

# Methods of the **Graphics** class in Java

University of Mount Union

CSC 120

Day 4

# Drawing or Filling a Rectangle or Square:

`g.drawRect( over, down, width, height );`

draws an outline of a rectangle in the current drawing color  
pixels inside the rectangle are unchanged

*over, down*: screen position of upper-left corner of shape

*width*: width of shape

*height*: height of shape

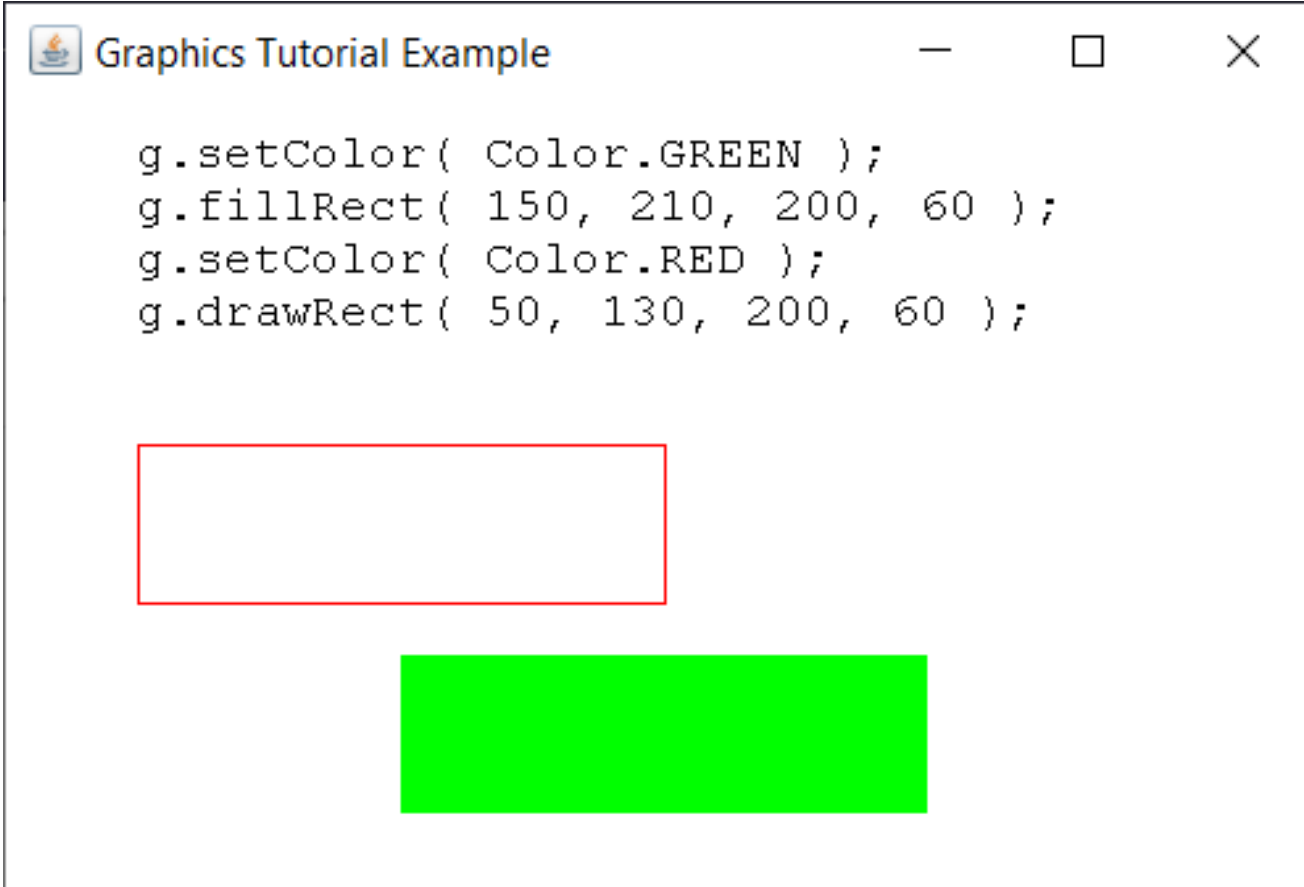
`g.fillRect( over, down, width, height );`

fills a rectangle in the current drawing color, including interior

# Drawing or Filling a Rectangle or Square:

```
Graphics Tutorial Example
```

```
g.setColor( Color.GREEN );  
g.fillRect( 150, 210, 200, 60 );  
g.setColor( Color.RED );  
g.drawRect( 50, 130, 200, 60 );
```



