Object Oriented Programming
CS F213

J. Jennifer Ranjani

email: jennifer.ranjani@pilani.bits-pilani.ac.in

Chamber: 6121 P, NAB

Consultation: Appointment by e-mail

BITS Pilani

Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

Query asked during the previous class

Priority Queue - Example

```
class test {
public static void main(String[] args) {
PriorityQueue<Account> al = new
        PriorityQueue<Account>(5, new AccCmp());

al.add(new Account(123,“Ryan",8.1f));
al.add(new Account(123,"Ankit",8.1f));
al.add(new Account(122,"Ryan",8.1f));
al.add(new Account(123,"Ankit",8.1f));

System.out.println("Acc. No. processed on their priority order");
while (!al.isEmpty()) {
System.out.println(al.peek().acc +" " +al.poll().name);  }
}
}
```

Output:
Acc. No. processed on their priority order
123 Ryan
123 Ankit
123 Ankit
122 Ryan

innovate achieve lead

BITS Pilani

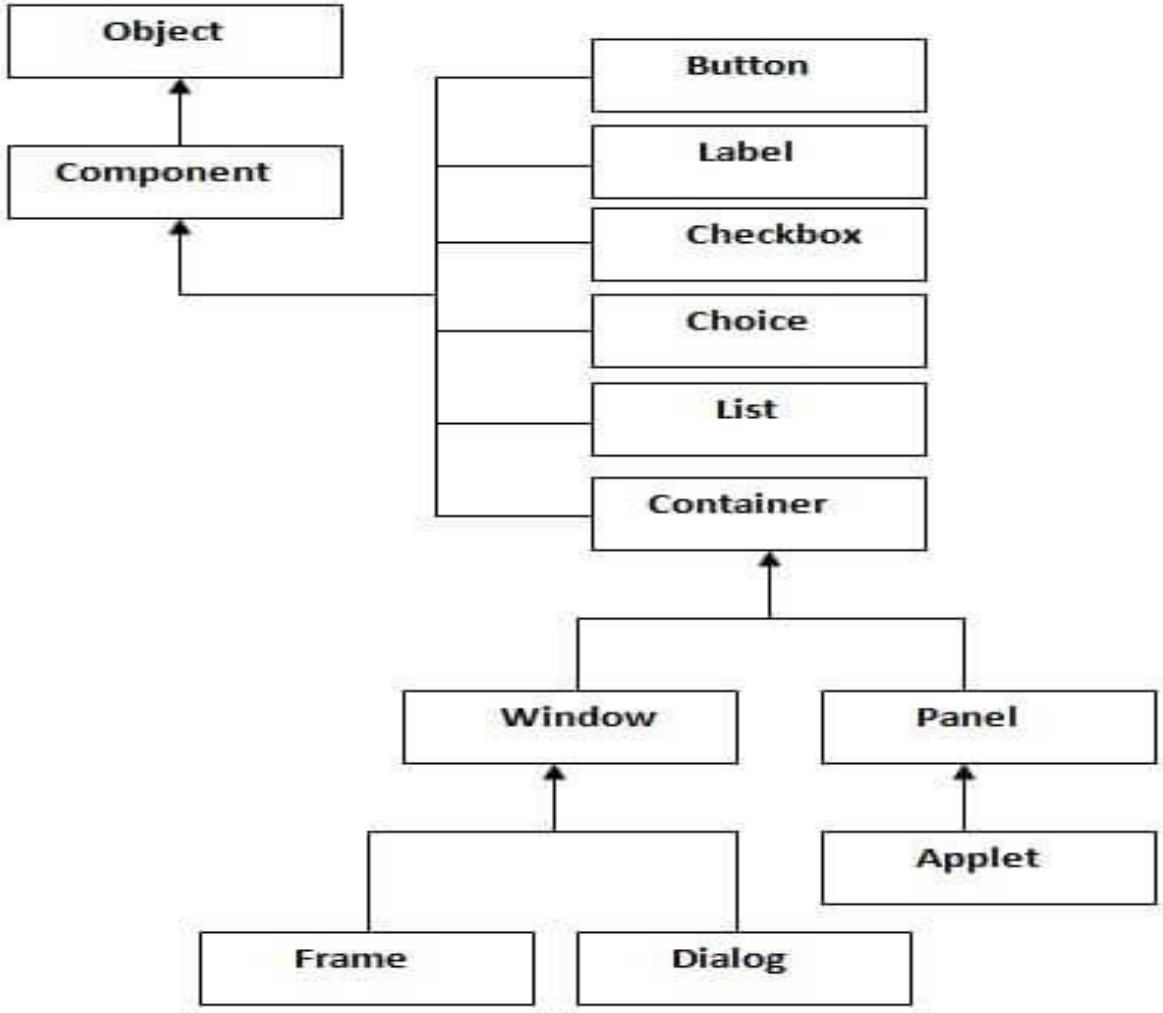
Pilani Campus

AWT

Abstract Window Toolkit

- Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.
- Java AWT components are platform-dependent
  - Java AWT calls native platform (Operating systems) subroutine for creating components such as textbox, checkbox, button etc.
- AWT is heavyweight i.e. its components uses the resources of OS.
- The java.awt package provides classes for AWT APIs such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

AWT Hierarchy



Key Terminologies

- **Container –** component in AWT that can contain another components like buttons, textfields, labels etc. The classes such as Frame, Dialog and Panel extends the Container class.
- **Window -** The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.
- **Panel -** The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.
- **Frame -** The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

