextListener – Defines 1 method to recognize when a text value changes.

    void textChanged(TextEvent te)

Event Listener Interface

11. WindowListener – Defines 7 methods to recognize when a window is activated, closed, deactivated, iconified, deiconified, opened or quit.

    void windowActivated(WindowEvent we)

    void windowClosed(WindowEvent we)

    void windowClosing(WindowEvent we)

    void windowDeactivated(WindowEvent we)

    void windowIconified(WindowEvent we)

    void windowDeiconified(WindowEvent we)

    void windowOpened(WindowEvent we)

Example Program

- Handling Mouse Events
- Handling Keyboard Events

AWT: Frame Class

- The Frame class is derived from the base class Window.
- Constructors

1. Frame() - Creates a standard window that does not contain a title.
2. Frame(String title) – Creates a standard window with the title specified by *title*.

AWT: Frame Class

- We cannot specify the dimensions of the window. Instead we must set the size of the window after it has been created.
- setSize() – this method is used to set the dimensions of the window.

void setSize(int newWidth,int newHeight)

Example

```
Frame f=new Frame()
f.setSize(400,300);
```

AWT: Frame Class

- Hiding and Showing a Window – After a frame window has been created, it will not be visible until we call setVisible().

  void setVisible(boolean visibleFlag)

Example

```
Frame f=new Frame();
f.setVisible(true);
```

The component is visible, if the argument is True. Otherwise it is hidden.

AWT: Frame Class

- Setting a Window's Title – We can change the title in a frame window using setTitle().

  void setTitle(String newTitle)

Example

```
Frame f= new Frame();
f.setTitle("Java Window");
```

AWT Controls

1. Label
2. Button
3. Checkbox
4. Choice
5. List
6. TextField
7. TextArea

1. AWT Controls - Labels

- Labels are passive controls that do not support any interaction with the user.

Constructors

1. Label()
2. Label(String str)
3. Label(String str, int how)

The value of how is one of the three constants: Label.LEFT, Label.RIGHT or Label.CENTER

Example Program

2. AWT Controls - Button

- Most widely used control.
- Generates an event when it is pressed.
- Push buttons are objects of type Button

Constructors

1. Button()
2. Button(String str)

After a button is created, we can set the label by calling setLabel() and retrieve its label by calling getLabel()

2. AWT Controls - Button

void setLabel(String str)

String getLabel()

Example

```
Button b1  =new Button();
 b1.setLabel("OK");
 String lab=b1.getLabel();
```

2. AWT Controls - Button

- Each time a button is pressed, an ActionEvent is generated.
- ActionListener interface defines the actionPerformed() method.
- The label of the button that is pressed is obtained by calling getActionCommand().

Example Program

3. AWT Controls - Checkbox

- A control that is used to turn an option on or off.
- Consists of a small box that can either contain a check mark or not.
- There is a label associated with each Checkbox.
- They can be used individually or as a part of a group.

3. AWT Controls - Checkbox

Constructors

1. Checkbox()
2. Checkbox(String str)
3. Checkbox(String str, boolean on)
4. Checkbox(String str, boolean on, CheckboxGroup cbgroup)
5. Checkbox(String str, CheckboxGroup cbgroup, boolean on)

3. AWT Controls - Checkbox

Methods

1. boolean getState() – retrieve the current state of a Checkbox.

   boolean b=c1.getState()

1. void setState(boolean on) – to set the state of a Checkbox.

   c1.setState(false)

1. String getLabel() – obtains the current label associated with a Checkbox.

   String s=c1.getLabel()

1. void setLabel(String str) - set the new label with the string passed.

   c1.setLabel("Tennis")

3. AWT Controls - Checkbox

Handling Checkboxes

- Each time, a check box is selected or deselected, an ItemEvent is generated.
- This is send to ItemListener interface which implements itemStateChanged() method.