The image shows a Java Swing window titled "Graphics Tutorial Example". The window contains the following code:

```
g.setColor( Color.GREEN );  
g.fillRect( 150, 210, 200, 60 );  
g.setColor( Color.RED );  
g.drawRect( 50, 130, 200, 60 );
```

The output of the code is a window containing two rectangles. The first rectangle is a red outline, and the second rectangle is a solid green fill. The red rectangle is positioned at the top left, and the green rectangle is positioned below and to the right of the red rectangle.

# Drawing or Filling an Oval or Circle:

`g.drawOval( over, down, width, height );`

draws an outline of an oval in the current drawing color that is inscribed in an invisible bounding box for the shape

*over, down*: screen position of upper-left corner of bounding box for the shape

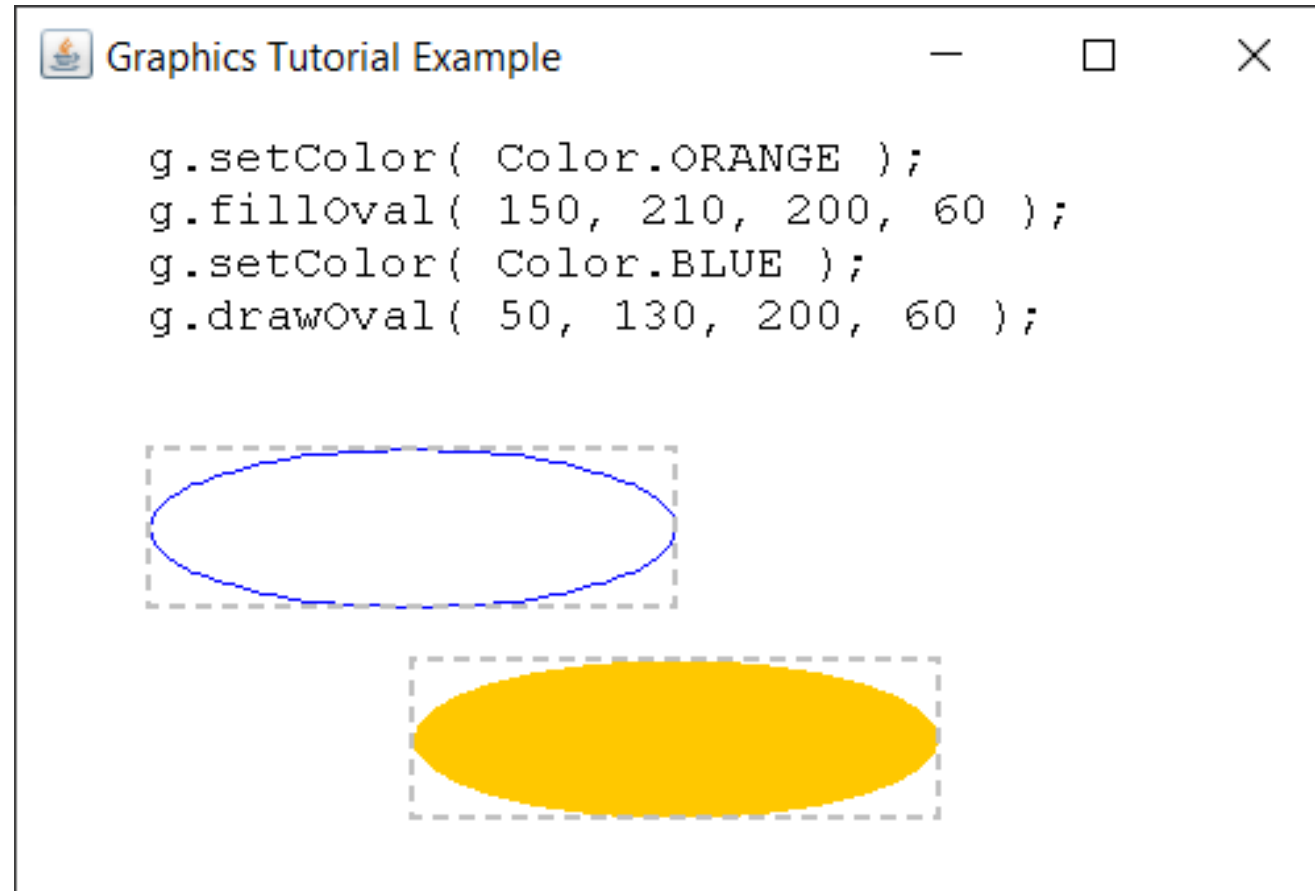
*width*: width of bounding box

*height*: height of bounding box

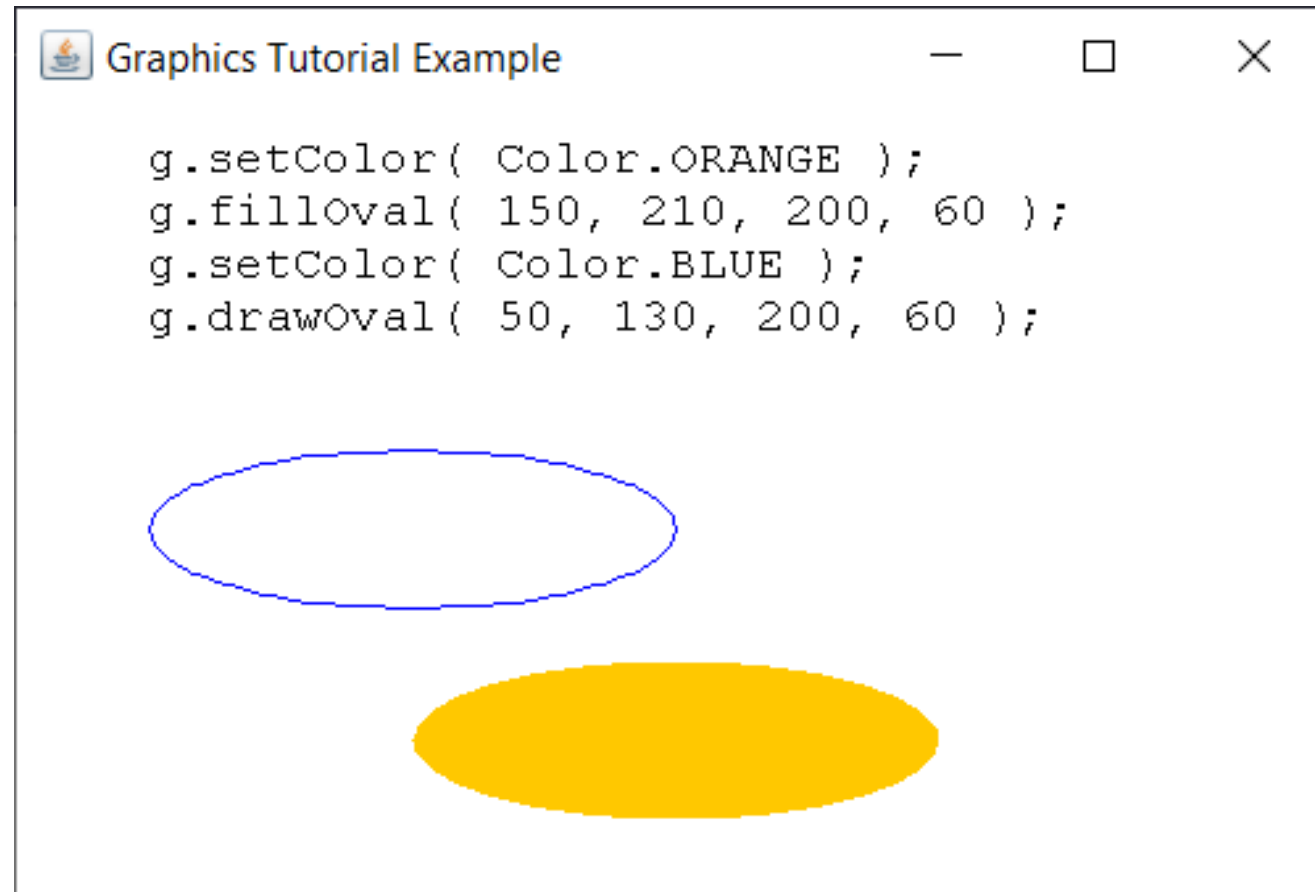
`g.fillOval( over, down, width, height );`

