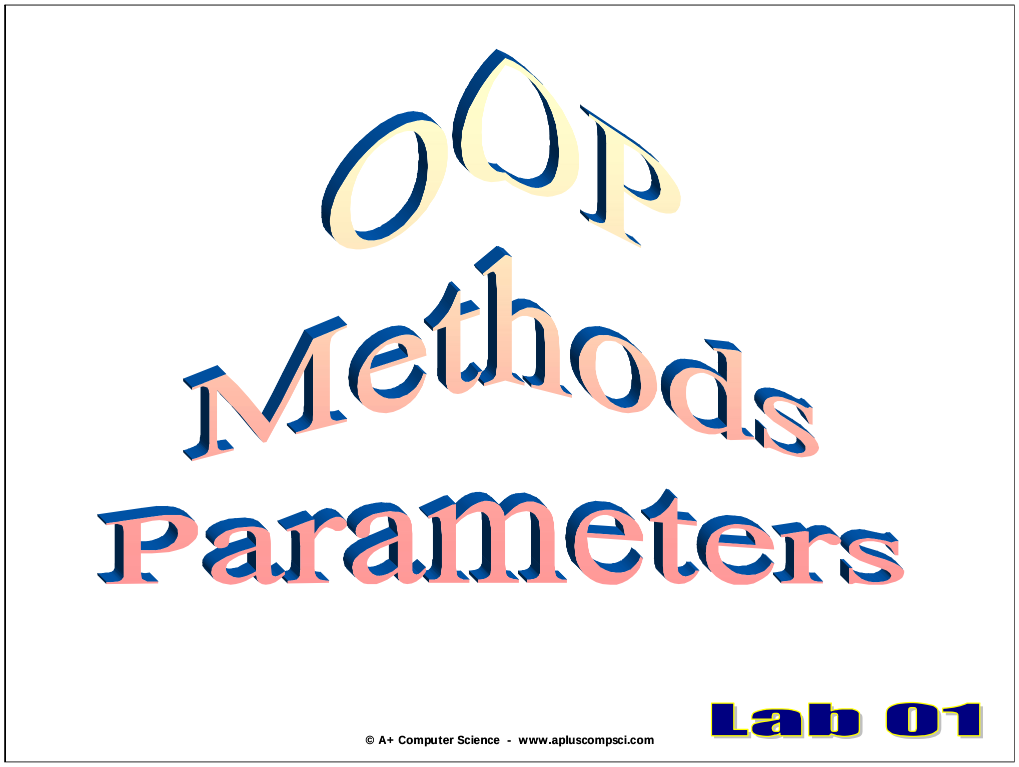
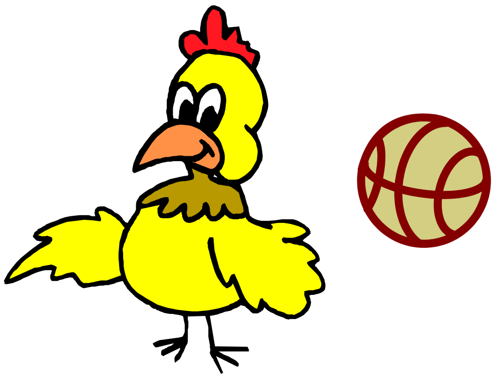
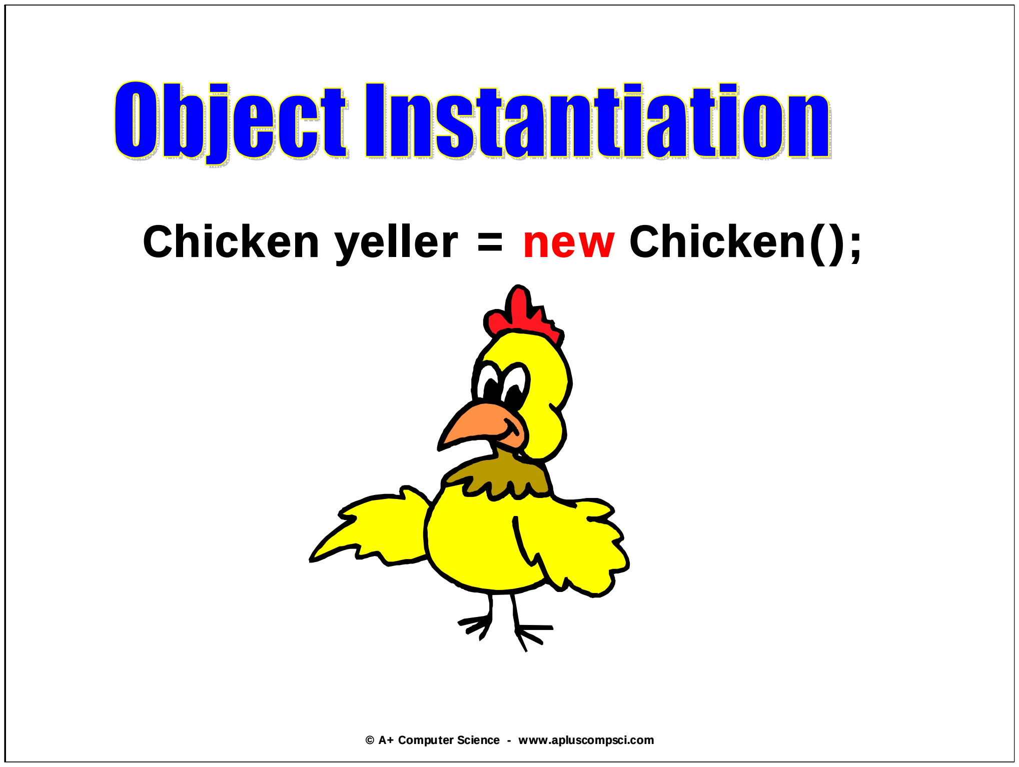


OOP Methods Parameters

Lab 01

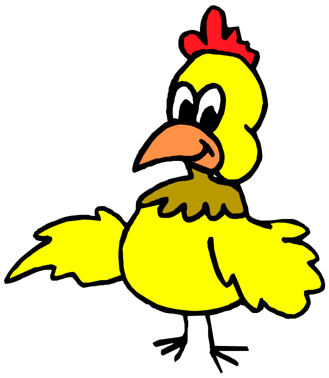
Objects





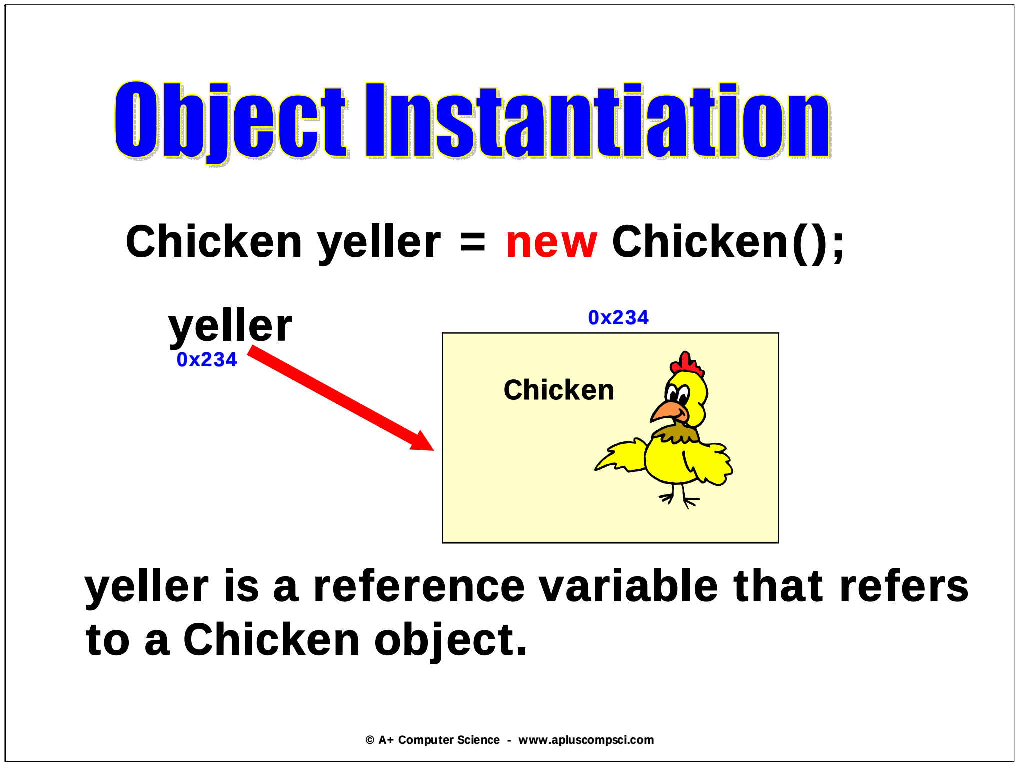
Object Instantiation

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



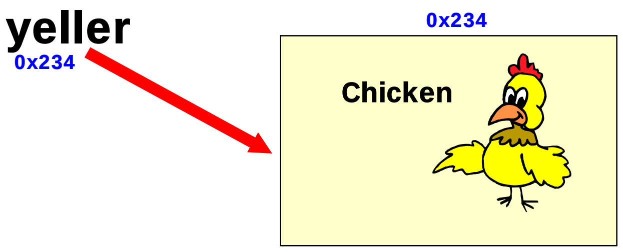
`yeller` is a Chicken reference.
`new Chicken()` creates a new Chicken Object out in memory.

`yeller` stores the location of that new Chicken Object.



Object Instantiation

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

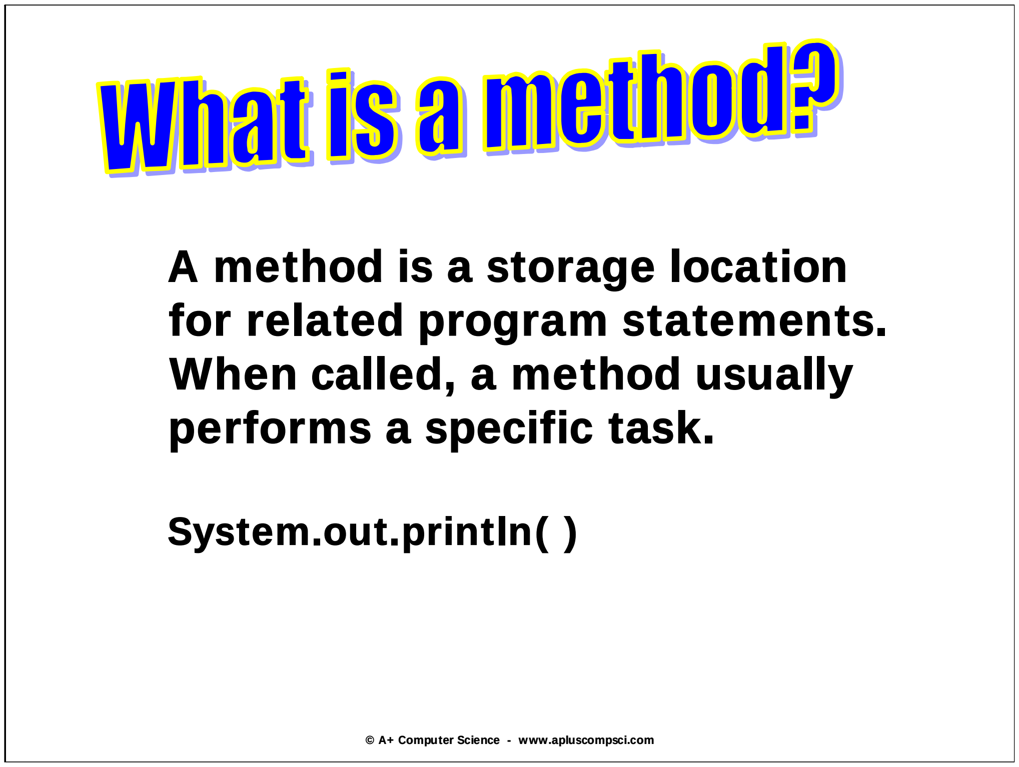


yeller is a reference variable that refers to a Chicken object.

