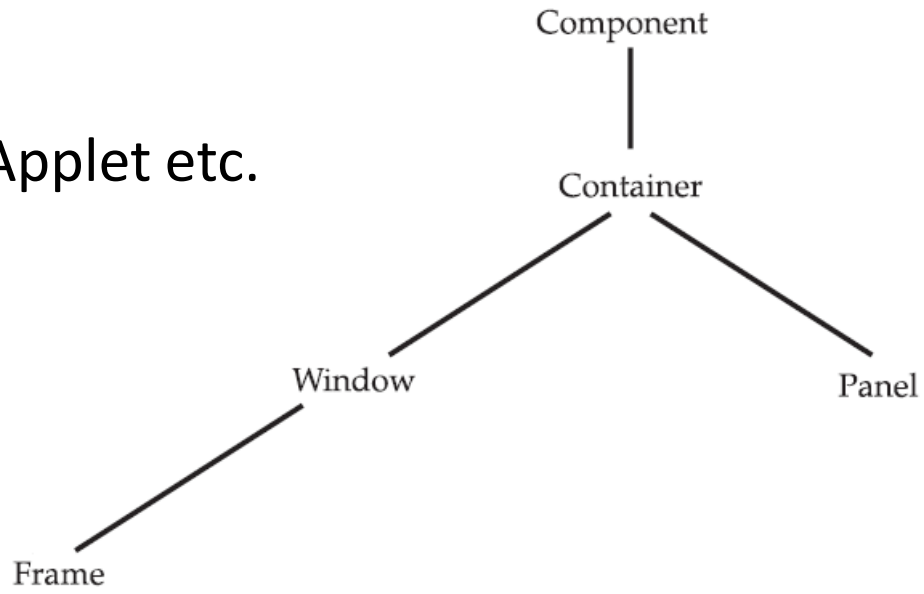
Java

Swings & Event Handling

Department of Data Science &
Engineering, DSCA, MIT

Swings

- Java **AWT** (Abstract Window Toolkit) contains numerous classes and methods for creating and managing windows
- **Java Swing** (***javax.swing***) is built on the AWT
- Swing Concepts
  - **Container**
    - Heavyweight: JFrame, JApplet etc.
    - Lightweight: JPanel
  - **Component**
    - Jbutton, JLabel etc.



A Simple Swing

- Two ways to create a simple swing application
- *By creating an object of JFrame class*
  - ***Example**: SimpleFrame1.java*
- *By extending the JFrame class*
  - ***Example**: SimpleFrame2.java*
    - HelloWorld.java (dialogue box)

JFrame with Simple Components

- Simple JFrame
  - with a JLabel
  - with a JButton
  - with Default Layout
  - ***Example**: LabelFrame1.java*
- Simple JFrame
  - with a JLabel
  - with a JButton
  - with FlowLayout
  - ***Example**: LabelFrame2.java*

Some Components

- **JLabel**
  - ***Example****: TestJLabelFrame.java*
- **JTextField** and **JPasswordField**
  - ***Example****: TestJTextFieldFrame.java*
- **JButton**
  - ***Example****: TestJButtonFrame.java*
    - UseOptionPanes.java

Some Components (contd..)

- **JCheckBox**
  - ***Example**: TestJCheckBoxFrame.java*
- **JRadioButton**
  - ***Example**: TestJRadioButtonFrame.java*
- **JComboBox**
  - ***Example**: TestJComboBoxFrame.java*

Event

- **Event** – state/behavior change
- Button – click/pressed
- Text Box 

- Mouse - Scroll

Delegation Event Model

DEM defines ***standard and consistent mechanisms to generate and process events.***

Concept :

- A **source generates** an event and sends it to one or more listeners.

- In this scheme, **the listener** simply waits until it receives an event.

- Once received, the **listener processes the event and then returns.**

Delegation Event Model(contd..)

- In the delegation event model, ***listeners must register with a source in order to receive an event notification***.
- Notifications are sent only to listeners that want to receive them.

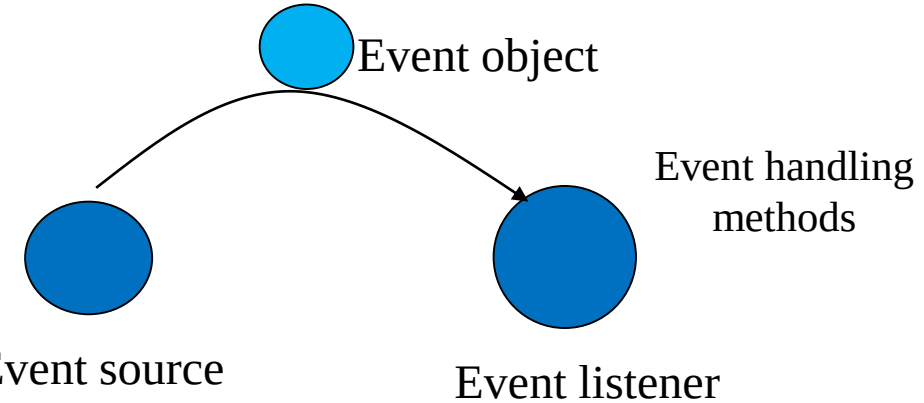
The general form:

public void add*Type*Listener(*Type*Listener *el)

Type **is the name of the event and** ***el*** **is a reference to the event listener.**

Ex.

source_object. addActionListener(Listener obj)



Event Handling

- Events are generated when user do some actions with the components (button click)
- Event handling are same for Swing and AWT(abstract Window Toolkit)
- The interface which is generally used for event handling - ***ActionListener***
- The class that implements the ActionListener interface must implement the following method

public void actionPerformed (ActionEvent ae)

Event Handling(contd..)

- The event name is ***ActionEvent***
  - getSource()
  - getActionCommand()
- Components registered to handle event by

  addActionListener (ActionListener al)
- ***Example****: EventFrame(1-3).java*

Keyboard Events

- In Swing we can also detect key and mouse events
- Interface for key event handling - ***KeyListener***
- The name of the functions are
  - keyTyped(KeyEvent ke)
  - keyPressed(KeyEvent ke)
  - keyReleased(KeyEvent ke)
- The event name is ***KeyEvent***
  - getKeyChar(), getKeyCode()
- ***Example****: TestKeyListener.java*

Mouse Events

- Interface for mouse event handling - ***MouseListener***
- The name of the functions are
  - mouseClicked(MouseEvent me)
  - mousePressed(MouseEvent me)
  - mouseReleased(MouseEvent me)
  - mouseEntered(MouseEvent me)
  - mouseExited(MouseEvent me)
- The event name is ***MouseEvent***
  - getX(), getY()
- ***Example****: TestMouseListener.java*

End of Chapter

Department of Data Science &
Engineering, DSCA, MIT