GUI Programming with Java

Graphics and Java 2D

- We will look at…
  - Basic Geometry
  - Drawing in SWING

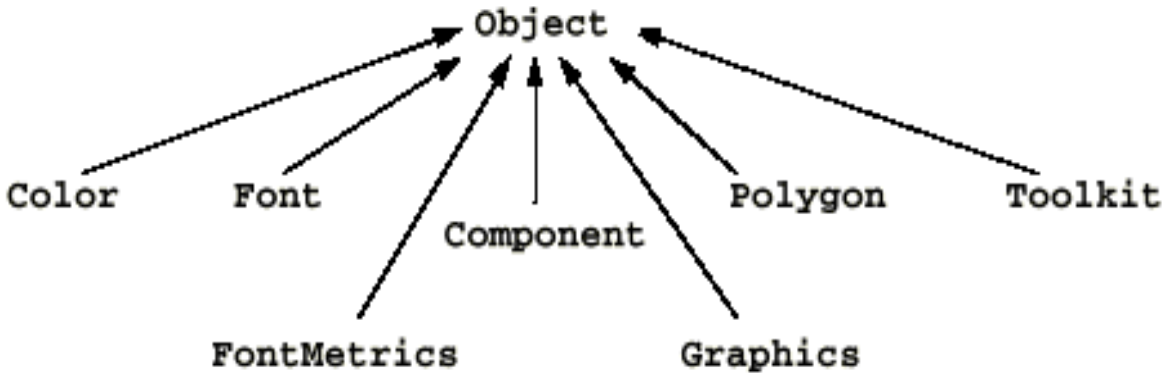
Outline

In this session we will look at

- The graphics capabilities of Java
- The color class
- Drawing simple shapes
- Drawing filled and 3D shapes
- Drawing arcs
- Drawing polygons and polylines

Introduction

- The graphics capabilities of Java for drawing on the screen are shown in the diagram below.



A portion of the **java.awt** hierarchy.

- This shows a portion of the **java.awt** class hierarchy.

- Each class in the figure inherits directly from the class Object.

Introduction

- Class **Color** contains methods and constants for manipulating colours.
- Class **Font** contains methods and constants for manipulating fonts.
- Class **FontMetrics** contains methods and constants for obtaining font information about the fonts on your system.
- Class **Polygon** contains methods and constants for creating polygon shapes.
- Class **Graphics** contains methods for drawing strings, lines, rectangles and other shapes.
- Class **Toolkit** provides methods for getting graphical information from a system such as the set of displayable fonts and the display screen resolution.

Color Control

- Color enhances the appearance of an application.
- Class Color defines methods and constants for manipulating colors in a Java program.
- Every color is created from a Red, Green and Blue component. Together these are called RGB values.
- The larger the RGB value, the greater the amount of particular color.
- Enables a programmer to choose between 256 X 256 X 256 colors (or 16.7 million colors).

Color Control

- Number of useful methods within the class

- public Color (int r, int g, int b)
  - Creates a color based on red, green and blue expressed as integers.

- public Color (float r, float g, float b)
  - Creates a color based on red, green and blue expressed as floating point values between 0.0 and 1.0

- int getRed()
  - Returns a value between 0 and 255 representing the red content.

Color Control

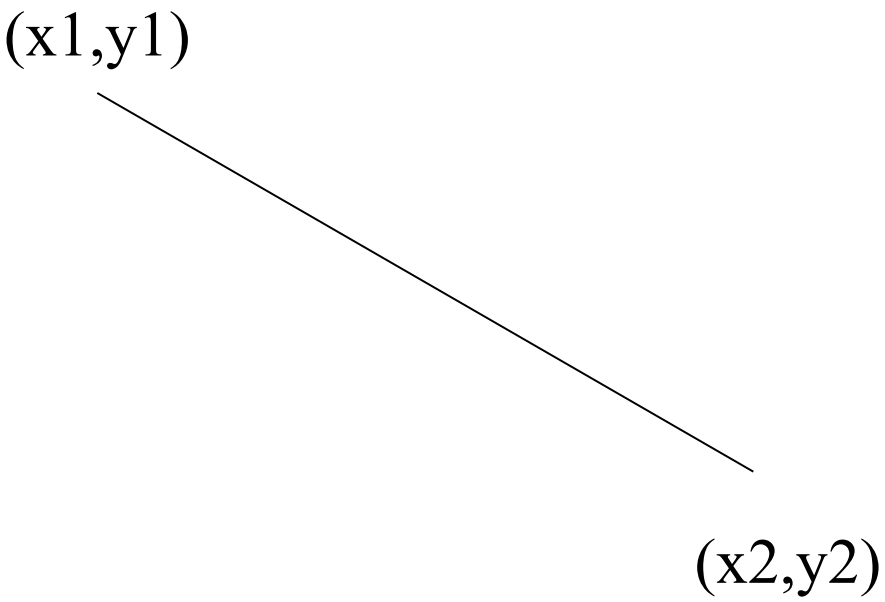
- public int getBlue()
  - Returns a value between 0 and 255 representing the blue content.
- public int getGreen()
  - Returns a value between 0 and 255 representing the green content.
- public Color getColor()
  - Returns a color object representing the current color in the graphics context.
- public void setColor(Color c)
  - Sets a color object representing the current color in the graphics context.