`yeller` is a Chicken reference.
`new Chicken()` creates a new Chicken Object out in memory.

`yeller` stores the location of that new Chicken Object.

Methods



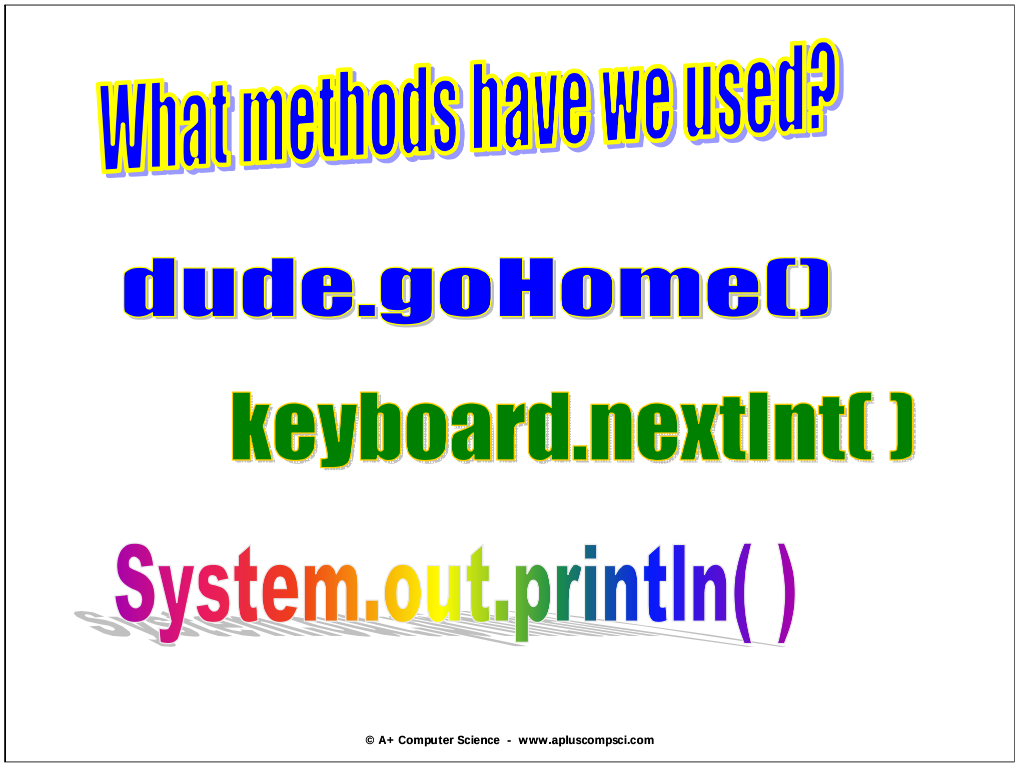
What is a method?

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

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

© A+ Computer Science - www.apluscompsci.com

Methods store commands / program statements. When called, the code inside the method is activated.

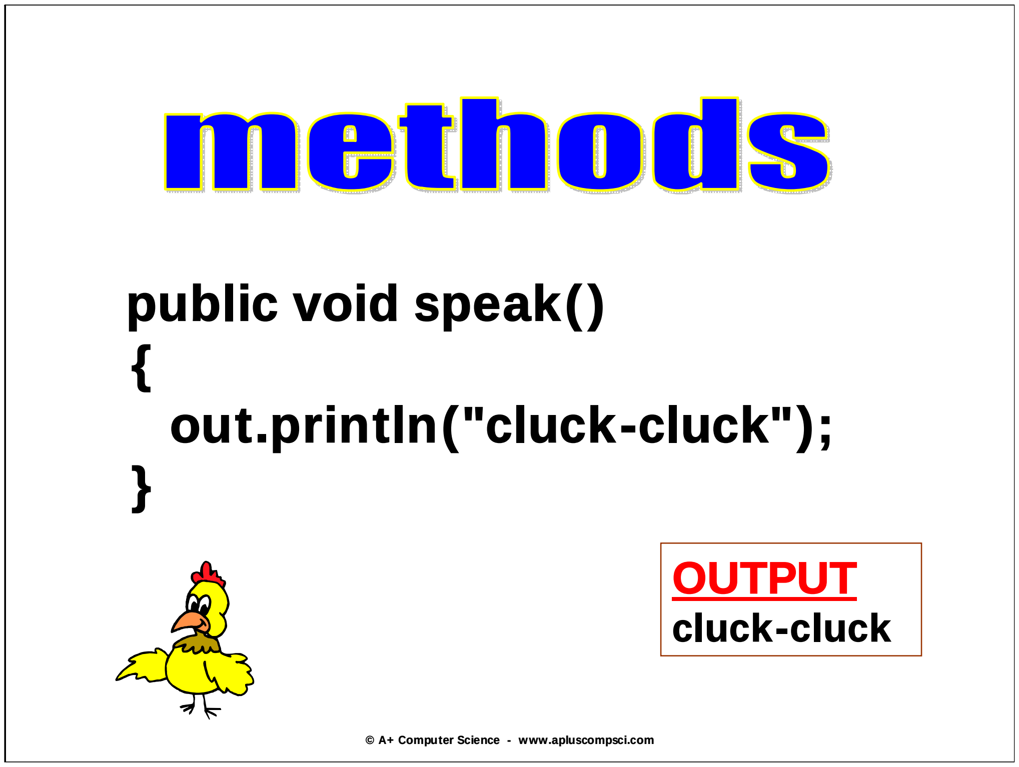


What methods have we used?

dude.goHome()

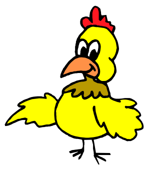
keyboard.nextInt( )

System.out.println( )



methods

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

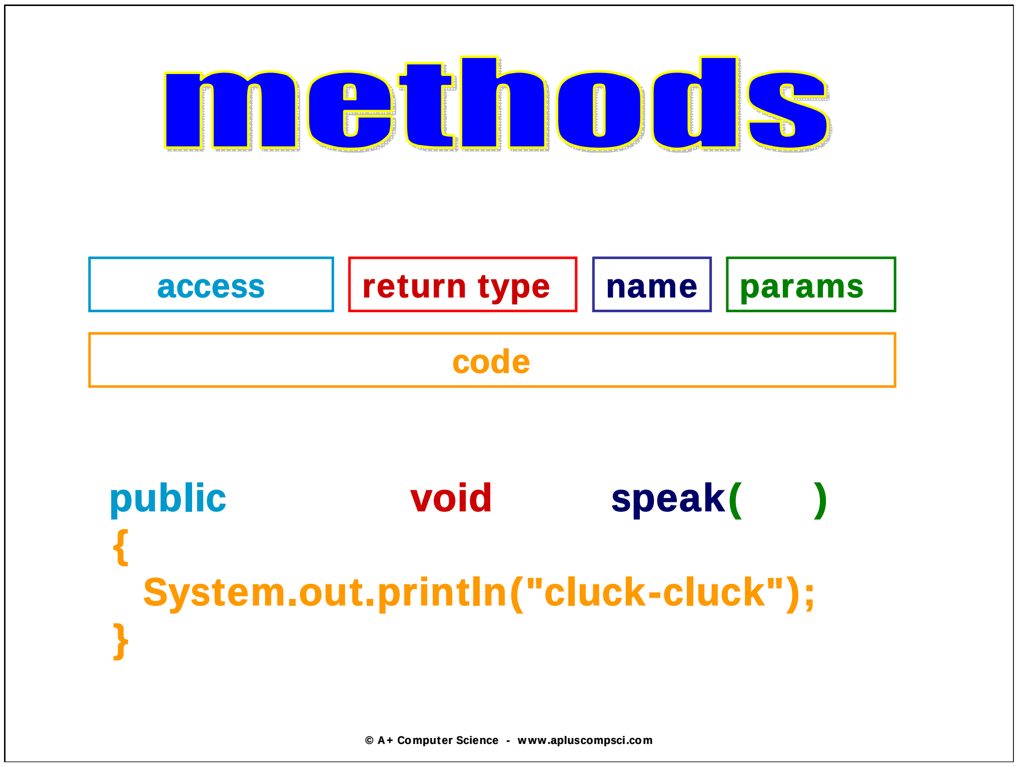


OUTPUT

```
cluck-cluck
```

The speak method shown above contains a single `println` command.

The speak method would print out cluck-cluck on the console window.



