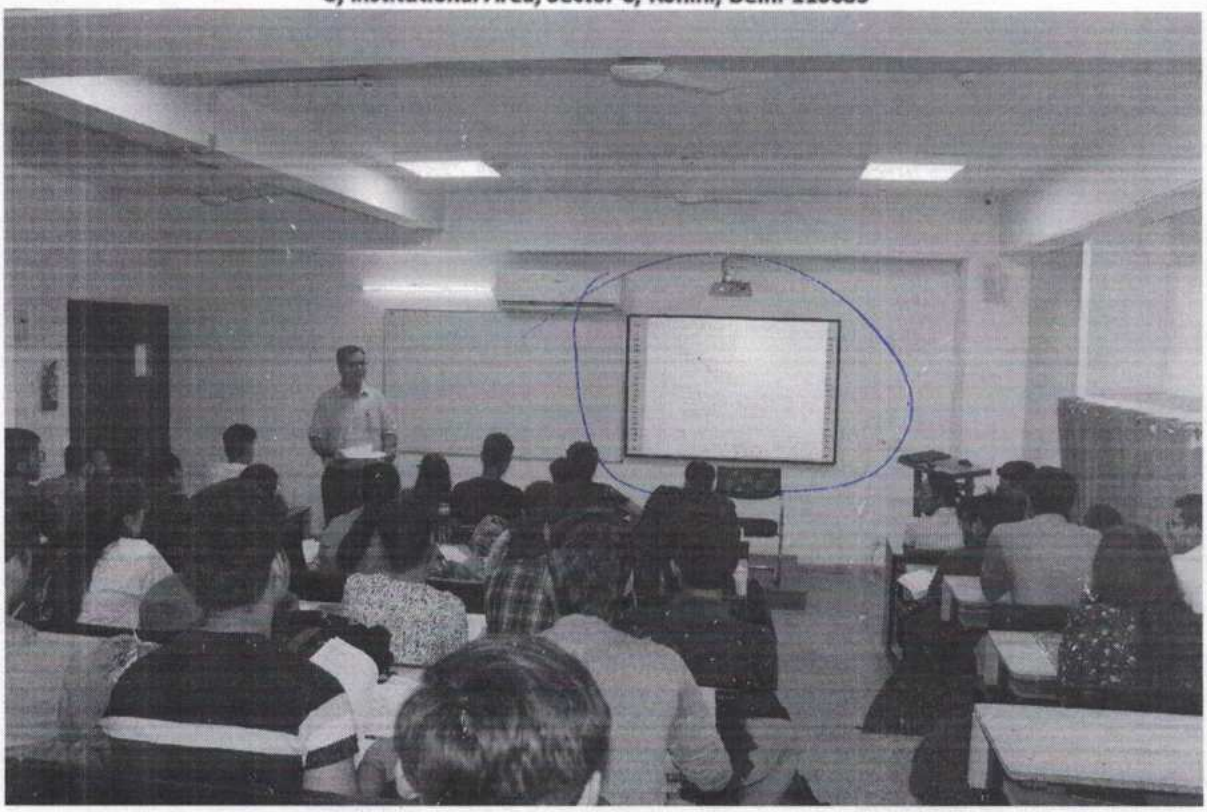
Sample copy of teachers using ICT for teaching (having PPTs)

