

Java Programming

Fall 2015 - Week 6

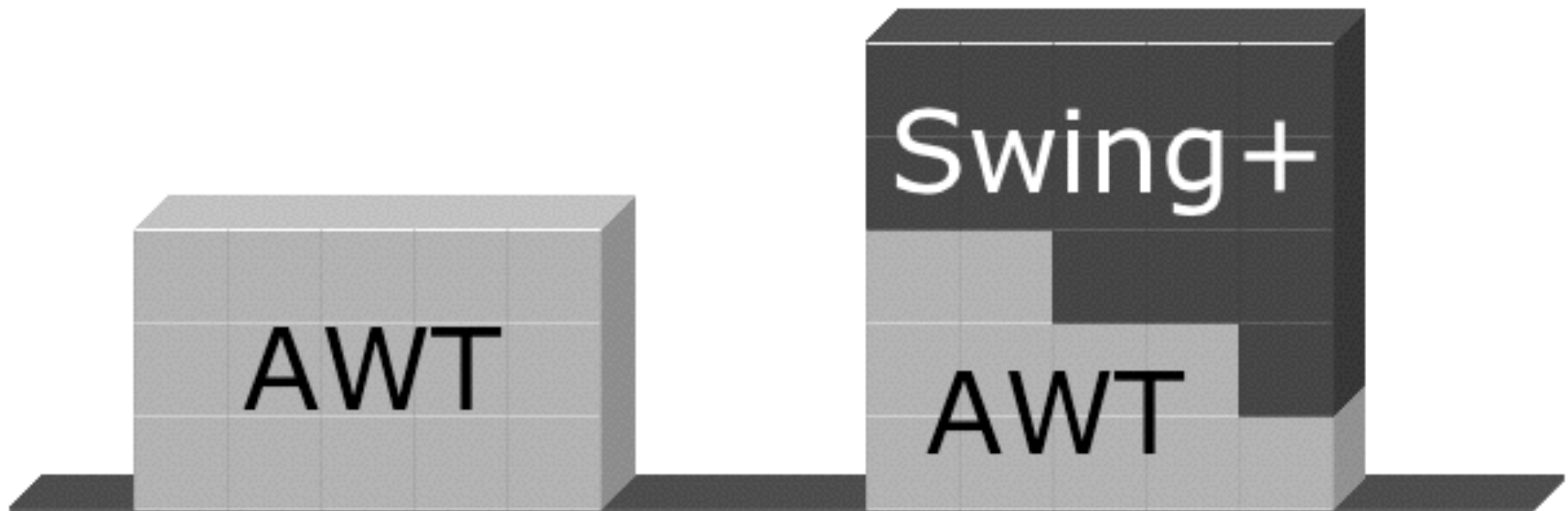
GUI

Weng Kai

<http://fm.zju.edu.cn>

A Little Story

AWT and Swing



About the JFC and Swing

- JFC is short for Java Foundation Classes, which encompass a group of features for building graphical user interfaces (GUIs) and adding rich graphics functionality and interactivity to Java applications.



