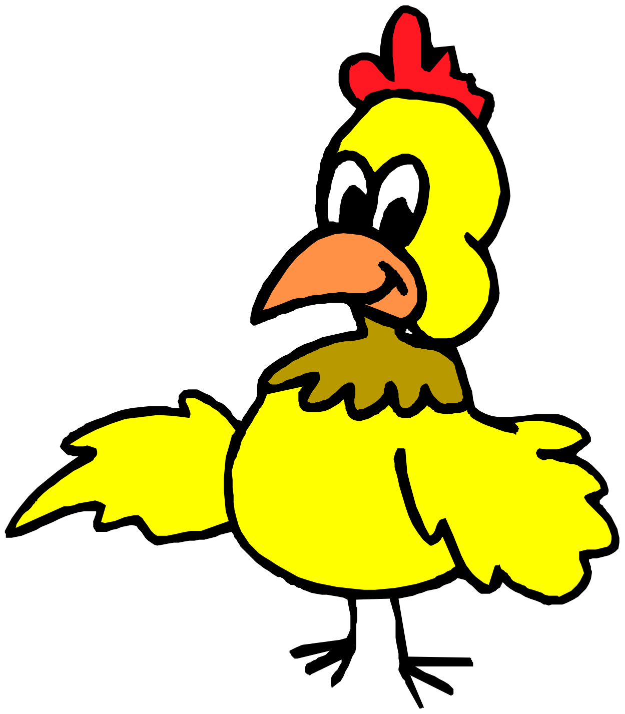
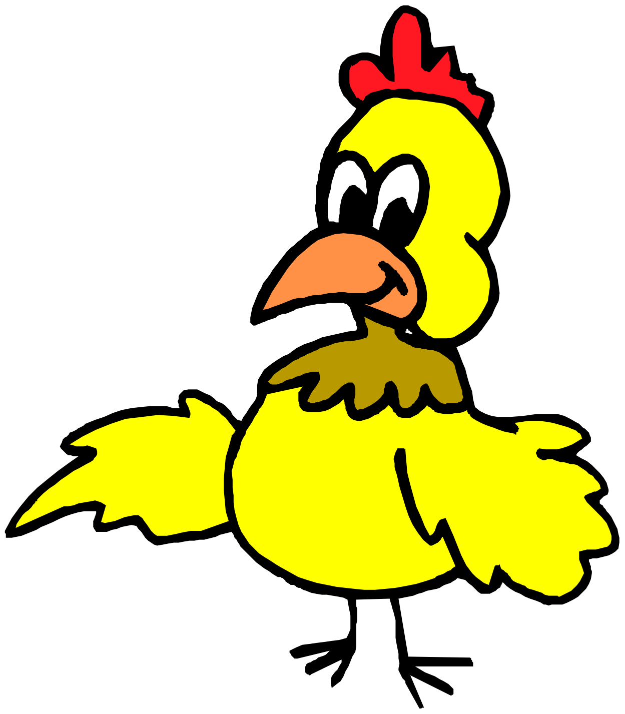
Methods Parameters Graphics

