Chapter 25 (part 2)

1. Creating a Frame-Based Application

In general, you create a subclass of Frame, override paint( ) to supply your output to the window, and implement the necessary event listeners. In all cases, you will need to implement the windowClosing( ) method of the WindowListener interface.

Once you have defined a Frame subclass, you can create an instance of that class. This causes a frame window to come into existence, but it will not be initially visible. You make it visible by calling setVisible(true).

1.1 Introducing Graphics

The AWT includes several methods that support graphics. All graphics are drawn relative to a window. This can be the main window of an application or a child window. The origin of each window is at the top-left corner and is (0,0). Coordinates are specified in pixels. All output to a window takes place through a graphics context.

A graphics context is encapsulated by the Graphics class. Here are two ways in which a graphics context can be obtained:

• It is passed to a method, such as paint( ) or update( ), as an argument.

• It is returned by the getGraphics( ) method of Component.

Graphics class defines a number of methods that draw various types of objects, such as lines, rectangles, and arcs. When a graphics object is drawn that exceeds the dimensions of the window, output is automatically clipped.

Drawing Lines

Lines are drawn by means of the drawLine( ) method, shown here:

- void drawLine(int startX, int startY, int endX, int endY)

Displays a line in the current drawing color that begins at startX, startY and ends at endX, endY.

Drawing Rectangles

The drawRect( ) and fillRect( ) methods display an outlined and filled rectangle, respectively. They are shown here:

- void drawRect(int left, int top, int width, int height)
- void fillRect(int left, int top, int width, int height)

The upper-left corner of the rectangle is at left, top. The dimensions of the rectangle are specified by width and height.

To draw a rounded rectangle, use drawRoundRect( ) or fillRoundRect( ), both shown here:

- void drawRoundRect(int left, int top, int width, int height, int xDiam, int yDiam)
- void fillRoundRect(int left, int top, int width, int height, int xDiam, int yDiam)
  A rounded rectangle has rounded corners. The diameter of the rounding arc along the X axis is specified by xDiam. The diameter of the rounding arc along the Y axis is specified by yDiam.

Drawing Ellipses and Circles

To draw an ellipse, use drawOval( ). To fill an ellipse, use fillOval( ). These methods are shown here:

- void drawOval(int left, int top, int width, int height)
- void fillOval(int left, int top, int width, int height)

The ellipse is drawn within a bounding rectangle whose upper-left corner is specified by left, top and whose width and height are specified by width and height. To draw a circle, specify a square as the bounding rectangle.

Drawing Arcs

Arcs can be drawn with drawArc( ) and fillArc( ), shown here:

- void drawArc(int left, int top, int width, int height, int startAngle, int sweepAngle)
- void fillArc(int left, int top, int width, int height, int startAngle, int sweepAngle)

The arc is bounded by the rectangle whose upper-left corner is specified by left, top and whose width and height are specified by width and height. The arc is drawn from startAngle through the angular distance specified by sweepAngle. Angles are specified in degrees. Zero degrees is on the horizontal, at the three o'clock position. The arc is drawn counterclockwise if sweepAngle is positive, and clockwise if sweepAngle is negative. Therefore, to draw an arc from twelve o'clock to six o'clock, the start angle would be 90 and the sweep angle 180.

Drawing Polygons

It is possible to draw arbitrarily shaped figures using drawPolygon( ) and fillPolygon( ), shown here:

- void drawPolygon(int x[ ], int y[ ], int numPoints)
- void fillPolygon(int x[ ], int y[ ], int numPoints)

Reference--Schildt, H. (2018). Java: The Complete Reference. 10th edition. McGraw-Hill Education.