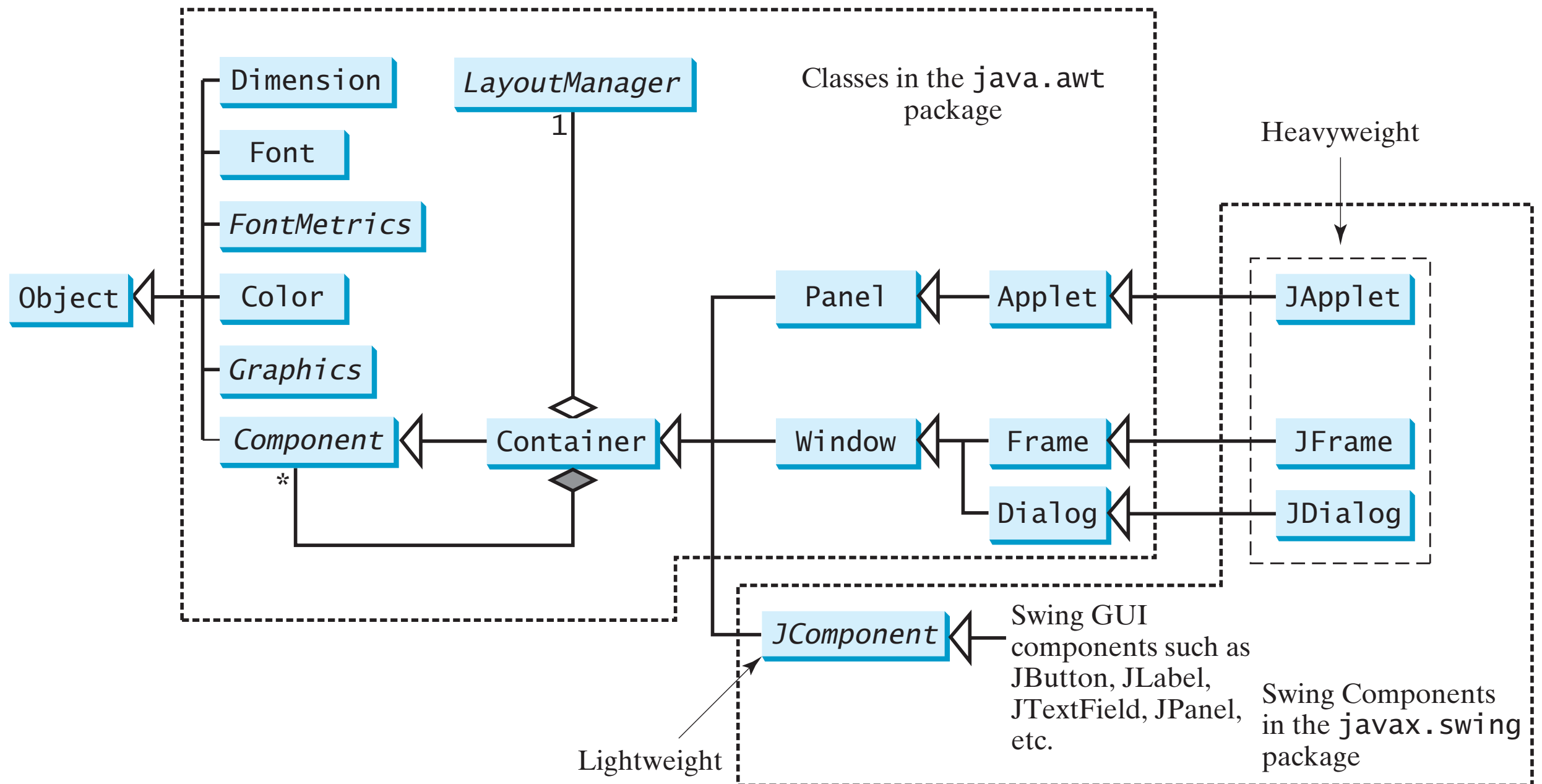
Features of the Java Foundation Classes

- Swing GUI Components
- Pluggable Look-and-Feel Support
- Accessibility API
- Java 2D API
- Internationalization

Swing Features

GUI Principles

- Components: GUI building blocks.
 - Buttons, menus, sliders, etc.
- Layout: arranging components to form a usable GUI.
 - Using layout *managers*.
- Events: reacting to user input.
 - Button presses, menu selections, etc.



Classes

`java.awt.Container`

`javax.swing.JFrame`

`javax.swing.JPanel`

`javax.swing.JApplet`

`javax.swing.JDialog`

`java.awt.Graphics`

`java.awt.Color`

`java.awt.Font`

`java.awt.FontMetrics`

`java.awt.Dimension`

`java.awt.LayoutManager`

JFrame

- A Frame is a top-level window with a title and a border.
- A frame, implemented as an instance of the JFrame class, is a window that typically has decorations such as a border, a title, and buttons for closing and iconifying the window.
Applications with a GUI typically use at least one frame.

Creating a frame

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

public class ImageViewer
{
    private JFrame frame;

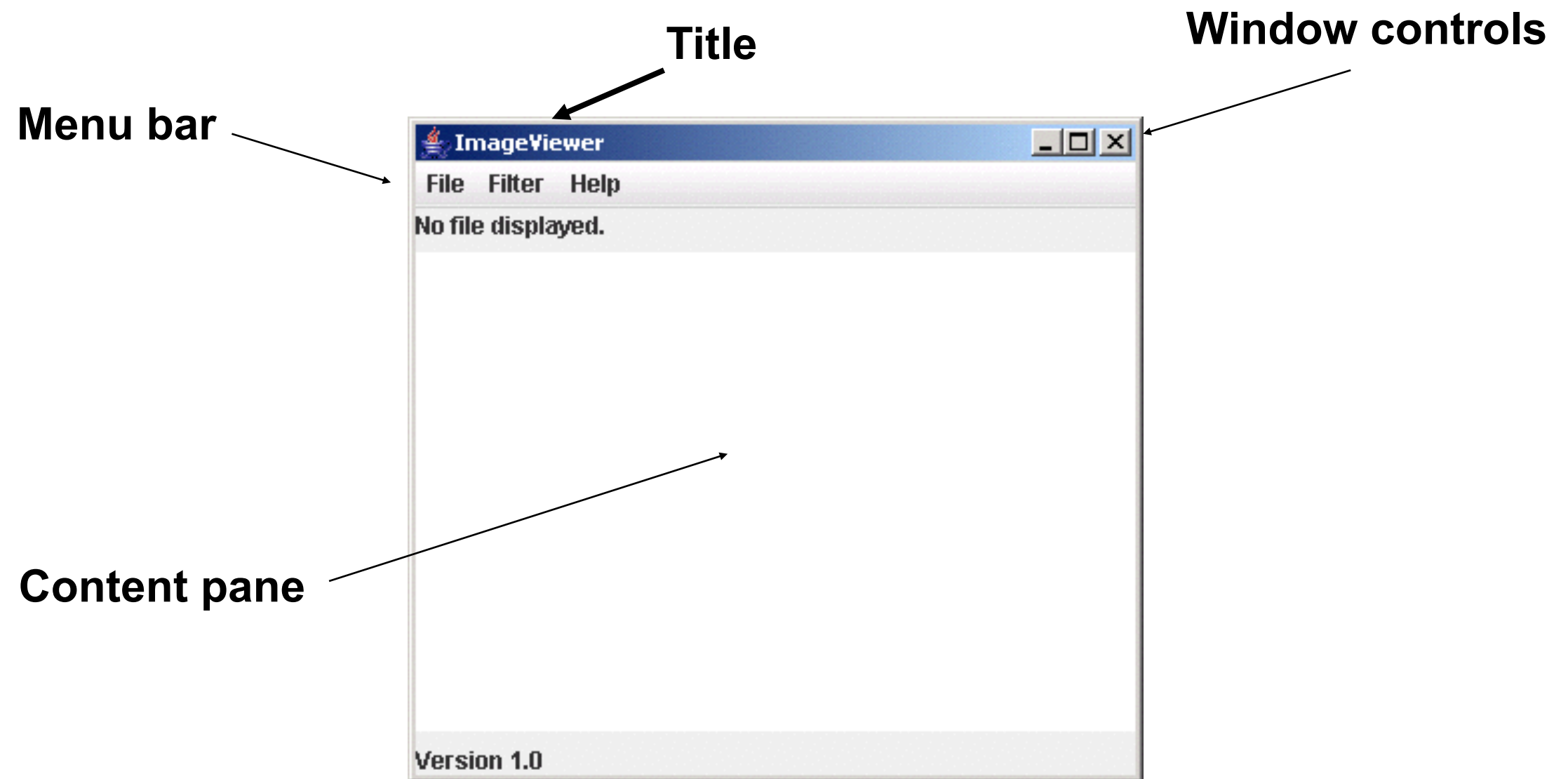
    /**
     * Create an ImageViewer show it on screen.
     */
    public ImageViewer()
    {
        makeFrame();
    }