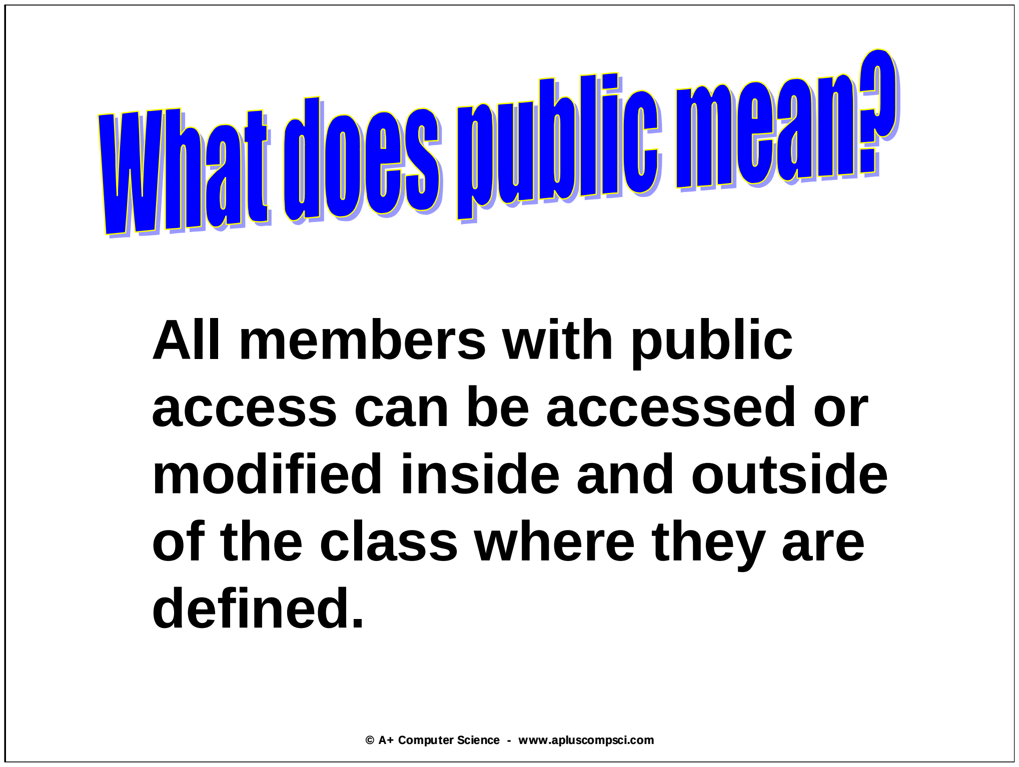
methods

access | return type | name | params

code

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

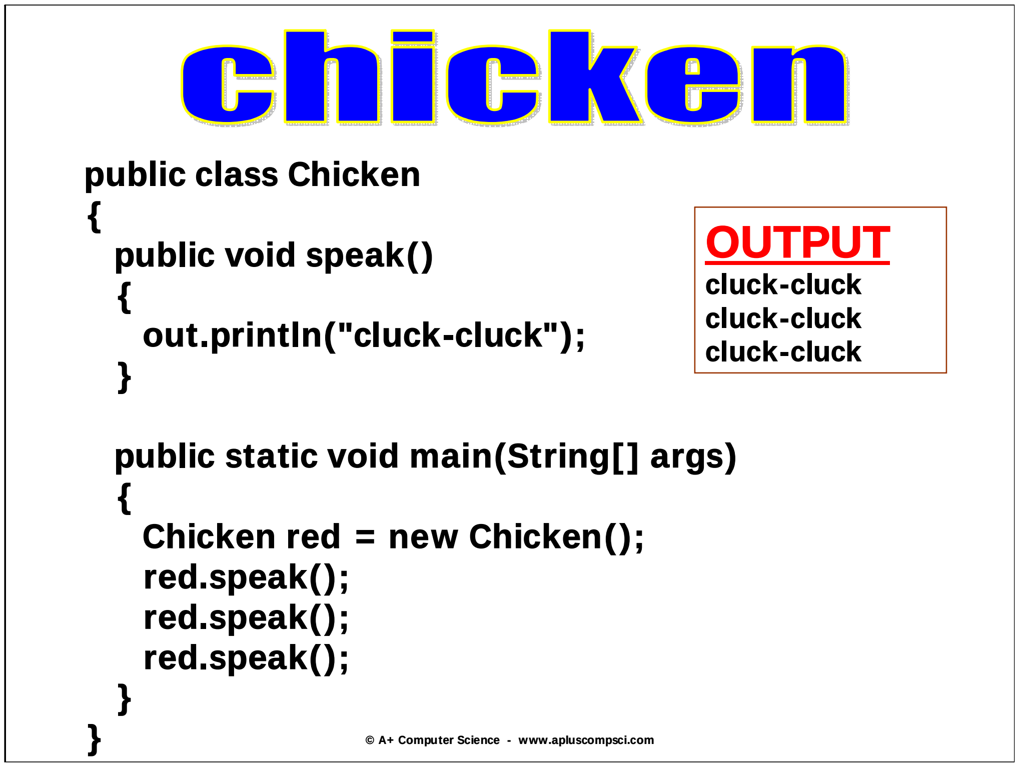
A method has a signature. The signature provides information about the method. The name is most used and recognizable part of the signature. The method shown above is named print. The return type states what the method will return. Method print has a return type of void which means the method does not return a value. The access of method print is public. This states that the method print can be called from any location.



What does public mean?

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

Public access simply means the member can be used anywhere inside or outside of the class.



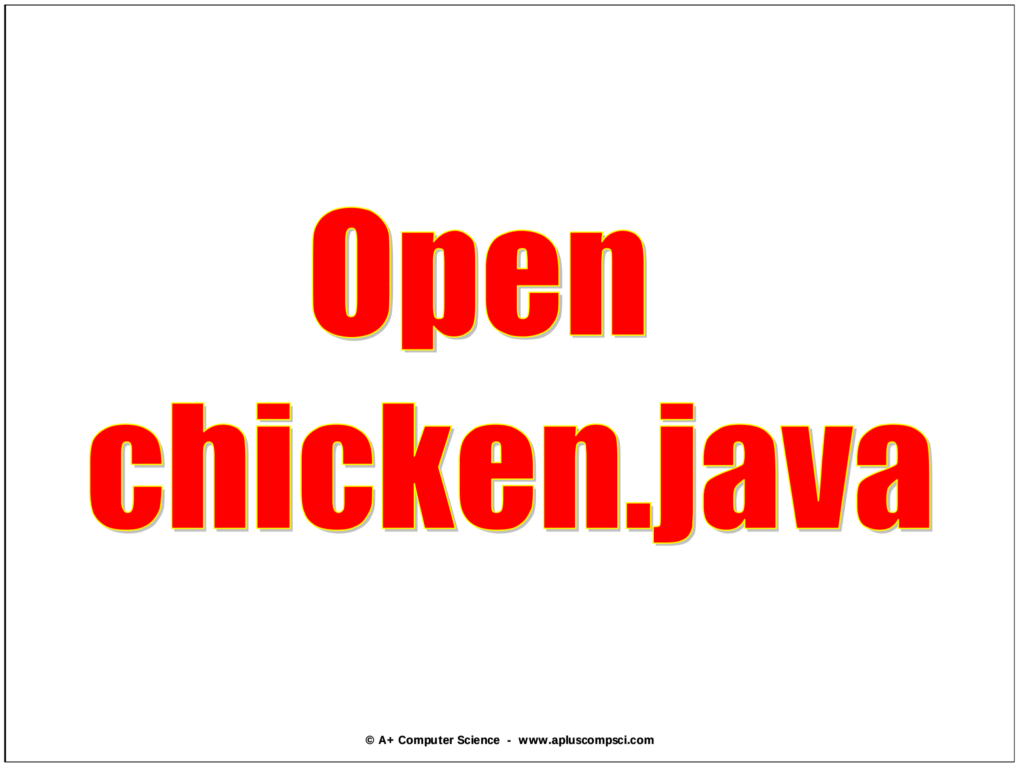
chicken

```
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();
  }
}
```

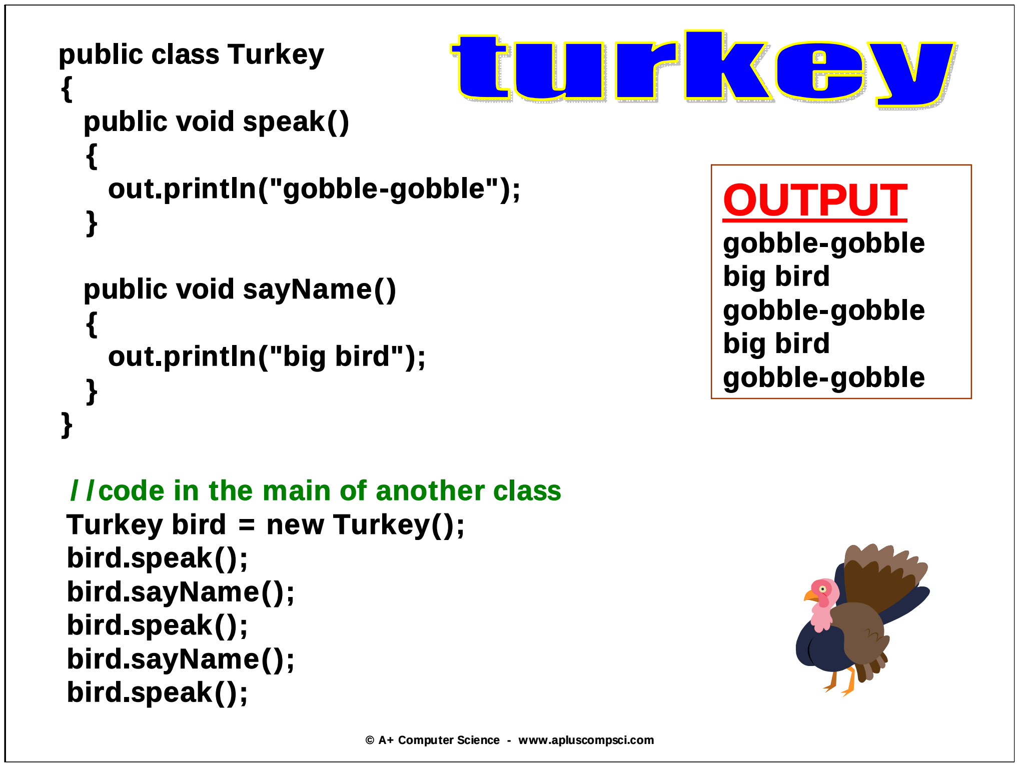
OUTPUT
```
cluck-cluck
cluck-cluck
cluck-cluck
```

In the `Chicken` example, method `speak()` prints out cluck-cluck each time it is called. Method `speak()` is called three times; thus, it prints out `cluck-cluck` three times.

OUTPUT
```
cluck-cluck
cluck-cluck
cluck-cluck
```



Open chicken.java

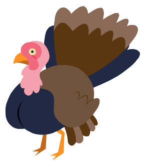


turkey

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

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

//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
big bird
gobble-gobble
```



In the `Turkey` example, `speak` is called which prints out gobble-gobble. sayName is called which prints out big bird.

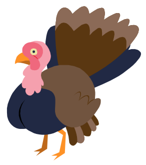
Then, speak is called again to print out gobble-gobble followed by a call to sayName to print big bird again. Last, speak is called to print out gobble-gobble.

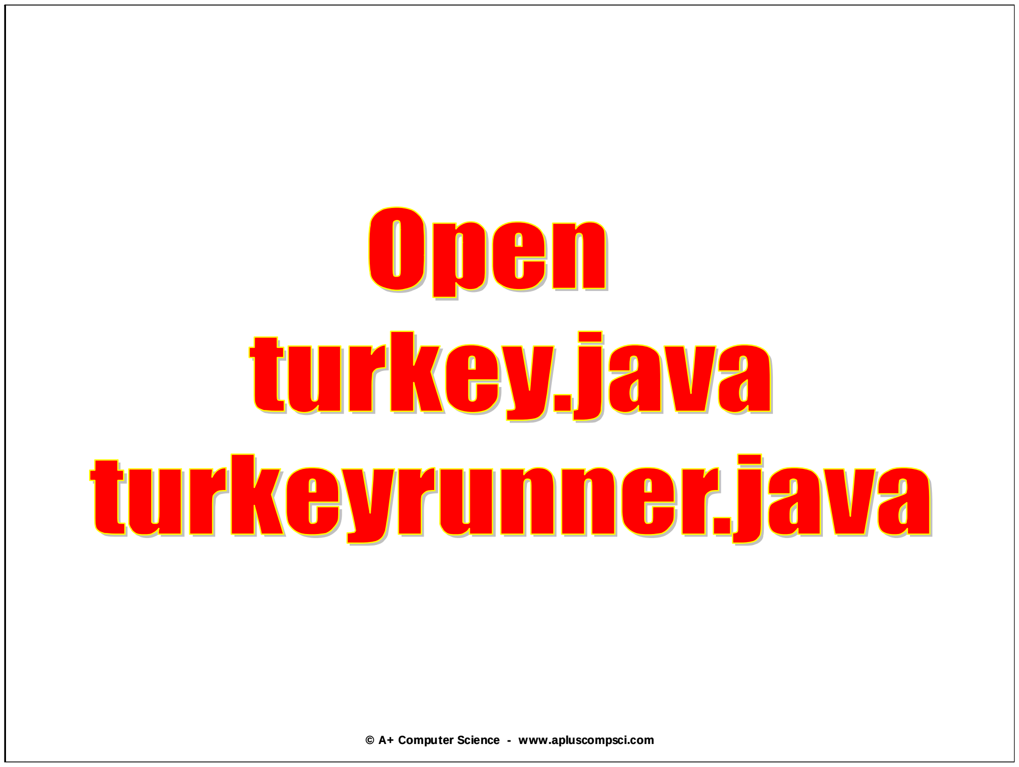
turkey

