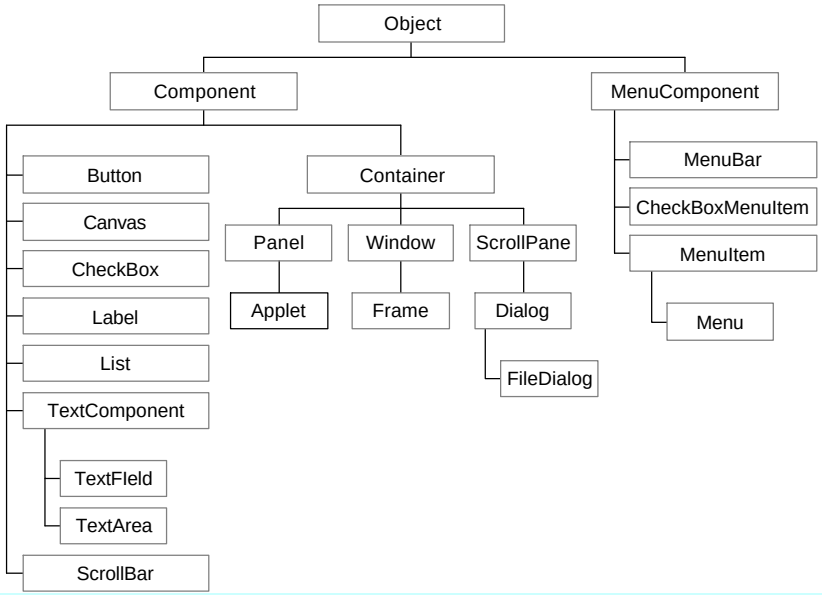
- Java Foundation Classes (JFC) provide 2 frameworks for building GUI-based applications
  - AWT: Abstract Windowing Toolkit
  - Swing

Java GUI



GUI

GUI = graphical user interface

⇒ an interface between a user and a computer, that makes use of input devices other than the keyboard, and presentation techniques other than alphanumeric characters

AWT = Abstract Windowing Toolkit

- 500 **pixel**s ≈ 6 inches (15.25 cm) on a 17 inch monitor
- An **event listener** acts as a communication between the Component and the Listener

The origin of (0,0) is the top left-hand corner of the container

Component	java.awt.**Component** public abstract class Component extends Object implements ImageObserver, MenuContainer, Serializable
	• void **setBackground(Color c)** • Color **getBackground()** • void **setBounds(int x, int y, int width, int height)** • Rectangle getBounds() • void setBounds(Rectangle r) • void **setFont(Font f)** • Font **getFont()** • void **setForeground(Color c)** • Color **getForeground()** Normally, the background color is used to fill the area occupied by the component, and text is rendered in the component's area using foreground color. o if foreground and background colors are not explicitly specified for a component, the corresponding values from this component's immediate container are used. • void **setLocation(int x, int y)** • Point **getLocation()**→ point specifying the component's top-left corner. • void setLocation(Point p) • void **setName(String name)** • String **getName()** • void setSize(Dimension d) • void **setSize(int width, int height)** • Dimension getPreferredSize() • Dimension **getSize()** //width, height) • void **setVisible(boolean b)** Default visibility is true for all components except for Window, Frame and Dialog classes
	• boolean hasFocus() → true if this Component has the keyboard focus. • void requestFocus() → Requests that this component get the input focus. • void **validate()** → Ensures that this component has a valid layout. • void invalidate() • boolean isDisplayable() • boolean isValid() • boolean **isVisible()**
	• float getAlignmentX() • float getAlignmentY() • int getHeight() • int getWidth() • Point getLocationOnScreen() • Dimension getMaximumSize() • Dimension getMinimumSize()