Objects



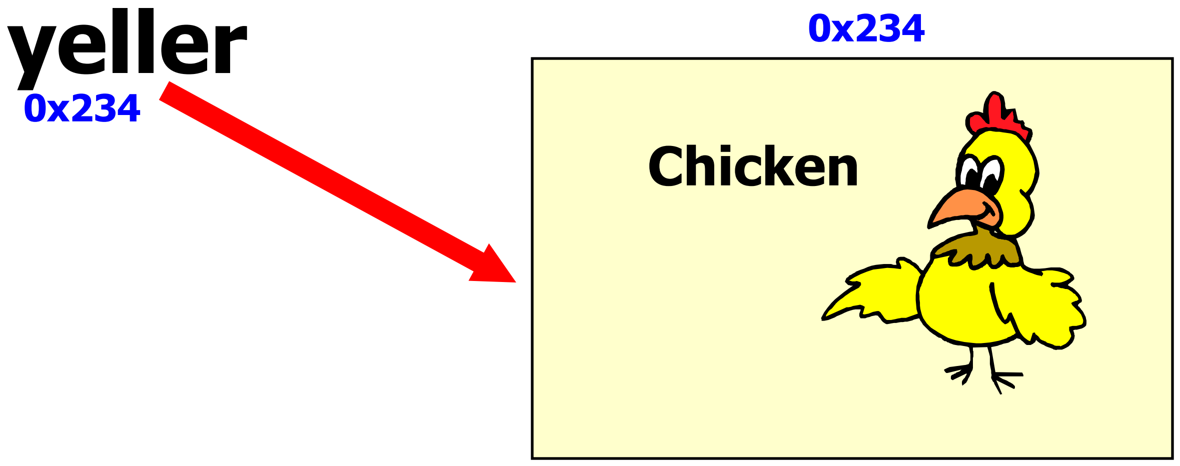
Object Instantiation

```
Chicken yeller = new Chicken();
```



Object Instantiation

```
Chicken yeller = new Chicken();
```



yeller is a reference variable that refers to a Chicken object.

Methods

What is a method?

A method is a storage location for related program statements. When called, a method usually performs a specific task.

```
System.out.println( )
```

What methods have we used?

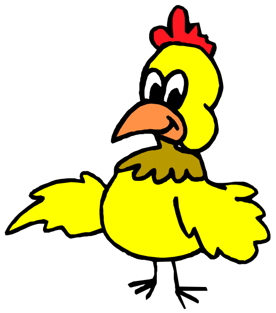
dude.goHome()

keyboard.nextInt( )

System.out.println( )

methods

```
public void speak()
{
  out.println("cluck-cluck");
}
```



OUTPUT

```
cluck-cluck
```

methods

access | return type | name | params

code

```
public            void       speak(    )
{
  System.out.println("cluck-cluck");
}
```

What does public mean?

All members with public access can be accessed or modified inside and outside of the class where they are defined.

chicken

```
public class Chicken
{
  public void speak()
  {
    out.println("cluck-cluck");
  }

  public static void main(String[] args)
  {
    Chicken red = new Chicken();
    red.speak();
    red.speak();
    red.speak();
  }
}
```

OUTPUT

```
cluck-cluck
cluck-cluck
cluck-cluck
```

Open chicken.java

```
//methods example 1

import static java.lang.System.*;

public class Chicken
{
  public void speak()
  {
    out.println("cluck-cluck");
  }

  public static void main(String[] args)
  {
    Chicken red = new Chicken();
    red.speak();
    red.speak();
    red.speak();
  }
}
```

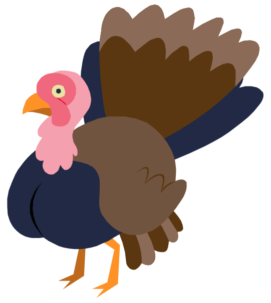
turkey

```
public class Turkey
{
   public void speak()
   {
      out.println("gobble-gobble");
   }

   public void sayName()
   {
      out.println("big bird");
   }
}
```

OUTPUT

```
gobble-gobble
big bird
gobble-gobble
big bird
gobble-gobble
```

```
//code in the main of another class
Turkey bird = new Turkey();
bird.speak();
bird.sayName();
bird.speak();
bird.sayName();
bird.speak();
```



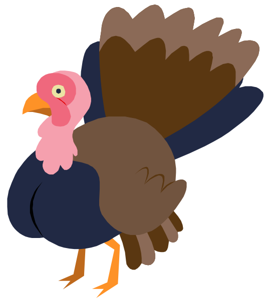
turkey

```
public class Turkey
{
   public void speak()
   {
      out.println("gobble-gobble");
   }

   public void sayName()
   {
      out.println("big bird");
      speak();
   }
}
```

```
//code in the main of another class
Turkey bird = new Turkey();
bird.speak();
bird.sayName();
bird.speak();
bird.sayName();
bird.speak();
```

