Java GUI Programming
Part 3

Chapter 13 (Graphics and Java 2D)
P. Deitel , H. Deitel - Java How To Program, 10th Edition

Edited by Ehsan Edalat

13.1 Introduction

- Overview capabilities for drawing two-dimensional shapes, controlling colors and controlling fonts.
- One of Java's initial appeals was its support for graphics that enabled programmers to visually enhance their applications.

13.1 Introduction (cont.)

13.1 Introduction (cont.)

- Class Color contains methods and constants for manipulating colors.
- Class `JComponent` contains method `paintComponent`, which is used to draw graphics on a component.
- Class Font contains methods and constants for manipulating fonts.
- Class FontMetrics contains methods for obtaining font information.
- Class Graphics contains methods for drawing strings, lines, rectangles and other shapes.
- Class Graphics2D, which extends class `Graphics`, is used for drawing with the Java 2D API.

13.1 Introduction (cont.)

- Class Polygon contains methods for creating polygons.
- Class BasicStroke helps specify the drawing characteristics of lines.
- Classes GradientPaint and TexturePaint help specify the characteristics for filling shapes with colors or patterns.
- Classes `GeneralPath`, `Line2D`, `Arc2D`, `Ellipse2D`, `Rectangle2D` and `RoundRectangle2D` represent several Java 2D shapes.

13.1 Introduction (cont.)

- Coordinate system (Fig. 13.2)
  - a scheme for identifying every *point* on the screen.
- The *upper-left corner* of a GUI component (e.g., a window) has the coordinates (0, 0).
- A coordinate pair is composed of an *x*-coordinate (the horizontal coordinate) and a *y*-coordinate (the vertical coordinate).
  - *x*-coordinates from left to right.
  - *y*-coordinates from top to bottom.
- The *x*-axis describes every horizontal coordinate, and the *y*-axis every vertical coordinate.
- Coordinate units are measured in pixels.
  - A pixel is a display monitor's smallest unit of resolution.

13.1 Introduction (cont.)

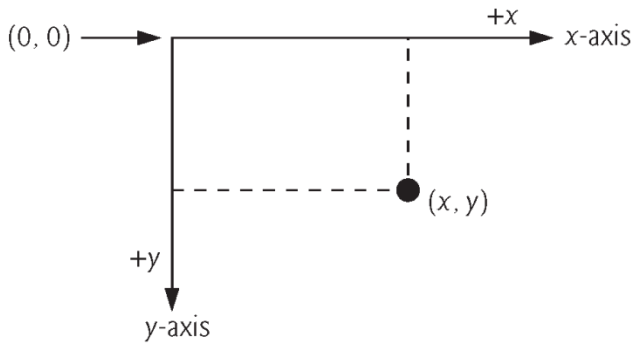

Fig. 13.2 | Java coordinate system. Units are measured in pixels.

13.2 Graphics Contexts and Graphics Objects

- A graphics context enables drawing on the screen.
- A `Graphics` object manages a graphics context and draws pixels on the screen.
- `Graphics` objects contain methods for *drawing*, *font manipulation*, *color manipulation* and the like.
- Class `JComponent` (package `javax.swing`) contains a `paintComponent` for drawing graphics.
  - Takes a `Graphics` object as an argument.
  - Passed to the `paintComponent` method by the system when a lightweight Swing component needs to be repainted.

13.3 Color Control

- `Graphics` method fillRect draws a *filled rectangle* in the current color.
- Four arguments:
  - The first two integer values represent the upper-left x-coordinate and upper-left *y*-coordinate, where the `Graphics` object begins drawing the rectangle.
  - The third and fourth arguments are nonnegative integers that represent the width and the height of the rectangle in pixels, respectively.
- A rectangle drawn using method `fillRect` is filled by the current color of the `Graphics` object.
- `Graphics` method drawString draws a `String` in the current color.

13.3 Color Control

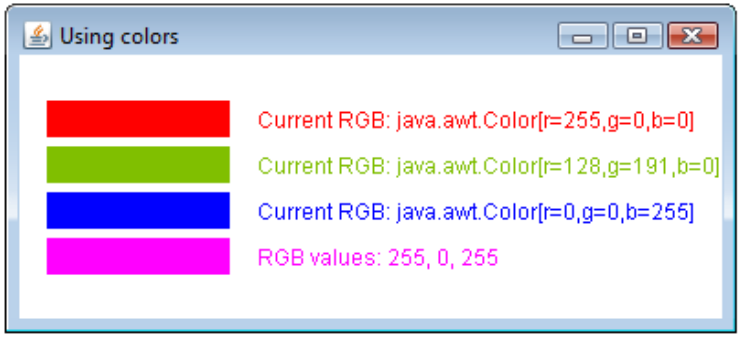

Fig. 13.6 | Demonstrating **Colors**. (Part 2 of 2.)

13.5 Drawing Lines, Rectangles and Ovals

- This section presents `Graphics` methods for drawing lines, rectangles and ovals.
- The methods and their parameters are summarized in Fig. 13.17.

