

# JavaBeans

- Formally introduced alongside Java 1.1
  - ◆ Bean Development Kit (BDK) 1.0
- Impetus for many of the new features of Java
  - ◆ inner classes
  - ◆ introspection
    - ☞ the Reflection class
- Also involves:
  - ◆ serialization
  - ◆ JAR files
  - ◆ etc.

# JavaBeans

- A bean is:
  - “A reusable software component that can be manipulated visually in a builder tool”
  - ◆ Not necessarily a visual thing itself
  - ◆ Not just a widget
    - ☞ a mini-application building block, possibly highly customizable
  - ◆ five common attributes: introspection, properties, events, customization, persistence
  - ◆ part of a larger component architecture that is evolving (and growing larger) all the time
- Another bullet in the war against ActiveX
  - ◆ designed to allow interoperability

# JavaBeans

The image displays four overlapping Java Swing windows, illustrating the use of JavaBeans in a graphical user interface.

**Very Meaningful Data (Bar Chart):** A window titled "Very Meaningful Data" showing a bar chart of Revenue (\$M) for the years 1996, 1997, and 1998. The chart compares four companies: World Wide Widgets, Yoyodyne, ACME, and Acumen Software. The Y-axis ranges from \$0.00 to \$80.00. The legend indicates: World Wide Widgets (yellow), Yoyodyne (purple), ACME (blue), and Acumen Software (light blue).

**OMT Diagrammer:** A window titled "OMT Diagrammer" showing a UML diagram. It includes a toolbar and a canvas with several objects: a Rectangle, a Point, and two instances of Point (p1 and p2). The Point class is defined with attributes int x and int y, and methods void setX(int x), void setY(int y), and Dimension distance().

**Word Processor:** A window titled "Word Processor" showing a text editor. The text content is as follows:

This application supports a variety of character formatting, including multiple styles (**bold**, *italic*, and underline), and **text coloring**. It also supports different fonts, like Helvetica and Courier as well as different point sizes like 8, 10, and **Twenty-Four**.

There is also plenty of paragraph formatting. You can set the horizontal alignment of the paragraph to left, right (for you poets), or centered, like this one. You can also adjust the left and right margins, as well as the amount of indent for the first line of the paragraph. These can be set by dragging the arrows in the ruler.

- Point 1, Bulleted paragraphs are supported.
- Point 2, Bulleted paragraphs are supported.

For Help, press F1

Paragraph 1, Line 1

**Table:** A window showing a table with the following data:

A1	A	B	C	D
1		World Wide Widgets	Yoyodyne	ACME
2	1996	\$10.00	\$30.00	\$80.00
3	1997	\$40.00	\$20.00	\$20.00

Sunday, July 05, 2009

# JavaBeans

- Beans can be used at three levels:
  - ◆ bean-aware tool developers
    - ☞ beans API targeted mainly at this group
  - ◆ bean author
    - ☞ needs to implement a subset of the API relevant to the bean being built
  - ◆ bean user
    - ☞ don't need to use the API at all, (perhaps) only the visual builder tool
    - ☞ domain experts assembling from a palette of tools

# JavaBeans

- Why JavaBeans?
  - ◆ reuse
  - ◆ ease of development
    - ☞ little more difficult than a 'normal' applet
  - ◆ scripting/user development
    - ☞ but don't believe all the hype about scripting & "snap together" application development... programming will **always** be *hard*
  - ◆ the internet has changed the rules regarding what a component should be like
  - ◆ "...the only architecture you should consider if you are developing in a Java environment."

# JavaBeans

- *“A Bean is any Java class which adheres to certain property and event interface conventions.”*
  - ◆ no Bean class: uses a protocol with other beans
    - ☞ a primitive, unformalised applet
  - ◆ heavy use of introspection
    - ☞ container or builder tool can determine the bean's characteristics by recognizing predefined patterns
- Protocol is discovered based on public characteristics:
  - ◆ properties, events & methods

# JavaBeans

## ■ Properties

### ◆ visible internal state

- ☞ inspected by a builder tool or container
- ☞ *not* instance variables
- ☞ always accessed via setters & getters

### ◆ simple property

- ☞ *PropertyName* is a simple variable

```
public void setPropertyName (PropertyType)  
public PropertyType getPropertyName ()  
public boolean isPropertyName ()
```

