# Swing GUI

# Java GUI History

- **Abstract Windowing Toolkit** (**AWT**): Sun's initial effort to create a set of cross-platform GUI classes. *(JDK 1.0 - 1.1)*
  - Maps general Java code to each operating system's real GUI system. Platform-dependent
  - It is heavy-weight in use because it is generated by the system's host operating system.

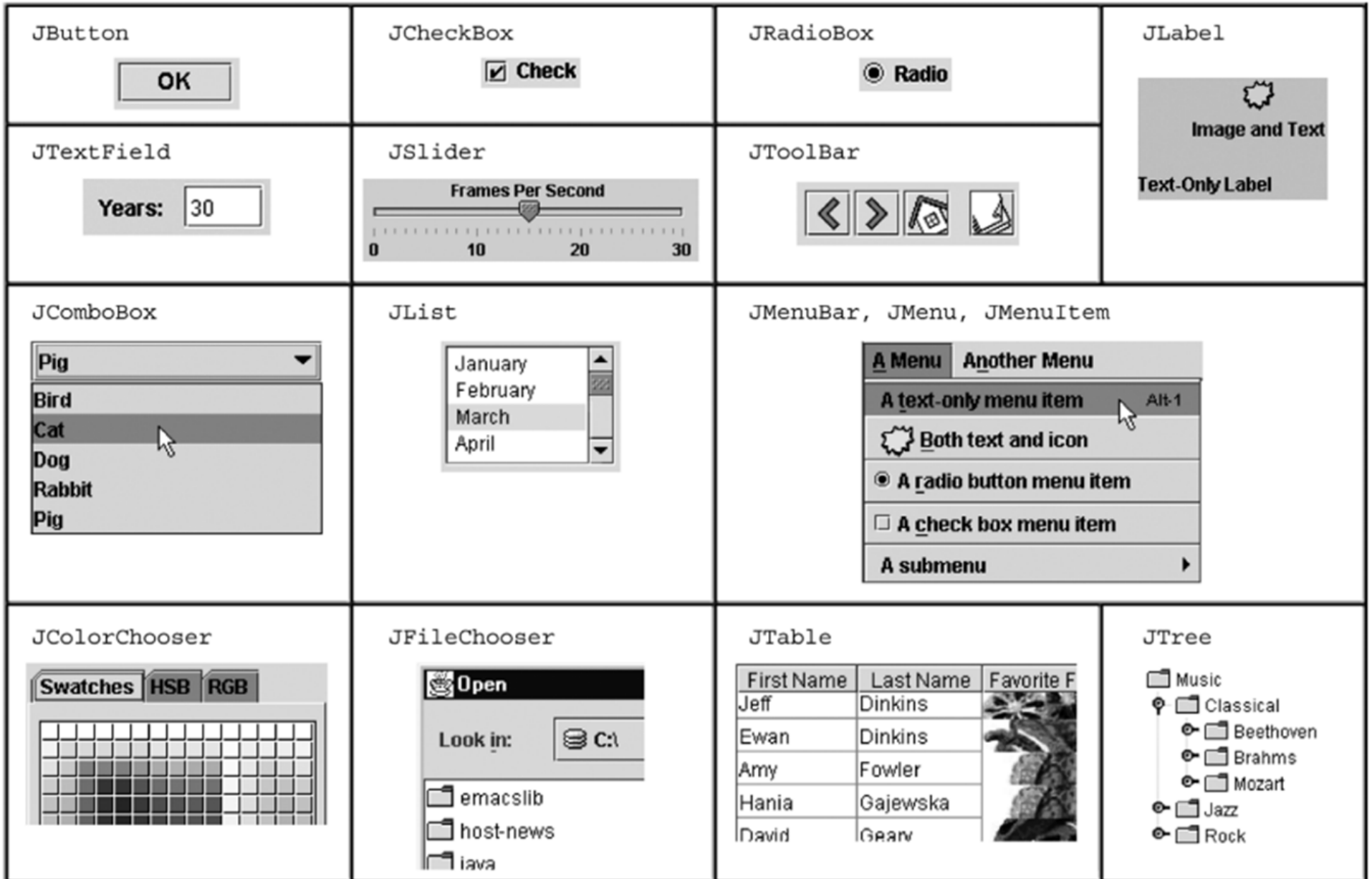
- **Swing**: A newer GUI library written from the ground up that allows much more powerful graphics and GUI construction. *(JDK 1.2+)*
  - Paints GUI controls itself pixel-by-pixel rather than handing off to OS. Platform-independent
  - It is lightweight Java graphical user interface (
  - *Benefits:* Features; compatibility; OO design.