    // rest of class omitted.
}
```

code for JFrame

```
JFrame frame = new ScoreBoard();  
frame.pack();  
frame.setVisible(true);
```

Elements of a frame



JFrame

javax.swing.JFrame

```
+JFrame()  
+JFrame(title: String)  
+setSize(width: int, height: int): void  
+setLocation(x: int, y: int): void  
+setVisible(visible: boolean): void  
+setDefaultCloseOperation(mode: int): void  
+setLocationRelativeTo(c: Component):  
    void  
+pack(): void
```

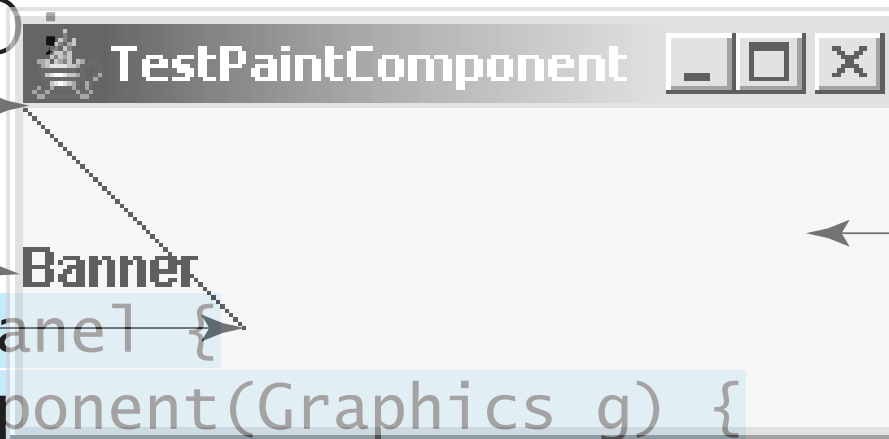
- JFrame is a top-level container to hold GUI components.

Graphics

Directly Render

```
1 import javax.swing.*;
2 import java.awt.Graphics;
3
4 public class TestPaintComponent extends JFrame {
5     public TestPaintComponent() {
6         add(new JPanel());
7     }
8
9     public static void main(String[] args) {
10         TestPaintComponent frame = new TestPaintComponent();
11         frame.setTitle("TestPaintComponent");
12         frame.setSize(200, 100);
13         frame.setLocationRelativeTo(null); // Center the frame
14         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15         frame.setVisible(true);
16     }
17 }
```

```
19 class JPanel extends JComponent {
20     protected void paintComponent(Graphics g) {
21         super.paintComponent(g);
22         g.drawLine(0, 0, 50, 50);
23         g.drawString("Banner", 0, 40);
24     }
25 }
```



This is a JPanel
object placed
inside a frame

TestPaintComponent.java

The Graphics

- The Graphics class contains the methods for drawing strings and shapes.

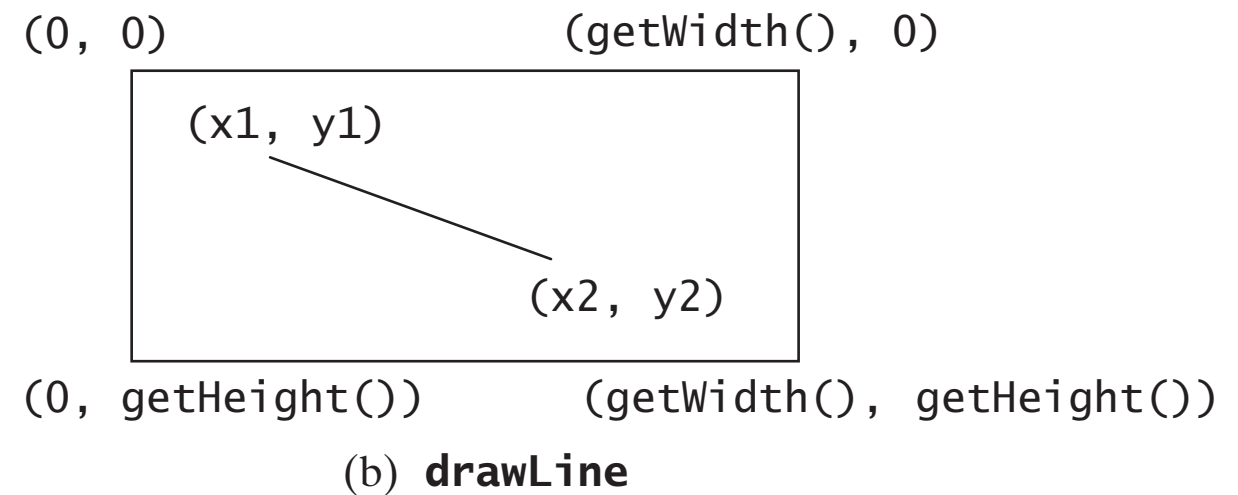
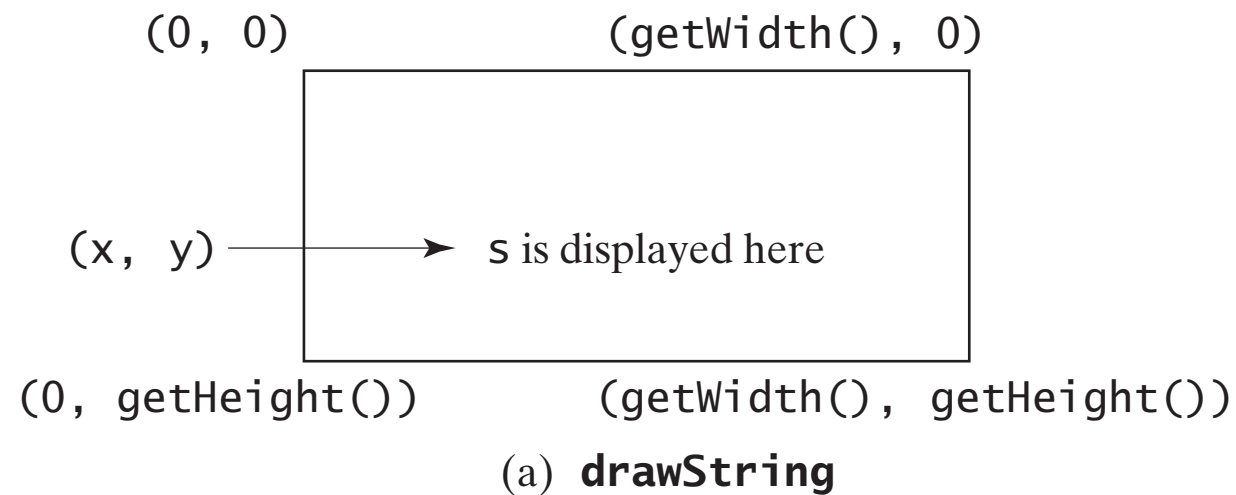
java.awt.Graphics

