TWO-DIMENSIONAL FIGURES

Two-dimensional (2D) figures can be rendered by a graphics context. Here are the Graphics methods for drawing common figures:

<table>
<thead>
<tr>
<th>java.awt.Graphics Methods to Draw Lines, Rectangles and Ovals</th>
</tr>
</thead>
<tbody>
<tr>
<td>void drawLine( int x1, int y1, int x2, int y2 )</td>
</tr>
<tr>
<td>void drawRect( int x, int y, int width, int height )</td>
</tr>
<tr>
<td>void fillRect( int x, int y, int width, int height )</td>
</tr>
<tr>
<td>void drawOval( int x, int y, int width, int height )</td>
</tr>
<tr>
<td>void fillOval( int x, int y, int width, int height )</td>
</tr>
</tbody>
</table>

Figures are drawn using the graphics context’s currently set color.

drawLine draws a line from points (x1, y1) to (x2, y2).

Example
g.drawLine( 5, 40, 50, 10 );

The drawX methods trace the outline of figure X; the fillX methods fill it with color.

Example
g.drawRect( 50, 40, 75, 75 );
g.fillRect( 150, 40, 75, 75 );
The arguments to the oval and rectangle methods describe this rectangle: the \((x, y)\) coordinate is that of its upper-left corner, its width is along the \(x\)-axis and its height along the \(y\)-axis. The rectangle methods trace this rectangle; the oval methods trace the oval surrounded by it.

**Example**

```java
        g.drawRect( 50, 40, 175, 75 );
        g.drawOval( 50, 40, 175, 75 );
```

**Resizing and Centering**

To resize two-dimensional figure, you must make its dimensions a proportion of the GUI component’s size, which can be accessed using these methods inherited by subclasses of `java.awt.Component`. These must be called within the `paint` method so that they are recalculated whenever the component is resized by the user.

<table>
<thead>
<tr>
<th>Two Dimension Methods of java.awt.Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>int <code>getWidth()</code></td>
</tr>
<tr>
<td>// Return the width of this component in pixels.</td>
</tr>
<tr>
<td>int <code>getHeight()</code></td>
</tr>
<tr>
<td>// Return the height of this component in pixels.</td>
</tr>
</tbody>
</table>

**Example**

These statements draw diagonal lines through the component’s center.

```java
        int \(w\) = getWidth();
        int \(h\) = getHeight();
        g.drawLine( 0, 0, \(w\), \(h\) );
        g.drawLine( 0, \(h\), \(w\), 0 );
```
Example
These statements draw a rectangle whose width and length span half the component’s width and length and whose upper-left corner starts ⅛ of the way across and down the component.

```java
int w = getWidth();
int h = getHeight();
g.fillRect( w/8, h/8, w/2, h/2 );
```

To center a figure, you must calculate the coordinates of its upper-left corner from the figure’s length and width as well as the length and the width of the component.

To determine the specific calculations needed, suppose \( w \) and \( h \) are the width and height of the component, respectively. \( fw \) and \( fh \) are the width and height, respectively, of the figure to be drawn.
The center point of the component is at \( \left( \frac{w}{2}, \frac{h}{2} \right) \). To be centered horizontally, the figure’s upper-left corner must start at \( x = \frac{w}{2} - \frac{fw}{2} \) so that half of it lies to the left of the component’s center point. To center it vertically, its baseline must start at \( y = \frac{h}{2} - \frac{fh}{2} \).

**Example**

These statements draw a rectangle that resizes with the component and is always vertically and horizontally centered.

```java
int w  = getWidth( );
int h  = getHeight( );
g.setColor( Color.blue );
g.drawRect( w/2 - fw/2, h/2 - fh/2, fw, fh );
```
## Exercises

1. Using the code from the resizing example on page 2, write a complete Java applet that draws diagonal lines through the applet’s center as shown to the right. Make sure your solution always connects the corners of the applet.

2. Using the code from the resizing example on page 3, write a complete Java applet that fills a rectangle whose width and length span half the component’s width and length and whose upper-left corner starts ⅛ of the way across and down the component. An example is shown to the right but make sure your solution resizes with the applet.

3. Using the code from the centering example on page 3, write a complete Java applet that draws a rectangle that resizes with the component and is always vertically and horizontally centered. An example is shown to the right but make sure your solution resizes with the applet.

4. Write a complete Java applet that draws two red lines, one vertical and one horizontal. They must resize in proportion to the applet window and always cross at the center point of the applet.
5. Write a complete Java applet that (1) Sets its background color to red, (2) draws a green filled rectangle of height 100 that spans the width of the applet and (3) draws a yellow filled circle of diameter 200 partially hidden by the rectangle.

6. Write a complete Java applet that asks the user for the value of a radius and then displays a red circle of that radius, centered on the applet background. For example:

The circle must always be in the center of the applet even if the applet window is resized.