fills an oval in the current drawing color, including interior pixels

# Drawing or Filling an Oval or Circle:



# Drawing or Filling an Oval or Circle: (this is how it would appear on the screen)



# Drawing a Line and Displaying a Text String:

```
g.drawLine( x1, y1, x2, y2 );
```

draws a line between the point (*x1*, *y1*) and the point (*x2*, *y2*) in the current drawing color

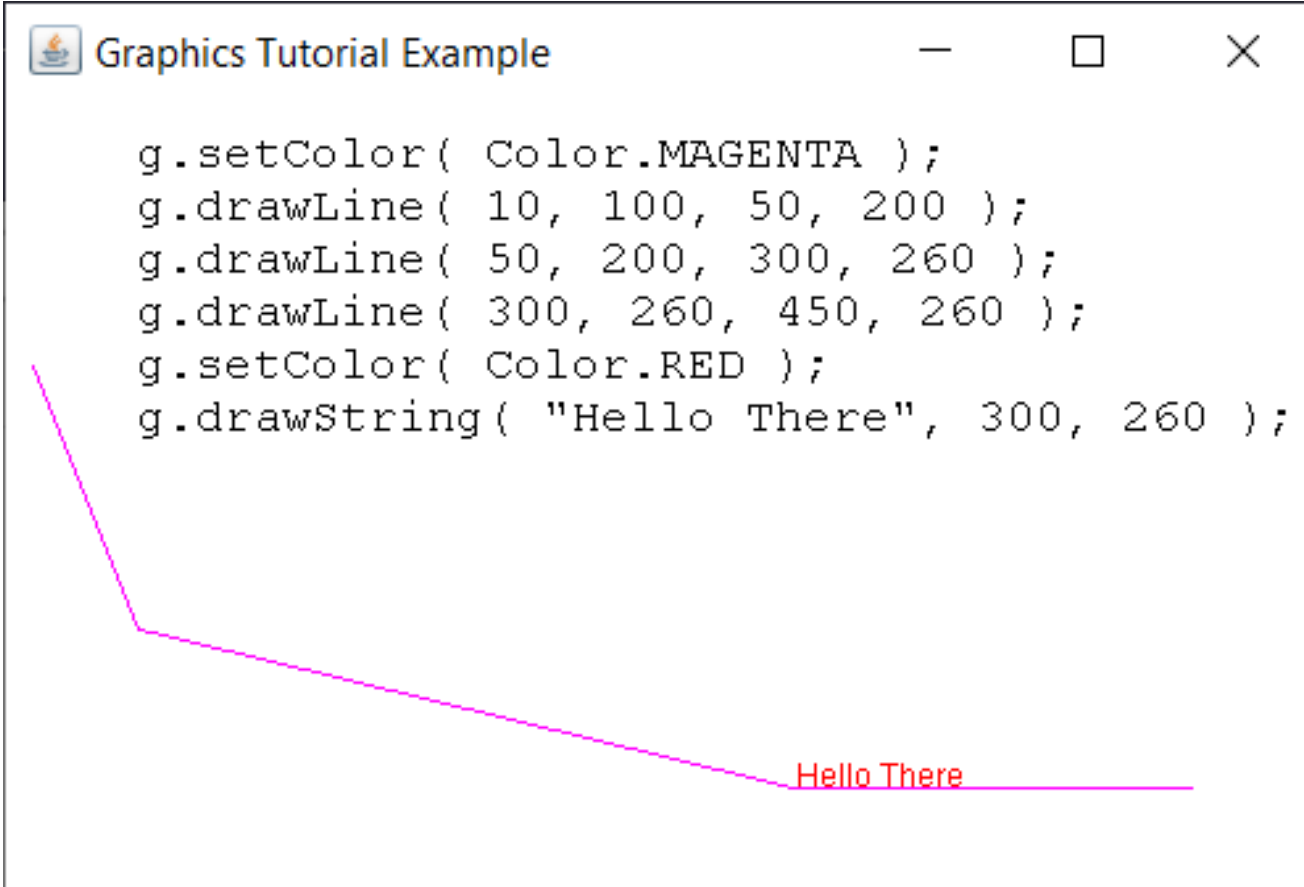
```
g.drawString( text, over, down );
```