```
+setColor(color: Color): void
+setFont(font: Font): void
+drawString(s: String, x: int, y: int): void
+drawLine(x1: int, y1: int, x2: int, y2:
    int): void
+drawRect(x: int, y: int, w: int, h: int):
    void
+fillRect(x: int, y: int, w: int, h: int): void

+drawRoundRect(x: int, y: int, w: int, h: int, aw:
    int, ah: int): void
+fillRoundRect(x: int, y: int, w: int, h: int,
    aw: int, ah: int): void
+draw3DRect(x: int, y: int, w: int, h: int,
    raised: boolean): void
+fill3DRect(x: int, y: int, w: int, h: int,
    raised: boolean): void
+drawOval(x: int, y: int, w: int, h: int):
    void
+fillOval(x: int, y: int, w: int, h: int): void

+drawArc(x: int, y: int, w: int, h: int,
    startAngle: int, arcAngle: int): void
+fillArc(x: int, y: int, w: int, h: int,
    startAngle: int, arcAngle: int): void
+drawPolygon(xPoints: int[], yPoints:
    int[], nPoints: int): void
+fillPolygon(xPoints: int[], yPoints: int[],
    nPoints: int): void
+drawPolygon(g: Polygon): void
+fillPolygon(g: Polygon): void
+drawPolyline(xPoints: int[], yPoints:
    int[], nPoints: int): void
```

String and Line

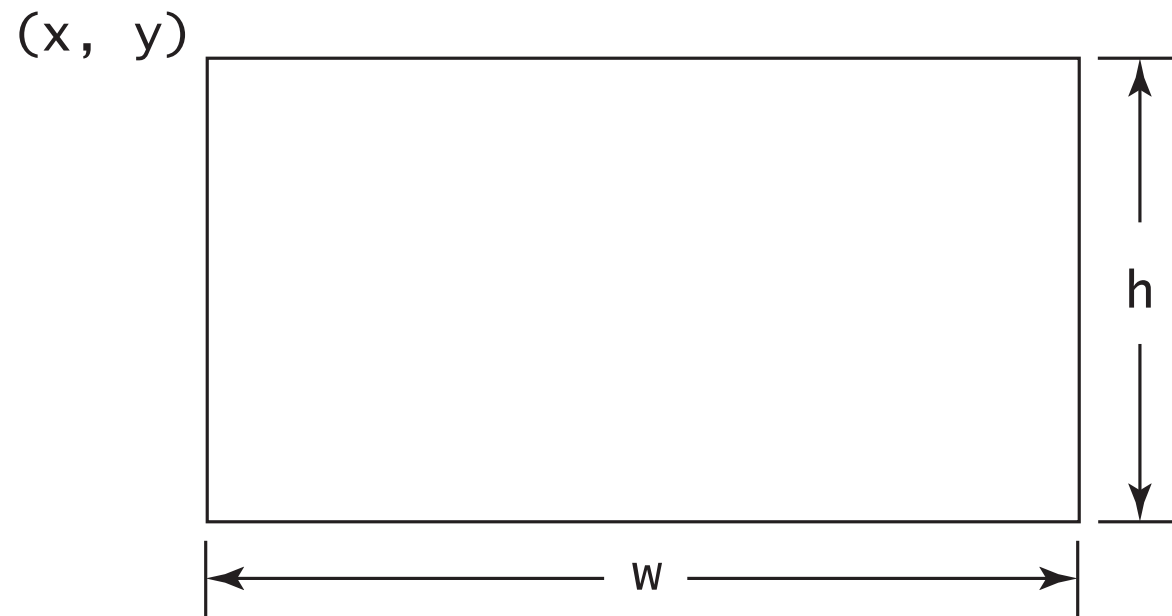


- `drawString(s, x, y)`
- `drawLine(x1, y1, x2, y2)`

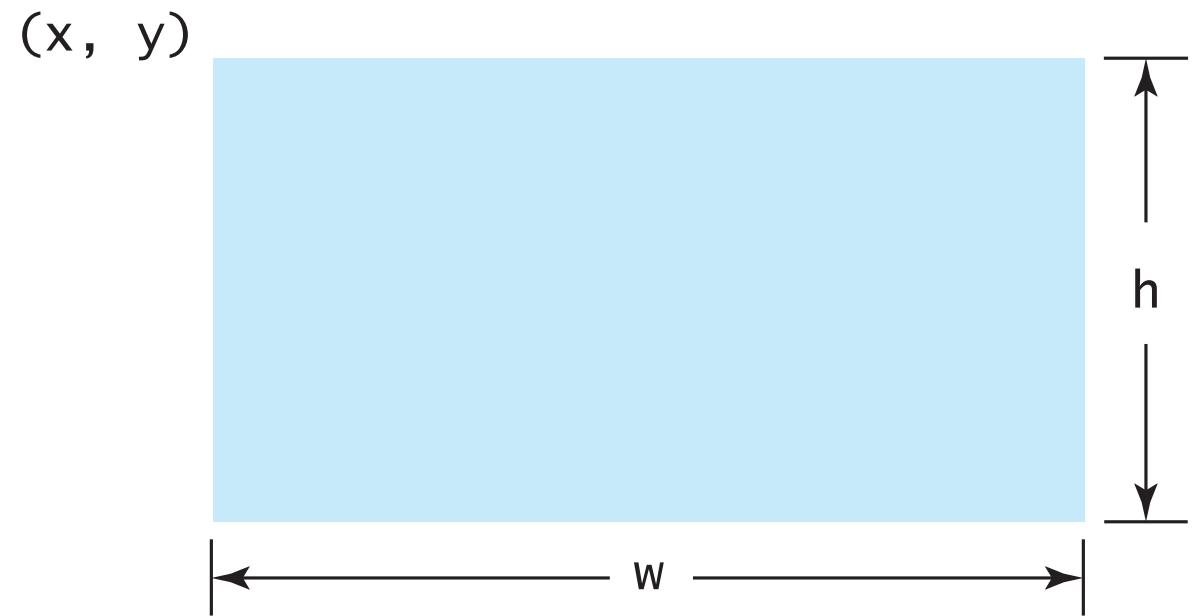
Color

- `public Color(int r, int g, int b);`
- `setColor(Color color);`
- `JButton jbtOK = new JButton("OK");`
`jbtOK.setBackground(color);`
`jbtOK.setForeground(new Color(100, 1, 1));`
- `jbtOK.setForeground(Color.RED);`

Draw and Fill Rectangle



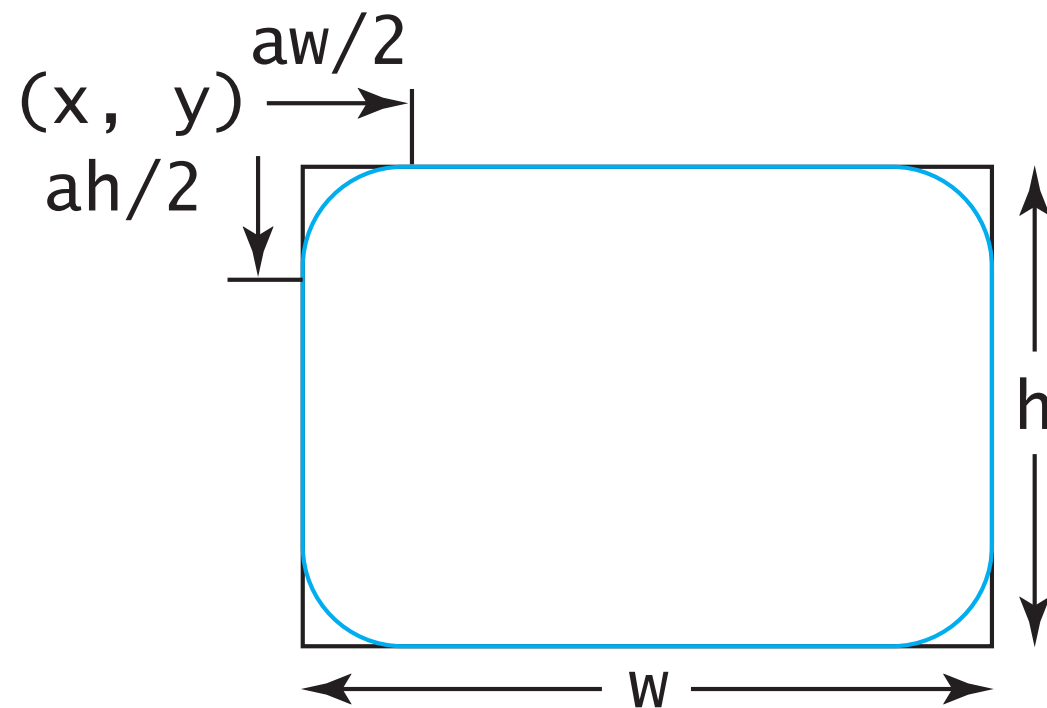
(a) Plain rectangle