Example Program

3. AWT Controls - Checkbox

Methods Example

```
Checkbox cb = new Checkbox("Course", true);
 boolean b= cb.getState();
 cb.setState(false);
 String s = cb.getLabel();
  cb.setLabel("Subject");
```

4. AWT Controls - CheckboxGroup

- Mutually exclusive check boxes in which one and only one checkbox in the group can be checked at any one time.
- Hence they are called “Radio Buttons”.

Methods

1. Checkbox getSelectedCheckbox()
2. void setSelectedCheckbox(Checkbox which)

Example Program

5. AWT Controls – Choice Control

- Used to create a pop-up list of items.
- When the user clicks on it, the whole list of choices pops up.
- Only one default constructor.

5. AWT Controls – Choice Control

Methods

1. void add(String name) – adds an item to the choice control.
2. String getSelectedItem() – determines which item is currently selected.
3. int getSelectedIndex() – returns the index of the item.
4. int getItemCount() – obtains the number of items in the list.

5. AWT Controls – Choice Control

Handling Choice Lists

- Each time a choice is selected, an ItemEvent is generated.
- The ItemListener interface defines the itemStateChanged() method.

Example Program

6. AWT Controls – List

- Provides a compact, multiple-choice, scrolling selection list.
- Allows multiple selection.

Constructors

1. List() – creates a List control that allows only one item to be selected at any one time.
2. List(int numRows) – numRows specifies the number of entries in the list that will always be visible.
3. List(int numRows, boolean multipleSelect) - if multipleSelect is True, then user can select two or more items.

6. AWT Controls – List

Methods

1. void add(String name) - name is the item added to the list.
2. void add(String name, int index) – adds the item to the specified index.
3. String getSelectedItem() – returns a string containing the name of the item.
4. int getSelectedIndex() – returns the index of the item.

6. AWT Controls – List

Methods

5. String[] getSelectedItems() – returns an array containing the names of the currently selected items.
6. int[] getSelectedIndexes() – returns an array containing the indexes of the currently selected item.
7. int getItemCount() – obtain the number of items in a list.
8. String getItem (int index) – index specifies the index of the desired item.

6. AWT Controls – List

Handling Lists

- To process the events in a List, we need to implement the ActionListener interface.

7. AWT Controls – TextField

- Single line text entry area.

Constructors

1. TextField()
2. TextField(int numChars) – creates a TextField numChars wide
3. TextField(String str) – creates a TextField and initializes that is str characters wide.
4. TextField(String str, int numChars) - creates a TextField numChars wide and initializes that is str characters wide.

7. AWT Controls – TextField

Handling TextField

1. String getText() – To obtain the string currently contained in the TextField.
2. void setText(String str)– To set the text to a TextField.
3. String getSelectedText() – returns the selected Text.
4. void setEchoChar(char ch) – disable the echoing of characters.
5. char getEchoChar() – ch specifies the character to be echoed.

Example Program

8. AWT Controls – TextArea

Handling TextArea

- Used for handling multiline texts

Constructors

1. TextArea()
2. TextArea(int numLines, int numChars) – numLines specifies the height, in lines of the TextArea and numChars specifies the width in characters.
3. TextArea(String str)

8. AWT Controls – TextArea

Constructors

4. TextArea(String str, int numLines, int numChars)
5. TextArea(String str, int numLines, int numChars, int sBars) – sBars must be one of these values.