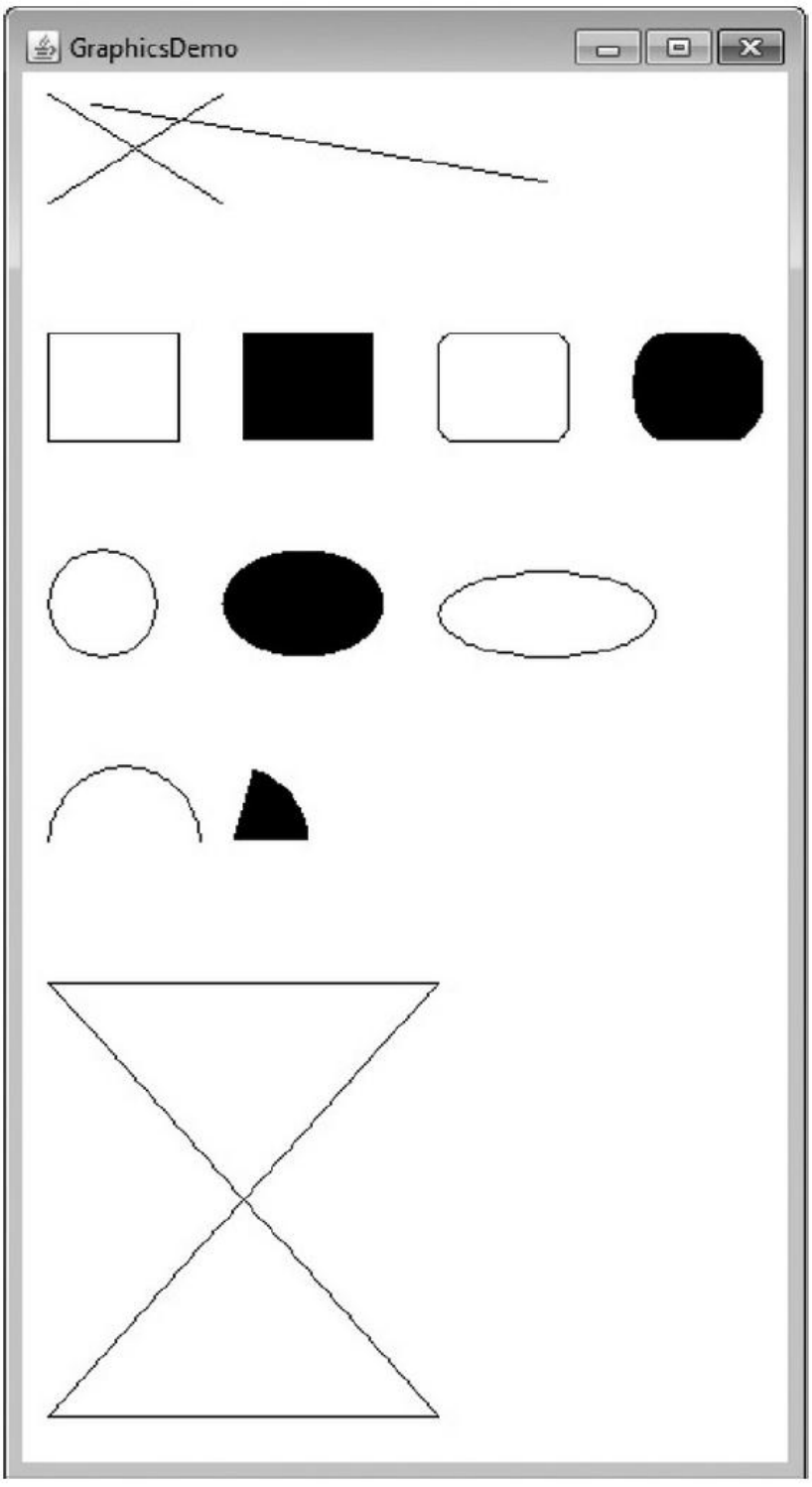
The polygon's endpoints are specified by the coordinate pairs contained within the x and y arrays. The number of points defined by these arrays is specified by numPoints. There are alternative forms of these methods in which the polygon is specified by a Polygon object.

Demonstrating the Drawing Methods

Reference--Schildt, H. (2018). Java: The Complete Reference. 10th edition. McGraw-Hill Education.

```
// Draw graphics elements.
import java.awt.event.*;
import java.awt.*;

public class GraphicsDemo extends Frame {

  public GraphicsDemo() {
    // Anonymous inner class to handle window close events.
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent we) {
        System.exit(0);
      }
    });
  }

  public void paint(Graphics g) {

    // Draw lines.
    g.drawLine(20, 40, 100, 90);
    g.drawLine(20, 90, 100, 40);
    g.drawLine(40, 45, 250, 80);

    // Draw rectangles.
    g.drawRect(20, 150, 60, 50);
    g.fillRect(110, 150, 60, 50);
    g.drawRoundRect(200, 150, 60, 50, 15, 15);
    g.fillRoundRect(290, 150, 60, 50, 30, 40);

    // Draw elipses and circles.
    g.drawOval(20, 250, 50, 50);
    g.fillOval(100, 250, 75, 50);
    g.drawOval(200, 260, 100, 40);

    // Draw arcs.
    g.drawArc(20, 350, 70, 70, 0, 180);
    g.fillArc(70, 350, 70, 70, 0, 75);

    // Draw a polygon.
    int xpoints[] = {20, 200, 20, 200, 20};
    int ypoints[] = {450, 450, 650, 650, 450};
    int num = 5;

    g.drawPolygon(xpoints, ypoints, num);
  }

  public static void main(String[] args) {
    GraphicsDemo appwin = new GraphicsDemo();
```

```
        appwin.setSize(new Dimension(370, 700));
        appwin.setTitle("GraphicsDemo");
        appwin.setVisible(true);
    }
}
```



Reference--Schildt, H. (2018). Java: The Complete Reference. 10th edition. McGraw-Hill Education.