OUTPUT

```
gobble-gobble
big bird
gobble-gobble
gobble-gobble
big bird
gobble-gobble
gobble-gobble
```



Open
turkey.java
turkeyrunner.java

```
//methods example 2 and 3

import static java.lang.System.*;

public class Turkey
{
  public void speak()
  {
    out.println("gobble-gobble");
  }

  public void sayName()
  {
    out.println("big bird");

    //what does the following line do??
    //speak();
  }
}
```

```
//methods example 2 and 3

import static java.lang.System.*;

public class TurkeyRunner
{
  public static void main(String[] args)
  {
    Turkey bird = new Turkey();
    bird.speak();
    bird.sayName();
    bird.speak();
    bird.sayName();
    bird.speak();
  }
}
```

Start work on the labs

Constructors and Graphics methods

Constructors

Constructors always have the same name as the class.

```
GraphOne test = new GraphOne();

Monster rob = new Monster();
```

Constructors

reference variable

```
Scanner keyboard =
                    new Scanner(System.in);
```

object instantiation / constructor call

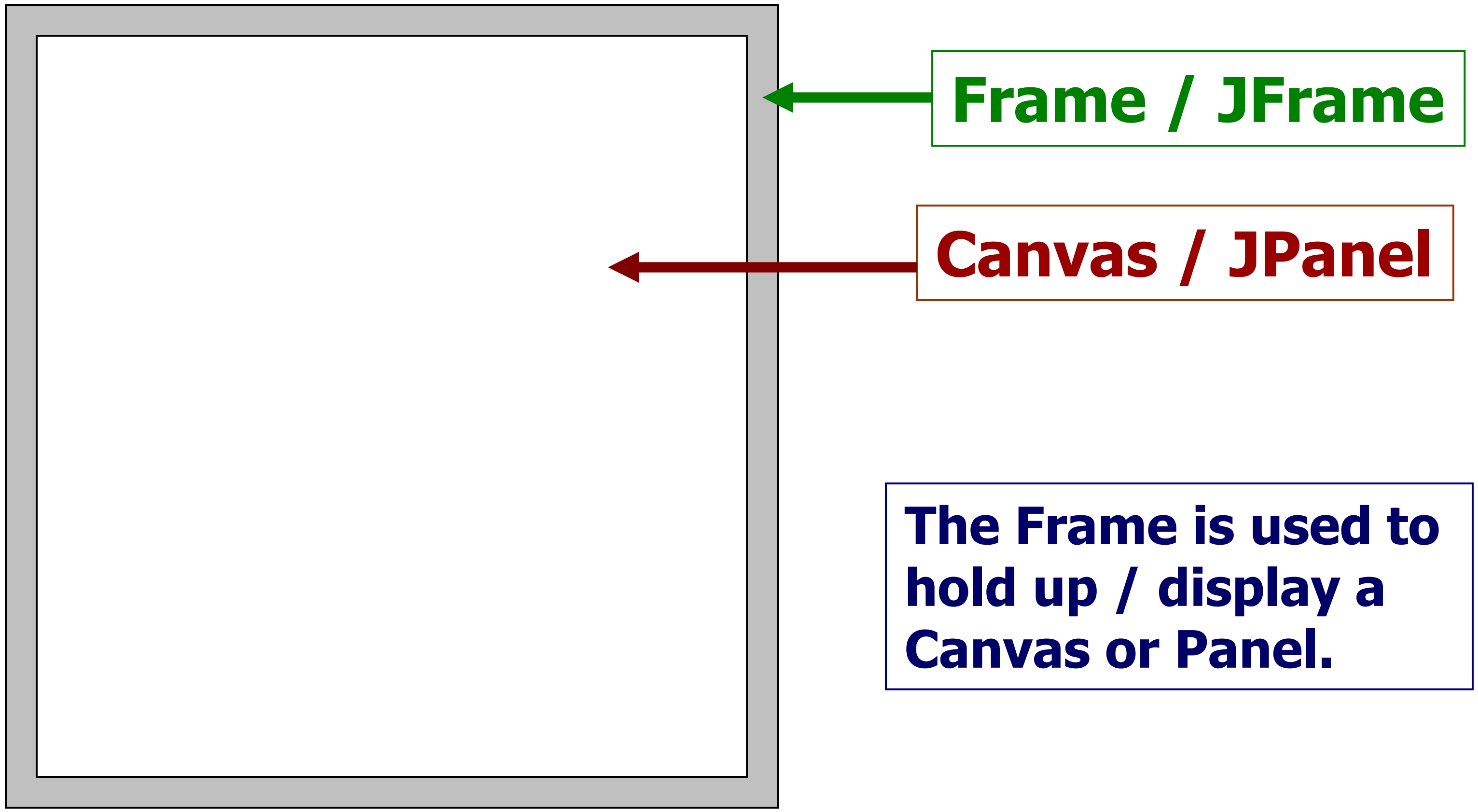
Constructors

```
public class GraphicsRunner extends JFrame
{
  private static final int WIDTH = 640;
  private static final int HEIGHT = 480;

  public GraphicsRunner()
  {
    setSize(WIDTH,HEIGHT);
    getContentPane().add(  new Circles()  );
    setVisible(true);
  }

  public static void main( String args[] )
  {
    GraphicsRunner run = new GraphicsRunner();
  }
}
```