  - *Problem:* Both exist in Java now; easy to get them mixed up; still have to use both in various places.

# GUI terminology

- **window**: A first-class citizen of the graphical desktop.
  - Also called a *top-level container*.
  - examples: frame, dialog box, applet
- **component**: A GUI widget that resides in a window.
  - Also called *controls* in many other languages.
  - examples: button, text box, label
- **container**: A logical grouping for storing components.
  - examples: panel, box

# Components


JButton
OK
JCheckBox
Check
JRadioBox
Radio
JLabel
Image and Text
Text-Only Label
JTextField
Years:
30
JSlider
Frames Per Second
0
10
20
30
JToolBar
JComboBox
Pig
Bird
Cat
Dog
Rabbit
Pig
JList
January
February
March
April
JMenuBar, JMenu, JMenuItem
A Menu
Another Menu
A text-only menu item
Alt-1
Both text and icon
A radio button menu item
A check box menu item
A submenu
JColorChooser
Swatches
HSB
RGB
JFileChooser
Open
Look in:
C:\
emacslib
host-news
JTable
First Name
Last Name
Favorite F
Jeff
Dinkins
Ewan
Dinkins
Amy
Fowler
Hania
Gajewska
David
Geary
JTree
Music
Classical
Beethoven
Brahms
Mozart
Jazz
Rock

# Swing inheritance hierarchy

```
import java.awt.*;
import javax.swing.*;
```

- `Component` (AWT)
  - `Window`
    - `Frame`
      - **`JFrame`** (Swing)
      - **`JDialog`**
  - `Container`
    - `JComponent` (Swing)
      - **`JButton`**
      - **`JComboBox`**
      - **`JMenuBar`**
      - **`JPopupMenu`**
      - **`JScrollPane`**
      - **`JSplitPane`**
      - **`JToolbar`**
      - **`JTextField`**
      - **`JColorChooser`**
      - **`JLabel`**
      - **`JOptionPane`**
      - **`JProgressBar`**
      - **`JSlider`**
      - **`JTabbedPane`**
      - **`JTree`**
      - `...`
      - **`JFileChooser`**
      - **`JList`**
      - **`JPanel`**
      - **`JScrollbar`**
      - **`JSpinner`**
      - **`JTable`**
      - **`JTextArea`**

# JFrame


*a graphical window to hold other components*

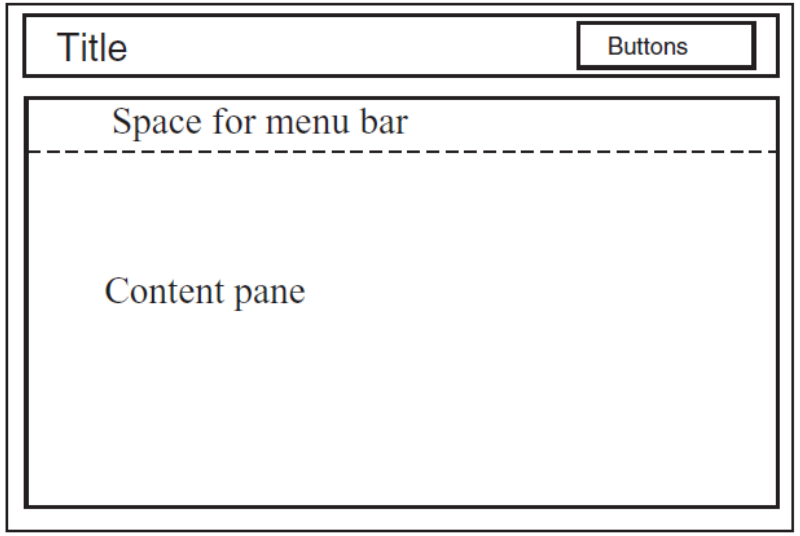
- `public JFrame()`
  `public JFrame(String title)`

  Creates a frame with an optional title.

- `setVisible(true)` makes a frame appear on the screen after creating it.
- `setTitle(String title)` sets the title appearing in the title bar to title.

- `public void add(Component comp)`

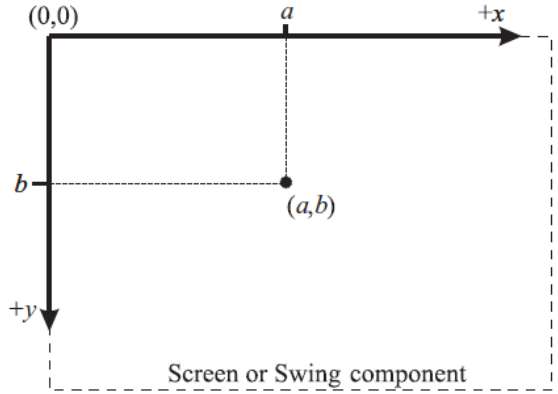
  Places the given component or container inside the frame.