displays the *text* on the screen at the point (*over*, *down*)  
(*over*, *down*) is the lower-left corner of the text box

# Drawing a Line and Displaying a Text String:

```
Graphics Tutorial Example
```

```
g.setColor( Color.MAGENTA );  
g.drawLine( 10, 100, 50, 200 );  
g.drawLine( 50, 200, 300, 260 );  
g.drawLine( 300, 260, 450, 260 );  
g.setColor( Color.RED );  
g.drawString( "Hello There", 300, 260 );
```

The image shows a Java Swing window titled "Graphics Tutorial Example" with standard window controls (minimize, maximize, close). The window contains a 2D coordinate system where a magenta line is drawn. The line starts at (10, 100), goes to (50, 200), then to (300, 260), and finally to (450, 260). The text "Hello There" is drawn in red at the coordinates (300, 260).



# Drawing or Filling an Arc:

```
g.drawArc( over, down, width, height, startAngle, arcAngle );
```

draws an outline of an arc of an oval in the current drawing color that is inscribed in an invisible bounding box for the shape, starting at *startAngle* degrees and continuing around the oval for *endAngle* degrees

*over, down*: screen position of upper-left corner of bounding box for the shape

*width*: width of bounding box

*height*: height of bounding box

```
g.fillArc( over, down, width, height, startAngle, arcAngle );
```