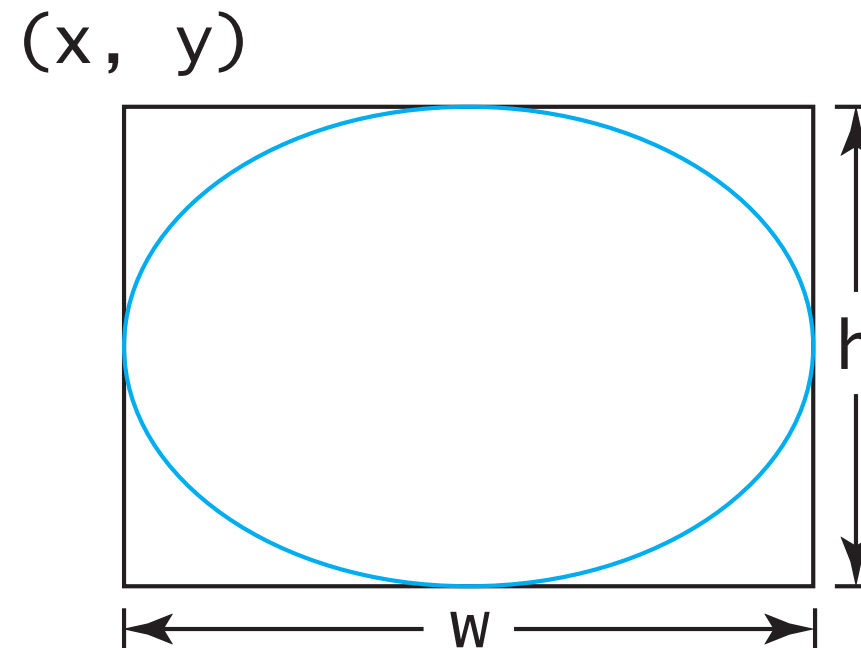
(b) Filled rectangle

- `drawRect(x, y, w, h)`
- `fillRect(x, y, w, h)`

RoundRect and Oval



(a) **drawRoundRect**

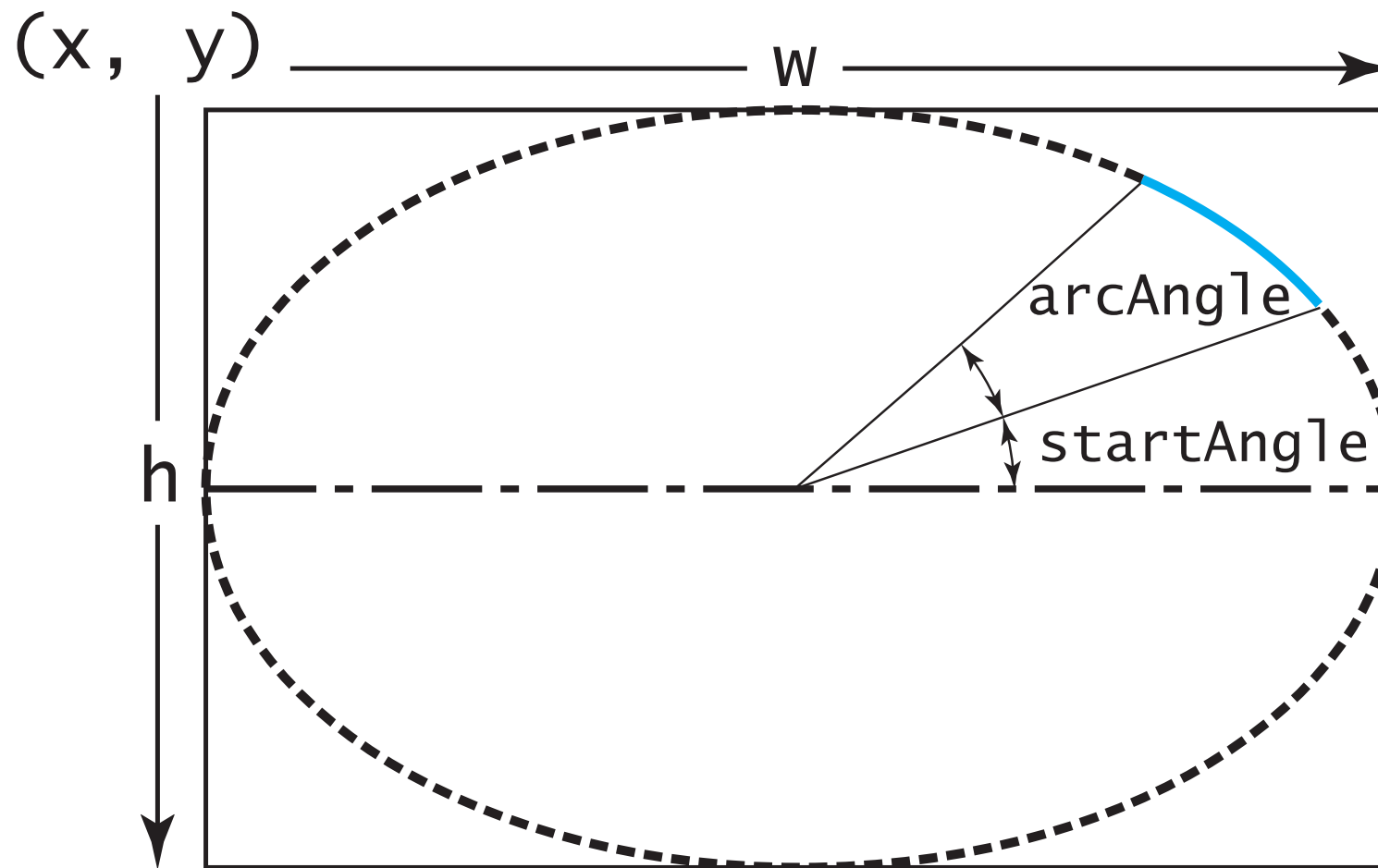


(b) **drawOval**

- **drawRoundRect**(x, y, w, h, aw, ah)
- **drawOval**(x, y, w, h)

FigurePanel.java
TestFigurePanel.java

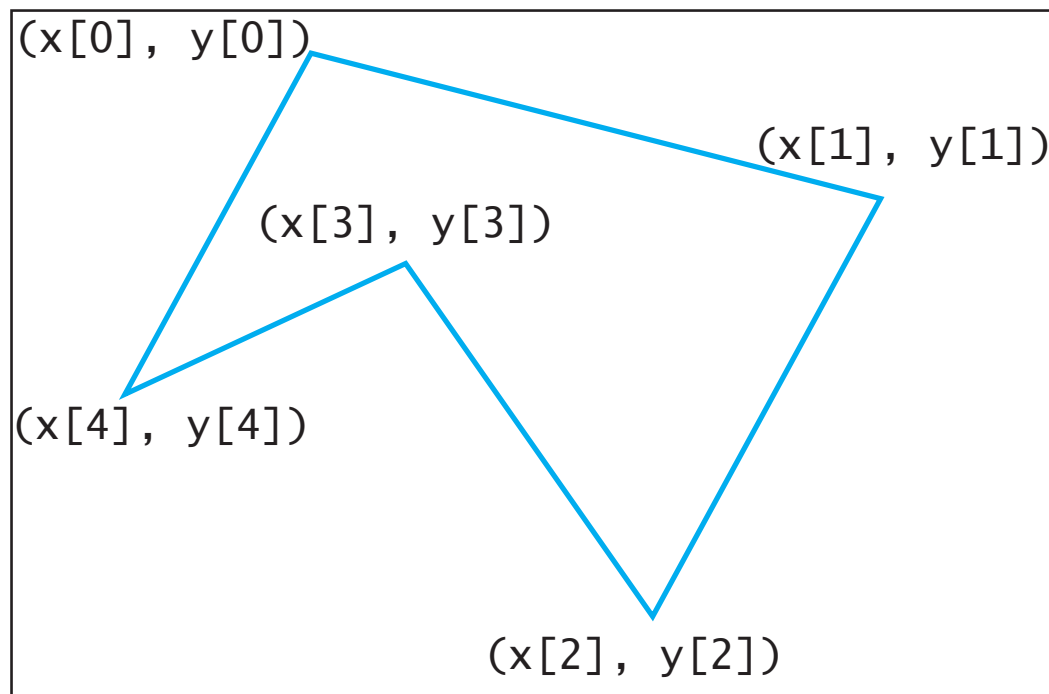
Arcs



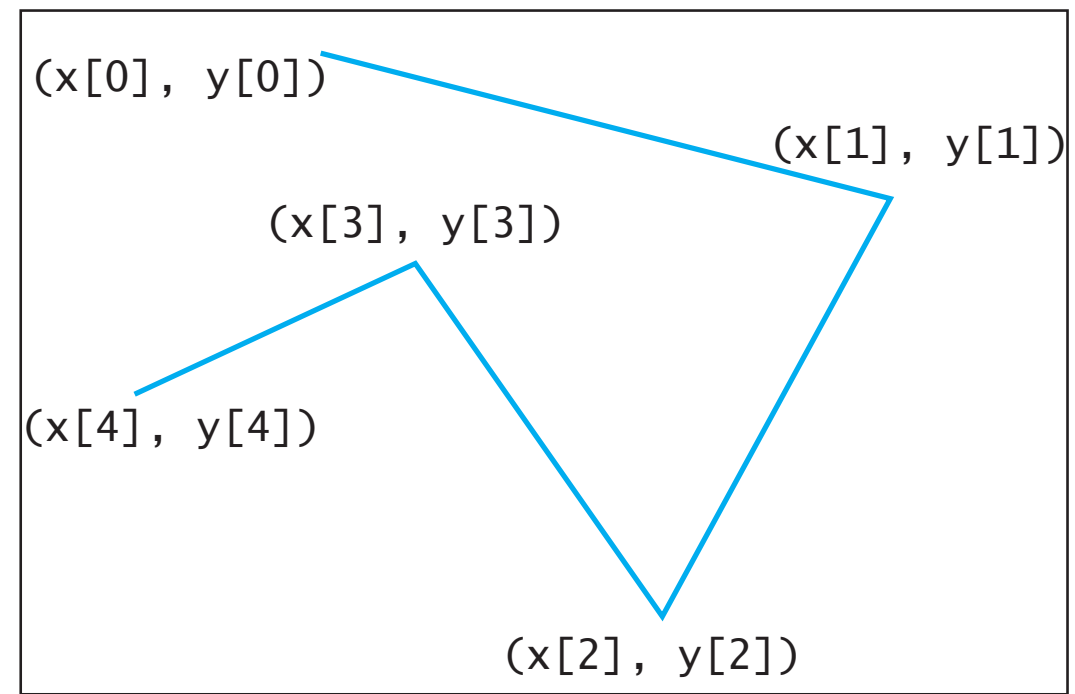
- `drawArc(int x, int y, int w, int h, int startAngle, int arcAngle);`
- `fillArc(int x, int y, int w, int h, int startAngle, int arcAngle);`

DrawArcs.java

Polygon and Polyline



(a) Polygon



(b) Polyline

- `Polygon polygon = new Polygon();`
`polygon.addPoint(40, 20);`
- `drawPolygon(Polygon polygon);`
`fillPolygon(Polygon polygon);`

Another way polygon

