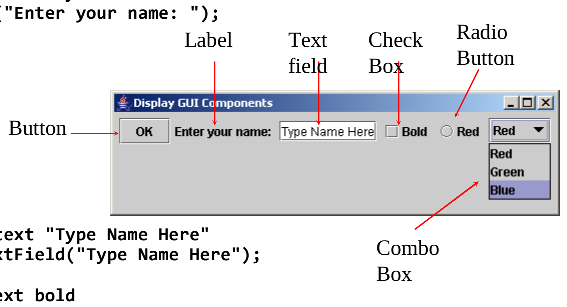
Chapter 12 GUI Basics

- API for Java GUI programming is an excellent example of object-oriented principles applied
- In the chapters that follow, you will learn the framework of Java GUI API and use the GUI components to develop user-friendly interfaces for applications and applets
- AWT is Abstract Windows Toolkit
  - Considered Heavy weight due to reliance on OS
  - Limited capability but provides foundation
- Swing
  - Lightweight with Similar rendering on all OS's
  - Swing built upon AWT classes and does not replace
- JavaFX is the next step in GUI Tookits

Creating GUI Objects

```
// Create a button with text OK
JButton jbtOK = new JButton("OK");

// Create a label with text "Enter your name: "
JLabel jlblName = new JLabel("Enter your name: ");
```

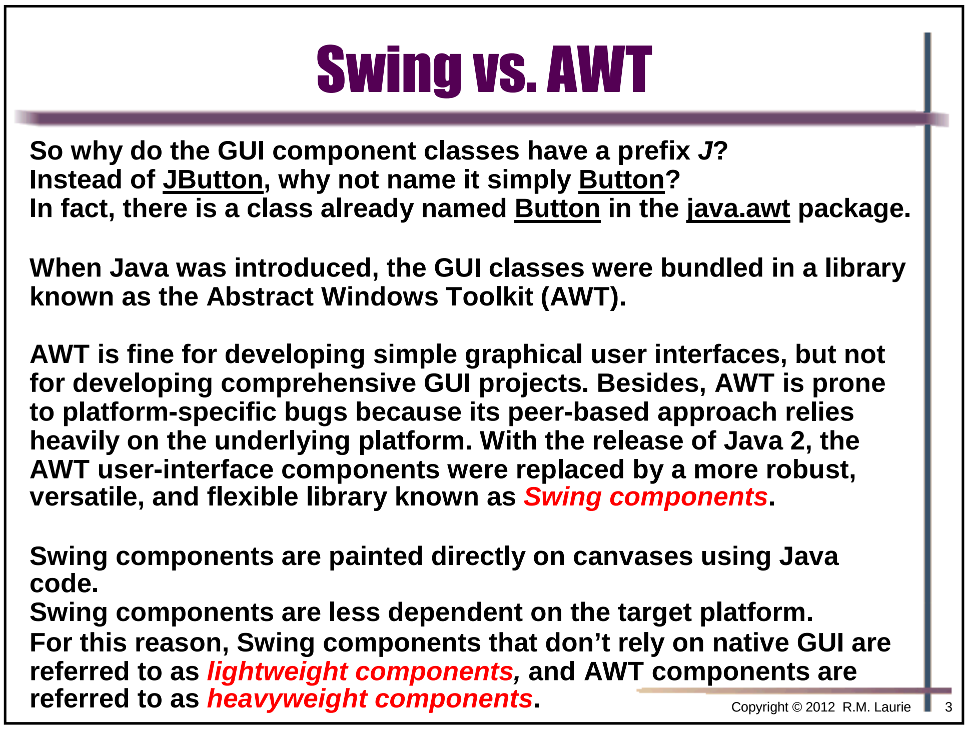


```
// Create a text field with text "Type Name Here"
JTextField jtfName = new JTextField("Type Name Here");

// Create a check box with text bold
JCheckBox jchkBold = new JCheckBox("Bold");

// Create a radio button with text red
JRadioButton jrbRed = new JRadioButton("Red");

// Create a combo box with choices red, green, and blue
JComboBox jcboColor = new JComboBox(new String[]{"Red",
  "Green", "Blue"});
```



Swing vs. AWT

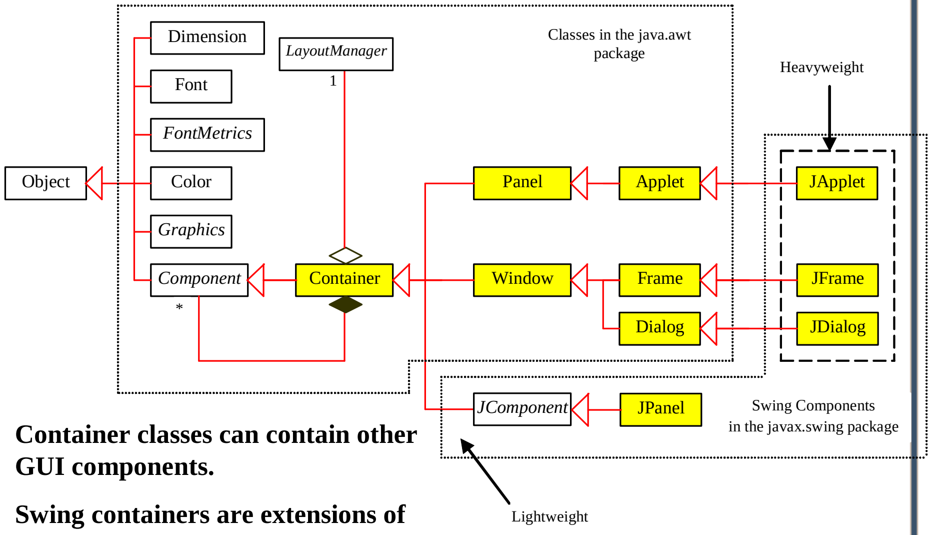
So why do the GUI component classes have a prefix *J*?
Instead of JButton, why not name it simply Button?
In fact, there is a class already named Button in the java.awt package.

When Java was introduced, the GUI classes were bundled in a library known as the Abstract Windows Toolkit (AWT).