Annexure IV-K

Jagan Institute of Management Studies jims

NAAC A++ Grade & NBA Accredited MCA Program

3, Institutional Area, Sector-5, Rohini, Delhi-110085

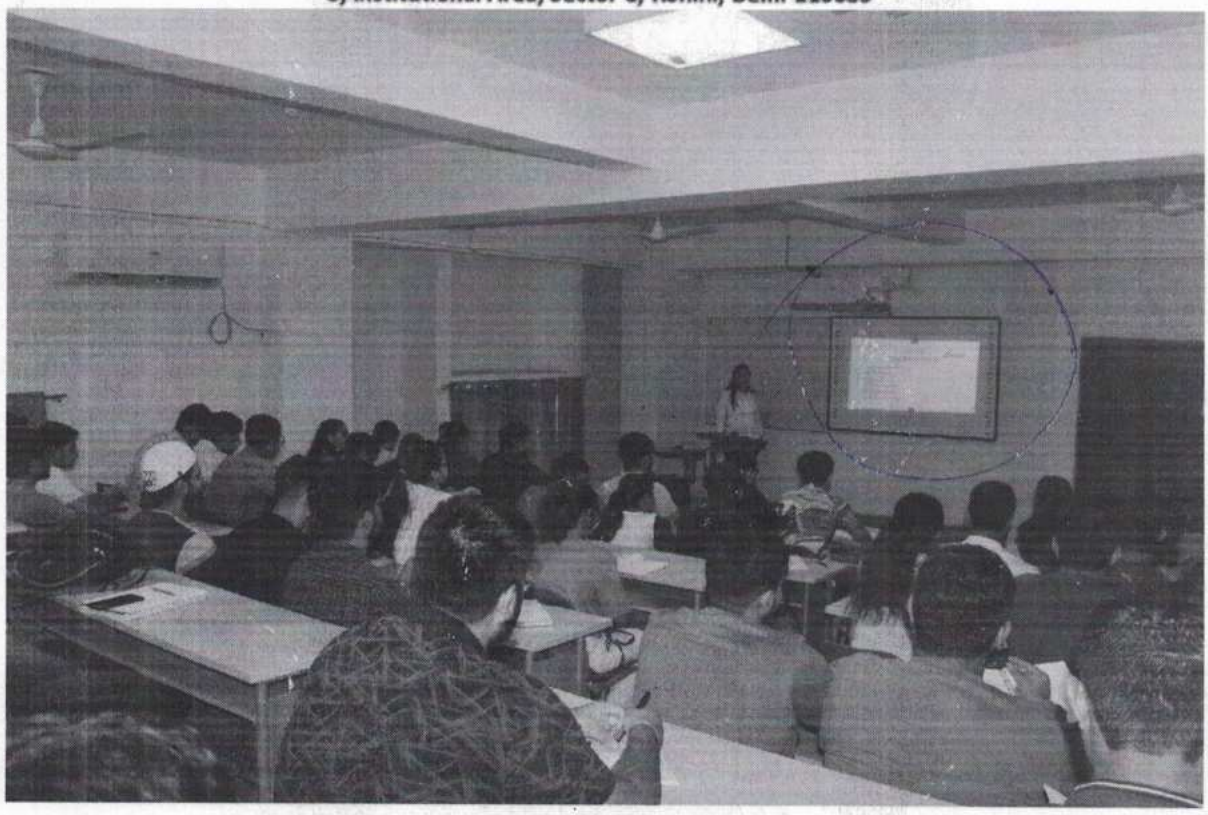


Jagan Institute of Management Studies Rohini, Sector-5, Delhi-85

Jagan Institute of Management Studies jims

NAAC A++ Grade & NBA Accredited MCA Program

3, Institutional Area, Sector-5, Rohini, Delhi-110085

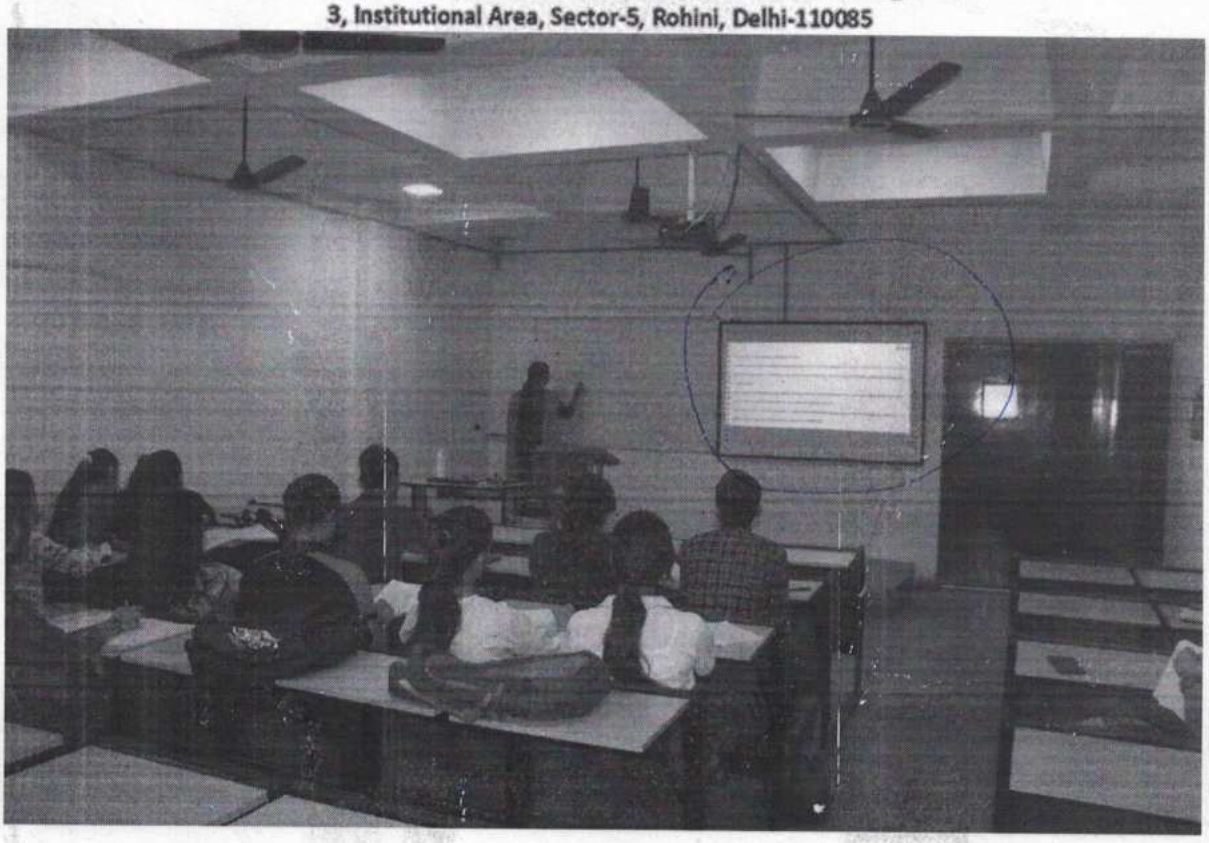


Jagan Institute of Management Studies