the constructor

constructor call

Frame



The Frame is used to hold up / display a Canvas or Panel.

paint()

```
public class Circles extends Canvas
{

   //constructors

   public void paint( Graphics window )
   {
      window.setColor(Color.BLACK);
      window.drawString("Circles", 50, 50);

      window.setColor(Color.BLUE);
      window.drawOval(500,300,40,40);
   }

   //other methods

}
```

paint

paint() is called automatically when you instantiate the class containing the paint method.

When an event is triggered that requires a redraw, paint is called again.

To call paint() without a Graphics parameter, you can use the repaint() method.

Open

graphicsrunner.java

circles.java

```
//graphics example for circles/ovals

import java.awt.Graphics;
import java.awt.Color;
import java.awt.Canvas;

public class Circles extends Canvas
{
public Circles()
{
    setBackground(Color.WHITE);
}

    public void paint( Graphics window )
                {
                window.setColor(Color.BLACK); window.drawString("Circles - Ovals", 50, 50);

                                window.setColor(Color.BLUE);

                                //drawOval(int x1, int y1, int width, int height)
                                window.drawOval(500,300,40,40);

                                window.setColor(Color.GREEN);
                                window.drawOval(400,100,100,50);

                                window.setColor(Color.YELLOW);
                                window.fillOval(250,250,90,90);

                                window.setColor(Color.RED);
                                window.fillOval(50,150,50,50);

                                window.setColor(Color.BLUE);
                                window.fillOval(150,350,120,80);                    }    }
```

```
//graphics frame to run graphics examples

import javax.swing.JFrame;

public class GraphicsRunner extends JFrame
{
        private static final int WIDTH = 800;
        private static final int HEIGHT = 600;


        public static void main( String args[] )
        {
                GraphicsRunner run = new GraphicsRunner();
        }
```

```
public GraphicsRunner()
	{
		super("Graphics Runner");

		setSize(WIDTH,HEIGHT);

		//getContentPane().add(new Circles());

		//getContentPane().add(new Rectangles());

		//getContentPane().add(new Lines());

		//getContentPane().add(new Polygons());

		//getContentPane().add(new Arcs());

		//getContentPane().add(new Colors());

		//getContentPane().add(new Fonts());

		//getContentPane().add(new ImageOne());

		//getContentPane().add(new DoubleBuffer());

		//getContentPane().add(new Animation());

		getContentPane().add(new Sounds());

		setVisible(true);

		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
```