	• Container getParent() • int getX() • int getY()
	• public synchronized void **add(PopupMenu popup)** • public synchronized void **remove(MenuComponent popup)** • void addNotify() • void removeNotify()
	• void addXlistener(Xlistener l) • void removeXlistener(Xlistener l) X = **Focus,** InputMethod, **Key, Mouse, MouseMotion,** PropertyChange
	• void **paint(Graphics g)** • void paintAll(Graphics g) → Paints this component and all of its subcomponents. • void **repaint()** • void repaint(int x, int y, int width, int height) • Graphics **getGraphics()**
Container	java.awt.**Container** • public Component **add(Component comp)** • public void **remove(Component comp)** • public void **removeAll()** • public void **setLayout(LayoutManager mgr)** • **LayoutManager getLayout()**
Window	• void **pack()** • void **show()** → visible and bring to front • void dispose()
Frame	java.awt.**Frame** public class Frame extends Window implements MenuContainer 🕮 a top-level window 🕮 with a title and a border. 🕮 The default layout for a frame is BorderLayout. 🕮 are capable of generating the following types of window events: WindowOpened, WindowClosing, WindowClosed, WindowIconified, WindowDeiconified, WindowActivated, WindowDeactivated. • public **Frame()** • public **Frame(String title)** • public void **dispose()** o This method must be called to release the resources that are used for the frame. All components contained by the frame and all windows owned by the frame will also be destroyed. • public synchronized void setTitle(String title) • public void **remove(MenuComponent m)** • public synchronized void **setResizable**(boolean resizable)
Panel	java.awt.**Panel** public class Panel extends Container

	• the simplest container class. • provides space in which an application can attach any other component, including other panels. • The default layout manager for a panel is the FlowLayout layout manager. • public **Panel()** • public **Panel(LayoutManager layout)**
Button	public class Button extends Component java.awt.Button • public **Button()** • public **Button(String label)** • public synchronized void **setLabel(String label)** • public synchronized void **addActionListener(ActionListener l)** • public synchronized void **removeActionListener(ActionListener l)** • public String **getActionCommand()** o Returns the command name of the action event fired by this button • public void **setActionCommand(String command)** o Sets the command name for the action event fired by this button. By default this action command is set to match the label of the button.
Label	public class Label extends Component java.awt.Label • public static final **int CENTER/RIGHT/LEFT** • public **Label()** • public **Label(String text)** • public **Label(String text, int alignment)** • public **String getText()** • public synchronized void **setText(String text)**
Checkbox	public class Checkbox extends Component implements ItemSelectable java.awt.Checkbox • public **Checkbox()** • public **Checkbox(String label)** • public **Checkbox(String label, boolean state)** • public **Checkbox(String label, boolean state, CheckboxGroup group)** • public **Checkbox(String label, CheckboxGroup group, boolean state)** • public synchronized void **addItemListener(ItemListener l)** • public synchronized void **removeItemListener(ItemListener l)** • public String **getLabel()** • public synchronized void **setLabel(String label)** • public void **setCheckboxGroup(CheckboxGroup g)** o Sets this check box's group to be the specified check box group. If this

	check box is already in a different check box group, it is first taken out of that group. o g - the new check box group, or null to remove this check box from any check box group. • public void **setState(boolean state)**
TextComponent	public class TextComponent extends Component java.awt.TextComponent • public synchronized String getSelectedText() • public synchronized String **getText()** • public boolean isEditable() • public synchronized void **setEditable(boolean b)** • public synchronized void **setText(String t)** • public synchronized void selectAll()
TextField	public class TextField extends TextComponent java.awt.TextField • public **TextField()** • public **TextField(String text)** • public **TextField(int columns)** • public **TextField(String text, int columns)** • public synchronized void **addActionListener(ActionListener l)** • public synchronized void **removeActionListener(ActionListener l)**
TextArea	public class TextArea extends TextComponent java.awt.TextArea • public static final **int SCROLLBARS_BOTH/SCROLLBARS_VERTICAL_ONLY/SCROLLBARS_HORIZONTAL_ONLY/SCROLLBARS_NONE** • public **TextArea()** • public **TextArea(String text)** • public **TextArea(int rows, int columns)** • public **TextArea(String text, int rows, int columns)** • public **TextArea(String text, int rows, int columns, int scrollbars)** • public void **append(String str)** • public void insert(String str, int pos)
List	public class List extends Component implements ItemSelectable java.awt.List • Clicking on an item that isn't selected selects it. Clicking on an item that is already selected deselects it. • public **List()** • public **List(int rows)** • public **List(int rows, boolean multipleMode)** • public void **add(String item)** • public synchronized void **addActionListener(ActionListener l)**