Rohini, Sector-5, Delhi-85

Jagan Institute of Management Studies jims

NAAC A++ Grade & NBA Accredited MCA Program

3, Institutional Area, Sector-5, Rohini, Delhi-110085

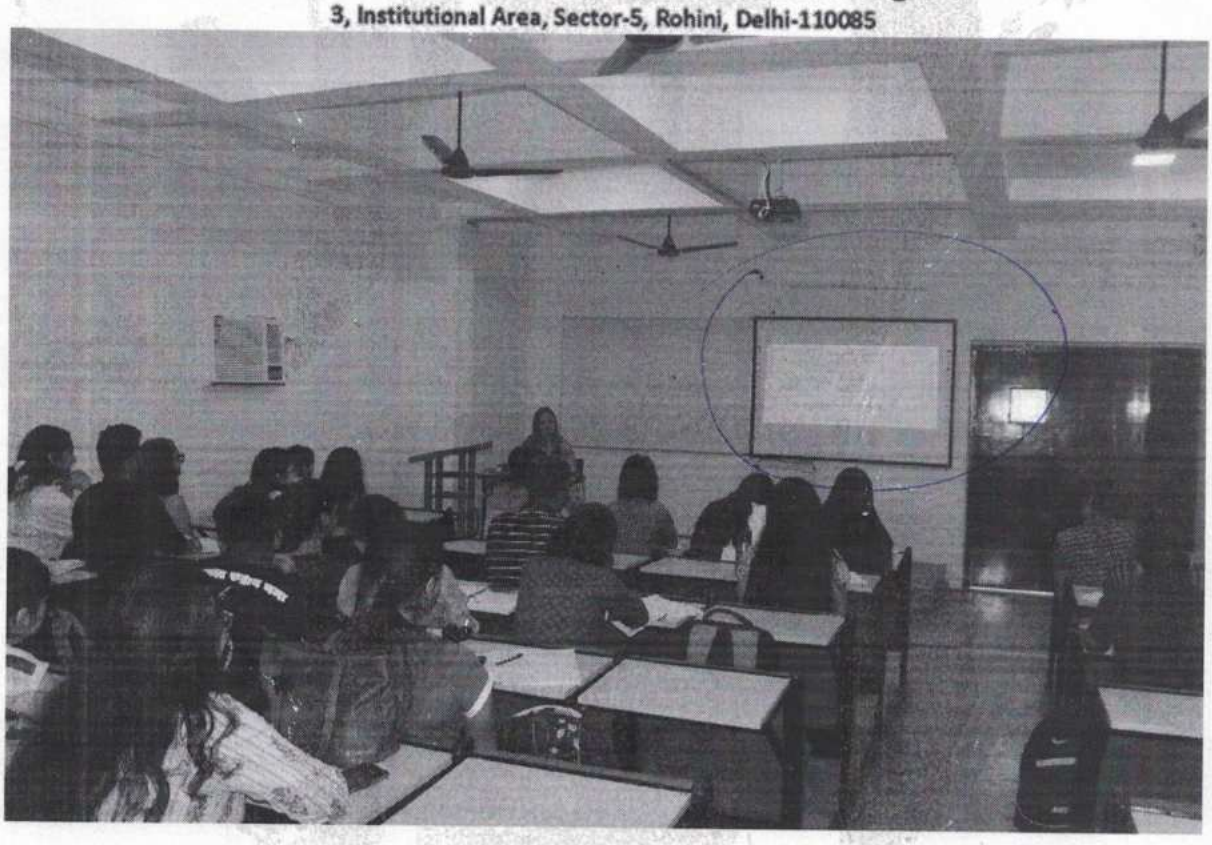


Jagan Institute of Management Studies
Rohini, Sector-5, Delhi-85

Jagan Institute of Management Studies jims

NAAC A++ Grade & NBA Accredited MCA Program

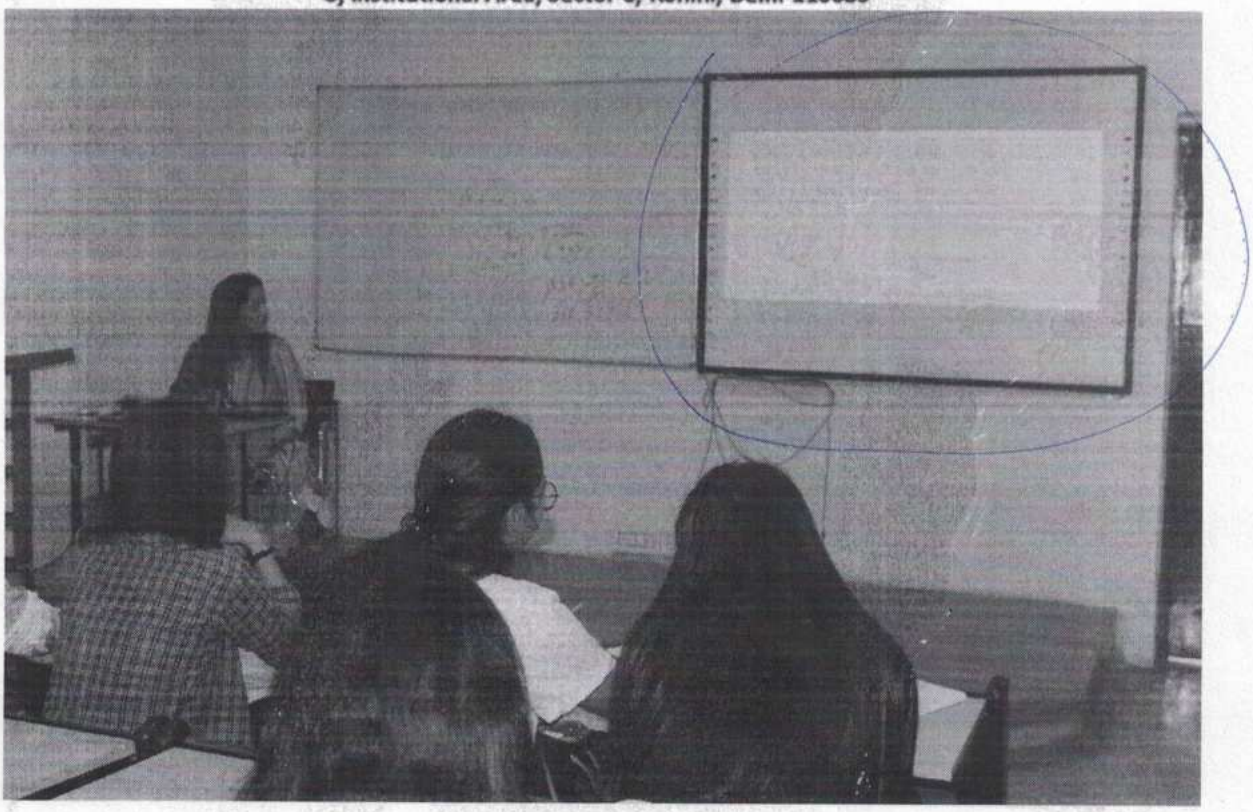
3, Institutional Area, Sector-5, Rohini, Delhi-110085