```
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

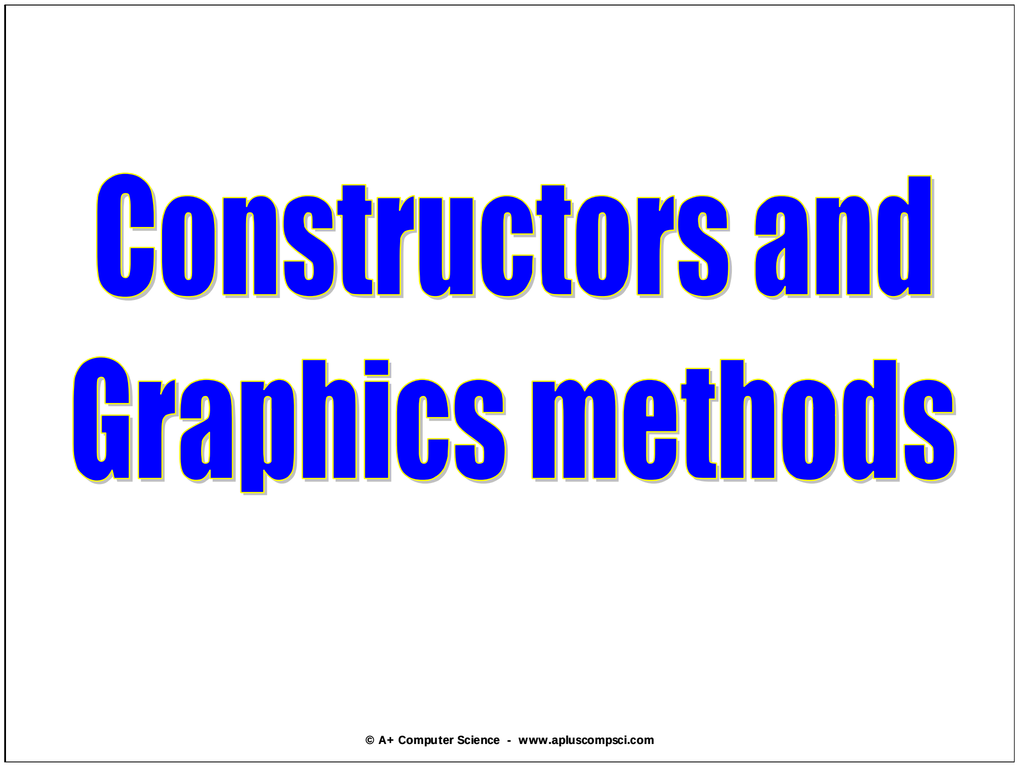
```
gobble-gobble
big bird
gobble-gobble
gobble-gobble
big bird
gobble-gobble
gobble-gobble
```



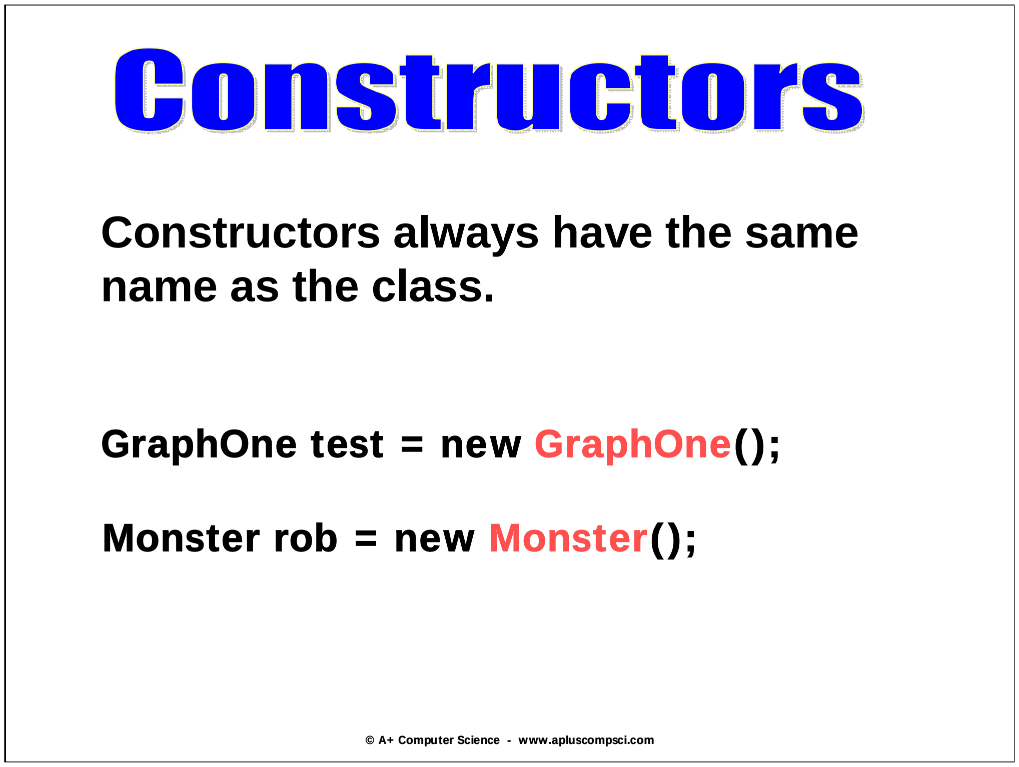


Open turkey.java turkeyrunner.java

Start work on Lab 01a



Constructors and Graphics methods

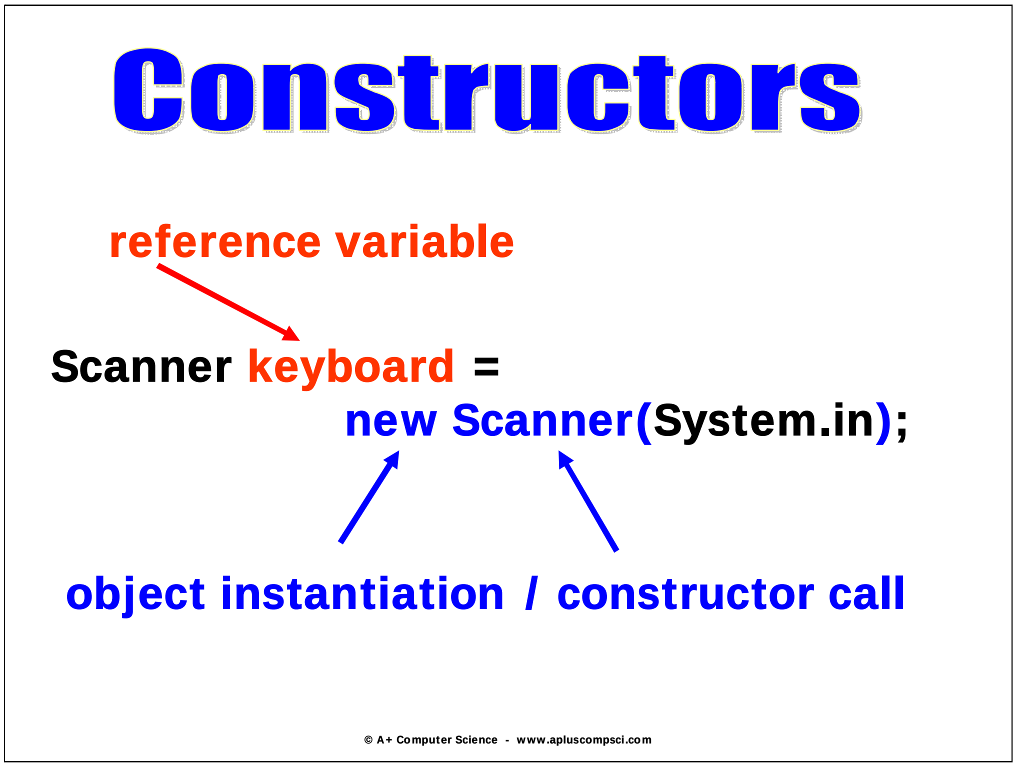


Constructors

Constructors always have the same name as the class.

```
GraphOne test = new GraphOne();

Monster rob = new Monster();
```

Constructors are used to initialize all of the data/properties inside the class. Constructors ensure that the Object is ready for use.



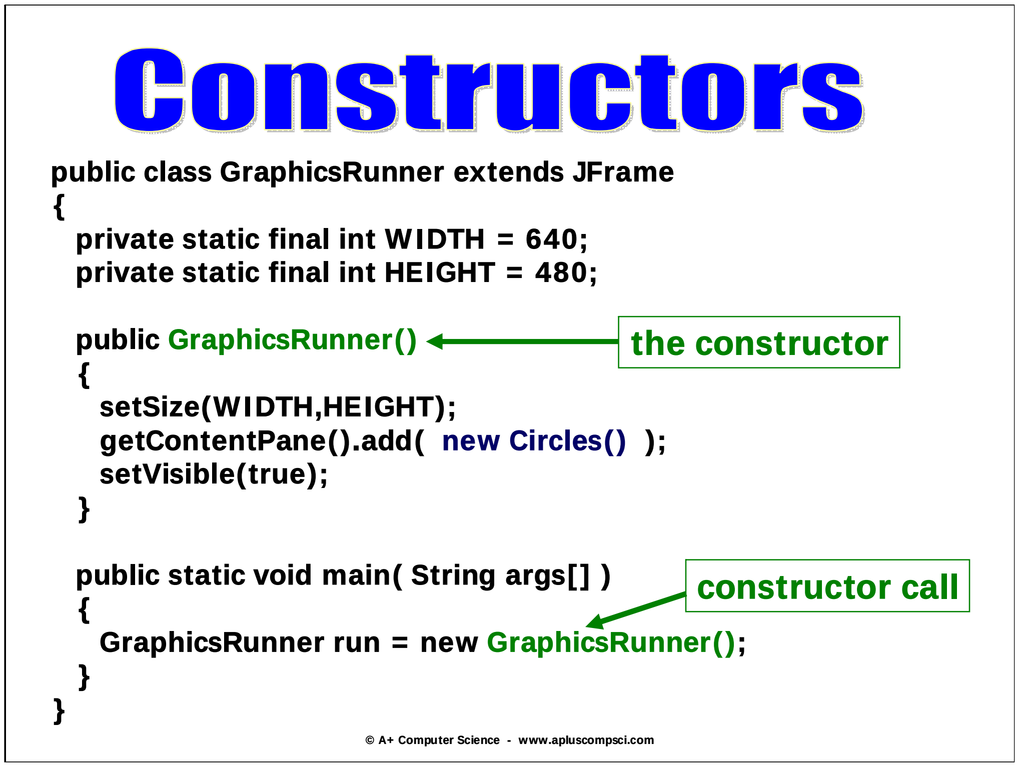
Constructors

reference variable
object instantiation / constructor call

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

`Scanner` is a class which must be instantiated before it can be used. In other words, you must make a new `Scanner` if you want to use `Scanner`. A reference must be used to store the location in memory of the `Scanner` object created.