Color Control

- Lets take a look at an example of using the Color class in java.
  - example1\ShowColors.java
- One of the new features of Java is the predefined GUI component JColorChooser. Lets take a look at it in
  - example2\ShowColors2.java

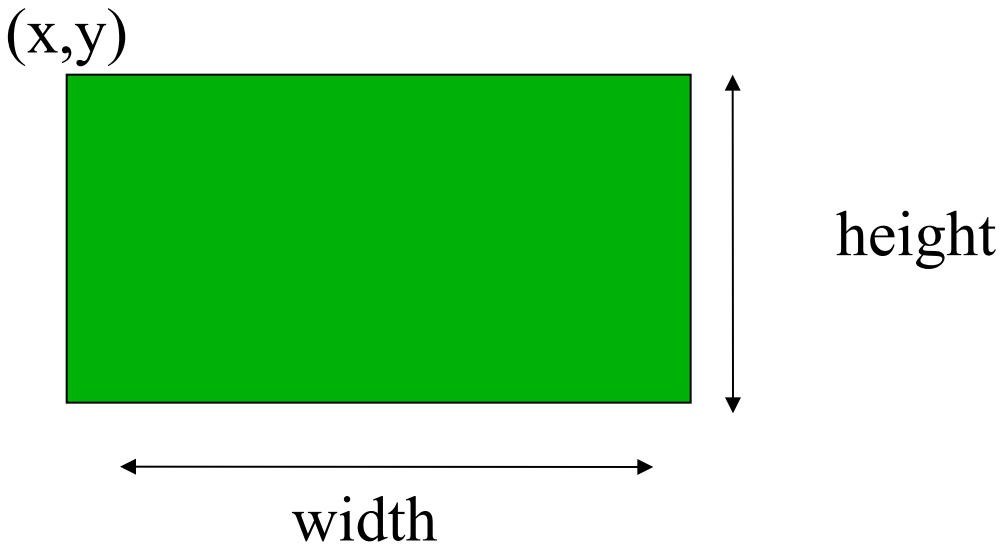
Drawing Shapes - Line

- public void drawLine (int x1, int y1, int x2, int y2)
  - Draws a line between the pint (x1,y1) and (x2,y2)



Drawing Shapes - Rectangle

- public void drawRect (int x, int y, int width, int height)
  - Draws a rectangle of specified width and height. The top left corner of the rectangle has the coordinates (x,y)



Drawing Shapes - Rectangle

- There are several other types of rectangle we can draw

- public void fillRect (int x, int y, int width, int height)
  - Draws a **solid** rectangle of specified width and height. The top left corner of the rectangle has the coordinates (x,y)

- public void clearRect (int x, int y, int width, int height)
  - Draws a **clear** rectangle of specified width and height. The top left corner of the rectangle has the coordinates (x,y)

- public void drawRoundRect (int x, int y, int width, int height, int arcWidth, int arcHeight)
  - Draws a rectangle of specified width and height. The top left corner of the rectangle has the coordinates (x,y). The values of arcWidth and arcHeight determine the rounding of the corners.

Drawing Shapes - Rectangle

- public void fillRoundRect (int x, int y, int width, int height, int arcWidth, int arcHeight)
  - Draws a **solid** rectangle of specified width and height. The top left corner of the rectangle has the coordinates (x,y). The values of arcWidth and arcHeight determine the rounding of the corners.