AWT is fine for developing simple graphical user interfaces, but not for developing comprehensive GUI projects. Besides, AWT is prone to platform-specific bugs because its peer-based approach relies heavily on the underlying platform. With the release of Java 2, the AWT user-interface components were replaced by a more robust, versatile, and flexible library known as *Swing components*.

Swing components are painted directly on canvases using Java code.
Swing components are less dependent on the target platform.
For this reason, Swing components that don't rely on native GUI are referred to as *lightweight components*, and AWT components are referred to as *heavyweight components*.

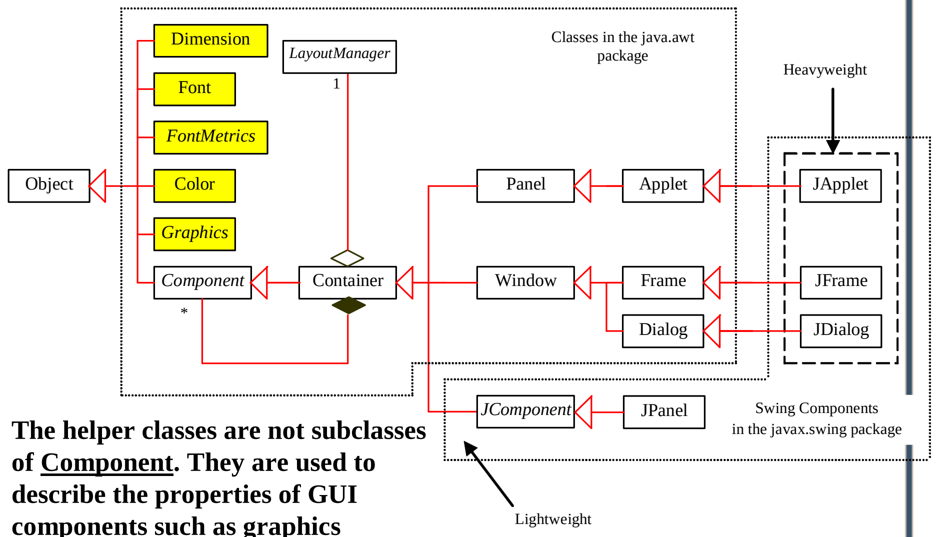
Swing GUI Container Classes



Container classes can contain other GUI components.

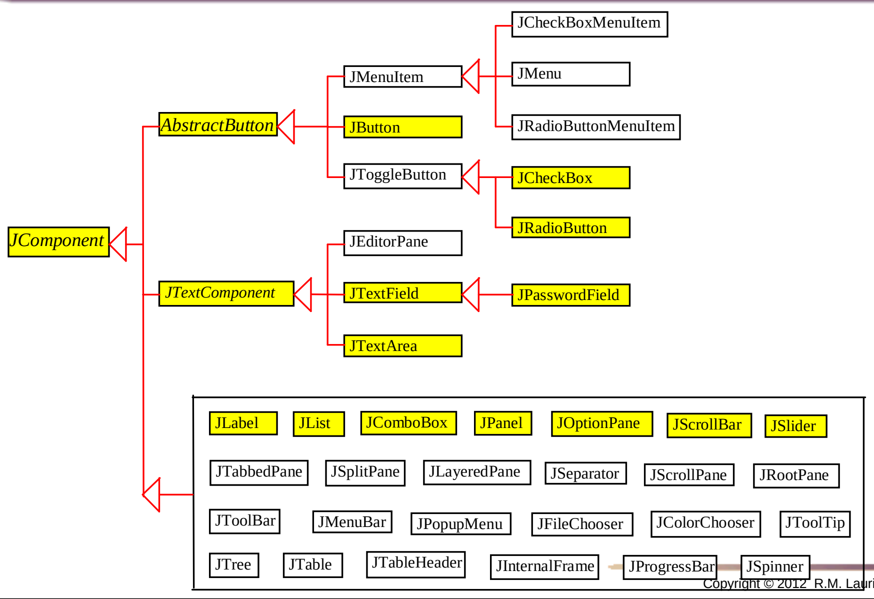
Swing containers are extensions of Abstract Windows Toolkit

GUI Helper Classes



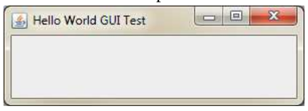
The helper classes are not subclasses of <u>Component</u>. They are used to describe the properties of GUI components such as graphics context, colors, fonts, and dimension.

JComponents Covered in this Class



JFrame is a container for GUI components

javax.swing.JFrame	
+JFrame()	Creates a default frame with no title.
+JFrame(title: String)	Creates a frame with the specified title.
+setSize(width: int, height: int): void	Specifies the size of the frame.
+setLocation(x: int, y: int): void	Specifies the upper-left corner location of the frame.
+setVisible(visible: boolean): void	Sets true to display the frame.
+setDefaultCloseOperation(mode: int): void	Specifies the operation when the frame is closed.
+setLocationRelativeTo(c: Component): void	Sets the location of the frame relative to the specified component. If the component is null, the frame is centered on the screen.
+pack(): void	Automatically sets the frame size to hold the components in the frame.



```java
import javax.swing.*;
public class HelloWorldGUI {
  public static void main(String[] args) {
    JFrame fraHello = new JFrame("Hello World GUI Test");
    fraHello.setSize(300, 100);  // Size in Pixels, do first
    fraHello.setLocationRelativeTo(null); // Do second
    fraHello.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraHello.setVisible(true);   // Show Window
  }
}
```

JComponent ← JLabel for Text or Image

17.5: JLabel is a display area for short text, image, or both

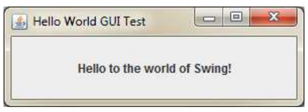
javax.swing.JComponent

The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

javax.swing.JLabel	
-text: String	The label's text.
-icon: javax.swing.Icon	The label's image icon.
-horizontalAlignment: int	The horizontal alignment of the text and icon on the label.
-horizontalTextPosition: int	The horizontal text position relative to the icon on the label.
-verticalAlignment: int	The vertical alignment of the text and icon on the label.
-verticalTextPosition: int	The vertical text position relative to the icon on the label.
-iconTextGap: int	The gap between the text and the icon on the label (JDK 1.4).
+JLabel()	Creates a default label with no text and icon.
+JLabel(icon: javax.swing.Icon)	Creates a label with an icon.
+JLabel(icon: Icon, hAlignment: int)	Creates a label with an icon and the specified horizontal alignment.
+JLabel(text: String)	Creates a label with text.
+JLabel(text: String, icon: Icon, hAlignment: int)	Creates a label with text, an icon, and the specified horizontal alignment.
+JLabel(text: String, hAlignment: int)	Creates a label with text and the specified horizontal alignment.