`System.in` is the parameter passed to the `Scanner` constructor so that Java will know to connect the new `Scanner` to the keyboard. keyboard is a reference that will store the location of newly created `Scanner` object.



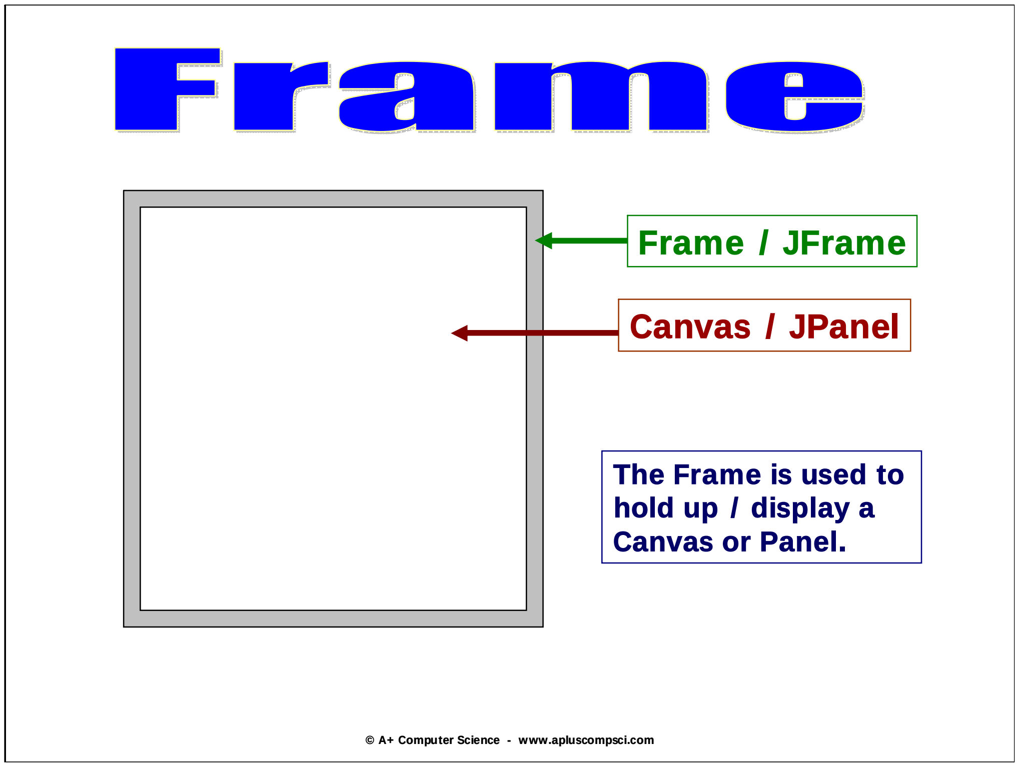
Constructors

```
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();
  }
}
```

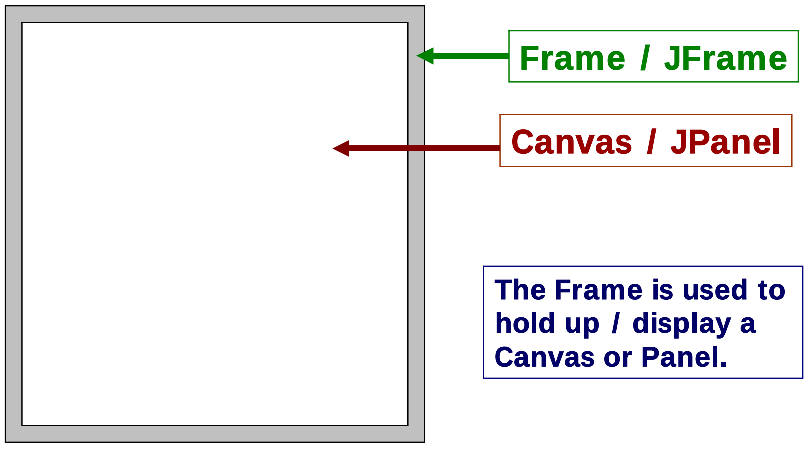
the constructor
constructor call

When a `GraphicsRunner` class is instantiated, the size of the JFrame is set and the visibility is also set. The `setSize()` method sets the width and height of the JFrame. The `setSize()` method tells the simply to either show the JFrame or hide the Frame.

The add() method adds a Component to the JFrame. A new `Circles()` Object is being instantiated and added to the JFrame.

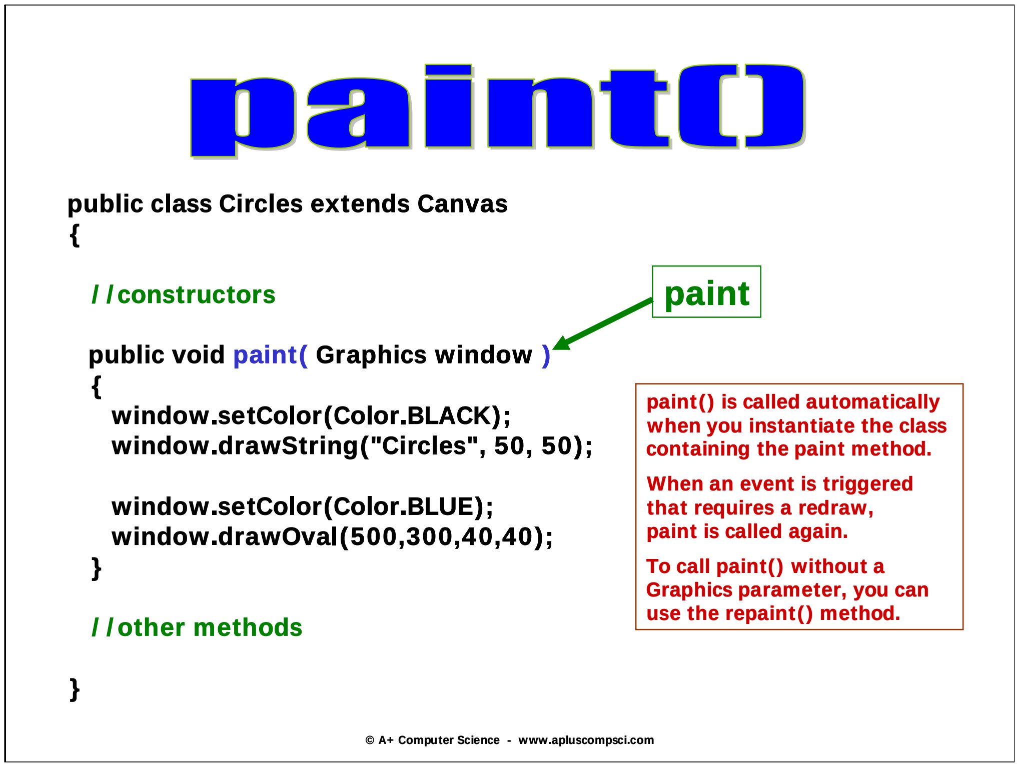


Frame



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

`Frame / JFrame` Objects are used to hold up / display `Canvas` and `JPanel` Objects. All drawing occurs on the `Canvas / JPanel`. The `JFrame` simply provides a place to show `Canvas / JPanel` after the drawing has occurred.



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() 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.

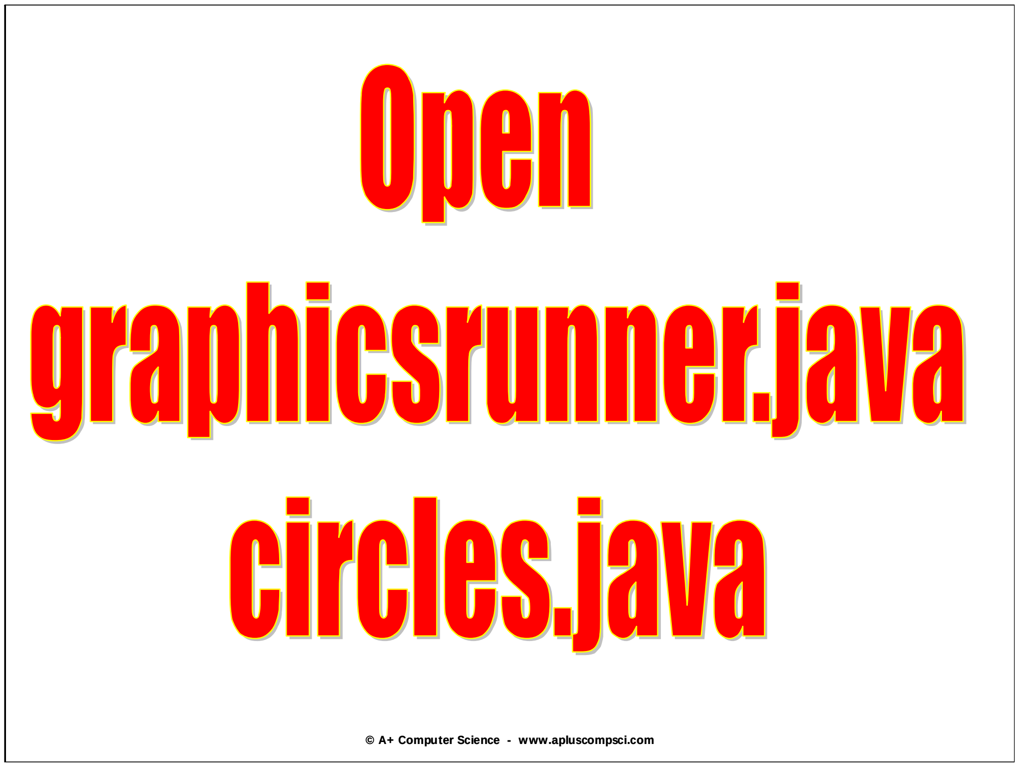
© A+ Computer Science - www.apluscompsci.com

`paint()` is the method typically used to draw Graphics on the window. There are other methods that could be used, but `paint()` is used most frequently.