# Jframe (cont.)

- `setDefaultCloseOperation(int op)`
  determines what happens if the ‘close’ button of the frame is clicked.
  - Common value passed: JFrame.EXIT_ON_CLOSE
  - If not set, the program will never exit even if the frame is closed.

- `setSize(int width, int height)`
  Gives the frame a fixed size in pixels.
- `setLocation(int horizontal, int vertical)` moves the frame, so that its upper left corner is at position (horizontal,vertical)
- `pack()` resizes the frame so that it tightly fits around components embedded its content pane.



# Simple Jframe Example

```
import javax.swing.JFrame;

public class SimpleFrame extends JFrame
{
  public SimpleFrame()
  {
     this.setSize(200,200);
     this.setLocation(200,200);
     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }

  // Makes the frame visible.
  public void showIt(){
    this.setVisible(true);
  }

}
```

# Simple Jframe Example (cont.)

```
// Makes the frame visible and sets the title text.
  public void showIt(String title){
    this.setTitle(title);
    this.setVisible(true);
  }

  // Makes the frame visible and sets the title text and the position of the window.

  public void showIt(String title,int x, int y){
    this.setTitle(title);
    this.setLocation(x,y);
    this.setVisible(true);
  }

  // Makes the frame invisible.
  public void hideIt(){
    this.setVisible(false);
  }
```

# Simple Jframe Example (cont.)

```
public class SimpleFrameDriver
{
 public static void main(String[] args)
  {
    SimpleFrame sFrame1 = new SimpleFrame();
    SimpleFrame sFrame2 = new SimpleFrame();
    sFrame1.showIt("SimpleFrame 1");
    sFrame2.showIt("SimpleFrame 2",300,300);
  }
}
```

# JFrame as container

A `JFrame` is a container. Containers have these methods:

- `public void add(Component comp)`
  `public void add(Component comp, Object info)`
  Adds a component to the container, possibly giving extra information about where to place it.

- `public void remove(Component comp)`

- `public void setLayout(LayoutManager mgr)`
  Uses the given layout manager to position components.

- `public void validate()`
  Refreshes the layout (if it changes after the container is onscreen).

# JPanel

The default container class in Swing. A rectangular component, which serves two main purposes: it can be used as a canvas that one draws on or it can be used as containers to embed further graphical components.

- `JPanel()` The size of panel is set to default values, 10 by 10 pixel on most systems.
  `JPanel(LayoutManager mgr)`
  Constructs a panel with the given layout (default = flow layout).
- `setBackground(Color c)` sets the background colour of the panel. The class `Color` is from the AWT library. There, some colours are predefined, e.g. red by `Color.red`. By default the background colour is grey.
- `setPreferredSize(Dimension d)` sets the size of the panel.
  - The class `Dimension` is from the AWT library. The constructor has the syntax `Dimension(int width, int height)`. Both `width` and `height` are in pixels.

# Simple Jpanel Example

```
import java.awt.*;
import javax.swing.JPanel;
public class ColorPanel extends JPanel
{
  // Generate a JPanel with background color col .
  public ColorPanel(Color col)
  {
    this.setBackground(col);
  }

  // Generate a JPanel with background color col, width width, and height height .
  public ColorPanel(Color col,int width,int height)
  {
    this.setPreferredSize(new Dimension(width,height));
    this.setBackground(col);
  }

}
```

# Simple Jpanel Example (cont.)

```
public class SimplePanelFrame extends SimpleFrame
{
  public SimplePanelFrame()
  {
    ColorPanel CPWest = new ColorPanel(Color.white,50,20);
    ColorPanel CPEast = new ColorPanel(Color.red);
    ColorPanel CPNorth = new ColorPanel(Color.yellow);
    ColorPanel CPSouth = new ColorPanel(Color.green);
    ColorPanel CPCenter = new ColorPanel(Color.blue);
    this.getContentPane().add(CPWest,BorderLayout.WEST);
    this.getContentPane().add(CPEast,BorderLayout.EAST);
    this.getContentPane().add(CPNorth,BorderLayout.NORTH);
    this.getContentPane().add(CPSouth,BorderLayout.SOUTH);
    this.getContentPane().add(CPCenter,BorderLayout.CENTER);
  }
}
```