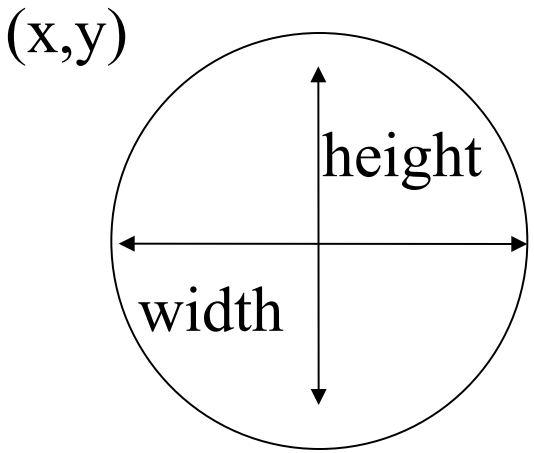
- public void draw3DRect (int x, int y, int width, int height, boolean b)
  - Draws a **3 dimensional** rectangle of specified width and height. The top left corner of the rectangle has the coordinates (x,y). The rectangle appears raised when b is true and lowered when b is false.

Drawing Shapes - Rectangle

- public void fill3DRect (int x, int y, int width, int height, boolean b)
  - Draws a **solid 3 dimensional** rectangle of specified width and height. The top left corner of the rectangle has the coordinates (x,y). The rectangle appears raised when b is true and lowered when b is false.

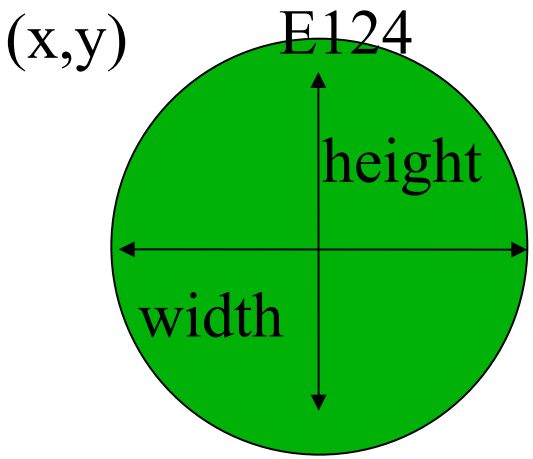
Drawing Shapes - Ovals

- public void drawOval (int x, int y, int width, int height)
  - Draws an oval in the current color of specified width and height. The bounding rectangles top left corner of the oval has the coordinates (x,y)



Drawing Shapes - Ovals

- public void fillOval (int x, int y, int width, int height)
  - Draws a filled oval in the current color of specified width and height. The bounding rectangles top left corner of the oval has the coordinates (x,y)



Drawing Shapes

- Lets take a look at an example of drawing shapes in java.
  - example3\LinesRectsOvals.java

Drawing Arcs

- An arc is a portion of a circle.
- Arc angles are measured in degrees.
- An arc is drawn between two angles: a starting angles and an arc angle
- The **starting angle** is the degree where the arc begins
- The **arc angle** is the last degree of the arc.
- Arcs **sweep** between their starting angle and their arc angle.
- Arcs that sweep in a:
- counter-clockwise direction are measured in positive degrees.
- clockwise direction are measured in negative degrees.

Drawing Arcs

- Arcs that sweep in a:
  - counter-clockwise direction are measured in positive degrees.
  - clockwise direction are measured in negative degrees.

Positive and negative arc angles.