Parameters and Graphics methods

Graphics

frequently used methods

Name	Use
setColor(x)	sets the current drawing color to x
drawString(s,x,y)	draws String s at spot x,y
drawOval(x,y,w,h)	draws an unfilled oval at spot x,y that is w wide and h tall
fillOval(x,y,w,h)	draws a filled oval at spot x,y that is w wide and h tall

```
import java.awt.Graphics;
import java.awt.Color;
import javax.swing.JFrame;
```

passing parameters

A parameter/argument is a channel used to pass information to a method. setColor() is a method of the Graphics class the receives a Color.

```
void setColor(Color theColor)
```

```
window.setColor( Color.RED );
```

method call with parameter

passing parameters

```
void fillRect (int x, int y, int width, int height)
```

```
window.fillRect( 10, 50, 30, 70 );
```

method call with parameters

passing parameters

```
void fillRect(int x, int y, int width, int height)
```

$$\begin{tikzcd}[column sep=small]
\text{int x} & \text{int y} & \text{int width} & \text{int height} \\
10 \arrow[u] & 50 \arrow[u] & 30 \arrow[u] & 70 \arrow[u]
\end{tikzcd}$$

```
window.fillRect( 10, 50, 30, 70 );
```

The call to fillRect would draw a rectangle at position 10,50 with a width of 30 and a height of 70.