# Angle Definition for Java Arcs:

Imagine an invisible set of axes  
inside the invisible bounding box

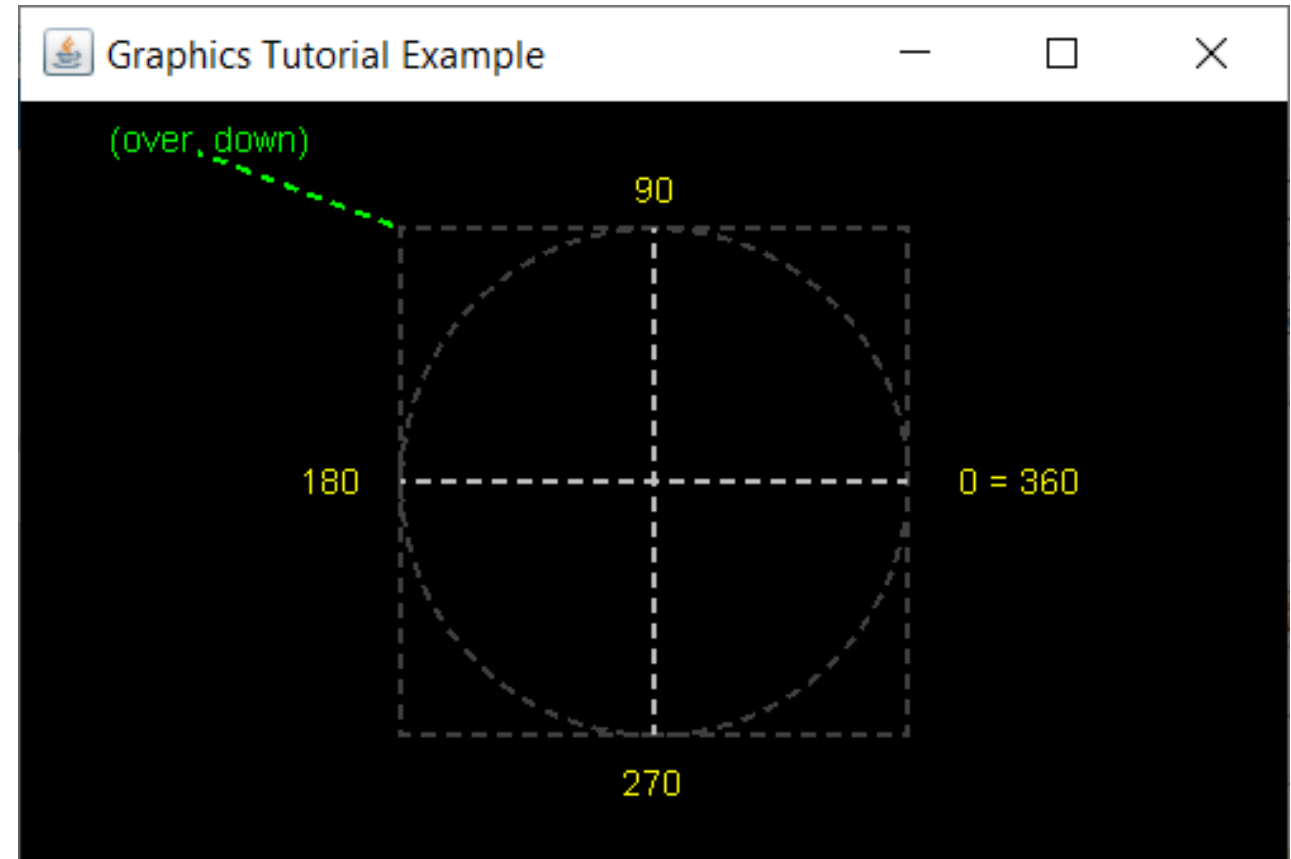
0 degrees = EAST on a compass

90 degrees = NORTH

180 degrees = WEST

270 degrees = SOUTH

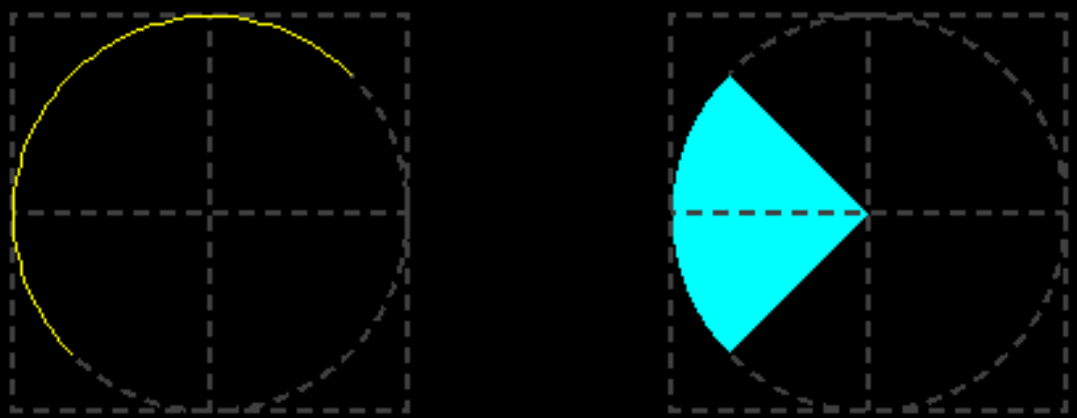
proceed COUNTER-CLOCKWISE  
around the oval as degrees  
increase



# Drawing or Filling an Arc:

```
Graphics Tutorial Example
```

```
g.setColor( Color.YELLOW );  
g.drawArc( 50, 130, 150, 150, 45, 180 );  
g.setColor( Color.CYAN );  
g.fillArc( 300, 130, 150, 150, 135, 90 );
```