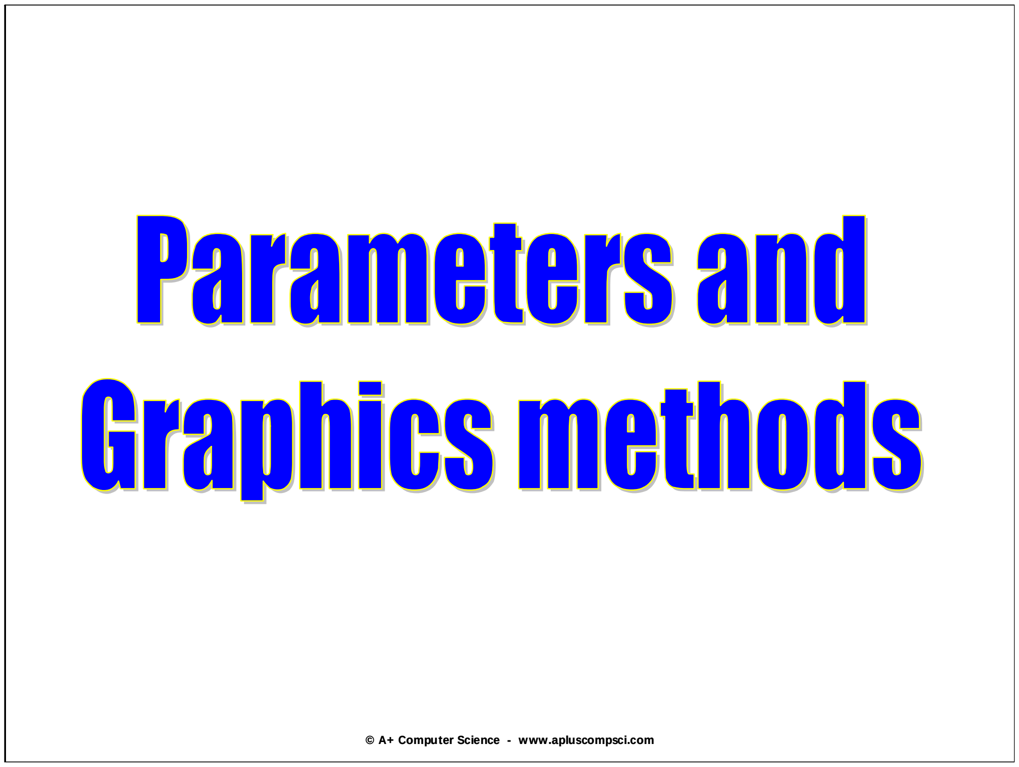
`paint()` is called when the window needs to be redrawn. If an event occurs that requires the window be updated, the system will call `paint()`.

`paint()` can be called without a Graphics parameter by simply using the `repaint()` method.

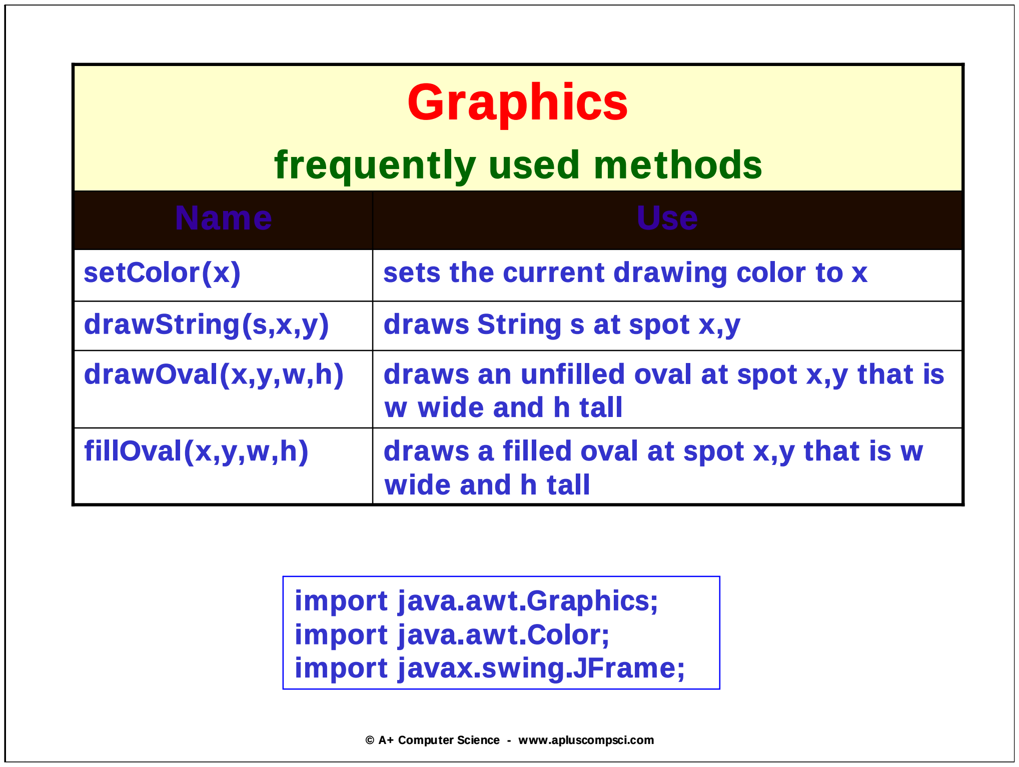
`paintComponent()` is another method used for drawing / redrawing the window.



Open graphicsrunner.java circles.java



Parameters and Graphics methods



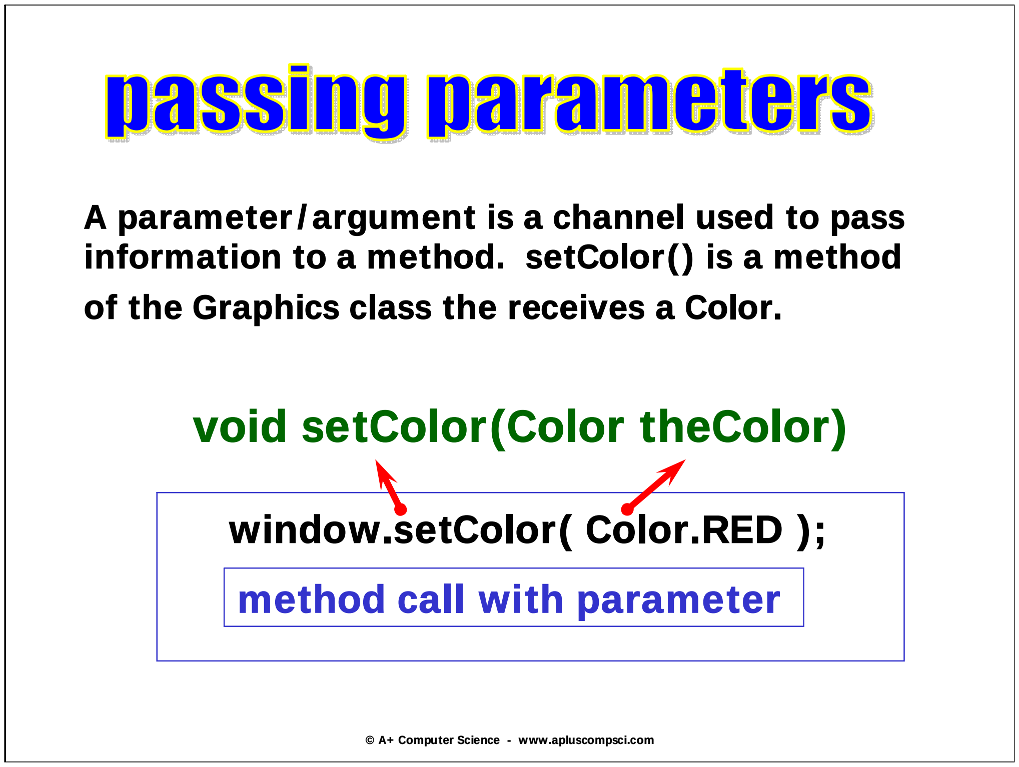
Graphics

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;
```

The Java Graphics class has many useful methods. The chart above lists the most common methods we will be using.



passing parameters

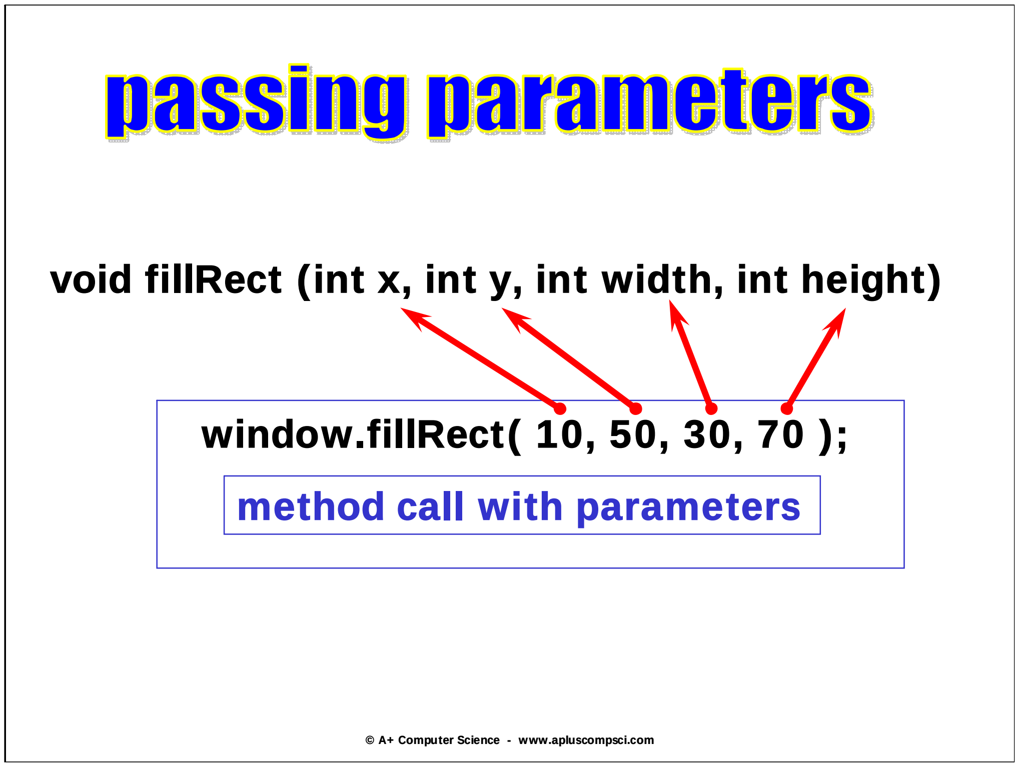
A parameter/argument is a channel used to pass information to a method. setColor() is a method of the Graphics class the receives a Color.