### ◆ indexed property

- ☞ *PropertyName* is an array

```
public void setPropertyName (PropertyElement [])  
public PropertyElement [] getPropertyName ()  
public void setPropertyName (int, PropertyElement)  
public PropertyElement getPropertyName (int)
```

a class is  
inspected looking  
for patterns  
like these

# JavaBeans

## ◆ bound property

```
public void setSalary (Float newSalary)
{
    Float oldSalary = this.salary;
    this.salary = newSalary;
    vListeners.firePropertyChange ("salary", oldSalary, newSalary);
}
```

☞ sends a `PropertyChangeEvent` when updated

- `PropertyChangeEvent` event
- `PropertyChangeSupport` class
- `PropertyChangeListener` interface
  - `propertyChange` method

☞ events reported at bean, not property, level: need to determine event/property correspondence



# JavaBeans

## ◆ constrained property

```
public void setSalary (Float newSalary) throws PropertyVetoException
{
    Float oldSalary = this.salary;
    vListeners.fireVetoableChange ("salary", oldSalary, newSalary);
    this.salary = newSalary;
    pListeners.firePropertyChange ("salary", oldSalary, newSalary);
}
```

☞ as bound, but changes can be vetoed

- VetoableChangeSupport class
- VetoableChangeListener interface
  - vetoableChange
  - examines a PropertyChangeEvent
- PropertyVetoException

```
public void vetoableChange (PropertyChangeEvent e)
    throws PropertyVetoException
{
    float new = ((Float) e.getNewValue ().floatValue ()),
        old = ((Float) e.getOldValue ().floatValue ());

    if (new > (old + (old * 0.1)))
        throw new PropertyVetoException ("Too generous!", e);
}
```

# JavaBeans

## ■ Methods

- ◆ all public methods available for use by container

- ☞ “pattern matching” used to determine properties, etc.

- ☞ unless alternative view specified:

- BeanInfo interface and SimpleBeanInfo class
      - get{Method, Property, EventSet}Descriptors ()
      - getIcon ()
      - getDefault{Property, Event}Index ()
      - etc.

- ◆ all public methods should be synchronized

- ☞ since may be multiple views in a container

# JavaBeans

```
public class SimplestBeanInfo extends SimpleBeanInfo
{
    private final static Class beanClass = SimplestBean.class;

    public java.awt.Image getIcon (int k)
    {
        if (k == BeanInfo.ICON_MONO_16x16 || k == BeanInfo.ICON_COLOR_16x16)
            return (loadImage ("SimplestBeanIcon16.gif"));

        else if (k == BeanInfo.ICON_MONO_32x32 || k == BeanInfo.ICON_COLOR_32x32)
            return (loadImage ("SimplestBeanIcon32.gif"));
        return null;
    }

    public PropertyDescriptor [] getPropertyDescriptors ()
    {
        try
        {
            PropertyDescriptor col = new PropertyDescriptor ("Color", beanClass);
            col.setshortDescription ("The color of this SimplestBean");
            return (new PropertyDescriptor [] { col });
        }
        catch (IntrospectionException e)
        { return (super.getPropertyDescriptors ()); }
    }

    public int getDefaultPropertyIndex () { return (0); }

    ( other code elided... )
}
```

# JavaBeans

## ■ Events

### ◆ beans can use the standard Java 1.1 events

- ☞ might simulate a mouse click, say
- ☞ all beans know how to deal with these events

### ◆ can also define custom events

- ☞ to say “hey! I’ve just done something really good and I want you to know about it...”
  - target must want to know about this specific event in the first place (have registered a listener for the event type)

*EEvent*

*EListener*

```
public void methodName (EEvent)
```

```
public void addEListener (EListener)
```

```
public void removeEListener (EListener)
```