   SCROLLBARS_BOTH(4),

   SCROLLBARS_NONE(3),
   SCROLLBARS_HORIZONTAL_ONLY(2),

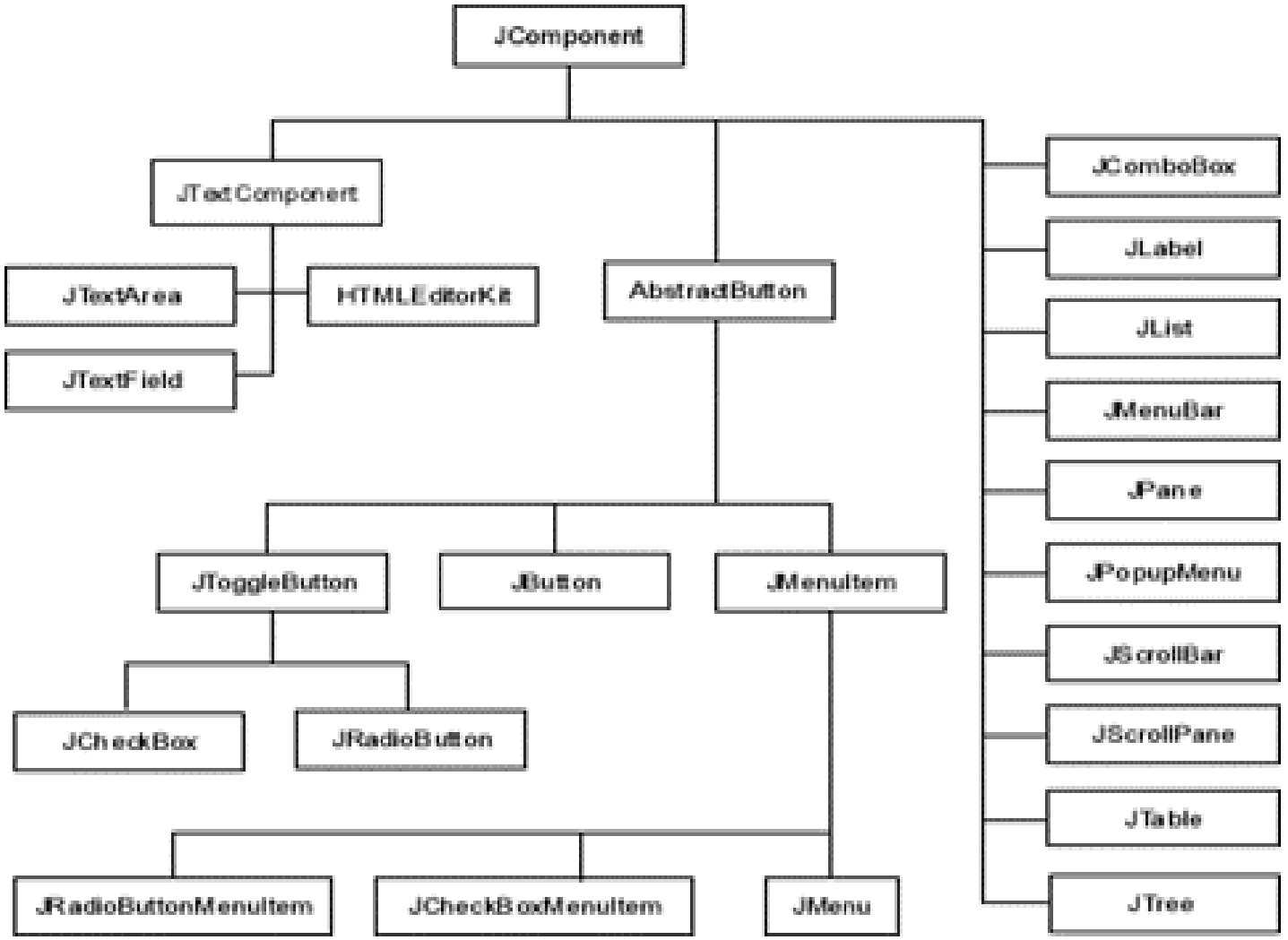
   SCROLLBARS_VERTICAL_ONLY (1)

Example Program

SWINGS

- Swing is a set of classes that provides more powerful and flexible components such as buttons, checkboxes, labels etc.
- For example a button may have both an image and a text string associated with it.

SWINGS - Architecture



Components of SWING - JLabel

- Swing labels are instances of the JLabel class. It can display text and icon.
- Constructors for JLabel

✓ JLabel(Icon i)

✓ JLabel(String s)

✓ JLabel(String s, Icon i, int align)

- where s is the text and i is the icon for the label. The align argument is either LEFT, RIGHT, CENTER, LEADING or TRAILING.

Components of SWING - JTextField

- Swing version of TextField.
- Constructors for JTextField

✓JTextField()

✓JTextField(int cols)

✓JTextField(String s, int cols)

✓JTextField(String s)

- where s is the string passed to the text field and cols is the number of cols in the text field.

