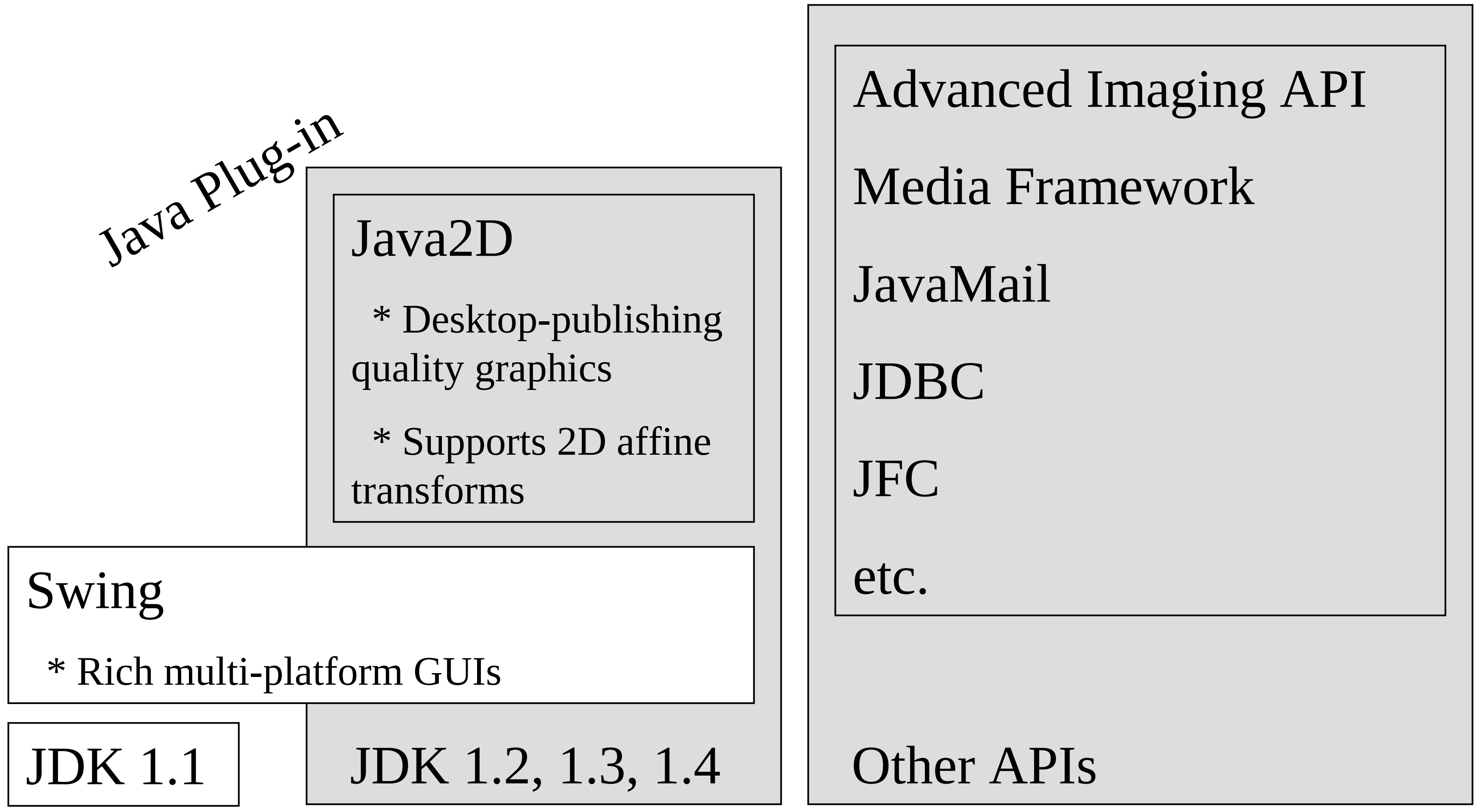
Advanced Graphics and GUIs in Java

Java, Swing, and Java2D

Ben Bederson
University of Maryland
Human-Computer Interaction Lab
Computer Science Department

Java Components



Java Overview

- Fully buzzword compliant
  - Usable
  - Simple (multiple inheritance, operator overloading, file name restrictions, include files, pointers, memory)
  - Object-oriented
  - Byte-compiled
  - Portable
  - Dynamic (reflection, dynamic loading)
  - Distributed
  - Robust (strongly typed, exceptions)
  - Secure (sandbox, signatures, verifier)
  - Multithreaded
  - Standard Libraries (UI, network, I/O, math, security...)