Graphics methods for drawing arcs

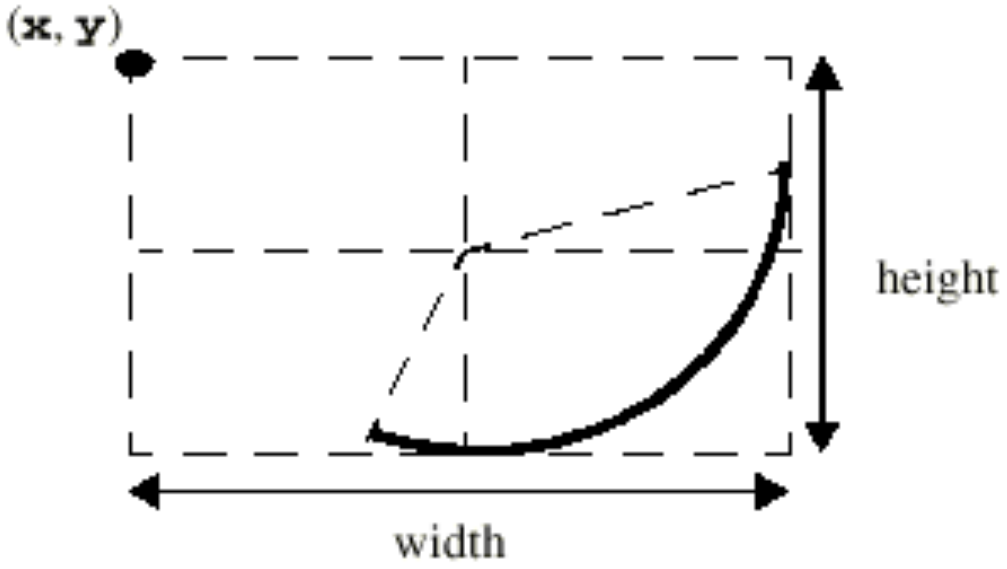
```
public abstract void drawArc(          // Graphics class
    int x,             // x coordinate
    int y,             // y coordinate
    int width,         // arc width
    int height,        // arc height
    int startAngle,    // beginning angle
    int arcAngle )     // arc angle
```

Draws an arc in the current color relative to the bounding rectangle's top-left coordinates (**x**, **y**) with the specified **width** and **height**. The arc segment is drawn from the starting angle to the arc angle.

```
public abstract void fillArc(          // Graphics class
    int x,             // x coordinate
    int y,             // y coordinate
    int width,         // arc width
    int height,        // arc height
    int startAngle,    // beginning angle
    int arcAngle )     // arc angle
```

Draws a solid arc in the current color relative to the bounding rectangle's top-left coordinates (**x**, **y**) with the specified width and height. The arc segment is drawn from the starting angle to the arc angle.

Drawing Arcs



An arc bounded by a rectangle.

- Take a look at example4\ArcExample.java

Drawing Polygons and PolyLines

- Polygons are multisided shapes.
- Polylines are a series of connected points.
- Lets take a look at some of the methods we can use to create these in java.
- When you are familiar with these take a look at example5\DrawPolygons.java

`Graphics` methods for drawing polygons and the `Polygon` constructors

```
public abstract void drawPolygon(     // Graphics class
    int xPoints[],    // x coordinates
    int yPoints[],    // y coordinates
    int points )      // number of points
```

Draws a polygon of **`points`** in the current color. The *x* coordinate of each point is specified in the **`xPoints`** array and the *y* coordinate of each point is specified in the **`yPoints`** array.

```
public abstract void drawPolyline(    // Graphics class
    int xPoints[],    // x coordinates
    int yPoints[],    // y coordinates
    int points )      // number of points
```

Draws a series of connected lines in the current color. The *x* coordinate of each point is specified in the **`xPoints`** array and the *y* coordinate of each point is specified in the **`yPoints`** array. The last argument specifies the number of **`points`**.

```
public void drawPolygon( Polygon p ) // Graphics class
```

Draws a polygon in the current color.

& .....

```
public abstract void fillPolygon(     // Graphics class
    int xPoints[],    // x coordinates
    int yPoints[],    // y coordinates
    int points )      // number of points
```

Draws a solid polygon of **points** in the current color. The *x* coordinate of each point is specified in the **xPoints** array and the *y* coordinate of each point is specified in the **yPoints** array.

```
public void fillPolygon( Polygon p ) // Graphics class
```

Draws a filled polygon in the current color.

```
public Polygon()              // Polygon class
```

Constructs a new polygon object. The polygon does not contain any points.

```
public Polygon(               // Polygon class
    int xValues[],            // x coordinates
    int yValues[],            // y coordinates
    int numberOfPoints )      // number of points
```

Constructs a new polygon object. The polygon has **numberOfPoints** sides with each point consisting of an *x* coordinate from **xValues** and a *y* coordinate from **yValues**.

Summary

In this session we have looked at

- The 2D graphics capabilities of Java
- The color class
- Drawing simple shapes
- Drawing filled and 3D shapes
- Drawing arcs
- Drawing polygons and polylines