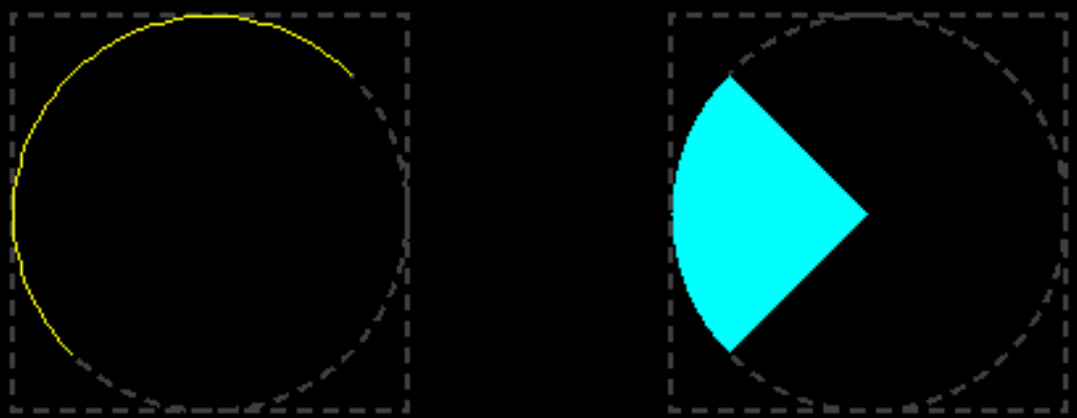


The image shows a window titled "Graphics Tutorial Example" with a black background. It contains four lines of Java code. The first line sets the color to yellow. The second line draws an arc with a bounding box of (50, 130, 150, 150) and an angle range of 45 to 180 degrees. The third line sets the color to cyan. The fourth line fills an arc with a bounding box of (300, 130, 150, 150) and an angle range of 135 to 90 degrees. Below the code, two diagrams illustrate the results. The left diagram shows a yellow arc drawn within a dashed square bounding box. The right diagram shows a cyan arc filled within a dashed square bounding box.

# Drawing or Filling an Arc:

```
Graphics Tutorial Example
```

```
g.setColor( Color.YELLOW );  
g.drawArc( 50, 130, 150, 150, 45, 180 );  
g.setColor( Color.CYAN );  
g.fillArc( 300, 130, 150, 150, 135, 90 );
```

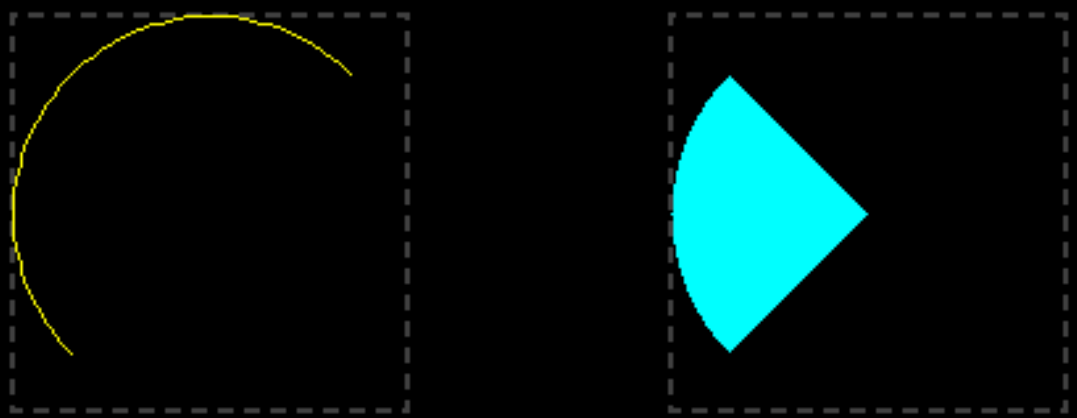


The image shows two side-by-side graphics windows on a black background. The left window displays a yellow arc drawn within a dashed square bounding box. The right window displays a cyan filled arc within a dashed square bounding box.