Method	Description
`public void drawLine(int x1, int y1, int x2, int y2)`	Draws a line between the point (`x1`, `y1`) and the point (`x2`, `y2`).
`public void drawRect(int x, int y, int width, int height)`	Draws a rectangle of the specified `width` and `height`. The rectangle's top-left corner is located at (`x`, `y`). Only the outline of the rectangle is drawn using the `Graphics` object's color—the body of the rectangle is not filled with this color.
`public void fillRect(int x, int y, int width, int height)`	Draws a filled rectangle in the current color with the specified `width` and `height`. The rectangle's top-left corner is located at (`x`, `y`).
`public void clearRect(int x, int y, int width, int height)`	Draws a filled rectangle with the specified `width` and `height` in the current background color. The rectangle's *top-left* corner is located at (`x`, `y`). This method is useful if you want to remove a portion of an image.

Method	Description
`public void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)`	Draws a rectangle with rounded corners in the current color with the specified `width` and `height`. The `arcWidth` and `arcHeight` determine the rounding of the corners (see Fig. 13.20). Only the outline of the shape is drawn.
`public void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)`	Draws a filled rectangle in the current color with rounded corners with the specified `width` and `height`. The `arcWidth` and `arcHeight` determine the rounding of the corners (see Fig. 13.20).
`public void draw3DRect(int x, int y, int width, int height, boolean b)`	Draws a three-dimensional rectangle in the current color with the specified `width` and `height`. The rectangle's *top-left* corner is located at (`x`, `y`). The rectangle appears raised when `b` is true and lowered when `b` is false. Only the outline of the shape is drawn.

Method	Description
`public void fill3DRect(int x, int y, int width, int height, boolean b)`	Draws a filled three-dimensional rectangle in the current color with the specified `width` and `height`. The rectangle's *top-left* corner is located at (`x`, `y`). The rectangle appears raised when `b` is true and lowered when `b` is false.
`public void drawOval(int x, int y, int width, int height)`	Draws an oval in the current color with the specified `width` and `height`. The bounding rectangle's *top-left* corner is located at (`x`, `y`). The oval touches all four sides of the bounding rectangle at the center of each side (see Fig. 13.21). Only the outline of the shape is drawn.
`public void fillOval(int x, int y, int width, int height)`	Draws a filled oval in the current color with the specified `width` and `height`. The bounding rectangle's *top-left* corner is located at (`x`, `y`). The oval touches the center of all four sides of the bounding rectangle (see Fig. 13.21).

Fig. 13.17 | `Graphics` methods that draw lines, rectangles and ovals. (Part 3 of 3.)

13.5 Drawing Lines, Rectangles and Ovals

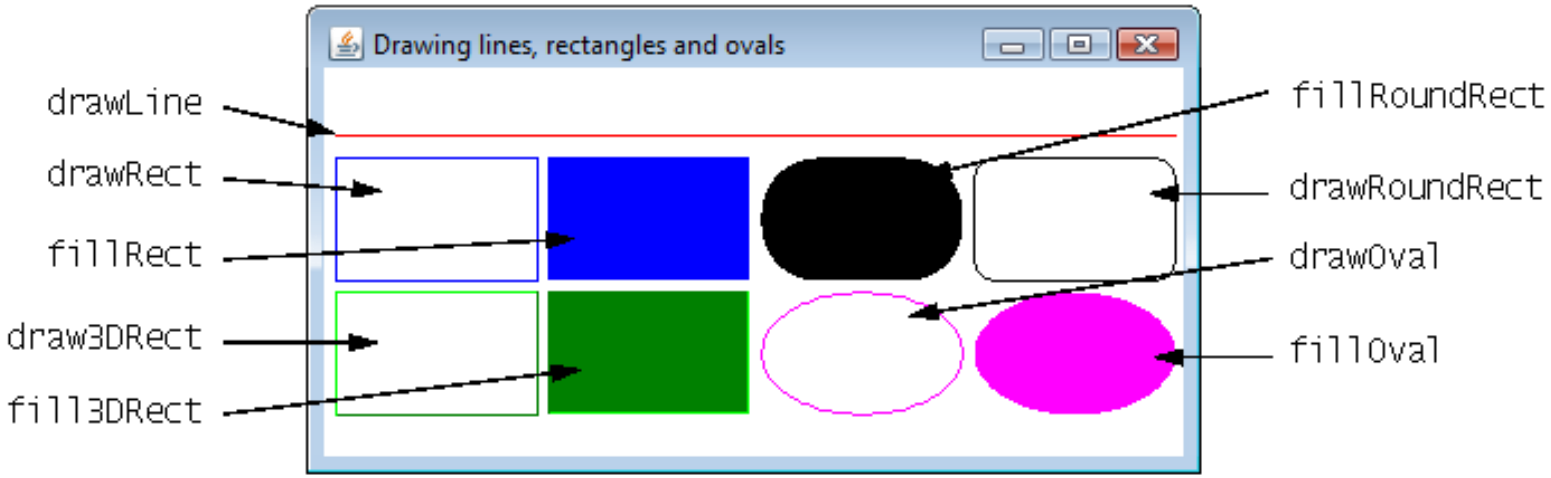

Fig. 13.19 | Testing `LinesRectsOvalsJPanel`. (Part 2 of 2.)

