DRAWING OVALS, ARCS AND POLYGONS

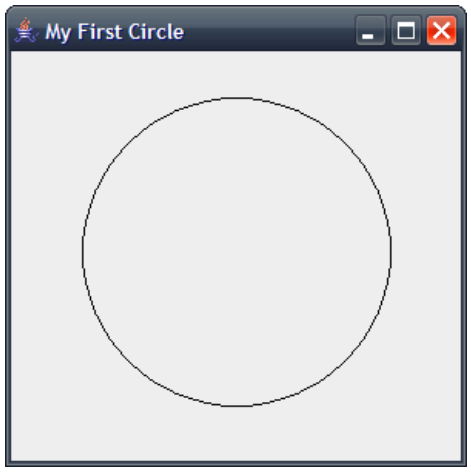
DRAWING OVALS

Drawing oval shapes is very similar to drawing rectangles. The **Ellipse2D.Double** class is what you will need to use to draw an oval shape. Drawing an ellipse is very similar to drawing a rectangle in that you need to specify the upper-left coordinate (x, y) and the width and height of the shape as follows:

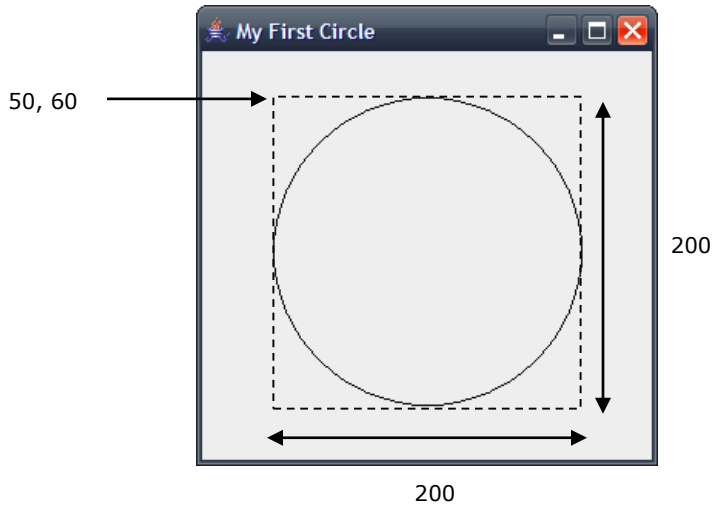
```
g2.draw(new Ellipse2D.Double(x, y, width, height));
```

The following line of code will draw a 200 x 200 oval at the centre of the frame:

```
g2.draw(new Ellipse2D.Double(50, 60, 200, 200));
```



The **Ellipse2D.Double** class draws an oval within a rectangle whose upper-left coordinate is specified by the x- and y-coordinates as follows:



DRAWING ARCS

An arc is a segment of an ellipse that is contained within a rectangular area. So to draw an arc you need to specify the upper-left coordinate (x, y) coordinate of the rectangular area, the width and height of the area, the angle (in degrees) at which the arc begins and the size (in degrees) of the arc, and the type of arc you want to create:

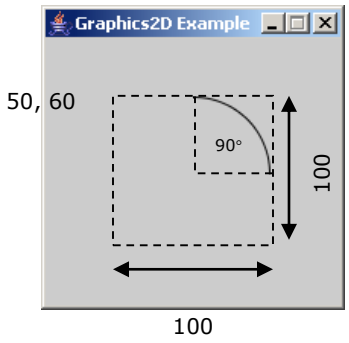
```
g2.draw(new Arc2D.Double(x, y, width, height, startAngle,
   arcAngle, arcType));
```

There are three arc types to choose from:

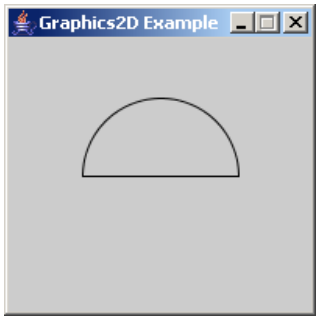
Arc2D.OPEN	Draws an open-ended arc.
Arc2D.CHORD	Draws a closed arc whereby the ends are connected with a straight line.
Arc2D.PIE	Draws a closed arc whereby the ends are connected with a straight line that meet at the center of the ellipse, thus creating a shape that looks like a piece of pie.