Jagan Institute of Management Studies jims

NAAC A++ Grade & NBA Accredited MCA Program

3, Institutional Area, Sector-5, Rohini, Delhi-110085

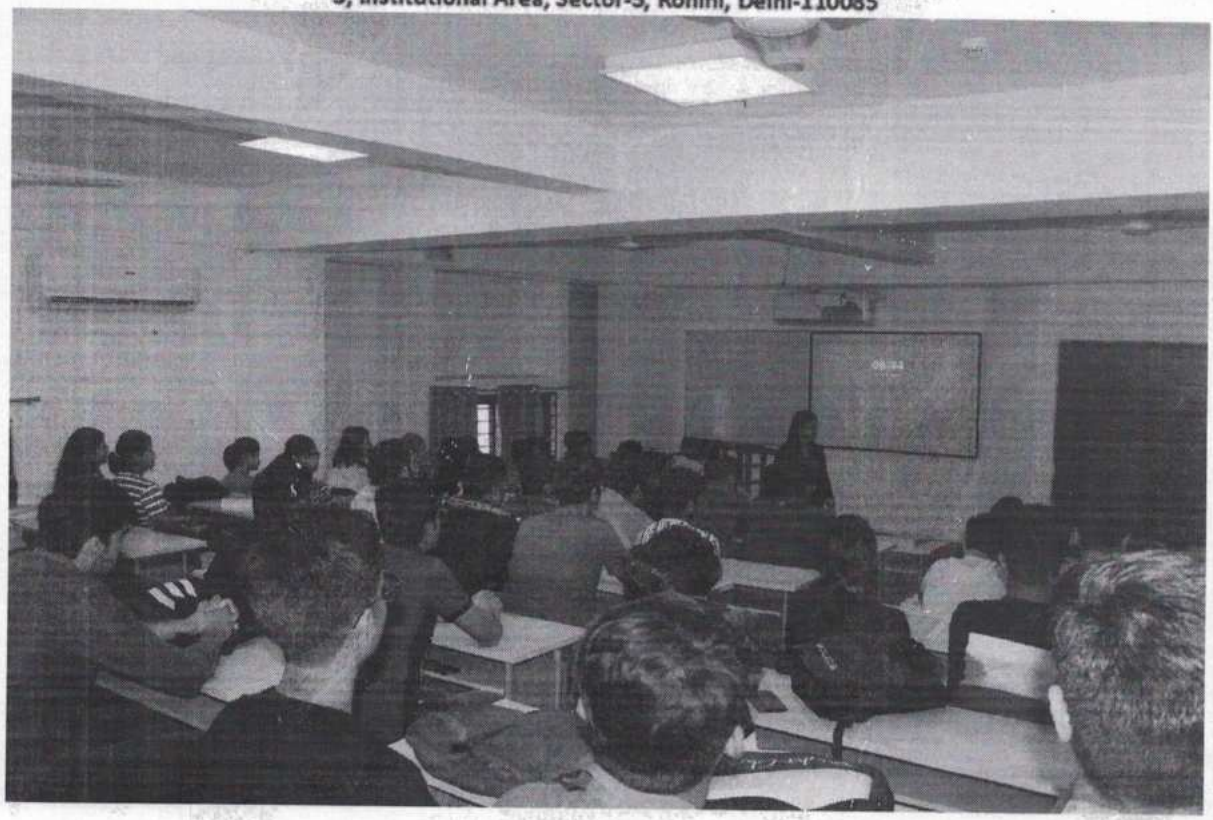


Jagan Institute of Management Studies
Rohini, Sector-5, Delhi-85

Jagan Institute of Management Studies jims

NAAC A++ Grade & NBA Accredited MCA Program

3, Institutional Area, Sector-5, Rohini, Delhi-110085



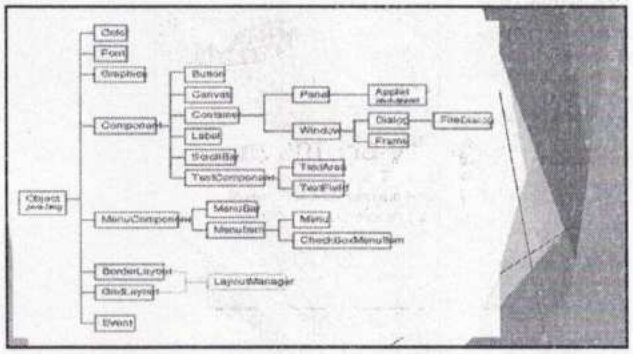
Jagan Institute of Management Studies
Rohini, Sector-5, Delhi-85

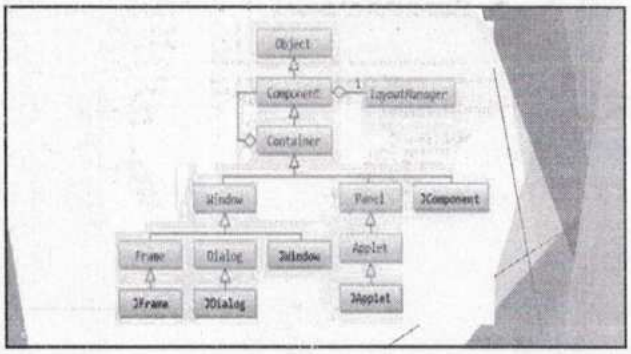
Event Handling

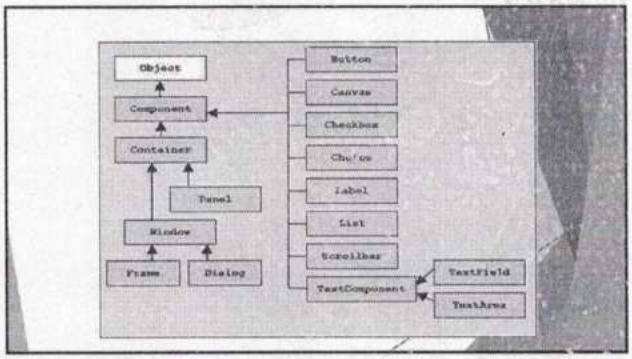
Dr. Archana B Saxena
Professor, Department of Information Technology
Archanab.saxena@jimsindia.org, archanabsaxena@gmail.com