	• public synchronized void **removeActionListener(ActionListener l)** • public void **addItem(String item)** • public void **delItem(int position)** • public synchronized **int getSelectedIndex()** • public synchronized **int[] getSelectedIndexes()** • public synchronized **String getSelectedItem()** • public synchronized **String[] getSelectedItems()** • public void **select(int index)** • public synchronized void **deselect(int index)** • public String **getItem(int index)** • public synchronized **String[] getItems()** • public void **remove(int position)** • public synchronized void **remove(String item)** • public void **removeAll()** • public synchronized void **replaceItem(String newValue, int index)** • public **void setMultipleMode(boolean b)**
MenuComponent	public abstract class MenuComponent extends Object implements Serializable java.awt.MenuComponent • public void **setFont(Font f)**
MenuItem	public class MenuItem extends MenuComponent java.awt.MenuItem • public MenuItem() • public MenuItem(String label) • public MenuItem(String label, MenuShortcut s) o Note that use of "-" in a label is reserved to indicate a separator between menu items. By default, all menu items except for separators are enabled. • public String getLabel() • public synchronized void setLabel(String label) • public void setActionCommand(String command) • public String getActionCommand() • public synchronized void addActionListener(ActionListener l) • public synchronized void removeActionListener(ActionListener l)
Menu	public class Menu extends MenuItem implements MenuContainer java.awt.Menu • public **Menu()** • public **Menu(String label)** • public Menu(String label, boolean tearOff) • public MenuItem add(MenuItem mi) • public void **add(String label)** • public void insert(MenuItem menuitem, int index) • public void insert(String label, int index)

	• public void addSeparator() • public void insertSeparator(int index) • public void remove(int index) • public void remove(MenuComponent item) • public void removeAll()
MenuBar	public class MenuBar extends MenuComponent implements MenuContainer java.awt.MenuBar • public **MenuBar()** • public Menu **add(Menu m)** • public void remove(int index) • public void remove(MenuComponent m) • public void setHelpMenu(Menu m)

Color	java.awt.**Color** • public static final **Color** black/red/pink/orange/yellow/green/magenta/cyan/blue/darkGray,gray,lightGray/white • public **Color(int r, int g, int b)** → 0-255 • public **Color(float r, float g, float b)** → 0.0-1.0
Font	java.awt.**Font** • public static final **int BOLD/ITALIC/PLAIN** • public **Font(String name, int style, int size)** o name: **Serif**, **SansSerif**, **Monospaced**, **Dialog**, **DialogInput**

FlowLayout	java.awt.**FlowLayout** public class FlowLayout extends Object implements LayoutManager, Serializable • arranges components in a left-to-right flow, much like lines of text in a paragraph. • default layout manager for the Panel and Applet classes • default: Each line is centered. • public static final **int CENTER/LEFT/RIGHT** • public **FlowLayout()** o Constructs a new Flow Layout with a centered alignment and a default 5-unit horizontal and vertical gap. • public **FlowLayout(int align)** o alignment : FlowLayout.LEFT, FlowLayout.RIGHT, or FlowLayout.CENTER • public **FlowLayout(int align, int hgap, int vgap)** • public void layoutContainer(Container target)