```
int x[] = {40, 70, 60, 45, 20};  
int y[] = {20, 40, 80, 45, 60};  
g.drawPolygon(x, y, x.length);  
g.drawPolyline(x, y, x.length);
```


英文字体从单字上分两种，一种有衬线（serif），一种无衬线（sans-serif）。从26个字母比较上又分两种，一种自然宽度，一种等宽（monospace）。衬线字体笔画有粗细变化，且首尾带装饰线，无衬线字体笔画粗细均匀无变化，也没有装饰线。自然宽度字体 m 最宽，i 最窄，等宽字体26个字母全部一样宽。

... a style from Font.PLAIN (0), Font.BOLD (1), Font.ITALIC (2), and Font.BOLD + Font.ITALIC (3),

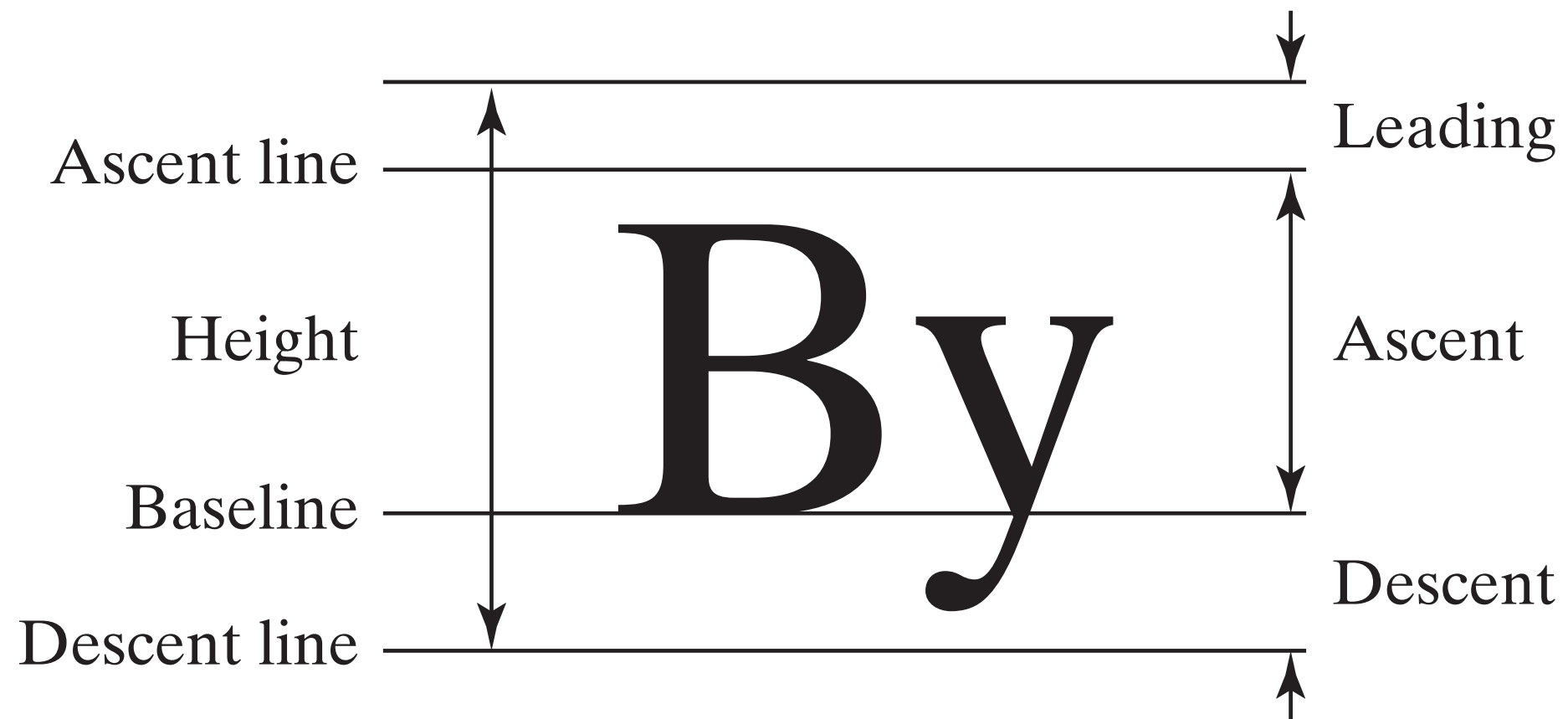
- and specify a font size of any positive integer.

- `Font font1 = new Font("SansSerif", Font.BOLD, 16);`
`Font font2 = new Font("Serif", Font.BOLD + Font.ITALIC, 12);`
- `JButton jbtOK = new JButton("OK");`
`jbtOK.setFont(font1);`

Enum Fonts