Adding JLabel JComponent to Frame

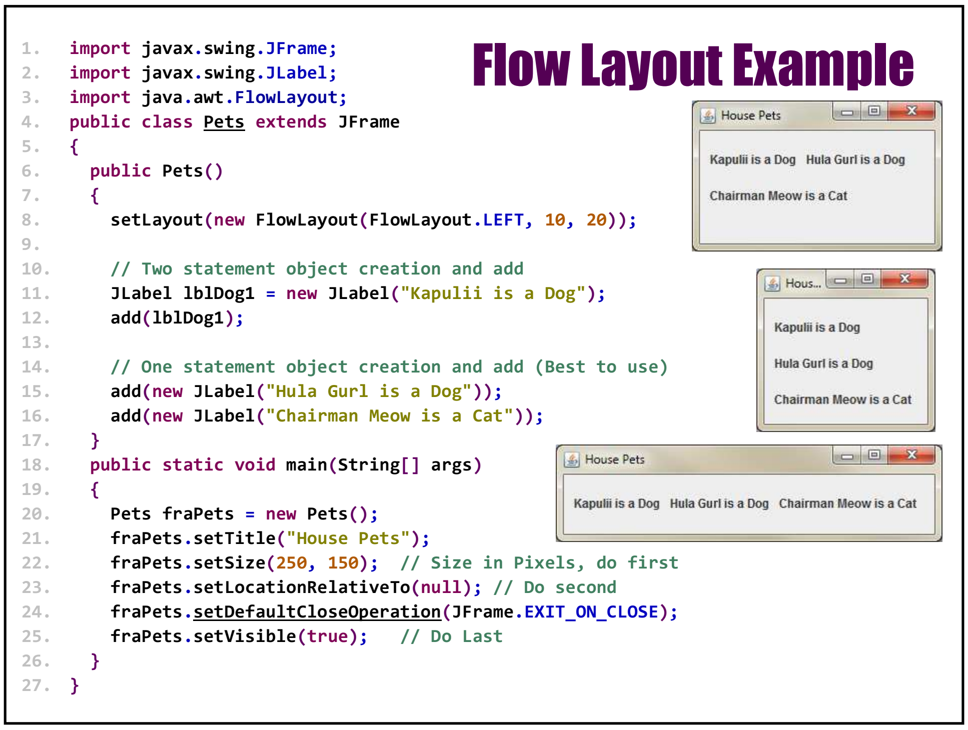
```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
public class HelloWorldGUI
{
  public static void main(String[] args)
  {
    JFrame fraHello = new JFrame("Hello World GUI Test");
    JLabel lblHello = new JLabel("Hello to the world of Swing!");
    lblHello.setHorizontalAlignment(SwingConstants.CENTER);
    fraHello.add(lblHello);
    fraHello.setSize(300, 100);  // Size in Pixels, do first
    fraHello.setLocationRelativeTo(null); // Do second
    fraHello.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraHello.setVisible(true);   // Show Window
  }
}
```



Hello to the world of Swing!

Layout Managers

- Placing multiple JComponent objects such as JLabel objects in a frame will result in a stack with only the last object displayed on top
- Java's layout managers provide a level of abstraction to map Components on all OS systems into a 2D layout
  - JFrame can utilize a layout manager to arrange multiple JComponents objects within the container
  - AWT Basic Layout managers
    - Extends JFrame using setLayout(LayoutManager) method
    - Three layouts available from AWT
      - FlowLayout
      - GridLayout
      - BorderLayout



Flow Layout Example

```java
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.FlowLayout;
public class Pets extends JFrame
{
  public Pets()
  {
    setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));

    // Two statement object creation and add
    JLabel lblDog1 = new JLabel("Kapulii is a Dog");
    add(lblDog1);

    // One statement object creation and add (Best to use)
    add(new JLabel("Hula Gurl is a Dog"));
    add(new JLabel("Chairman Meow is a Cat"));
  }
  public static void main(String[] args)
  {
    Pets fraPets = new Pets();
    fraPets.setTitle("House Pets");
    fraPets.setSize(250, 150);  // Size in Pixels, do first
    fraPets.setLocationRelativeTo(null); // Do second
    fraPets.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraPets.setVisible(true);   // Do Last
  }
}
```

House Pets

```
Kapulii is a Dog   Hula Gurl is a Dog

Chairman Meow is a Cat
```

Hous...

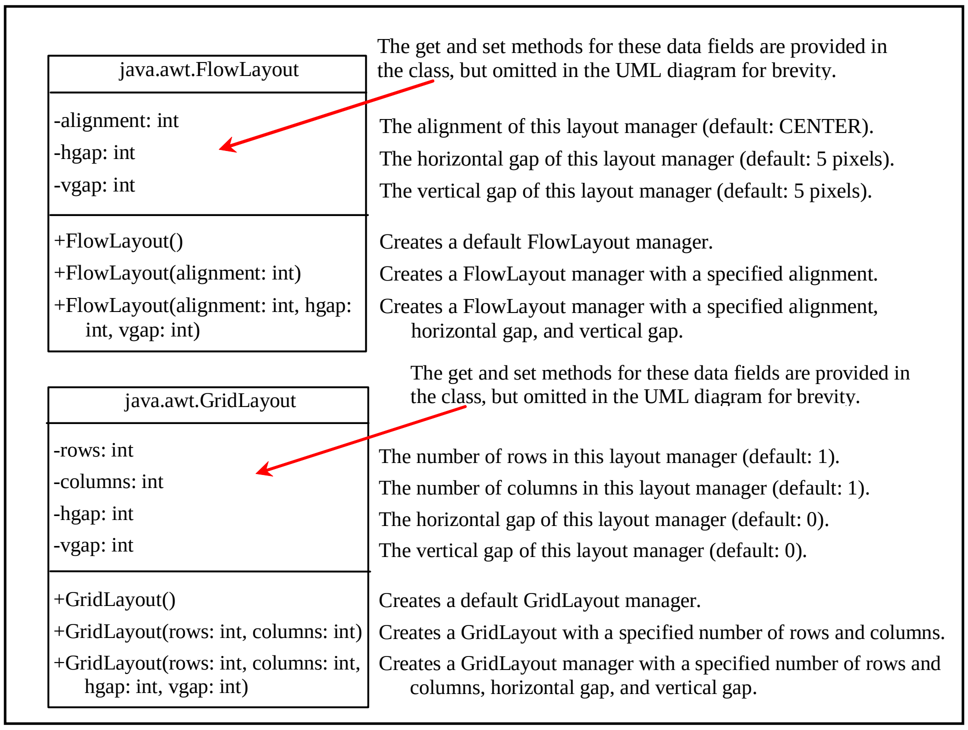
```
Kapulii is a Dog

Hula Gurl is a Dog

Chairman Meow is a Cat
```