Overview

- Java2D
  - Graphics
  - Text
  - Images
  - Devices
  - Color Mgmt
  - Consistency
  - Transforms

- Swing
  - Extensive Widgets
  - PLAF
  - Accessibility
  - 100% Java Implementation

Power vs. Simplicity trade-off

Java moves towards the power end

Programming Java I

http://java.sun.com/docs/books/tutorial/

“Application” – standalone, starts w/ “main”

“Applet” – extends JApplet, runs in sandbox

“References” are memory-safe and type-safe pointers

```
Foo foo = new Foo();
int a = foo.getA();
Foo bar;
int b = bar.getB();   // null-ptr exception
```

Primitive types accessed by value

Runtime checker will catch array bounds problems, illegal casting, null pointer access, math problems, other exceptions

Garbage collection manages memory for you. No “delete”, and no destructors

Programming in Java II

Foo.java

```
public class Foo {
  static public void main (String args[]) {
   System.out.println(“Hello world!”);
  }
}
```

javac Foo.java   =>   Foo.class

java Foo

  => Hello world!

“Packages” define namespaces.
Access with “import”
Implicit “import java.lang.*”

Programming in Java - Exceptions

- Bad things (errors or exceptions) get caught by the JVM. You can also **throw** them yourself

```
try {
    // some stuff
} catch (SomeException e) {
    //
} finally {
  // Always called
}
```

```
try {
  foo[i] = bar;
} catch (ArrayIndexOutOfBoundsException e) {
  increaseCapacity(foo, i);
  foo[i] = bar;
}
```

Programming in Java - Interfaces

Interfaces are like a completely abstract class:

```
interface Foo {
  int bar();
}
class Dog implements Foo {
  int bar { // do something }
}
…
Dog dog = new Dog();
Foo foo = (Foo)Dog;
foo.bar();
```

Swing

- `http://java.sun.com/products/jfc/tsc`
- `http://java.sun.com/docs/books/tutorial/uiswing`
- Swing is a 100% Java Implementation
- All “lightweight” components, except top-level windows
- Built on top of (and backwards compatible) with AWT (Abstract Windowing Toolkit)

Main Swing Features

- Pluggable Look and Feel
- Extensive Widgets
- Accessibility

The “Swing” name: Someone mentioned that “Swing” music was enjoying a comeback at the 1997 JavaOne convention