```
GraphicsEnvironment e =  
GraphicsEnvironment.getLocalGraphicsEnvironment();  
String[] fontnames = e.getAvailableFontFamilyNames();  
for (int i = 0; i < fontnames.length; i++)  
    System.out.println(fontnames[i]);
```

FontMetrics



- In Graphics:
 - `FontMetrics getFontMetrics(Font font)`
 - `FontMetrics getFontMetrics()`
- TestCenterMessage.java

FontMetrics

- `public int getAscent() // Return the ascent`
- `public int getDescent() // Return the descent`
- `public int stringAscent(String str) // Return the ascent of the string`
- `public int stringWidth(String str) // Return the width of the string`
- `public int getHeight() // Return the height`

This is a MessagePanel object



```
(xCoordinate, yCoordinate)
xCoordinate = getWidth / 2 - stringWidth / 2;
yCoordinate = getHeight / 2 - stringAscent / 2;
```

MessagePanel.java
TestMessagePanel.java

Images as Icons

- `ImageIcon icon = new ImageIcon("image/us.gif");
JLabel lblImage = new JLabel(icon);`
- `Image image = icon.getImage();`
- `g.drawImage(image, 0, 0, getWidth(), getHeight(),
this);`

drawImage

java.awt.Graphics

```
+drawImage(image: Image, x: int, y: int,  
    bgcolor: Color, observer:  
    ImageObserver): void  
  
+drawImage(image: Image, x: int, y: int,  
    observer: ImageObserver): void  
  
+drawImage(image: Image, x: int, y: int,  
    width: int, height: int, observer:  
    ImageObserver): void  
  
+drawImage(image: Image, x: int, y: int,  
    width: int, height: int, bgcolor: Color,  
    observer: ImageObserver): void
```

- JPanel is a kind of ImageObserver

DisplayImage.java