# JavaBeans

```
import java.awt.*;

public class SimplestBean extends Canvas
{
    private Color ourColor = Color.orange;

    public SimplestBean () { setSize (60, 40); }

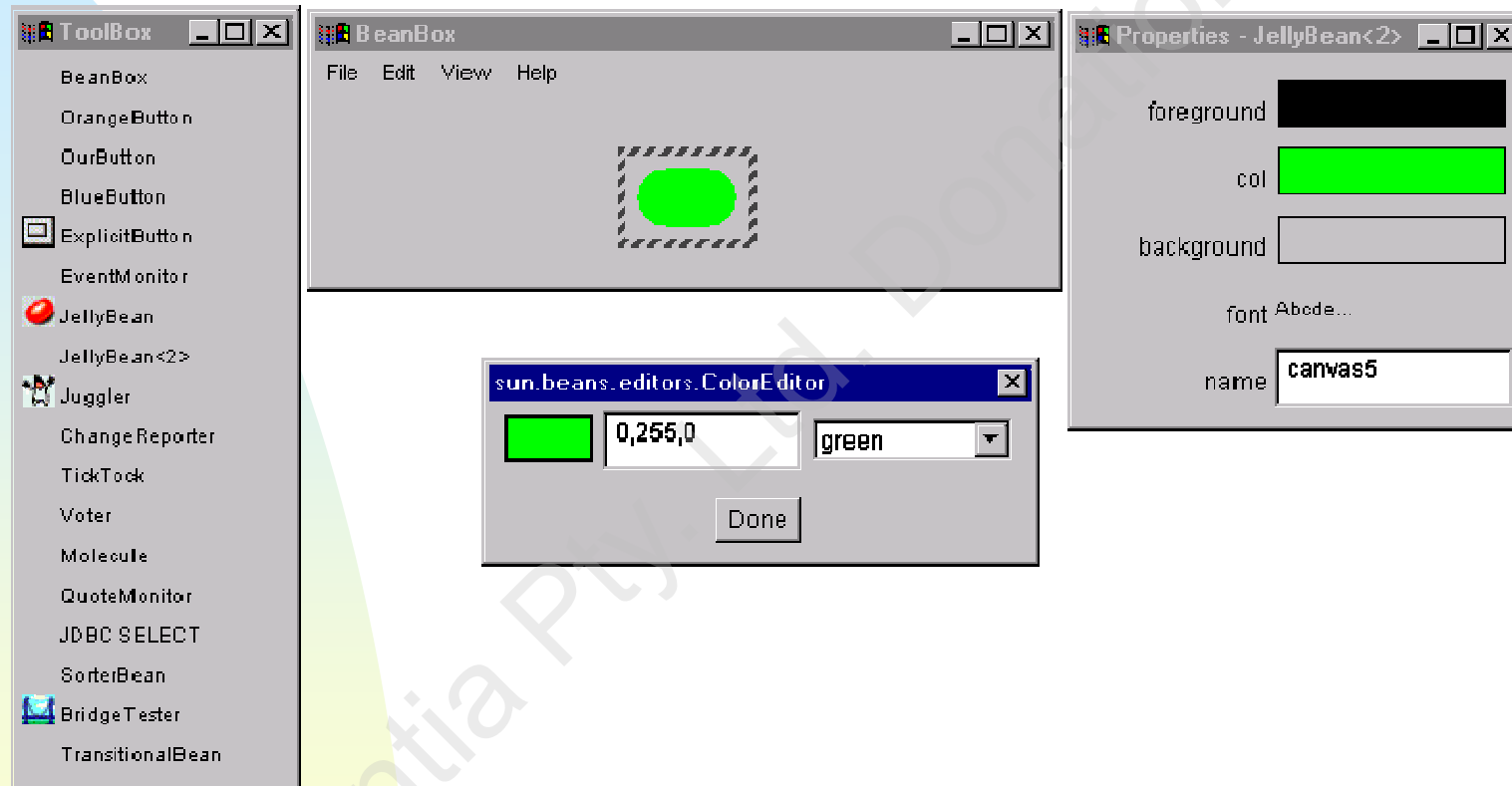
    public void paint (Graphics g)
    {
        g.setColor (ourColor);
        g.fillArc (5, 5, 30, 30, 0, 360);
        g.fillArc (25, 5, 30, 30, 0, 360);
        g.fillRect (20, 5, 20, 30);
    }

    public synchronized Color getCol ()
    { return ourColor; }

    public synchronized void setCol (Color newColor)
    {
        ourColor = newColor;
        repaint();
    }
}
```

Defines the bean's  
simple *col*/  
property

# JavaBeans



# JavaBeans

- Property editors
  - ◆ for bean-defined, non-standard property types
- Customizers
  - ◆ allow properties to be established *en-masse* via a specialized dialog, rather than piecemeal through numerous property editors
  - ◆ may be more user (programmer) friendly