AWT by Inheritance

```
import java.awt.*;
class MyGui extends Frame {
MyGui(){
setSize(1000,1000);
setLayout(null);
setVisible(true);
setTitle("Core Banking");
Button b=new Button("Submit");
setBackground(Color.cyan);
b.setBounds(50,100,80,30);
add(b);}
}
class test {
public static void main(String[] args) {
MyGui mi = new MyGui();
}}
```

AWT by Association


```
class test {
public static void main(String[] args) {

    Frame f=new Frame("Core Banking");
    Button b=new Button("Submit");
    b.setBounds(50,100,80,30);
    f.add(b);
    f.setSize(1000,1000);
    f.setBackground(Color.cyan);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

Event Delegation Model

- **Event:** It is an object that describes a state change in a source.
  - Pressing a button, entering a character via keyboard, clicking the mouse etc.
  - Indirect interactions with the user interface may cause events to occur. Eg. timer expires, s/w or h/w failure etc.

- **Event Source:** It is an object that generates an event.
  - A source may generate **more than one type** of event.
  - A source must **register listeners** in order for the listeners to receive notification about the specific type of the event
  - public void add*Type*Listener(*Type*Listener *el*)
    - Eg. **addKeyListener(), addMouseMotionListener()**
  - When an event occurs, the registered listeners are notified and receive a copy of the event object. (Multicasting)
  - **Some source allow only one listener** to register
  - To unregister: public void remove*Type*Listener(*Type*Listener *el*)

Event Delegation Model

- **Event Listener:** It is an object that is notified when an event occurs.
  - It must be registered with one or more sources to receive notifications
  - It must **implement methods to receive and process** these notifications. (Event handlers)
  - These methods are **defined in a set of interfaces** in **java.awt.event**
  - The event handler must **return the control to the run-time system quickly** it should not maintain the control for an extended period of time

Event Classes:

- The root of the event class hierarchy is the EventObject class which contains two methods
  - **getSource() - returns the source of the event**
  - **toString() – returns the string equivalent of the event**
- **AWTEvent** – is the superclass of all the AWT events handled by the event delegation model

Event Class

Event Class	Description
ActionEvent	Generated when a button is pressed, a list item is double-clicked, or a menu item is selected.
AdjustmentEvent	Generated when a scroll bar is manipulated.
ComponentEvent	Generated when a component is hidden, moved, resized, or becomes visible.
ContainerEvent	Generated when a component is added to or removed from a container.
FocusEvent	Generated when a component gains or loses keyboard focus.
InputEvent	Abstract superclass for all component input event classes.
ItemEvent	Generated when a check box or list item is clicked; also occurs when a choice selection is made or a checkable menu item is selected or deselected.
KeyEvent	Generated when input is received from the keyboard.
MouseEvent	Generated when the mouse is dragged, moved, clicked, pressed, or released; also generated when the mouse enters or exits a component.
MouseWheelEvent	Generated when the mouse wheel is moved.
TextEvent	Generated when the value of a text area or text field is changed.
WindowEvent	Generated when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

Sources of Events

Event Source	Description
Button	Generates action events when the button is pressed.
Check box	Generates item events when the check box is selected or deselected.
Choice	Generates item events when the choice is changed.
List	Generates action events when an item is double-clicked; generates item events when an item is selected or deselected.
Menu item	Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected.
Scroll bar	Generates adjustment events when the scroll bar is manipulated.
Text components	Generates text events when the user enters a character.
Window	Generates window events when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

Event Listener Interfaces

innovate achieve lead

INTERFACE	INTERFACE METHODS	ADD METHOD	EVENT CLASS
ActionListener	actionPerformed (ActionEvent)	addActionListener()	ActionEvent
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)	addAdjustmentListener()	AdjustmentEvent
ComponentListener	componentHidden(ComponentEvent)	addComponentListener()	ComponentEvent
	componentMoved(ComponentEvent)		
	componentResized(ComponentEvent)		
	componentShown(ComponentEvent)		
ContainerListener	componentAdded(ComponentEvent)	addContainerListener()	ContainerEvent
	componentRemoved(ComponentEvent)		

Event Listener Interfaces

innovate achieve lead

INTERFACE	INTERFACE METHODS	ADD METHOD	EVENT CLASS
ItemListener	itemStateChanged(ItemEvent)	addItemListener()	ItemEvent
KeyListener	keyPressed(KeyEvent)	addKeyListener()	KeyEvent
	keyReleased(KeyEvent)		
	keyTyped(KeyEvent)		
MouseListener	mouseClicked(MouseEvent)	addMouseListener()	MouseEvent
	mouseEntered(MouseEvent)		
	mouseExited(MouseEvent)		
	mousePressed(MouseEvent)		
	mouseReleased(MouseEvent)		
MouseMotionListener	mouseDragged(MouseEvent)	addMouseMotionListener()	MouseEvent
	mouseMoved(MouseEvent)		

Event Listener Interfaces

innovate achieve lead

INTERFACE	INTERFACE METHODS	ADD METHOD	EVENT CLASS
TextListener	textValueChanged(TextEvent)	addText:Listener()	TextEvent
WindowListener	windowActivated(WindowEvent)	addWindowListener()	WindowEvent
	windowClosed(WindowEvent)		
	windowClosing(WindowEvent)		
	windowDeactivated(WindowEvent)		
	windowDeiconified(WindowEvent)		
	windowIconified(WindowEvent)		
	windowOpened(WindowEvent)		