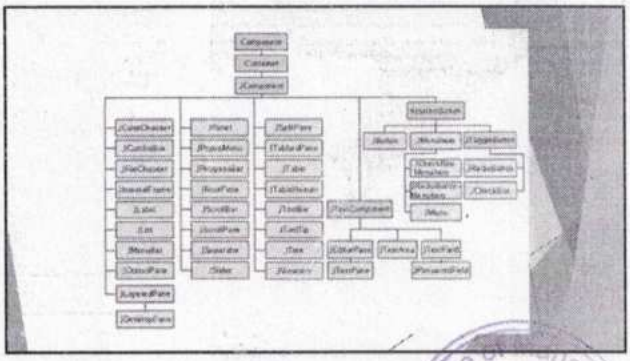
- When developing a Java program it is important to select the appropriate Java Graphical User Interface (GUI) components. There are two basic sets of components that you will most likely build your Java programs with. These two groups of components are:
  - Abstract Window Toolkit (AWT)
  - Swing.

  Both of these groups of components are part of the Java Foundation Classes (JFC).
- The **Java Foundation Classes** (JFC) are a graphical framework for building portable Java-based graphical user interfaces (GUIs).
- (JFC) dramatically simplify the development and deployment of commercial-quality desktop and Internet/intranet applications.
- JFC consists of the Abstract Window Toolkit (AWT) and Swing . Together, they provide a consistent user interface for Java programs, regardless whether the underlying user interface system is Windows, Mac OS X or Linux.
- Using the Java programming language, Java Foundation Classes (JFC) are pre-written code in the form of class libraries (coded routines) that give the programmer a comprehensive set of graphical user interface (GUI) routines to use.
- The Java Foundation Classes are a superset that contains AWT.









Control Fundemaentals

- AWT supports following types of controls:
  - Label
  - Button
  - Check Boxes
  - Choice lists
  - Lists
  - Text editing

These controls are subclass of component class.

Label: A label is an object of type Label, and it contains a string which it displays. Label are passive controls that do not support an interaction with user.

Label();

Label(String str)

Button: Most widely used control is the push button. Push buttons are object of type Buttons.

Button()

Button(String str)

TextField: the TextField class implements a single line text entry area, usually called and edit control. Text field allow the user to enter strings and to edit the text field using the arrow keys, cut and paste keys and mouse selections.

TextField()

TextField(int numchars)

TextField(String str, int numchars)

TextArea: Sometimes a single line of text input is not enough for a given task. To handle these situations, the AWT includes a simple multiline editor called Textarea.

TextArea()

TextArea(String str)

TextArea(int numlines, int numarea)

TextArea(String str,int numlines, int numarea)

Check Boxes: check box is a control that is used to turn an option on or off. It consist of a small box that can either contain a check mark or not. There is an label associated with each check box that describes what option the box represents.

Checkbox();

Checkbox(String str);

Checkbox(String str, Boolean on);

Checkbox(String str, Boolean on, CheckboxGroup cbgroup);

CheckboxGroup: it is possible to create a set of mutually exclusive checkboxes in which one and only one check box in the group can be checked at any one time. These check boxed are often called radio buttons. Chekbox group are object of CheckboxGroup .

CheckboxGroup();

Choice control: the choice control is used to create a pop up list of items from which a user may choose. When inactive Choice component takes up only enough space to show the currently selected items.

Lists: List class provides a compact, multiple choice, scrolling selection list. Unlike choice object, which shows only the single selected item in the menu. A list object can be constructed to show any number of choices in the visible window. It can also be used to allow multiple selections. List provides three constructor:

List()

List(int numrows)

List(int numrows boolean multipleselect)

Layout Managers

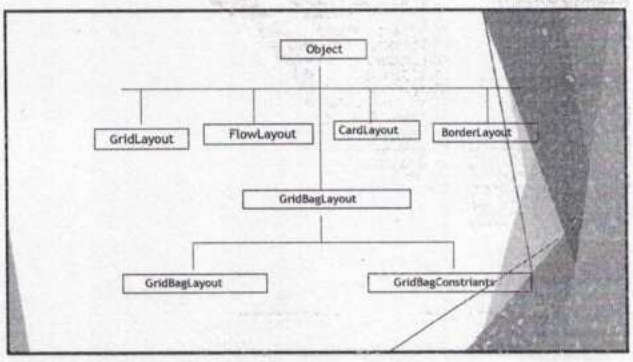
Layout

Layout Management is the process that determines the size and the position of component in a container. The layout manager manages the layout of the components in a container. It determines the size and position of the components of the container. Every container has a default layout manager.

Layout managers are the special objects that determines how the component of a container are organized. When you create a container , java automatically creates and assigns a default layout manager. The default layout manager determines the placing of controls in the display area of the container. You can create different types of layout managers in order to control how your applet or frame looks.