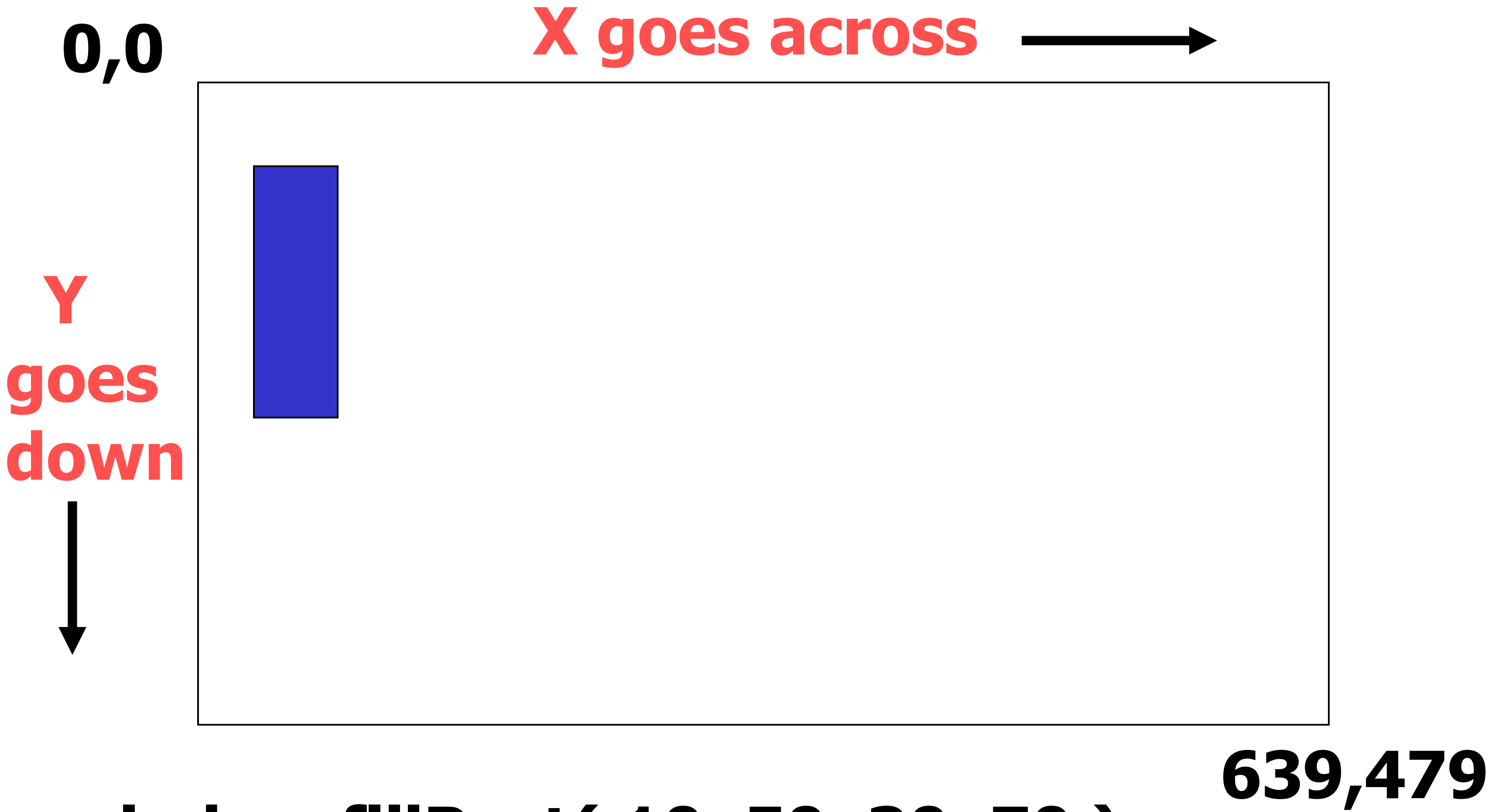
Graphics

frequently used methods

Name	Use
drawLine(a,b,c,d)	draws a line starting at point a,b and going to point c,d
drawRect(x,y,w,h)	draws an unfilled rectangle at spot x,y that is w wide and h tall
fillRect(x,y,w,h)	draws a filled rectangle at spot x,y that is w wide and h tall

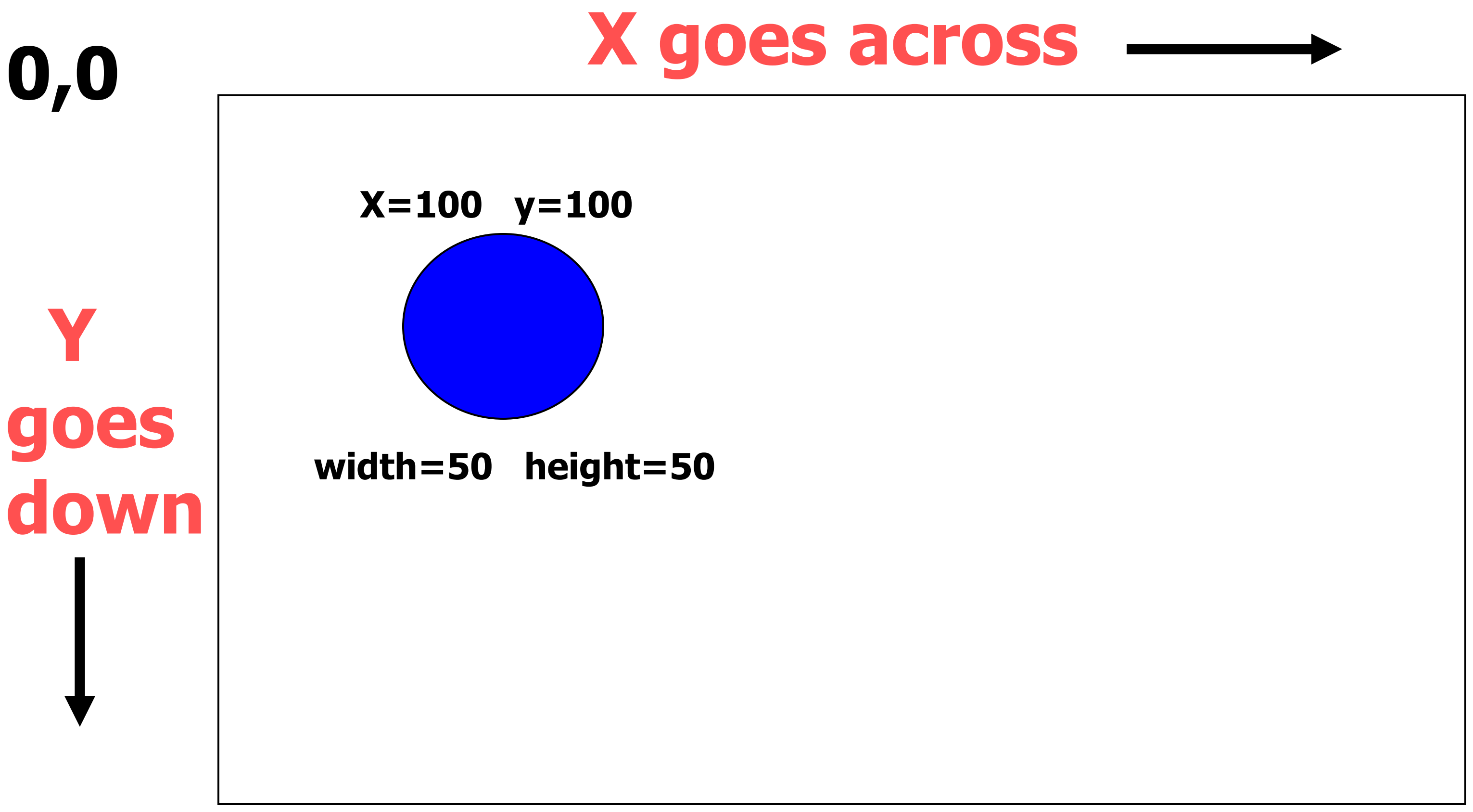
```
import java.awt.Graphics;
import java.awt.Color;
import javax.swing.JFrame;
```