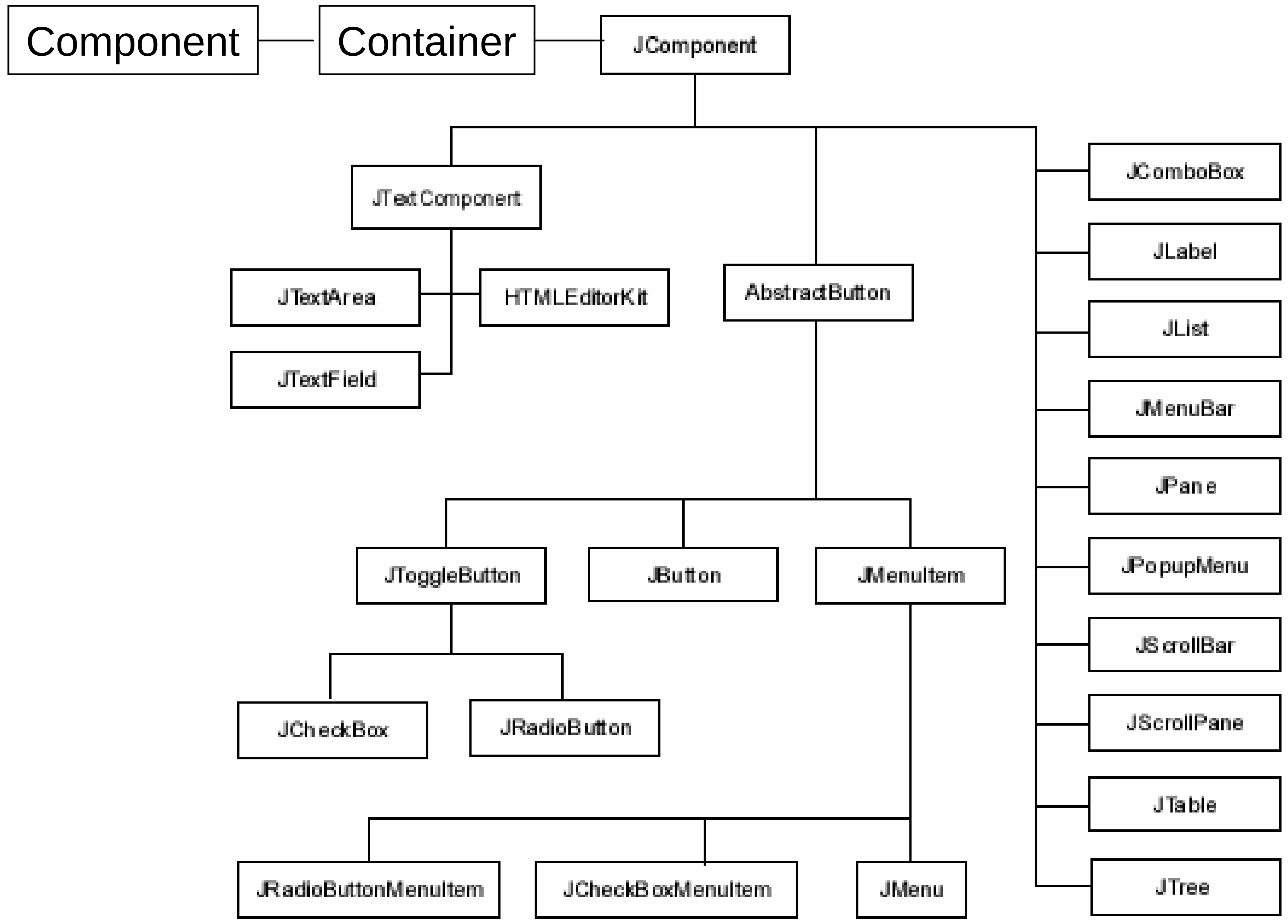
Pluggable Look and Feel

- AWT had Java wrapper classes and “peers” that created a native widget unique for each platform.
- Each component has a “model” and a PLAF.
- =>Choose your look and feel at run-time
- Currently: Windows, Motif, Java, and Mac
- Write your own…

Additional Features

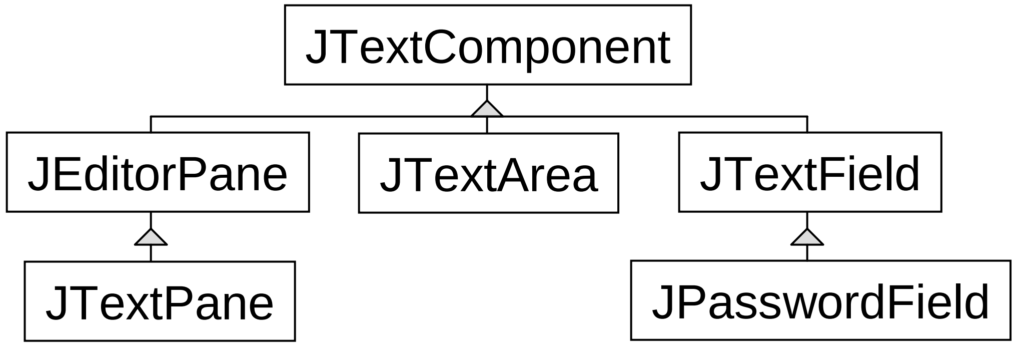
- Extensibility
- Handle just keyboard events you care about
- Customize component borders
- Tool tips
- Autoscrolling
- Support for localization
- Drag and drop
- Dockable toolbars