```
void setColor(Color theColor)

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

method call with parameter

© A+ Computer Science - www.apluscompsci.com

Most, if not all, of the `Graphics` class methods require parameters. The parameters communicate to the `Graphics` methods information about what needs to be done. The `setColor()` method changes the current drawing color to the color passed in. `setColor()` cannot be called without a color parameter.



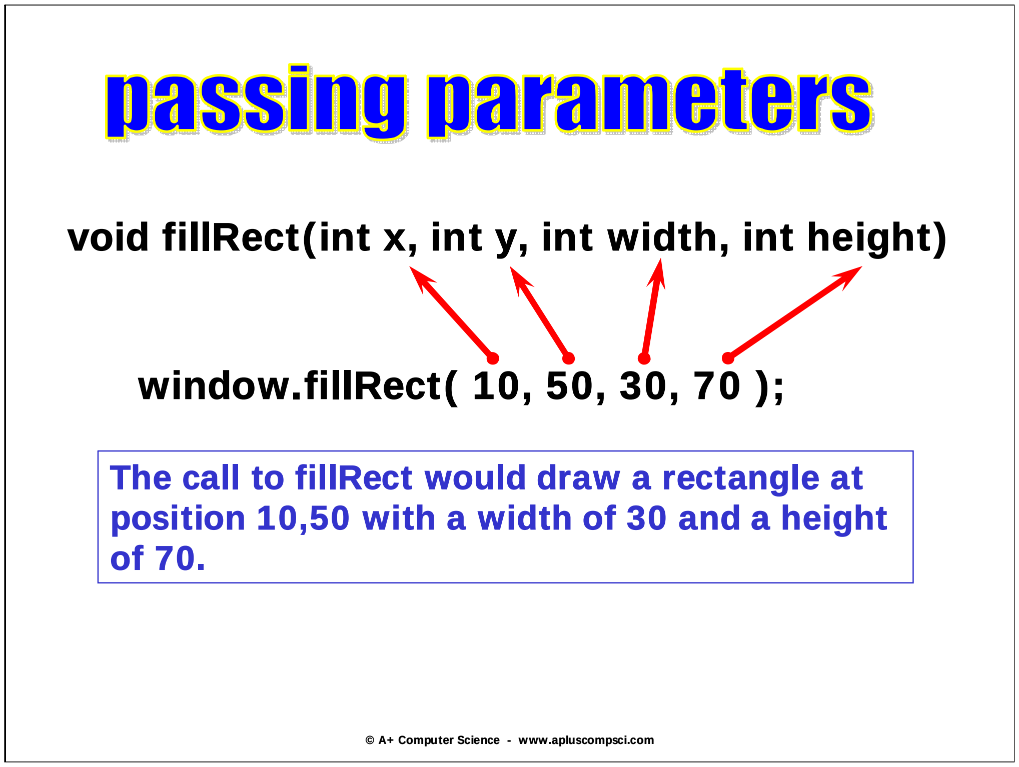
passing parameters

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

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

method call with parameters

The `fillRect()` method requires four pieces of information. `fillRect()` needs an x value, a y value, a width, and a height. `fillRect()` will draw a filled rectangle on the window at x,y with height and width as stated by the parameters.

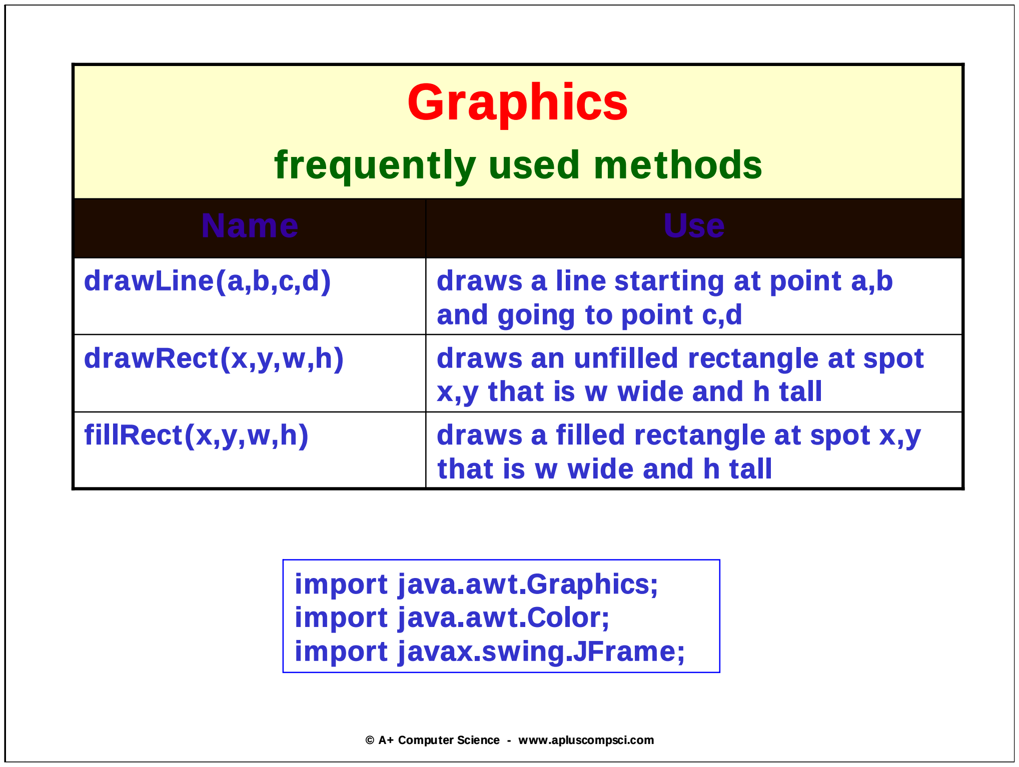


passing parameters

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

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.

The `fillRect()` method requires four pieces of information. `fillRect()` needs an x value, a y value, a width, and a height. `fillRect()` will draw a filled rectangle on the window at x,y with height and width as stated by the parameters.