Components of SWING - JTextArea

- Swing version of TextArea.
- Constructors for JTextArea

✓ JTextArea()-Constructs a new TextArea.

✓ JTextArea(int rows, int columns)-Constructs a new empty TextArea with the specified number of rows and columns.

✓ JTextArea(String text)-Constructs a new TextArea with the specified text displayed.

✓ JTextArea(String text, int rows, int columns)-Constructs a new TextArea with the specified text and number of rows and columns.

Components of SWING - JButton

- Swing version of Button. More powerful than buttons in AWT.
- For example, we can place an icon on the swing button.
- Constructors for JButton

✓ JButton(Icon i)

✓ JButton(String s)

✓ JButton(String s, Icon i)

- where s and i are the string and icon used for the button.
- On clicking the button, ActionEvent will be generated which will be handled by ActionListener interface.

Components of SWING - JCheckBox

- Swing version of Checkbox.
- It has two states either checked or unchecked.
- Constructors for JCheckBox

✓ JCheckBox(Icon i)
✓ JCheckBox(Icon I, boolean state)
✓ JCheckBox(String S)
✓ JCheckBox(String s, boolean state)
✓ JCheckbox(String s, Icon i)
✓ JCheckbox(String s, Icon i, boolean state)

- where s and i are the string and icon used for the button.
- When a check box is selected or deselected, an ItemEvent is generated. This is handled by itemStateChanged() method of ItemListener. The getItem() method gets the JCheckBox object that generated the event.

Components of SWING - JRadioButton

- Swing version of RadioButton.
- The button has only two states
- Constructors for JRadioButton

✓ JRadioButton(Icon i)
✓ JRadioButton(Icon i, boolean state)
✓ JRadioButton(String s)
✓ JRadioButton(String s, boolean state)
✓ JRadioButton(String s, Icon i)
✓ JRadioButton(String s, Icon i, boolean state)

- where s and i are the string and icon used for the button.
- When a radio button is pressed, it generates ActionEvent that is handled by actionPerformed() method of ActionListener interface. The getActionCommand() method gets the text that is associated with a radio button.

Components of SWING - JComboBox

- Swing version of Combo box.
- A combo box normally displays one entry.
- It also displays a drop-down list that allows a user to select a different item.
- Constructors for JComboBox

✓ JComboBox()

✓ JComboBox(Vector v)

- Here v is a vector that initializes the combo box. Items are added to the combo box using addItem() method. The syntax of addItem() is as follows.

➢ void addItem(Object obj)

- where obj is the object added to the combo box.

Components of SWING - JList

- Swing version of List.
- Displays a list of objects and allows the user to select one or more items.
- Constructors for JList

✓ JList()-Constructs a JList with an empty, read-only, model.

✓ JList(Object[] listData)- Constructs a JList that displays the elements in the specified array.

- JList generates a ListSelectionEvent when the user makes or changes a selection.
- Also generated when the user deselects an item.
- It is handled by implementing ListSelectionListener. This listener specifies only one method, called valueChanged( ), which is shown below.
- void valueChanged(ListSelectionEvent le)

Components of SWING - JList

- JList allows the user to select multiple ranges of items within the list.
- We can change this behavior by calling setSelectionMode( ) which is defined by JList. The syntax is shown below.
- void setSelectionMode(int mode)
- mode specifies the selection mode. It must be one of these values SINGLE_SELECTION, SINGLE_INTERVAL_SELECTION or MULTIPLE_INTERVAL_SELECTION.
- The default is the MULTIPLE_INTERVAL_SELECTION. It lets the user to select multiple ranges of items within a list.
- With SINGLE_INTERVAL_SELECTION, the user can select a range of items.
- With SINGLE_SELECTION, the user can select only a single item.

Components of SWING - JFrame

- Swing version of Frame.
- Top-level window with a title and a border.
- Constructors for JFrame

✓ JFrame()-Constructs a new frame that is initially invisible.