House Pets

```
Kapulii is a Dog   Hula Gurl is a Dog   Chairman Meow is a Cat
```



java.awt.FlowLayout

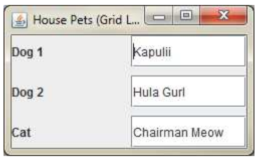
	The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.
-alignment: int	The alignment of this layout manager (default: CENTER).
-hgap: int	The horizontal gap of this layout manager (default: 5 pixels).
-vgap: int	The vertical gap of this layout manager (default: 5 pixels).
+FlowLayout()	Creates a default FlowLayout manager.
+FlowLayout(alignment: int)	Creates a FlowLayout manager with a specified alignment.
+FlowLayout(alignment: int, hgap: int, vgap: int)	Creates a FlowLayout manager with a specified alignment, horizontal gap, and vertical gap.

java.awt.GridLayout

	The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.
-rows: int	The number of rows in this layout manager (default: 1).
-columns: int	The number of columns in this layout manager (default: 1).
-hgap: int	The horizontal gap of this layout manager (default: 0).
-vgap: int	The vertical gap of this layout manager (default: 0).
+GridLayout()	Creates a default GridLayout manager.
+GridLayout(rows: int, columns: int)	Creates a GridLayout with a specified number of rows and columns.
+GridLayout(rows: int, columns: int, hgap: int, vgap: int)	Creates a GridLayout manager with a specified number of rows and columns, horizontal gap, and vertical gap.

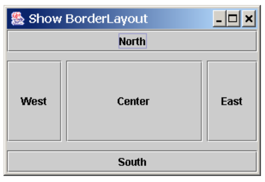
Grid Layout Example

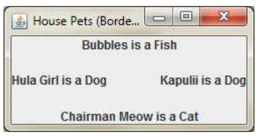
```java
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTestField;
import java.awt.GridLayout;
public class PetsGrid extends JFrame
{
  public PetsGrid()
  {
    setLayout(new GridLayout(3, 2, 4, 8));
    add(new JLabel("Dog 1"));
    add(new JTextField("Kapulii", 20));
    add(new JLabel("Dog 2"));
    add(new JTextField("Hula Gurl"));
    add(new JLabel("Cat"));
    add(new JTextField("Chairman Meow"));
  }
  public static void main(String[] args)
  {
    PetsGrid fraPets = new PetsGrid();
    fraPets.setTitle("House Pets (Grid Layout)");
    fraPets.setSize(250, 150);  // Size in Pixels, do first
    fraPets.setLocationRelativeTo(null); // Do second
    fraPets.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraPets.setVisible(true);   // Do Last
  }
}
```

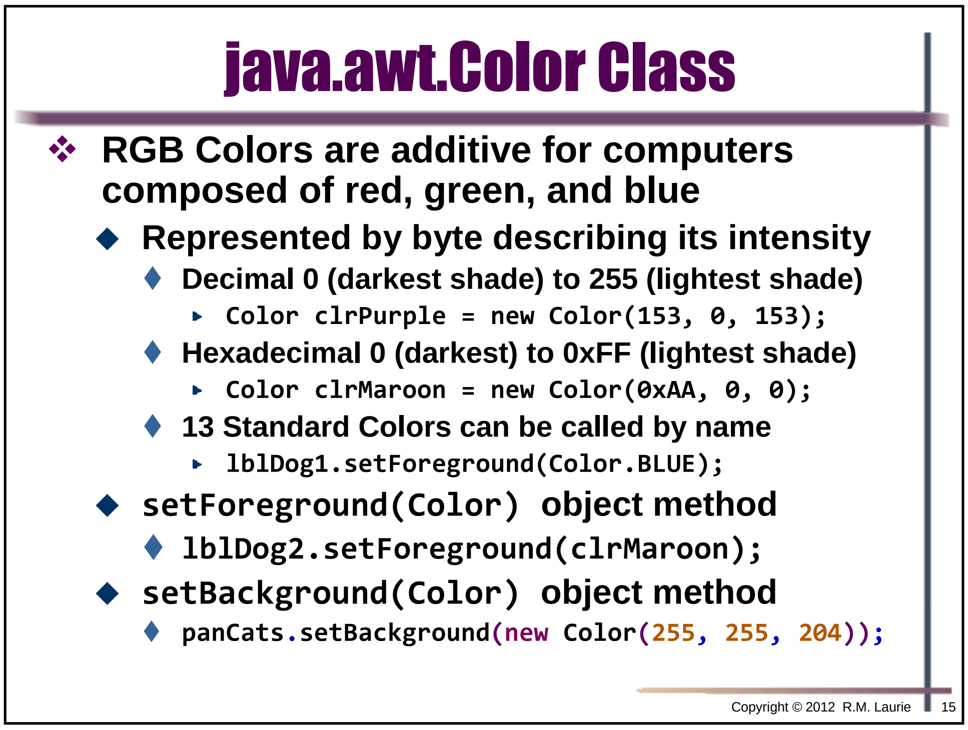


Border Layout Example

```java
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.BorderLayout;
public class PetsBorder extends JFrame
{
  public PetsBorder()
  {
    setLayout(new BorderLayout(5, 10));
    JLabel lblFish = new JLabel("Bubbles is a Fish");
    lblFish.setHorizontalAlignment(SwingConstants.CENTER);
    add(lblFish, BorderLayout.NORTH);
    add(new JLabel("Kapulii is a Dog"), BorderLayout.EAST);
    add(new JLabel("Hula Girl is a Dog"), BorderLayout.WEST);
    JLabel lblCat = new JLabel("Chairman Meow is a Cat");
    lblCat.setHorizontalAlignment(SwingConstants.CENTER);
    add(lblCat, BorderLayout.SOUTH);
  }
  public static void main(String[] args)
  {
    PetsBorder fraPets = new PetsBorder();
    fraPets.setTitle("House Pets (Border Layout)");
    fraPets.setSize(250, 125);  // Size in Pixels, do first
    fraPets.setLocationRelativeTo(null); // Do second
    fraPets.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraPets.setVisible(true);   // Do Last
  }
}
```