Partial Class Hierarchy



Text Components

- Goals:
  - Model-view separation
  - Pluggable look-and-feel (PLAF)
  - Scalability
  - Extensibility
  - Blurs text-component boundaries



JEditorPane

- Supports:
  - Plain text
  - RTF
  - HTML

Model-view separation

```
Jslider slider = new JSlider();
BoundedRangeModel myModel =
  new DefaultBoundedRangeModel() {
      public void setValue(int n) {
        System.out.println(“SetValue: “ +n);
        super.setValue(n)
      }
  });
slider.setModel(myModel);
```

Model change notification

- Models can notify interested parties that they have changed with either:
  - stateless notification
  - stafefull notification

```
Jslider slider = new JSlider();
BoundedRangeModel model = slider.getModel();
model.addChangeListener(new ChangeListener() {
  public void stateChanged(ChangeEvent e) {
    BoundedRangeModel m =
          (BoundedRangeModel)e.getSource();
    System.out.println(“model changed: “ +
          m.getValue();
  }
});
```

Or, ignore the model

```
Jslider slider = new Jslider();
int value = slider.getValue();
   // What’s a model anyway?
```

Layout Managers

- Automatically layout widgets based on various models
- Powerful, but often hard to use
- See demos\applets\CardTest

Documentation

- API available as HTML distributed w/ doc
- Guides for major packages distributed w/ documentation
- Tutorials online

Swing

Demo

Java2D

- **`http://java.sun.com/products/java-media/2D`**
- AWT’s goal was to provide “Web” graphics
- Java2D’s goal is to provide “desktop publishing” graphics

Graphics: Antialiased, Bezier, Transforms, Compositing, Richer text attributes, Arbitrary fill styles, Stroke Parameters

Text: Extended font support, Advanced text layout, Antialiased text rendering

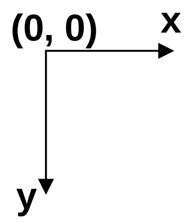
Images: Flexible in-memory image layouts, Extended image filters, Lookup tables, and affine transformation

Devices: Hooks for supporting arbitrary graphics devices such as printers and screens

Color Management: ICC profile support, Color spaces, Color conversion

Graphics2D

- Graphics2D extends Graphics
- Rendering model similar to 3D
  - User Space -> Device Space
  - Coord space includes 2D Affine transforms for everything



$$\begin{bmatrix} a & c & t_x \\ b & d & t_y \end{bmatrix}$$

$$x' = ax + cy + t_x$$

$$y' = bx + dy + t_y$$

Rotated red rectangle

```
public void paint(Graphics g) {
    AffineTransform rot30 = new AffineTransform();
    rot30.setToRotation(30.0f * Math.PI / 180.0f);
    Graphics2D g2 = (Graphics2D)g;
    g2.setTransform(rot30);
    Rectangle2D rect =
        new Rectangle2D.Float(10.0f, 10.0f, 20.0f, 30.0f);
    g2.setColor(color.red);
    g2.draw(rect);
}
```

Geometries

- java.awt.geom
  - Dimension2D
  - Point2D
  - Line2D
  - Rectangle2D
  - RoundRectangle
  - Arc2D
  - Ellipse2D
  - CubicCurve2D
  - QuadCurve2D
  - GeneralPath
  - Area