The list of Commonly used layout managers in Java is given:

- Flow Layout (Default)
- Grid Layout
- Border Layout
- Card Layout
- GridBag Layout



Flow Layout Manager: the default manager of an applet is the flow layout manager. The flow layout manager places control in the order in which they are added, when the layout manager reaches the right border of the applet , it begins placing the controls on the next row. In its default state, the flow layout manager centers the control in each row.

The GRID Layout Manager: the flow layout manager is the simplest of the layout manager, it may not give you the control you need to create sophisticated frames and applets. When more control is needed over the placement of components , the grid layout can be used.

The grid layout manager organizes the display into a rectangular grid, java then places the component you create into each grid working from left to right and top to bottom.

The Border Layout Manager: the border layout manager enables you to position components using the directions : North, South, East, West and center.

The Card Layout Manager: the card layout manager is one of the most complex layout manager. Using this manager, you can create stack of cards and then flip from one layout to another. Like tabbed property in Windows

GridBag Layout Manager: the gridBag layout manager is the most flexible and complex layout manager that AWT provides. It places components in rows and columns, allowing specified components to span multiple rows and columns. You can resize the components by assigning weights to individual components in the gridBag layout.

When you specify the size and position of components , you also need to specify the constraints for each component. To specify constraints , you need to set variables in a GridBagConstriants object and specify the GridBagLayout manager object with the setConstraints() method, to associate the constraints with the component.

The GridBagLayout class has a single constructor that does not take any arguments. Since the position of each component in a layout is controlled by a GridBagLayout object is determined by the currently set GridBagConstriants object you need to create the GridBagConstriants object before the layout can built. The object is built by calling constructor of the class using the code:

GridBagConstriants con = new GridBagConstriants()

Like the GridBagLayout class, the GridBagConstriants constructor requires no arguments. However since the attributes of the class start off initialized to default values, you will usually need to change of those values before adding component to the layout.

Specifying Constraints : you can reuse a GridBagConstriants object for multiple components, even if the component have different constraints . You can assign the following values to GridBagConstriants attribute : anchor, fill, gridx, gridy,weightx and weighty

- anchor: this attribute is used when a component is smaller than its display area, in order to determine where to place the component in the display area. The valid values are:

GridBagConstriants.CENTER (default)

GridBagConstriants.NORTH

GridBagConstriants.NORTHEAST

GridBagConstriants.EAST

GridBagConstriants.SOUTHEAST

GridBagConstriants.SOUTH

GridBagConstriants.SOUTHWEST

GridBagConstriants.WEST

GridBagConstriants.NORTHWEST

- fill: this attribute is used when display area of a component is larger than the size required by the component, in order to determine whether (and how) to resize the component. The valid values are :

GridBagConstriants.NONE (default)

GridBagConstriants.HORIZONTAL

GridBagConstriants.VERTICAL

GridBagConstriants.BOTH

gridx and gridy: these attributes respectively specify the row and column at the upper left of the display area of the component. The leftmost column has the address gridx=0 and the topmost cell has the address gridy=0. use GridBagConstriants. RELATIVE (default value) to specify that the component that is added to the container before this component.

weightx and weighty: these attributes determine whether the component stretch horizontally to fill the display area of the applet (weightx) or vertically (weighty). The default value is 0 for both.

Event Handling

- What is an Event?

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

Types of Event: The events can be broadly classified into two categories:

- **Foreground Events** - Those events which require the direct interaction of user.They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page etc.
- **Background Events** - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

Event Handling

- What is Event Handling?

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

- **Source** - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide as with classes for source object.
- **Listener** - It is also known as event handler.Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns.

Delegation Event Model

Modern approach to handling events is based on the delegation event Model, which defines standard and consistent mechanism to generate and process events. Its Concepts says:

- A source generates an event and sends it to one or more listeners
- In this scheme listeners simple waits until it receives an event.
- Once event is received listener process it and return

In this delegation model , listener must register with a source in order to receive an event notification.

Components of an event: An event compromises of three components:

- **Events:** when a user interacts with an application by pressing a key or clicking a mouse button an event is generated. The operating system traps this event and the data associated with it, for example time at which the event occurred, the type of event (key press, mouse move). This data is passed to the application to which the event belongs. Events may also occur that are not directly caused by interactions with a user interface. like: when time expires, when counter exceeds a value , H/W or S/W failure.
- **Event Source:** an event source is an object that generates the event. For example, if you click on a button, or move mouse. Source may generate more than one type of event. A source may register listeners in order for the listeners to receive notifications about a specific event.
- **Event Handler/listeners:** an event handler is a method that understands the event and processed it. The event handler method takes an event object as parameter. It has two major requirements
  - First, it must register with one or more source to receive notifications about specific type of events.
  - it must implement methods to receive and process these notifications.

Steps involved in event handling

- The User clicks the button and the event is generated.
- Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
- Event object is forwarded to the method of registered listener class.
- the method is now get executed and returns.

Points to remember about listener

- In order to design a listener class we have to develop some listener interfaces. These Listener interfaces forecast some public abstract callback methods which must be implemented by the listener class.
- If you do not implement the any if the predefined interfaces then your class can not act as a listener class for a source object.

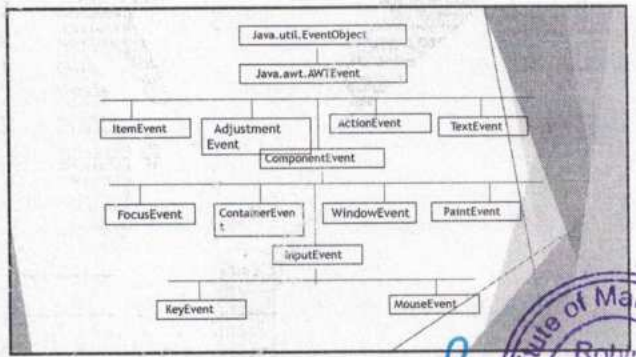
JDK 1.2 Event Model (Delegation)

The delegation model came into existence with JDK 1.1 . In this model , you can specify the objects that are to be notified when a specific event occurs. If the event is irrelevant , it is discarded. The JDK 1.2 model is based on following components:

- Event Classes
- Event Listeners
- Explicit event handling
- Adapters
- Inner Classes

Event Classes

- The EventObject class is at the top of the event class hierarchy. It belongs to the java,util package. It is superclass for all events. Its one constructor :
- EventObject(Object src). (src is the object that generates this event.)
- EventObject contains two method:
  - getSource(): method returns the source of the event.
  - toString(): returns the string equivalent of the event.
- The class AWTEvent, defined within the java.awt package is a subclass of EventObject class, and super class of all awt events that are handled by the delegation event model.