13.8 Java 2D API

- The Java 2D API provides advanced two-dimensional graphics capabilities for programmers who require detailed and complex graphical manipulations.
- For an overview, visit
  - http://docs.oracle.com/javase/7/docs/technotes/guides/2d/
- Drawing with the Java 2D API is accomplished with a Graphics2D reference (package `java.awt`).
- To access `Graphics2D` capabilities, we must cast the `Graphics` reference (`g`) passed to `paintComponent` into a `Graphics2D` reference with a statement such as
  - `Graphics2D g2d = ( Graphics2D ) g;`

13.8 Java 2D API (cont.)

- Example demonstrates several Java 2D shapes from package `java.awt.geom`, including Line2D.Double, Rectangle2D.Double, RoundRectangle2D.Double, Arc2D.Double and Ellipse2D.Double.

13.8 Java 2D API (cont.)

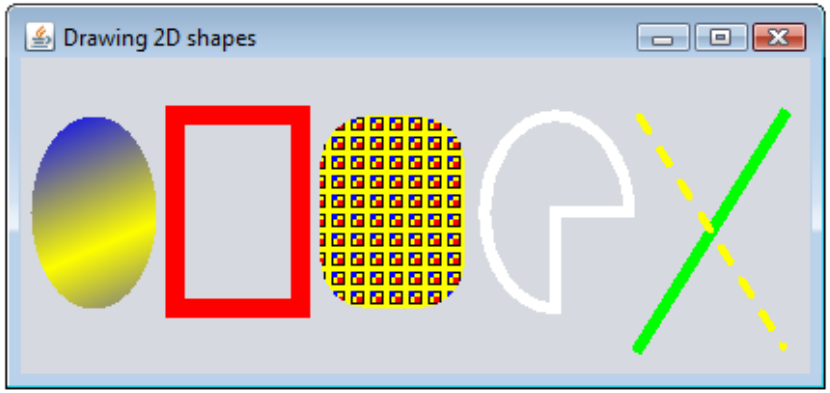

Fig. 13.30 | Testing ShapesJPanel. (Part 2 of 2.)

13.8 Java 2D API (cont.)

- `Graphics2D` method setPaint sets the Paint object that determines the color for the shape to display.
- A `Paint` object implements interface `java.awt.Paint`.
  - Can be something one of the predeclared `Color`, or it can be an instance of the Java 2D API's `GradientPaint`, SystemColor, `TexturePaint`, `LinearGradientPaint` or `RadialGradientPaint` classes.
- Class `GradientPaint` helps draw a shape in *gradually changing colors*—called a gradient.
- `Graphics2D` method fill draws a filled Shape object—an object that implements interface `Shape` (package `java.awt`).

13.8 Java 2D API (cont.)

- `Graphics2D` method setStroke sets the characteristics of the shape's border (or the lines for any other shape).
  - Requires as its argument an object that implements interface Stroke (package `java.awt`).
- Class `BasicStroke` provides several constructors to specify the width of the line, how the line ends (called the end caps), how lines join together (called line joins) and the dash attributes of the line (if it's a dashed line).
- `Graphics2D` method draw draws a `Shape` object.

13.8 Java 2D API (cont.)

- Class BufferedImage (package `java.awt.image`) can be used to produce images in color and grayscale.
- The third argument BufferedImage.TYPE_INT_RGB indicates that the image is stored in color using the RGB color scheme.
- `BufferedImage` method create-Graphics creates a `Graphics2D` object for drawing into the `BufferedImage`.
- A `TexturePaint` object uses the image stored in its associated `BufferedImage` (the first constructor argument) as the fill texture for a filled-in shape.

13.8 Java 2D API (cont.)

- Constant Arc2D.PIE indicates that the arc is closed by drawing two lines—one line from the arc's starting point to the center of the bounding rectangle and one line from the center of the bounding rectangle to the ending point.
- Constant Arc2D.CHORD draws a line from the starting point to the ending point.
- Constant Arc2D.OPEN specifies that the arc should not be closed.

13.8 Java 2D API (cont.)

- BasicStroke.CAP_ROUND causes a line to have rounded ends.
- If lines join together (as in a rectangle at the corners), use BasicStroke.JOIN_ROUND to indicate a rounded join.

13.8 Java 2D API (cont.)

- General path—constructed from straight lines and complex curves.
- Represented with an object of class GeneralPath (package `java.awt.geom`).
- `GeneralPath` method moveTo moves to the specified point.
- `GeneralPath` method lineTo draws a line from the current point to the specified point.
- `GeneralPath` method closePath draws a line from the last point to the point specified in the last call to `moveTo`.
- `Graphics2D` method translate moves the drawing origin to the specified location.
- `Graphics2D` method rotate rotates the next displayed shape.
  - The argument specifies the rotation angle in radians (with 360° = 2π radians).

Simple Paint Application