✓ JFrame(String title)-Creates a new, initially invisible Frame with the specified title.

Components of SWING - JPanel

- Swing version of Panel.
- A Container is a component that can contain other SWING components.
- A container provides a space where a component can be located
- For example, JPanel, JFrame and JWindow are Containers.
- JPanel is the simplest container. It provides space in which any other component can be placed, including other panels.
- Constructors for JPanel

✓ JPanel()-Creates a new JPanel with a flow layout.

✓ JPanel(LayoutManager layout)-Creates a JPanel with the specified layout manager.

Layout Managers

- A Layout Manager automatically arranges the controls within a window.

1. Flow Layout
2. Border Layout
3. Grid Layout
4. Card Layout
5. Box Layout
6. Null Layout

Layout Managers

1. FlowLayout

- Default layout manager
- Components are laid from upper left corner, left to right and top to bottom.

Constructors

1. FlowLayout() – creates a default layout which centers components and leaves five pixels of space between each component.
2. FlowLayout(int how) – Specify how each line is aligned. (FlowLayout.LEFT,
   FlowLayout.CENTER,
   FlowLayout.RIGHT)
3. FlowLayout(int how, int horz, int vert) - specify the horizontal and vertical space left between components in *horz* and *vert*

Example Program

Layout Managers

2. BorderLayout

- Implements a common layout style for top-level windows.
- Four sides are referred to as north, south, east and west.
- Middle area is called the center.

Constructors

1. BorderLayout() – creates a default layout.
2. BorderLayout(int horz, int vert) – Specify the horizontal and vertical space left between the components in horz and vert respectively.

Layout Managers

2. BorderLayout

- Defines the following constants that specify the regions.

1. BorderLayout.CENTER
2. BorderLayout.EAST
3. BorderLayout.NORTH
4. BorderLayout.SOUTH
5. BorderLayout.WEST

Example Program

Layout Managers

3. GridLayout

- Lays out components in a two-dimensional grid.

Constructors

1. GridLayout() – creates a single-column grid layout.
2. GridLayout(int numRows, int numColumns) – creates a grid layout with the specified number of rows and columns.
3. GridLayout(int numRows, int numColumns, int horz, int vert) – allows to specify the horizontal and vertical space left between components in horz and vert respectively.

Layout Managers

3. GridLayout

- numRows or numColumns can be zero.
- Specifying numRows as zero allows for unlimited-length columns.
- Specifying numColumns as zero allows for unlimited length rows.

Example Program

Layout Managers

4. CardLayout

- Unique among other layout managers in that it stores several different layouts.
- Each layout can be thought of as a separate index card in a deck that can be shuffled so that any card is on top at a given time.

Layout Managers

4. CardLayout

Constructors

1. CardLayout() – creates a default card layout
2. CardLayout(int horz, int vert)- specify the horizontal and vertical space between the components.

Layout Managers

4. CardLayout

- After creating a deck, the program activates a card by calling one of the following methods.

1. void first(Container deck) – first card will be shown.
2. void last(Container deck)- last card will be shown.
3. void next(Container deck) – next card will be shown.
4. void previous(Container deck) – previous card will be shown.
5. void show(Container deck, String cardName) – displays the card whose name is passed in the cardName.

Example Program

Layout Managers

5. BoxLayout

- The BoxLayout is used to arrange the components either vertically or horizontally.
- For this purpose, BoxLayout provides four constants. They are X_AXIS, Y_AXIS, LINE_AXIS and PAGE_AXIS. The following shows the constructor for the BoxLayout class.
- BoxLayout(Container c, int axis)- creates a box layout that arranges the components with the given axis.

Example Program

Layout Managers

6. NullLayout

- Rather than using a layout manager that controls the size and position of all components in a container, we can set the layout manager to null.
- Each component then controls its own position and size using its bounds.
- Creating a container without a layout manager involves the following steps.

    1. Set the container's layout manager to null by calling setLayout(null).
    2. Call the Component class's setBounds() method for each of the container's children.

Example Program

End of Unit 4