GridLayout	public class GridLayout extends Object implements LayoutManager, Serializable java.awt.GridLayout • lays out a container's components in a rectangular grid. • The container is divided into equal-sized rectangles, and one component is placed in each rectangle. • Only one component can be placed in each cell • a component is resized to fill the cell • default gap = 0 • public **GridLayout()** • public **GridLayout(int rows, int cols)** o either rows or columns can be 0, but not both • public **GridLayout(int rows, int cols, int hgap, int vgap)** • public void layoutContainer(Container parent)
BorderLayout	• default layout manager for Window and its subclasses (Frame and Dialog)

Prototype:

```
class G extends Frame/Panel implements Xlistener, , , …
{
        public G(String s)
        {
            super(s);
            setBackground(Color c);
            setLayout(LayoutManager mgr);
            Component.addListener(this);
        }
        public void ListenerMethod(XEvent e)
        {
            if(e.getSource()==ComponentObject){}
        }
        public void paint(Graphic g)
        {
            g.graphicMethod();
        }
        public void terminate()
        {
            dispose(); System.exit(0);
        }

}
```

Event delegation

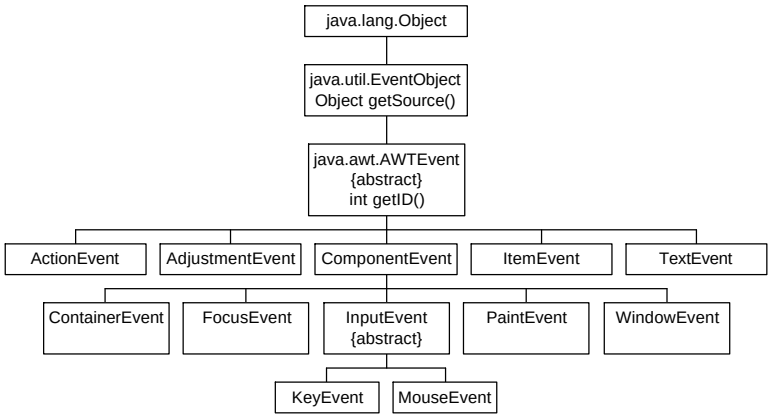
elements

- Event classes → encapsulate information about different types of user interaction
- Event source objects → inform event listener about events when these occur and supply the necessary information about these events
    - Each event source defines methods for registering

(addXListener()) and removing (removeXListener()) listeners, which implement a particular listener interface (Xlistener)

- Event listener objects → are informed by an event source when designated events occur, so that they can take appropriate action.
  - defines methods which accept a specific event type as argument
  - each listener interface extends the java.util.EventListener interface
- an event listener, which is interested in receiving events, is informed by an event source when certain types of events occur, so that it can take appropriate action
- all listeners of a particular event are notified, but the order in which they are notified is not necessarily the same as the order in which they were added as listeners
- Notification of all listeners is not guaranteed to occur in the same thread.
  - Access to any data shared between the listeners should be synchronized.
- same listener can be added to several event sources

Event Classes



Component	XEvent	XListener	Listener Methods
Button TextField MenuItem List	Action	Action	actionPerformed 🕮 if(e.getActionCommand() .equals("String")){}
Checkbox CheckboxMenuItem Choice	Item	Item	itemStateChanged 🕮 if(e.getItem().equals("String")){}

Window	Window	Window	windowActivated windowClosed windowClosing windowDeactivated windowDeactivated windowDeiconified windowIconified windowOpened
TextComponent	Text	Text	textValueChanged
Component	Component	Component	componentHidden componentMoved componentResized componentShown
	Focus	Focus	focusGained focusLost
Container	Container	Container	componentedAdded componentRemoved
[key]	Key	Key	keyPressed keyReleased keyTyped
[mouse]	Mouse	Mouse	mouseClicked mouseEntered mouseExiteded mousePresses mouseReleased
		MouseMotion	mouseDragged mouseMoved
Scrollbar	Adjustment	Adjustment	adjustmentValueChanged