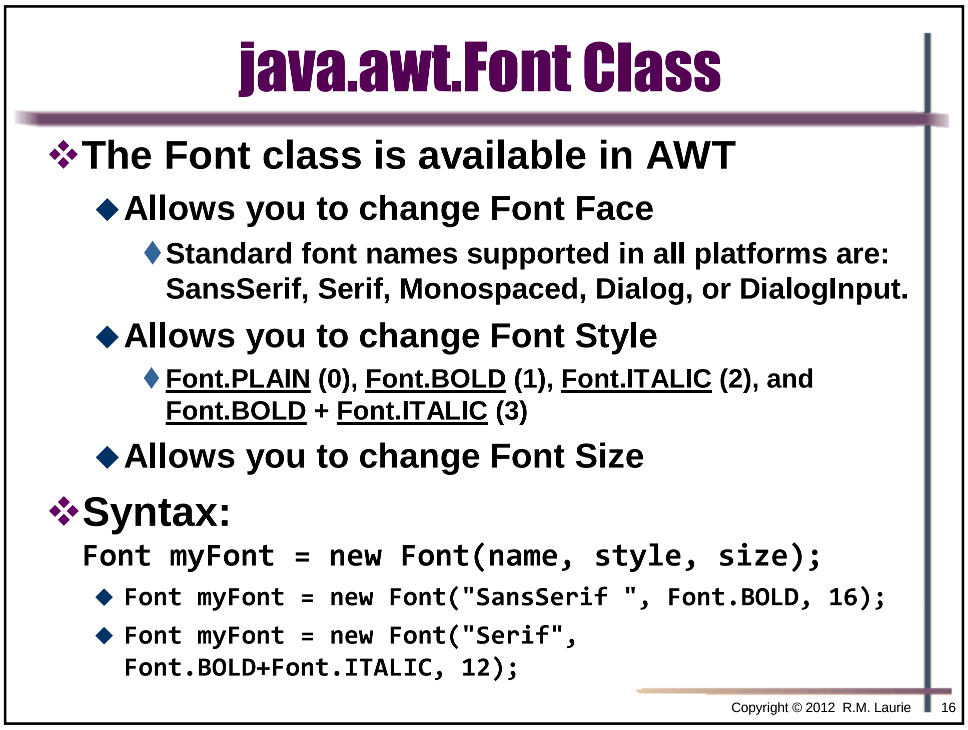




java.awt.Color Class

- RGB Colors are additive for computers composed of red, green, and blue
  - Represented by byte describing its intensity
    - Decimal 0 (darkest shade) to 255 (lightest shade)
      - `Color clrPurple = new Color(153, 0, 153);`
    - Hexadecimal 0 (darkest) to 0xFF (lightest shade)
      - `Color clrMaroon = new Color(0xAA, 0, 0);`
    - 13 Standard Colors can be called by name
      - `lblDog1.setForeground(Color.BLUE);`
  - `setForeground(Color)` object method
    - `lblDog2.setForeground(clrMaroon);`
  - `setBackground(Color)` object method
    - `panCats.setBackground(new Color(255, 255, 204));`

Copyright © 2012 R.M. Laurie 15



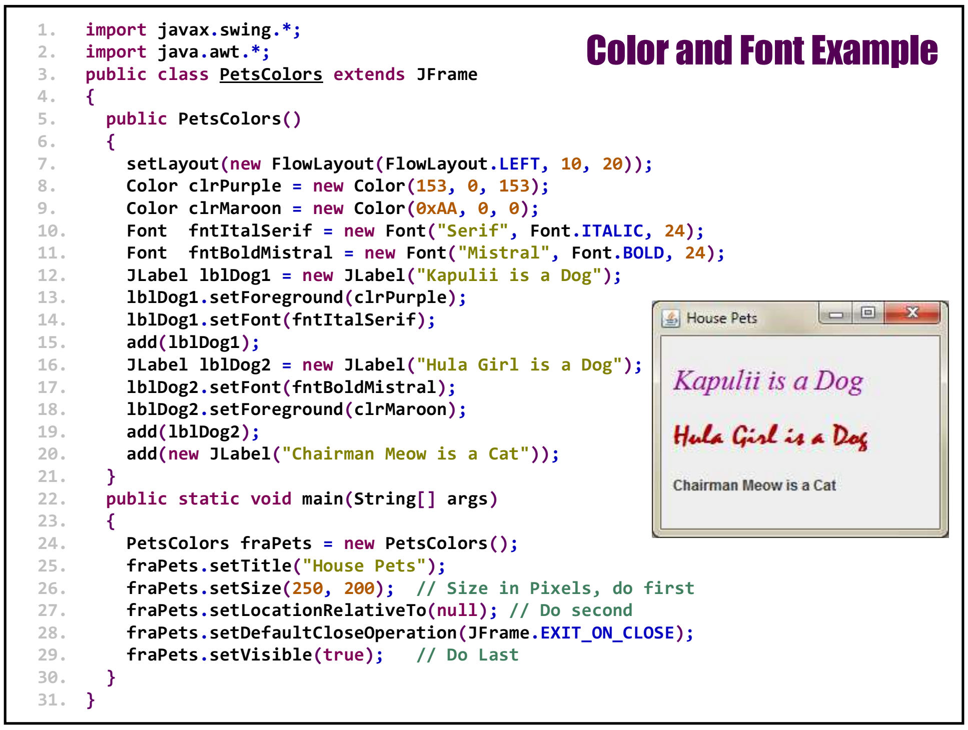
java.awt.Font Class

- The Font class is available in AWT
  - Allows you to change Font Face
    - Standard font names supported in all platforms are: SansSerif, Serif, Monospaced, Dialog, or DialogInput.
  - Allows you to change Font Style
    - Font.PLAIN (0), Font.BOLD (1), Font.ITALIC (2), and Font.BOLD + Font.ITALIC (3)
  - Allows you to change Font Size
- Syntax:

```
Font myFont = new Font(name, style, size);
```