Event Handling Table

Event	Event class	Interface	Interface Methods	Registering method	Program Name
• Click • Double click • Menu Selection	ActionEvent	ActionListener	Public void actionPerformed(ActionEvent ae)	addActionListener()	Event_Handling.java
• Focus gain • Focus Lost	FocusEvent	focusListerer	• Public void FocusGained(FocusEvent fe) • Public void FocusLost(FocusEvent fe)	addFocusListener()	Focus_Listener.java

Event	Event class	Interface	Interface methods	Registering method	Program Name
• Kerpressed • keyReleased • keyTyped	KeyEvent	KeyListener	Public void KeyPressed(KeyEvent ke) Public void KeyReleased(KeyEvent ke) Public void KeyTyped(KeyEvent ke)	addKeyListener()	Key_Listener.java
• Focus gain • Focus Lost	FocusEvent	focusListerer	• Public void FocusGained(FocusEvent fe) • Public void FocusLost(FocusEvent fe)	addFocusListener()	Focus_Listener.java

Event Classes

The package java.awt.event defines several types of events that are generated by various interface elements. following are event classes:

- The ActionEvent Class:
  - Events: action event is generated when a button is pressed, a list item is double clicked or a menu is selected.
  - Interface: The ActionListenerInterface is used to handle these events generated by ActionEvent class. This interface defines actionPerformed() method that is invoked when an action event occurs.
  - InterfaceMethod: Public Void actionPerformed(ActionEvent ae)

[Event_Handling.java]

The FocusEventClass:

- Event:the focus event is generated when a component gains or losses input focus.
- Interface: FocusListener
- Interface functions:
  - Public void FocusGained(FocusEvent fe)
  - Public void FocusLost(FocusEvent fe)

[Focus_Listener.java]

The InputEvent Class: Event: abstract InputEvent is a subclass of ComponentEvent and is super class of KeyEvent and MouseEvent.

ItemEventClass: the item event is generated when a checkable menu item is selected or deselected.

Interface: ItemListener

Interface Function: Public void ItemStateChanged(ItemEvent ie)

[Item_Listener.java]

KeyEventClass: a key event occurs when keyboard input occurs.

Interface: KeyListener

Interface functions:

Public void KeyPressed(KeyEvent ke)

Public void KeyReleased(KeyEvent ke)

Public void KeyTyped(KeyEvent ke)

[Key_Event_Listener.java]

Adjustment Event Class:

Events: an adjustment event is generated by a scroll bar.

Interface: AdjustmentListener

Interface Method: Public void adjustmentValuechanged(AdjustmentEvent ac)

The componentEvent Class:

- Event:a component event is generated when the size, position or visibility of a component is changed.
- Interface: ComponentListener
- Interface function: this listener has four methods that are invoked when a component is resized, moved, shown or hidden.
  - Public void ComponentResized(ComponentEvent ce)
  - Public void ComponentMoved(ComponentEvent ce)
  - Public void ComponentShown(ComponentEvent ce)
  - Public void ComponentHidden(ComponentEvent ce)

[Component_Event.java]

The ContainerEvent Class:

Event: a container event is generated when a component is added or removed from a container.

Interface: ContainerListener

Interface functions: this interface has two functions:

- Public void ComponentAdded(ContainerEvent ce)
- Public void ComponentRemoved(ContainerEvent ce)

[Container_Listener.java]

The MouseEvent Class:

Events: there are eight types of mouse events: mouse click, mouse drag, mouse enter a component, mouse exit a component, mouse moved, mouse pressed, mouse release and mouse wheel move.

Interface: The MouseListener Interface

Interface Functions: void mouseClicked(MouseEvent me)

Public void mouseEntered(MouseEvent me)

Public void mouseExited(MouseEvent me)

Public void mousePressed(MouseEvent me)

Public void mouseReleased(MouseEvent me) [Mouse_Listener.java]

Mouse WheelEvent Class: is subclass of MouseEvent class.
Interface: The MouseWheelListener
InterfaceFunction: public void mouseWheelMoved(MousEevent me)

The TextEvent Class: these are generated by text fields and text areas. When charaters are entered by user or program.
Interface: The TextListener
Interface Function: public void textValueChanged(TextEvent te)
[Text_Listener.java]

Window Event class: there are ten type of window event: window activated, window closed, window gain focus, window lost focus and many more.
Interface: The windowListener
Interface Functions: void windowActivated(WindowEvent we)
Public void windowClosed(WindowEvent we)
Public void windowDeactivated(WindowEvent we)
Public void windowOpened(WindowEvent we)
Public void windowIconified(WindowEvent we)

Difference between AWT and Swing

S.NO	AWT	SWING
1.	Package use is java.awt.*	Package used is javax.swing.*;
2.	The components of Java AWT are heavy weighted.	The components of Java Swing are light weighted.
3.	Java AWT has comparatively less functionality as compared to Swing.	Java Swing has more functionality as compared to AWT.
4.	The execution time of AWT is more than Swing.	The execution time of Swing is less than AWT.
5.	The components of Java AWT are platform dependent.	The components of Java Swing are platform independent.
6.	MVC pattern is not supported by AWT.	MVC pattern is supported by Swing.
7.	AWT provides comparatively less powerful components.	Swing provides more powerful components.