EventObject	java.util.EventObject public class EventObject extends Object implements Serializable • the abstract root class from which all event state objects shall be derived. • EventObject(Object source) • Object getSource() → object on which the Event initially occurred.
ActionEvent	• clicked Button, double-clicked List, selected MenuItem, ENTER key hit in TextField • String getActionCommand() → button label, list-item name, menu-item name/text • int getModifiers() → SHIFT_MASK, CTRL_MASK, META_MASK, ALT_MASK
MouseEvent	java.awt.event.MouseEvent public class MouseEvent extends InputEvent • public static final int MOUSE_PRESSED, MOUSE_RELEASED, MOUSE_CLICKED, MOUSE_DRAGGED, MOUSE_MOVED, MOUSE_ENTERED, MOUSE_EXITED • int **getClickCount()** • int **getX()** • int **getY()**

Painting

- The code for drawing the component is usually implemented in the overriding method paint()
- AWT will automatically call paint() when
  - the size of the component has changed
  - the component has been uncovered
- The paint() method is seldom called directly
- the repaint() method is usually called by the application for screen updating
  - → invoke update()
    - → invoke paint()

Graphics	public **abstract** class Graphics extends Object
	java.awt.Graphics
	• abstract void clearRect(int x, int y, int width, int height)
	• abstract void clipRect(int x, int y, int width, int height)
	• abstract void copyArea(int x, int y, int width, int height, int dx, int dy)
	• abstract Graphics create()
	• Graphics create(int x, int y, int width, int height)
	• abstract void **dispose()** → Disposes of this graphics context and releases any system resources that it is using.
	• void draw3DRect(int x, int y, int width, int height, boolean raised)
	• abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
	• void drawBytes(byte[] data, int offset, int length, int x, int y)
	• void drawChars(char[] data, int offset, int length, int x, int y)
	• abstract boolean drawImage
	  o (Image img, int x, int y, Color bgcolor, ImageObserver observer)
	  o (Image img, int x, int y, ImageObserver observer)
	  o (Image img, int x, int y, int width, int height, Color bgcolor, ImageObserver observer)
	  o (Image img, int x, int y, int width, int height, ImageObserver observer)
	  o (Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, Color bgcolor, ImageObserver observer)
	  o (Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer)
	• abstract void **drawLine(int x1, int y1, int x2, int y2)**
	• abstract void **drawOval(int x, int y, int width, int height)**
	• abstract void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
	• void drawPolygon(Polygon p)
	• abstract void drawPolyline(int[] xPoints, int[] yPoints, int nPoints)
	• void **drawRect(int x, int y, int width, int height)**
	• abstract void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)
	• abstract void drawString(AttributedCharacterIterator iterator, int x, int y)
	• abstract void **drawString(String str, int x, int y)** → the baseline of the first character is at the specified coordinates
	• void fill3DRect(int x, int y, int width, int height, boolean raised)
	• abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)
	• abstract void **fillOval(int x, int y, int width, int height)**
	• abstract void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)
	• void fillPolygon(Polygon p)
	• abstract void **fillRect(int x, int y, int width, int height)**
	• abstract void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)
	• void finalize() → Disposes of this graphics context once it is no longer referenced.
	• abstract Shape getClip()
	• abstract Rectangle getClipBounds()
	• Rectangle getClipBounds(Rectangle r)
	• abstract Color **getColor()**
	• abstract Font **getFont()**
	• FontMetrics getFontMetrics()
	• abstract FontMetrics getFontMetrics(Font f)
	• boolean hitClip(int x, int y, int width, int height)
	• abstract void setClip(int x, int y, int width, int height)
	• abstract void setClip(Shape clip)
	• abstract void **setColor(Color c)**
	• abstract void **setFont(Font font)**
	• abstract void setPaintMode()
	• abstract void setXORMode(Color c1)
	• String toString()
	• abstract void translate(int x, int y)

- The Graphics object passed as argument to the paint() method is only temporarily made available, and should not be disposed of.

Event Adapter

- implements stubs for all the methods of the corresponding interface
- Low-level event listener interface: Xlistener
  Low-level event listener adapter: Xadapter
  X : Component, Container, Focus, Key, Mouse, MouseMotion, **Window**