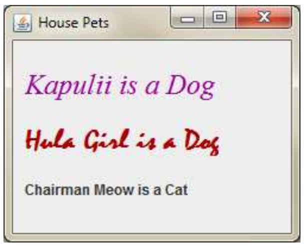
  - `Font myFont = new Font("SansSerif ", Font.BOLD, 16);`
  - `Font myFont = new Font("Serif", Font.BOLD+Font.ITALIC, 12);`

Copyright © 2012 R.M. Laurie 16



Color and Font Example

```
import javax.swing.*;
import java.awt.*;
public class PetsColors extends JFrame
{
  public PetsColors()
  {
    setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));
    Color clrPurple = new Color(153, 0, 153);
    Color clrMaroon = new Color(0xAA, 0, 0);
    Font  fntItalSerif = new Font("Serif", Font.ITALIC, 24);
    Font  fntBoldMistral = new Font("Mistral", Font.BOLD, 24);
    JLabel lblDog1 = new JLabel("Kapulii is a Dog");
    lblDog1.setForeground(clrPurple);
    lblDog1.setFont(fntItalSerif);
    add(lblDog1);
    JLabel lblDog2 = new JLabel("Hula Girl is a Dog");
    lblDog2.setFont(fntBoldMistral);
    lblDog2.setForeground(clrMaroon);
    add(lblDog2);
    add(new JLabel("Chairman Meow is a Cat"));
  }
  public static void main(String[] args)
  {
    PetsColors fraPets = new PetsColors();
    fraPets.setTitle("House Pets");
    fraPets.setSize(250, 200);  // Size in Pixels, do first
    fraPets.setLocationRelativeTo(null); // Do second
    fraPets.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraPets.setVisible(true);   // Do Last
  }
}
```