Adapter Classes

- Java provides a special feature called an adapter classes, that can simplify the creation of event handlers in certain situations. An adapter class provides an empty implementation of all methods in an event listener interface. Adapter classes are useful when you want to receive and process only some of the events that are handled by a particular event listener interface. You can define a new class to act as an event listener by extending one of the adapter classes and implementing only those events in which you are interested.

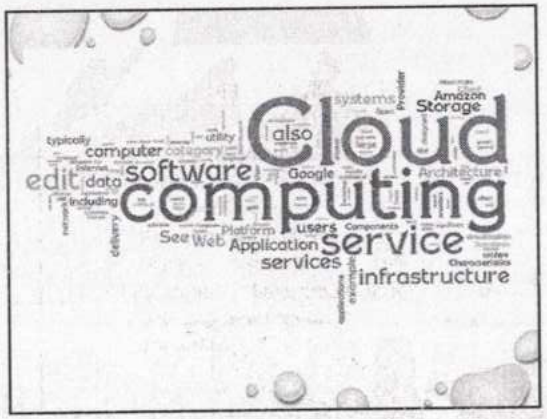
Commonly used Listener interfaces implemented by adapter classes:
- ComponentAdapter: - ComponentListener
- ContainerAdapter: - ContainerListener
- FocusAdapter: - FocusListener
- KeyAdapter: KeyListener
- MouseAdapter: MouseListener
- MouseMotionAdapater: MouseMotionListener
- WindowsAdapater: WindowsListener

[ComponentAdapter.java]

LECTURE 1- CLOUD COMPUTING

BY:

DR. ISHA SINGH



INTRODUCTION

- How many of you used cloud computing services ?
  - If you ever browsed YouTube for videos
  - If you streamed music online
  - Or just sent and received emails online
- "The cloud has been around us for a long time and its only now we have started realizing its full potential"

WHAT IS CLOUD COMPUTING?

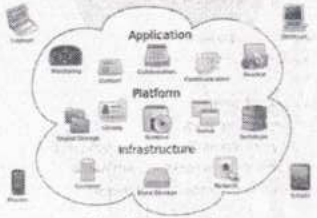
- IT IS **DELIVERY OF COMPUTING SERVICE OVER THE INTERNET.**

  (SERVICES LIKE SERVERS, DATABASES, NETWORKING, SOFTWARE ETC)

  PAY FOR WHAT YOU USE !!
- **STORING OR ACCESSING YOUR DATA OVER THE INTERNET.**
- E.G.: YAHOO!, GMAIL, HOTMAIL-

  Instead of running an e-mail program on your computer, you log in to a web e-mail account remotely. The software and storage for your account doesn't exist on your computer -- it's on the service's computer cloud.

WHAT IS CLOUD COMPUTING?



Cloud Computing

What is not cloud computing ?

- If you peel back the label and its says "Grid" or "OGSA" underneath... its not a cloud.
- If you need to send a 40 page requirements document to the vendor then... it is not cloud.
- If you can't buy it on your personal credit card... it is not a cloud
- If they are trying to sell you hardware... its not a cloud.
- If there is no API... its not a cloud.
- If you need to rearchitect your systems for it... its not a cloud.
- If it takes more than ten minutes to provision... its not a cloud.
- If you can't deprovision in less than ten minutes... its not a cloud.
- If you know where the machines are... its not a cloud.
- If there is a consultant in the room... its not a cloud.

What is not cloud computing ?

- If you need to specify the number of machines you want upfront... its not a cloud.
- If it only runs one operating system... its not a cloud.
- If you can't connect to it from your own machine... its not a cloud.
- If you need to install software to use it... its not a cloud.
- If you own all the hardware... its not a cloud.
- If it takes 20 slides to explain.... its not a cloud

HISTORY

- CONCEPT EVOLVED IN 1950(IBM) CALLED RJE (REMOTE JOB ENTRY PROCESS).
- IN 2006 AMAZON PROVIDED FIRST PUBLIC CLOUD AWS(AMAZON WEB SERVICE).

What is new ?

- Acquisition Model: Based on purchasing of services
- Business Model: Based on pay for use
- Access Model: Over the internet to ANY device
- Technical Model: Scalable, elastic, dynamic, multi-tenant, & sharable

Five Essential Characteristics

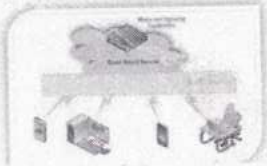
On-Demand self-service

A consumer can provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Five Essential Characteristics

Broad network access

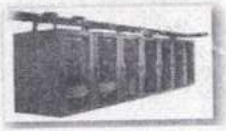
Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).



Five Essential Characteristics

Resource Pooling

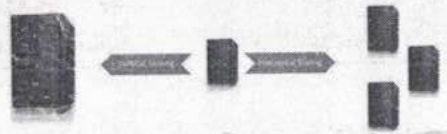
The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.



Five Essential Characteristics

Rapid elasticity

Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.



Five Essential Characteristics

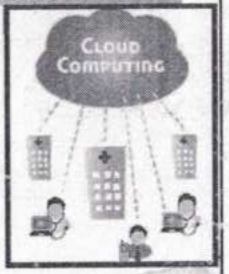
Measured service

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts).

Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

WHY USE THE CLOUD ??

- Cost Savings
- Reduction in IT Staff
- Increased availability
- Device & Location Independence
- Moves capital expense to operating expense
- Scalable and On-Demand
- Low Entry Point



THANK YOU !!

NEXT LECTURE : CLOUD COMPUTING SERVICE MODELS