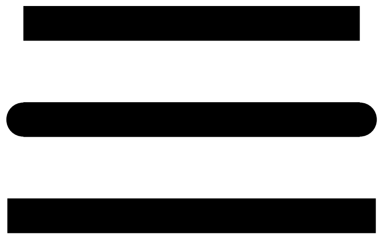
Controlling Rendering Quality

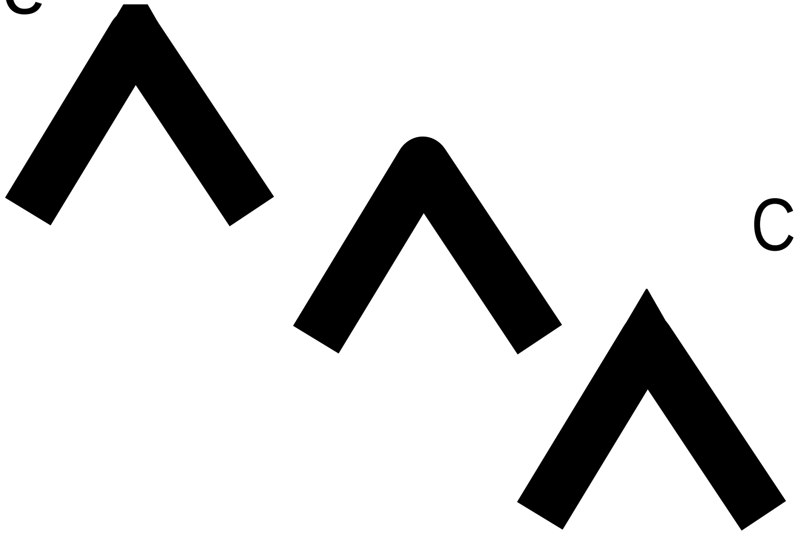
- Application can control: alpha, antialiasing, color, dithering, interpolation, general quality
- **`g2.setRenderingHint( RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_OFF);`**

```
g2.setRenderingHint(
RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_OFF);
```

Rendering Attributes

- Cap Style
  - Butt
  - Round
  - Square
- Join Style
  - Bevel
  - Round
  - Miter





Stroke
- Solid
- Outline
- Dashed

Fill
- Gradient
- Texture
- Pattern

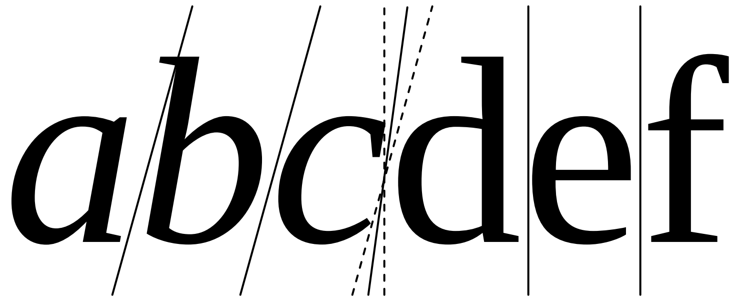
Clip

Fonts and Text

- The Font class includes
  - Families (Helvetica, Courier, etc.)
  - Faces (Bold, italics, etc.)
  - Attributes (weight, posture)
  - Metrics (Ascent, descent, bounding box)
  - Affected by current Affine Transform

Fonts and Text (2)

- Text supports:
  - Multidirectional fonts (Arabic, Chinese, etc.)
  - Ligatures (arabic, typesetting)
  - Caret control



Imaging

- Image, ImageProducer, ImageConsumer, BufferedImage classes
  - Enable networked image access. Start use before entire image is available
  - Good performance starting with JDK1.4b2
- JDK1.4 introduces new imageio package
  - Supports new image types and is extensible
  - Supports metadata
  - Supports multiple images per file, thumbnails, multi-resolution imagery, tiled imagery
  - Efficient about reading/writing metadata, very large images

Java2D Demo

The Future

- Sun’s motto with Java is: “First do it right, then do it fast”
- JDK1.2 introduced most of the new API
- JDK1.3 introduced reliability
- JDK1.4 is adding decent performance
- Java is getting good, but is it too little, too late? Microsoft’s .NET is biting at Suns heels...