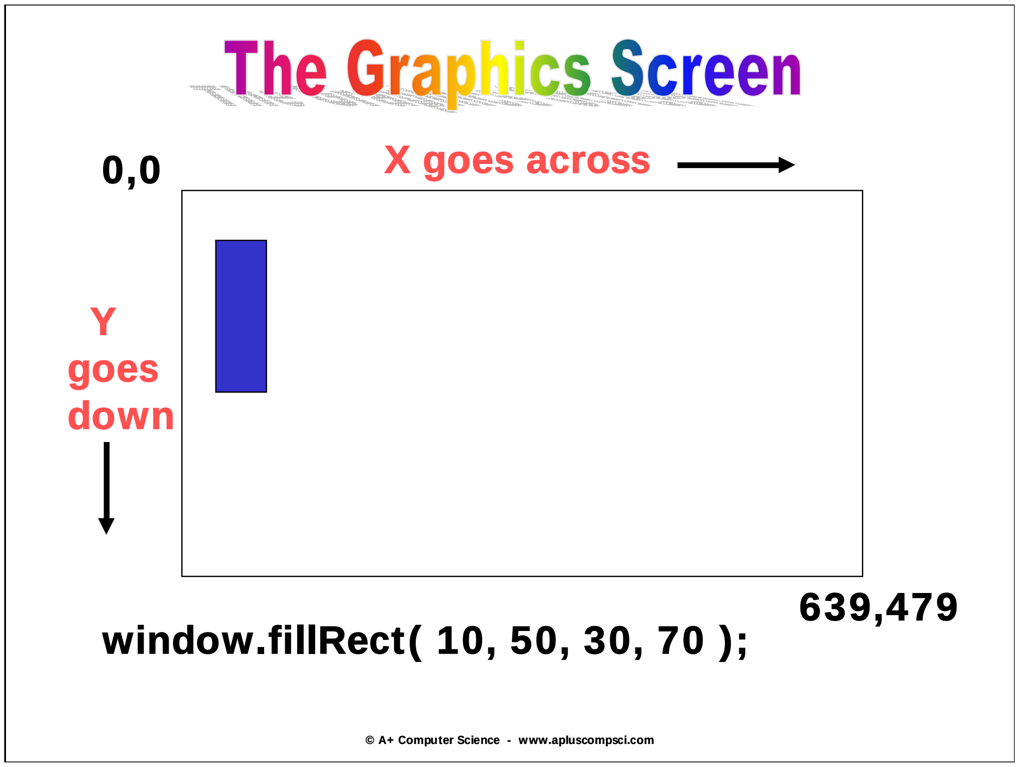


Graphics

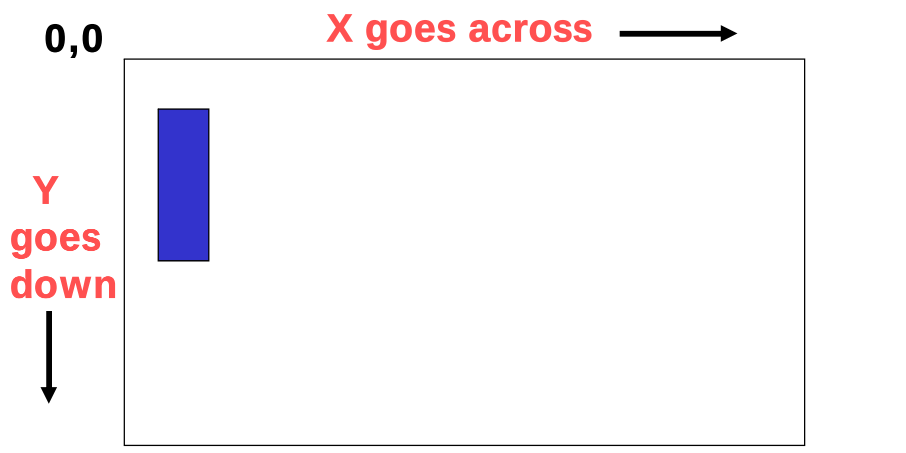
frequently used methods

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;
```



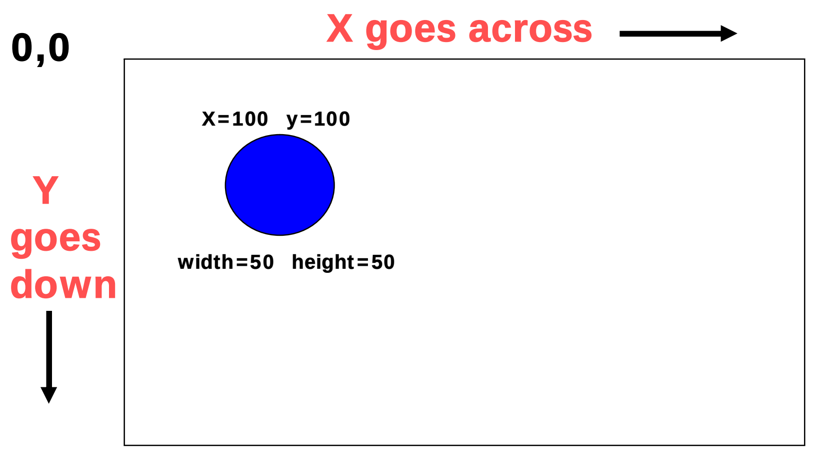
The Graphics Screen



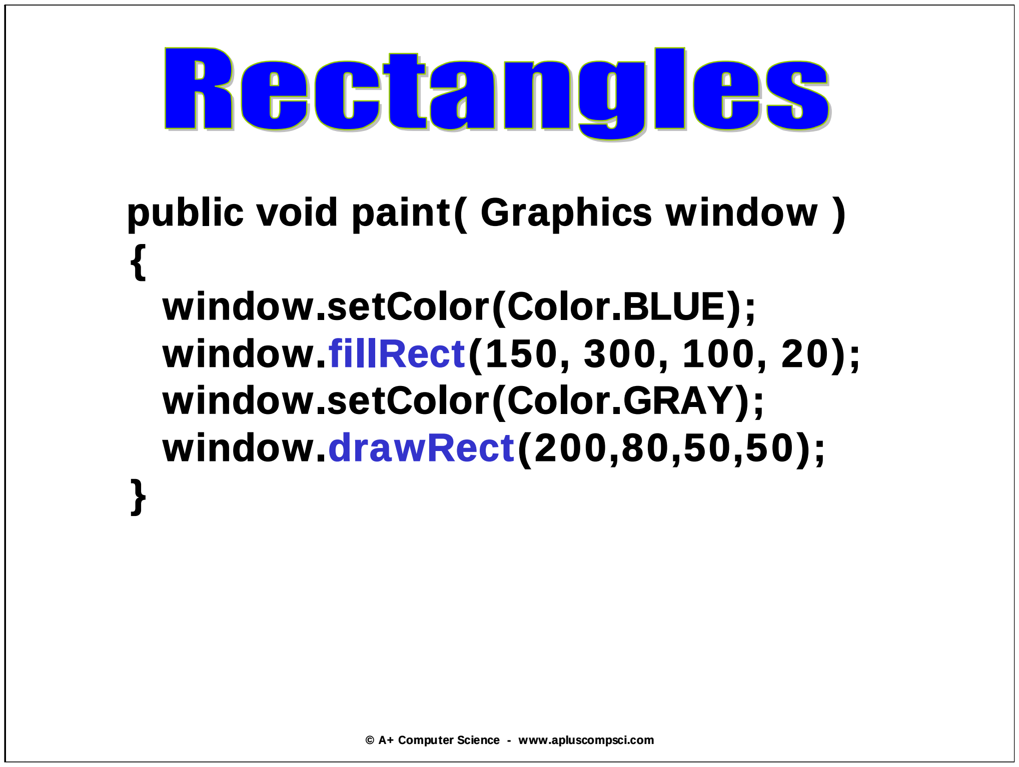
```
window.fillRect( 10, 50, 30, 70 );
```

Notice the Graphics screen being used with Graphics class does not use Cartesian coordinates. X goes across and Y goes down. X starts at 0 and goes to MAXX which in this case is 640. Y starts at 0 and goes down to MAXY which in this case is 479.

The Graphics Screen



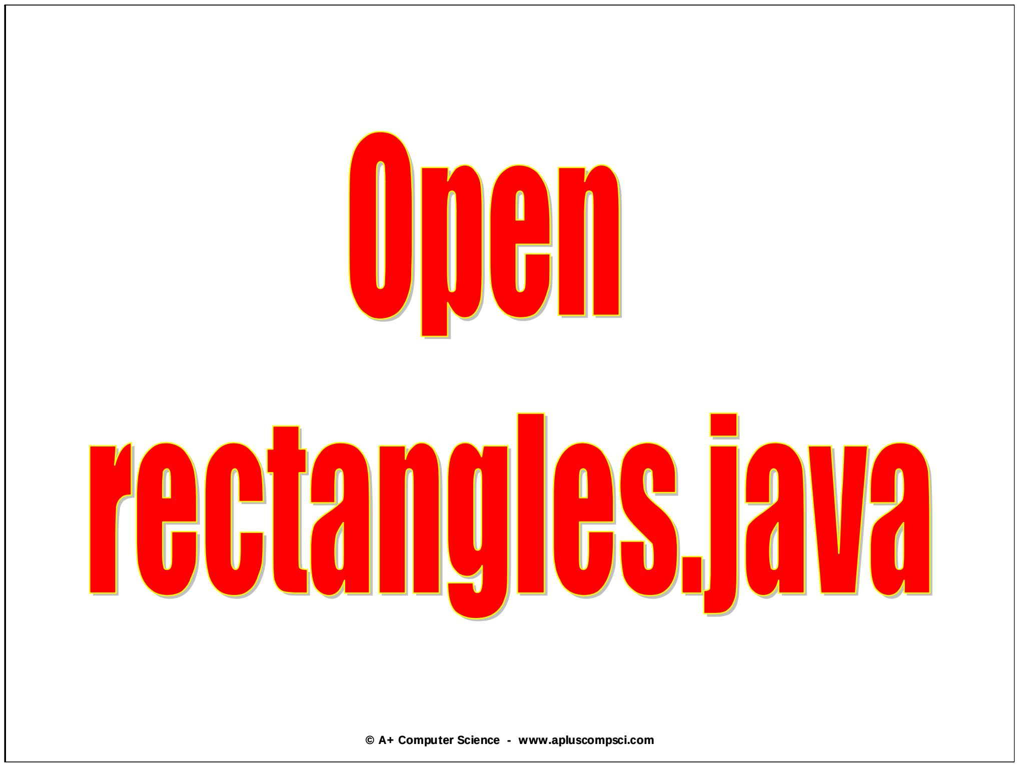
```
window.fillOval( 100, 100, 50, 50 );
```



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);
}
```

The `paint()` method is typically doing the most drawing. Other methods may be called from `paint()` as well.



Open rectangles.java

Open lines.java

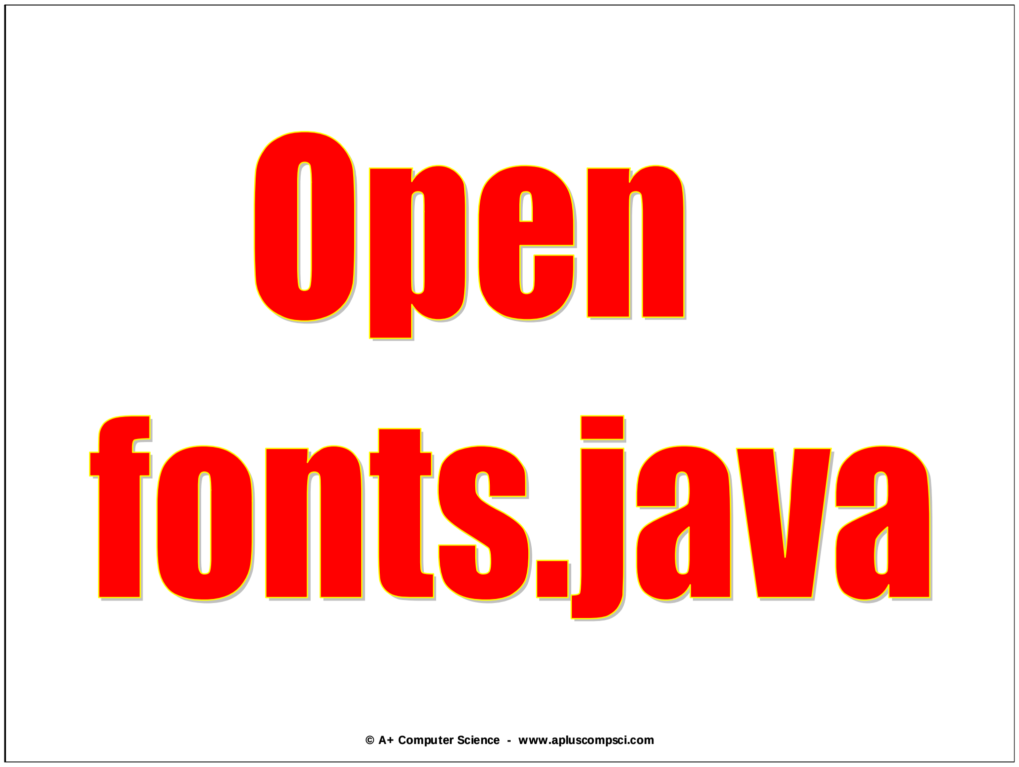
Graphics

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	

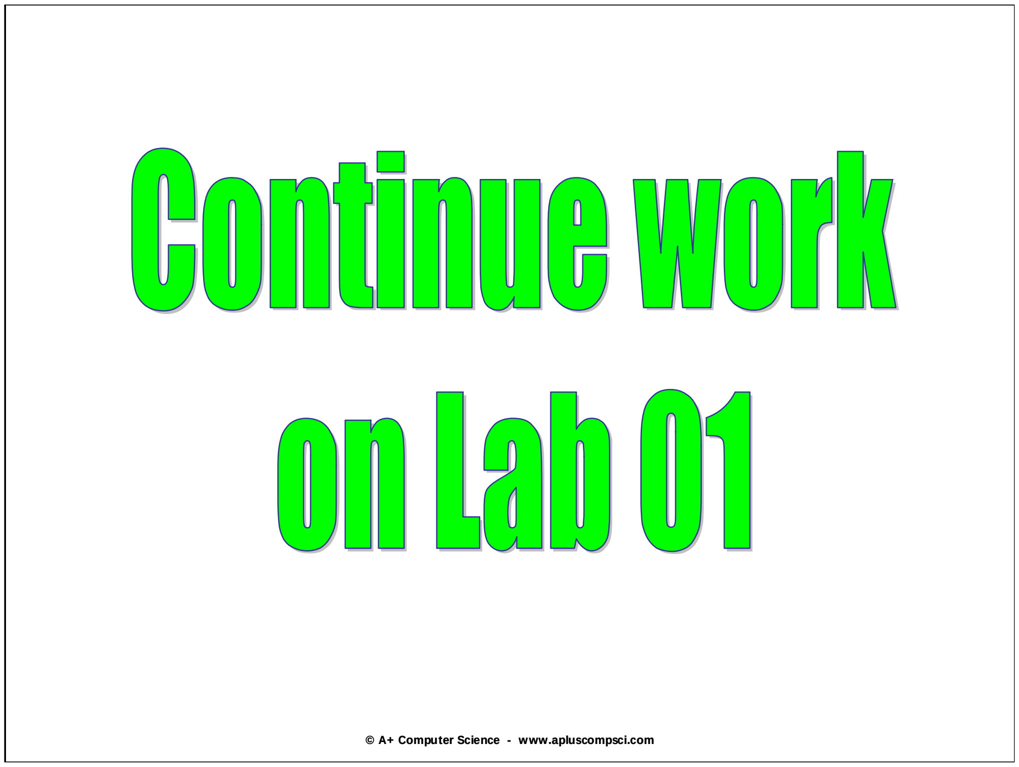
```
import java.awt.Graphics;
import java.awt.Color;
import javax.swing.JFrame;
```

Open arcs.java



Open fonts.java

Open colors.java



Continue work on Lab 01