The Graphics Screen



```
window.fillRect( 10, 50, 30, 70 );
```

The Graphics Screen



window.fillOval(100, 100, 50, 50);

Rectangles

```
public void paint( Graphics window )
{
   window.setColor(Color.BLUE);
   window.fillRect(150, 300, 100, 20);
   window.setColor(Color.GRAY);
   window.drawRect(200,80,50,50);
}
```

Open rectangles.java

```
//graphics example for rectangles

import java.awt.Graphics;
import java.awt.Color;
import java.awt.Canvas;

public class Rectangles extends Canvas
{
		public Rectangles()
		{
				setBackground(Color.WHITE);
		}

		public void paint( Graphics window )
		{
				window.setColor(Color.BLACK);
				window.drawString("Squares - Rectangles", 25, 50);

				window.setColor(Color.BLUE);
				//fillRect(int x1, int y1, int width, int height)
				window.fillRect(150, 300, 100, 20 );

				window.setColor(Color.GRAY);
				window.drawRect(200,80,50,50);

				window.setColor(Color.RED);
				window.fillRect(320,370,40,40);

				window.setColor(Color.BLUE);
				window.drawRect(100,180,50,50);

				window.setColor(Color.ORANGE);
				window.fillRect(520,250,90,20);
		}
}
```

Open

lines.java

```
//graphics example for lines

import java.awt.Graphics;
import java.awt.Color;
import java.awt.Canvas;

public class Lines extends Canvas
{
		public Lines()
		{
			setBackground(Color.WHITE);
		}

		public void paint( Graphics window )
		{
			window.setColor(Color.BLACK);
			window.drawString("Points - Lines", 25, 50);

			window.setColor(Color.YELLOW);
			//drawLine(int x1, int y1, int x2, int y2)
			window.drawLine(300,300,400,400);

			window.setColor(Color.RED);
			window.drawLine(50,100,50,300);

			window.setColor(Color.BLUE);
			window.drawLine(100,100,100,400);

			window.setColor(Color.ORANGE);
			window.drawLine(400,200,400,201);

			window.setColor(Color.GREEN);
			window.drawLine(50,400,500,400);
		}
}
```