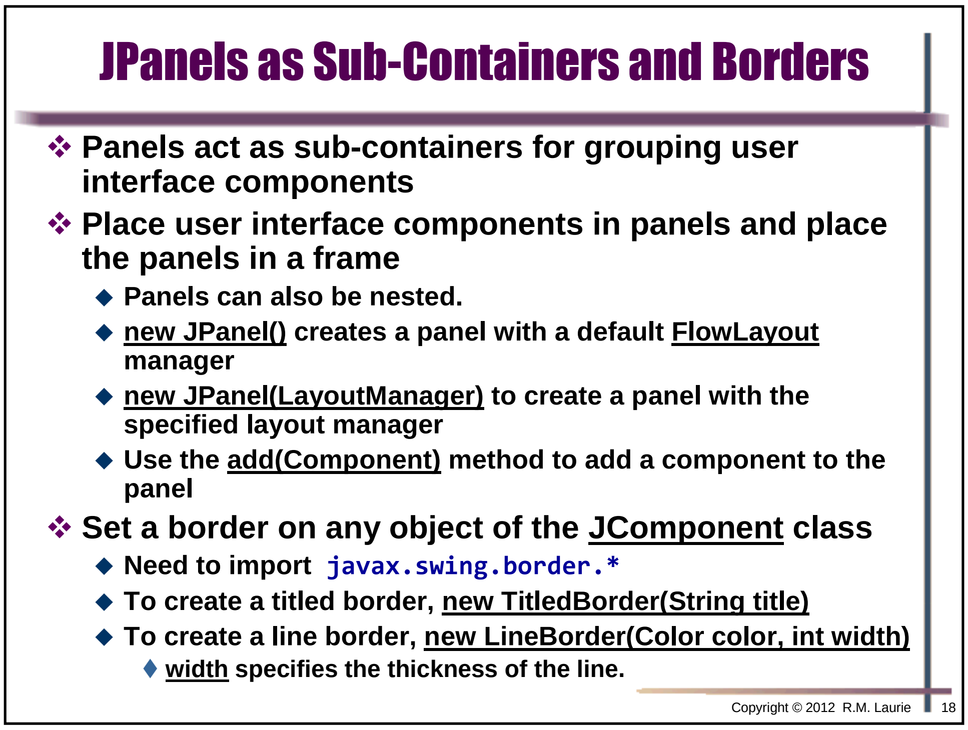


House Pets

Kapulii is a Dog

Hula Girl is a Dog

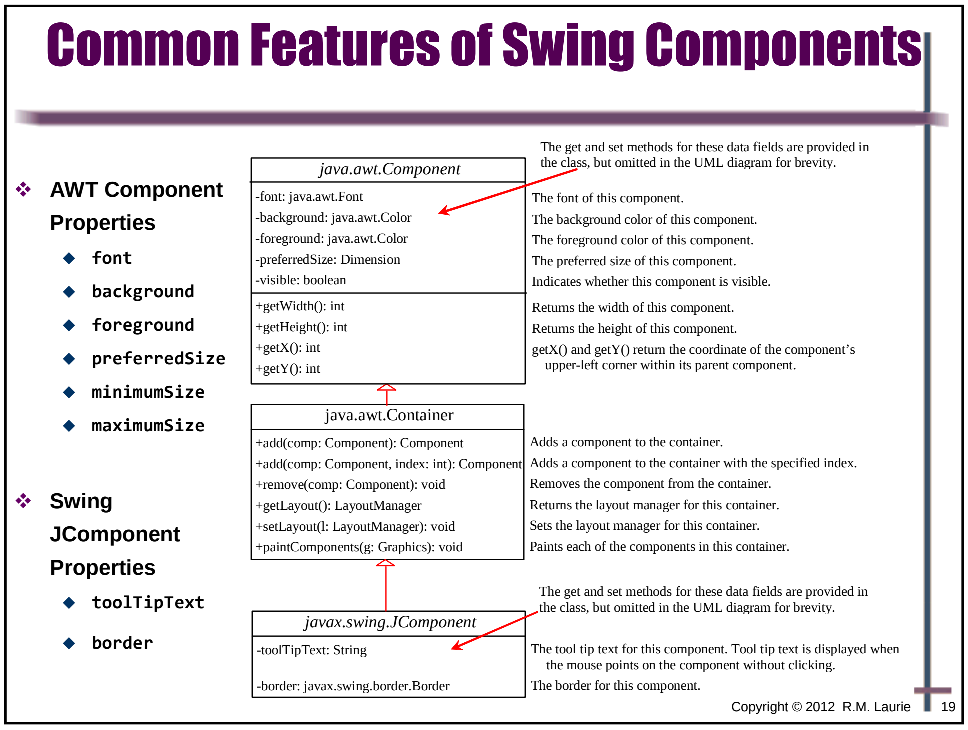
Chairman Meow is a Cat



JPanels as Sub-Containers and Borders

- Panels act as sub-containers for grouping user interface components
- Place user interface components in panels and place the panels in a frame
  - Panels can also be nested.
  - new JPanel() creates a panel with a default FlowLayout manager
  - new JPanel(LayoutManager) to create a panel with the specified layout manager
  - Use the add(Component) method to add a component to the panel
- Set a border on any object of the JComponent class
  - Need to import `javax.swing.border.*`
  - To create a titled border, new TitledBorder(String title)
  - To create a line border, new LineBorder(Color color, int width)
    - width specifies the thickness of the line.

Copyright © 2012 R.M. Laurie 18



Common Features of Swing Components

- AWT Component Properties
  - font
  - background
  - foreground
  - preferredSize
  - minimumSize
  - maximumSize
- Swing JComponent Properties
  - toolTipText
  - border

java.awt.Component

-font: java.awt.Font
-background: java.awt.Color
-foreground: java.awt.Color
-preferredSize: Dimension
-visible: boolean

+getWidth(): int
+getHeight(): int
+getX(): int
+getY(): int

The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

The font of this component.
The background color of this component.
The foreground color of this component.
The preferred size of this component.
Indicates whether this component is visible.
Returns the width of this component.
Returns the height of this component.
getX() and getY() return the coordinate of the component's upper-left corner within its parent component.

java.awt.Container

+add(comp: Component): Component
+add(comp: Component, index: int): Component
+remove(comp: Component): void
+getLayout(): LayoutManager
+setLayout(l: LayoutManager): void
+paintComponents(g: Graphics): void

Adds a component to the container.
Adds a component to the container with the specified index.
Removes the component from the container.
Returns the layout manager for this container.
Sets the layout manager for this container.
Paints each of the components in this container.

javax.swing.JComponent

-toolTipText: String
-border: javax.swing.border.Border

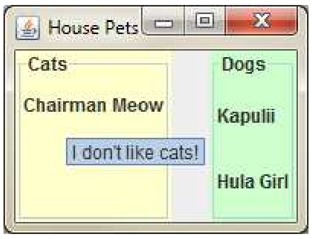
The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

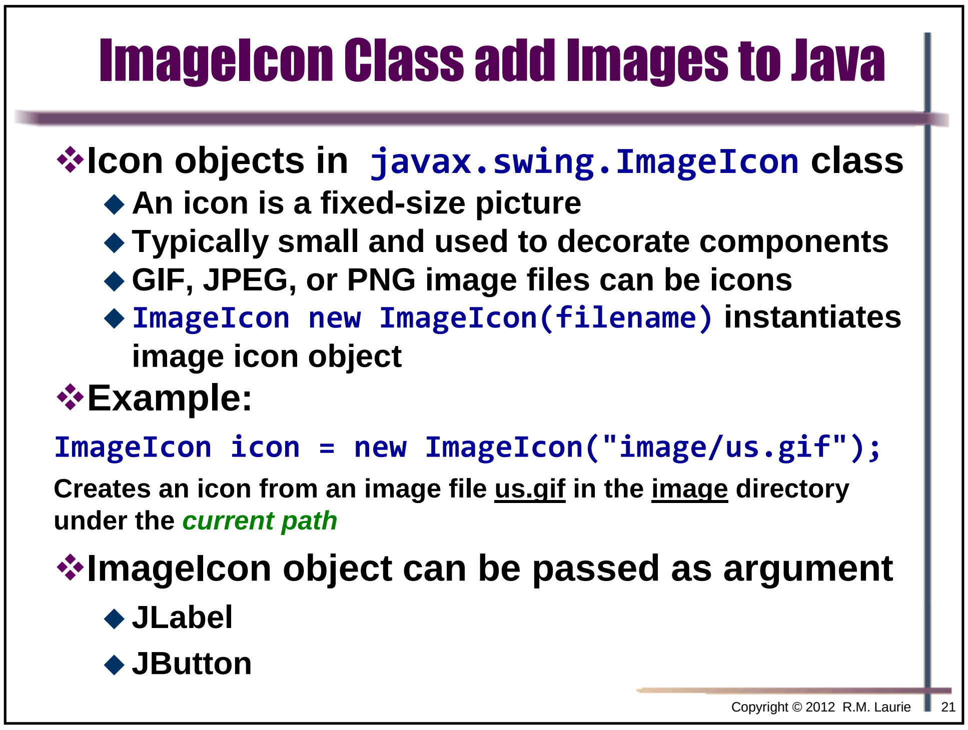
The tool tip text for this component. Tool tip text is displayed when the mouse points on the component without clicking.
The border for this component.