The following are examples of each type of arc:

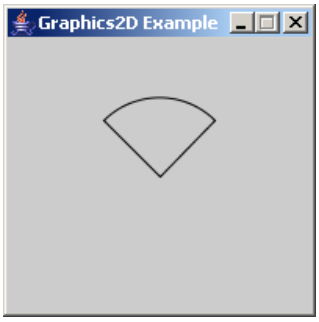
```
g2.draw(new Arc2D.Double(50, 60, 100, 100, 0, 90,
   Arc2D.OPEN));
```




```
g2.draw(new Arc2D.Double(50, 60, 100, 100, 0, 180,
   Arc2D.CHORD));
```



```
g2.draw(new Arc2D.Double(50, 60, 100, 100, 45, 90,
   Arc2D.PIE));
```



The important thing to remember about drawing arcs is that the arc travels counter clockwise from the starting point (which is the halfway point of the right side of the rectangular area).

DRAWING POLYGONS

Polygons are multisided shapes composed of straight-line segments. To draw a polygon you will need to first create a **Polygon** object:

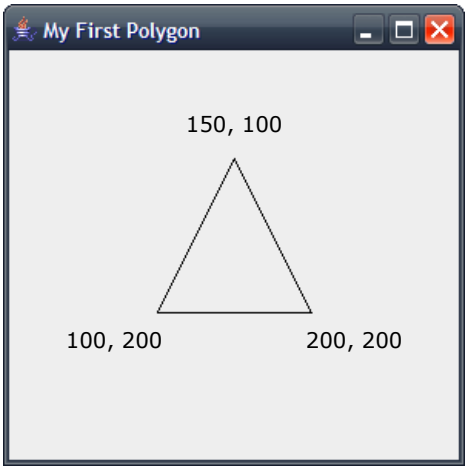
```
Polygon p = new Polygon();
```

You will then need to use the **addPoint()** method to add pairs of x- and y-coordinates to the polygon as follows:

```
p.addPoint(150, 100);
p.addPoint(100, 200);
p.addPoint(200, 200);
```

Once you've added all the points to your polygon you can use the **drawPolygon()** or **fillPolygon()** method to draw the polygon:

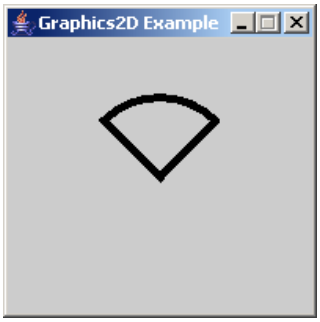
```
g2.drawPolygon(p);
```



SETTING STROKES

You can set the thickness of the lines that are used to draw a shape by using the **setStroke()** method and using the **BasicStroke** class as follows:

```
g2.setStroke(new BasicStroke(5));
g2.draw(new Arc2D.Double(50, 60, 100, 100, 45, 90,
   Arc2D.PIE));
```



The above line of code will set the thickness of my lines to 5.

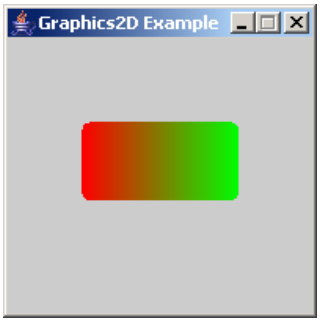
USING GRADIENT FILLS

You are not limited to just using solid colours to fill shapes. The **GradientPaint** class allows you to use a gradient fill by blending two colours. The constructor for creating a **GradientPaint** object is as follows:

```
GradientPaint(double x1, double y1, Color c1, double x2,
   double y2, Color c2)
```

The gradient is set by setting a colour at point x1 and y1 and another colour at point x2 and y2 and then smoothly blending the colours in between those two points. In the following example I will create a round rectangle that blends the colors red and green from left to right:

```
g2.setPaint(new GradientPaint(50, 75, Color.RED, 150, 75,
   Color.GREEN));
g2.fill(new RoundRectangle2D.Double(50, 75, 100, 50, 10,
   10));
```



DRAWING IMAGES

To draw an image onto the frame, you will need to create an **ImageIcon** object, you will need to use the **getImage()** method included in the **ImageIcon** class, and the **drawImage()** method included in the **Graphics2D** class. The constructor for the **ImageIcon** object is:

```
ImageIcon img = new ImageIcon("filename.gif");
```

Once you have created the **ImageIcon** object, you can call the **drawImage()** method which takes the following form:

```
g2.drawImage(img.getImage(), int x, int y, ImageObserver obs);
```

In the following example, I have drawn an image saved in a file called **familyguy.gif**:

```
public void paint(Graphics g) {

   super.paint(g);
   Graphics2D g2 = (Graphics2D) g;
   ImageIcon imgStewie = new ImageIcon("familyguy.gif");
   g2.drawImage(imgStewie.getImage(), 50, 50, this);
}
```

You can also set the size of an image when drawing it by doing the following:

```
g2.drawImage(imgStewie.getImage(), 50, 50,
   imgStewie.getIconWidth() / 2, imgStewie.getIconHeight() /
   2, this);
```

The above line of code takes the width and height of the original image and reduces it by half.