Graphics

frequently used methods

Name	Use
drawArc(x,y,w,h,startAngle,arcAngle)	draws an arc at spot x,y that is w wide and h tall
fillArc(x,y,w,h,startAngle,arcAngle)	draws a filled arc at spot x,y that is w wide and h tall
startAngle specifies the start of the arc arcAngle specifies the length of the arc	

```
import java.awt.Graphics;
import java.awt.Color;
import javax.swing.JFrame;
```

Open arcs.java

```
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Canvas;

public class Arcs extends Canvas
{
		public Arcs()
		{
				setBackground(Color.WHITE);
		}

		public void paint( Graphics window )
		{
				window.setColor(Color.BLACK);
				window.drawString("Arcs ", 50, 50);

				window.setColor(Color.BLUE);

				//drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
				window.drawArc(500,300,40,40,90,90);

				window.setColor(Color.GREEN);
				window.drawArc(100,100,50,50,0,-180);

				window.setColor(Color.RED);
				window.drawArc(250,100,50,50,0,270);

				window.setColor(Color.ORANGE);
				window.drawArc(50,200,50,50,180,-180);
		}
}
```

Open fonts.java

```
//graphics example for changing fonts

import java.awt.Graphics;
import java.awt.Color;
import java.awt.Font;
import java.awt.Canvas;

public class Fonts extends Canvas
{
		public Fonts()
		{
				setBackground(Color.WHITE);
		}

		public void paint( Graphics window )
		{
				window.setColor(Color.BLACK);
				window.drawString("Fonts", 50, 50);

				window.setColor(Color.BLUE);
				window.setFont(new Font("TAHOMA",Font.BOLD,12));
				window.drawString("Here is the new Tahoma Font!", 100, 100 );

				window.setColor(Color.GREEN);
				window.setFont(new Font("ARIAL",Font.BOLD,24));
				window.drawString("Here is the new Arial Font!", 200, 200 );
		}
}
```

Open

colors.java

```
//graphics example for colors

import java.awt.Graphics;
import java.awt.Color;
import java.awt.Canvas;

public class Colors extends Canvas
{
		public Colors()
		{
				setBackground(Color.WHITE);
		}

		public void paint( Graphics window )
		{
				window.setColor(Color.BLACK);
				window.drawString("Colors ", 50, 50);

				//Color( int red, int green, int blue )
				Color newColor = new Color(40,60,80);
				window.setColor(newColor);
				window.drawArc(100,100,50,50,0,-180);


				//the simple approach
				int red = (int)(Math.random()*256);
				int green = (int)(Math.random()*256);
				int blue = (int)(Math.random()*256);
				newColor = new Color(red, green, blue);
				window.setColor(newColor);
				window.fillRect(250,300,50,50);
//the not so simple approach
                  newColor = new
Color(((int)(Math.random()*256)),((int)(Math.random()*256)),((int)
(Math.random()*256)));
                  window.setColor(newColor);
                  window.fillOval(150,200,50,50);

                  newColor = new
Color(((int)(Math.random()*256)),((int)(Math.random()*256)),((int)
(Math.random()*256)));
                  window.setColor(newColor);
                  window.fillOval(550,100,10,50);

                  red = (int)(Math.random()*256);
                  green = (int)(Math.random()*256);
                  blue = (int)(Math.random()*256);
                  newColor = new Color(red, green, blue);
                  window.setColor(newColor);
                  window.fillRect(450,200,50,50);
         }
}
```

Continue work on the labs

Vocabulary Words

- method
- instantiation
- reference
- method signature
- access
- return type
- parameters
- public
- constructor
- JFrame
- JPanel
- Canvas
- paint()
- repaint()