Copyright © 2012 R.M. Laurie 19

Panel and Border Example

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
public class TestPanels extends JFrame {
  public TestPanels() {
    Color clrPaleGreen = new Color(204, 255, 204);
    Color clrPaleYellow = new Color(255, 255, 204);
    JPanel panDogs = new JPanel();
    panDogs.setLayout(new GridLayout(2,1));
    panDogs.setBorder(new TitledBorder("Dogs"));
    panDogs.setBackground(clrPaleGreen);
    JLabel lblDog1 = new JLabel("Kapulii");
    panDogs.add(lblDog1);
    JLabel lblDog2 = new JLabel("Hula Girl");
    panDogs.add(lblDog2);
    add(panDogs, BorderLayout.EAST);
    JPanel panCats = new JPanel();
    panCats.setLayout(new GridLayout(3,1));
    panCats.setBorder(new TitledBorder("Cats"));
    panCats.setBackground(clrPaleYellow);
    JLabel lblCat1 = new JLabel("Chairman Meow");
    lblCat1.setToolTipText("I don't like cats!");
    panCats.add(lblCat1);
    panCats.add(new JLabel(" "));
    add(panCats, BorderLayout.WEST);
  }
  public static void main(String[] args) {
    TestPanels fraPets = new TestPanels();
    fraPets.setTitle("House Pets");
    fraPets.setSize(200, 150);  // Size in Pixels, do first
    fraPets.setLocationRelativeTo(null); // Do second
    fraPets.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraPets.setVisible(true);   // Do Last
  }
}
```





ImageIcon Class add Images to Java

- Icon objects in `javax.swing.ImageIcon` class
  - An icon is a fixed-size picture
  - Typically small and used to decorate components
  - GIF, JPEG, or PNG image files can be icons
  - `ImageIcon new ImageIcon(filename)` instantiates image icon object
- Example:

```
ImageIcon icon = new ImageIcon("image/us.gif");
```

Creates an icon from an image file us.gif in the image directory under the *current path*

- ImageIcon object can be passed as argument
  - JLabel
  - JButton

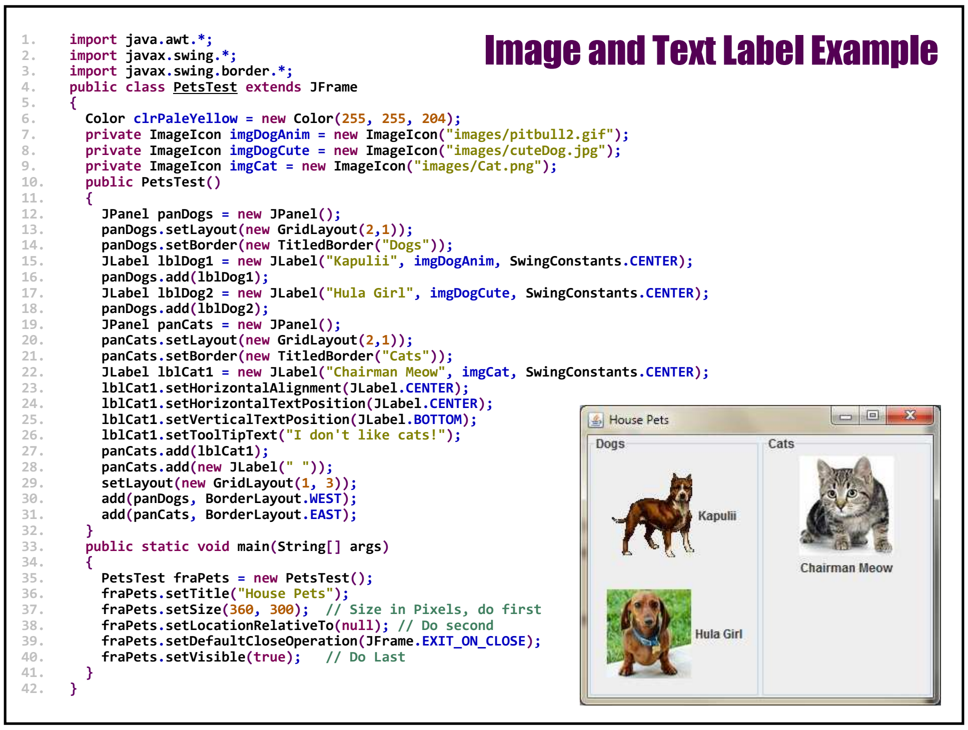
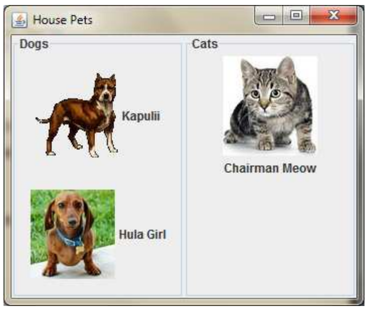


Image and Text Label Example

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
public class PetsTest extends JFrame
{
  Color clrPaleYellow = new Color(255, 255, 204);
  private ImageIcon imgDogAnim = new ImageIcon("images/pitbull2.gif");
  private ImageIcon imgDogCute = new ImageIcon("images/cuteDog.jpg");
  private ImageIcon imgCat = new ImageIcon("images/Cat.png");
  public PetsTest()
  {
    JPanel panDogs = new JPanel();
    panDogs.setLayout(new GridLayout(2,1));
    panDogs.setBorder(new TitledBorder("Dogs"));
    JLabel lblDog1 = new JLabel("Kapulii", imgDogAnim, SwingConstants.CENTER);
    panDogs.add(lblDog1);
    JLabel lblDog2 = new JLabel("Hula Girl", imgDogCute, SwingConstants.CENTER);
    panDogs.add(lblDog2);
    JPanel panCats = new JPanel();
    panCats.setLayout(new GridLayout(2,1));
    panCats.setBorder(new TitledBorder("Cats"));
    JLabel lblCat1 = new JLabel("Chairman Meow", imgCat, SwingConstants.CENTER);
    lblCat1.setHorizontalAlignment(JLabel.CENTER);
    lblCat1.setHorizontalTextPosition(JLabel.CENTER);
    lblCat1.setVerticalTextPosition(JLabel.BOTTOM);
    lblCat1.setToolTipText("I don't like cats!");
    panCats.add(lblCat1);
    panCats.add(new JLabel(" "));
    setLayout(new GridLayout(1, 3));
    add(panDogs, BorderLayout.WEST);
    add(panCats, BorderLayout.EAST);
  }
  public static void main(String[] args)
  {
    PetsTest fraPets = new PetsTest();
    fraPets.setTitle("House Pets");
    fraPets.setSize(360, 300);  // Size in Pixels, do first
    fraPets.setLocationRelativeTo(null); // Do second
    fraPets.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraPets.setVisible(true);   // Do Last
  }
}
```


House Pets
Dogs
Cats
Kapulii
Chairman Meow
Hula Girl