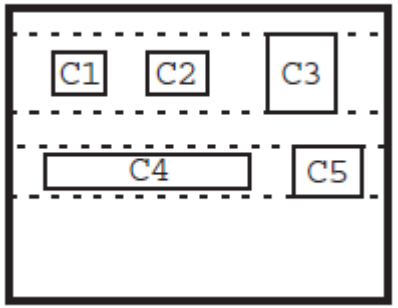
# Simple Jpanel Example (cont.)

```
public class SimplePanelFrameDriver
{
  public static void main(String[] args)
  {
    SimplePanelFrame spFrame = new SimplePanelFrame();
    spFrame.showIt("Simple Panel Frame");
  }
}
```

# FlowLayout

`public FlowLayout()`

- The default layout for containers other than JFrame
- treats container as a left-to-right, top-to-bottom "paragraph".
  - Components are given preferred size, horizontally and vertically.
  - Components are positioned in the order added.
  - If too long, components wrap around to the next line

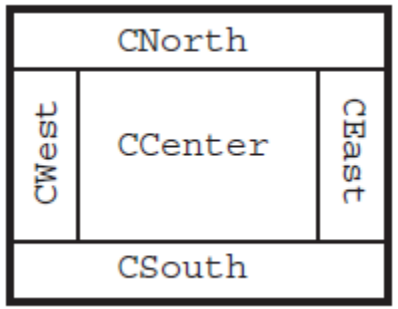



`FlowLayout()` generates a layout which by default has five pixels between the components in a row and five pixels between the rows.

`Flowlayout(int align)` specifies the alignment of the components, where `align` is one of `FlowLayout.RIGHT, FlowLayout.LEFT, FlowLayout.CENTER`

`Flowlayout(int align, int hdist, int vdist)` specifies the horizontal distance `hdist` between the components in a row and the vertical distance `vdist` between rows.

# BorderLayout

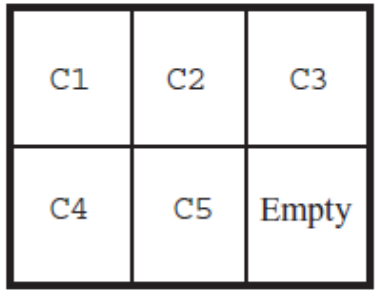



```
public BorderLayout()
```

- This is the default layout for a JFrame.
- Divides container into five regions:
  - NORTH and SOUTH regions expand to fill region horizontally, and use the component's preferred size vertically.
  - WEST and EAST regions expand to fill region vertically, and use the component's preferred size horizontally.
  - CENTER uses all space not occupied by others.

```
myFrame.setLayout(new BorderLayout());
```

# GridLayout




```
public GridLayout(int rows, int columns)
```

- Treats container as a grid of equally-sized rows and columns.
- Components are given equal horizontal / vertical size, disregarding preferred size.
- Can specify 0 rows or columns to indicate expansion in that direction as needed

- `setHgap(int hdist)`
- `setVgap(int vdist)`

The width of the gaps can be passed between the columns (`hdist`) and rows (`vdist`):

# Layout Example

```
import java.awt.LayoutManager;
import its.SimpleFrameWithPanels.ColorPanel;
import java.awt.Color;
public class LayoutFrame extends SimpleFrame
{
  public LayoutFrame(LayoutManager layout)
  {
    this.getContentPane().setLayout(layout);

    ColorPanel CP1 = new ColorPanel(Color.red,30,30);
    ColorPanel CP2 = new ColorPanel(Color.yellow,40,20);
    ColorPanel CP3 = new ColorPanel(Color.green);
    ColorPanel CP4 = new ColorPanel(Color.blue);
    ColorPanel CP5 = new ColorPanel(Color.white,80,20);
    this.getContentPane().add(CP1);
    this.getContentPane().add(CP2);
    this.getContentPane().add(CP3);
    this.getContentPane().add(CP4);
    this.getContentPane().add(CP5);
  }
}
```

# Layout Example (cont.)

```
import java.awt.FlowLayout;
import java.awt.GridLayout;
public class LayoutDriver
{
  public static void main(String[] args)
  {
    FlowLayout  flowLayout1 = new FlowLayout();
    LayoutFrame flow1Frame  = new LayoutFrame(flowLayout1);
    flow1Frame.showIt("Flow Layout 1",60,60);

    FlowLayout  flowLayout2 = new FlowLayout(FlowLayout.LEFT,40,30);
    LayoutFrame flow2Frame  = new LayoutFrame(flowLayout2);
    flow2Frame.showIt("Flow Layout 2",300,60);

    GridLayout  gridLayout  = new GridLayout(2,4);
    LayoutFrame gridFrame  = new LayoutFrame(gridLayout);
    gridFrame.showIt("Grid Layout",540,60);
  }
```