# Drawing or Filling an Arc:

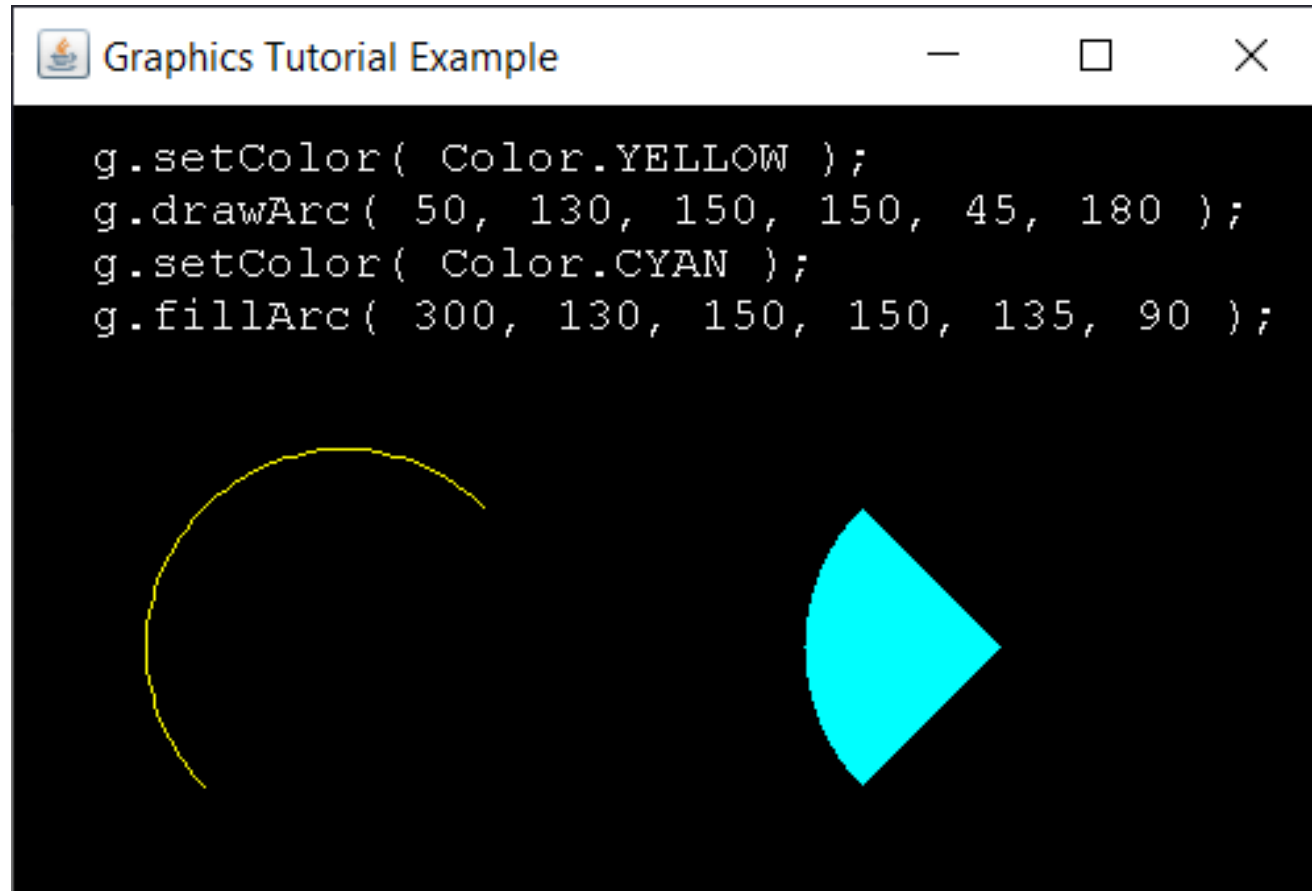
```
Graphics Tutorial Example
```

```
g.setColor( Color.YELLOW );  
g.drawArc( 50, 130, 150, 150, 45, 180 );  
g.setColor( Color.CYAN );  
g.fillArc( 300, 130, 150, 150, 135, 90 );
```



The image shows a window titled "Graphics Tutorial Example" with a black background. It contains two graphical outputs. The first is a yellow arc drawn on a black background, enclosed in a dashed white box. The second is a cyan filled arc on a black background, also enclosed in a dashed white box.

# Drawing or Filling an Arc: (this is how it would appear on the screen)



```
g.setColor( Color.YELLOW );  
g.drawArc( 50, 130, 150, 150, 45, 180 );  
g.setColor( Color.CYAN );  
g.fillArc( 300, 130, 150, 150, 135, 90 );
```

# Methods of the **Graphics** class in Java

University of Mount Union

CSC 120

Day 4