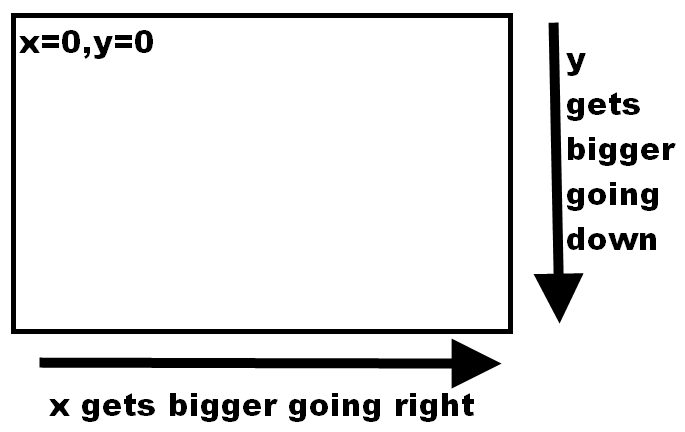
Java coordinate system  
x is 0 at the left edge of the screen  
x gets bigger as you move to the right  
y is 0 at the top edge of the screen  
y gets bigger as you move down



Begin photo  
Drawing of the Scratch's (x,y) coordinate system.  (0,0) is in the upper left corner.  X gets larger as we move to the right.  Y gets larger as we move down.  
End photo

## How do I draw a line?     put a dot on the page     put another dot on the page    draw a line between the 2 dots In this line:     blue dot:    x coordinate is 100, y coordinate is 100     green dot:  x coordinate is 540, y coordinate is 380

## A drawing of  a line.

#### Begin photo A drawing of  a line.   The upper left end has an x coordinate of 100 and a y coordinate is 100. The lower right end has an x coordinate of 540 and a y coordinate is 380. End photo

## We have to tell Java the (x,y) position of the first dot. We have to tell Java the (x,y) position of the second dot. The code drawLine(100,100, 540,380).  The "100,100" is the (x,y) coordinates of one of the line's end points.  The "540,380" is the (x,y) coordinates of the line's other end point.

#### Begin photo The code drawLine(100,100, 540,380).  The "100,100" is the (x,y) coordinates of one of the line's end points.  The "540,380" is the (x,y) coordinates of the line's other end point. End photo

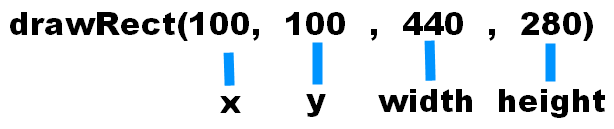
#### 

## Suppose I have this rectangle:  g.drawRect(100, 100 , 440 , 280);

## 

## Drawing of a rectangle.  The (x,y) coordinate of the rectangle's upper left corner is (100,100).  The rectangle has a width of 440.  The rectangle has a height of 280.

Begin photo  
Drawing of a rectangle.  The (x,y) coordinate of the rectangle's upper left corner is (100,100).  The rectangle has a width of 440.  The rectangle has a height of 280.  
End photo

  
Begin photo  
The code drawRect(100,100,440,280).  The 100 is the x coordinate of the rectangle.  The 2nd 100 is the y coordinate.  The 440 is the rectangle's width.  The 280 is the rectangle's height.  
End photo

## down 100 pixels g.drawRect(100, 200, 440, 280); up 100 pixels g.drawRect(100, 0, 440, 280); right 100 pixels g.drawRect(200, 100, 440, 280); left 100 pixels. g.drawRect(0, 100, 440, 280); 100 pixels wider. g.drawRect(100, 100, 540, 280); 100 pixels thinner g.drawRect(100, 100, 340, 280); 100 pixels taller g.drawRect(100, 100, 440, 380); 100 pixels shorter g.drawRect(100, 100, 440, 180);

#### 

## A dot is either a line with a length of 0 OR a circle with a width of 1 and a height of 1 The code for drawLine(100,100,100,100) and the code for drawOval(100,100,1,1).

Begin photo  
The code for drawLine(100,100,100,100) and the code for drawOval(100,100,1,1).  
The "100,100" in the drawLine code is the (x,y) coordinate of the line's left edge.  

#### [back](http://walshe.faculty.mjc.edu/virtual/introtoprogramming/lectures/10/035.html) [next](http://walshe.faculty.mjc.edu/virtual/introtoprogramming/lectures/10/037.html)

## An arc is a portion of an oval. Here is a 90 degree arc that starts at 225 degrees. [drawArc](http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html#drawArc%28int,%20int,%20int,%20int,%20int,%20int%29)(x, y, width, height, startAngle, arcAngle)

#### Drawing of an arc that looks like the smile in a smiley face.

Begin photo  
Drawing of an arc that looks like the smile in a smiley face.  
An arc is a portion of a circle.  The circle's upper left corner is at (x,y) coordinates (100,100).  
The left edge of the arc is at 225 degrees which is in the south east part of the circle.  
NOTE:  0 degrees is due east.  90 degrees is due north.  180 degrees is due west.  270 degrees is due south.  
The arc has a length of 90 degrees.  
End photo

## drawPolygon(int[] x, int[] y, int numberOfPoints)     int[] x is array of x coordinates     int[] y is array of y coordinates     numberOfPoints is the # of points

## The code of the x coordinate array for the circle is {100,200,300}.  The code for the y coordinate array for the circle is {300,100,300}.

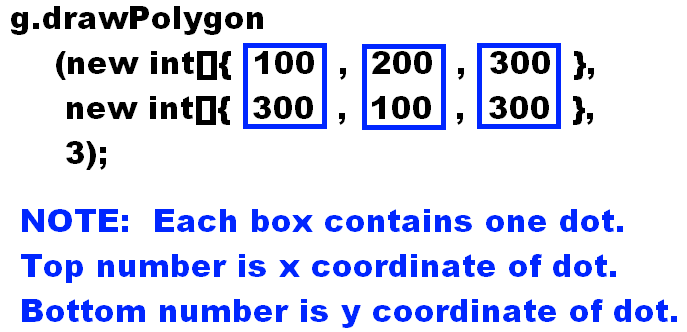
#### Begin photo The code of the x coordinate array for the circle is {100,200,300}.  The code for the y coordinate array for the circle is {300,100,300}. In the x coordinate array, the 100 is the x coordinate of the triangle's lower left vertex.  In the x coordinate array, the 200 is the x coordinate of the triangle's top vertex. In the x coordinate array, the 300 is the x coordinate of the triangle's lower right vertex. In the y coordinate array, the 300 is the y coordinate of the triangle's lower left vertex.  In the y coordinate array, the 100 is the y coordinate of the triangle's top vertex. In the y coordinate array, the 2nd 300 is the y coordinate of the triangle's lower right vertex. End photo

## The 1st number in the x array is the x coordinate of the 1st point The 1st number in the y array is the y coordinate of the 1st point IN THIS EXAMPLE:   the blue dot has x coordinate of 100 & y coordinate of 100

## The 2nd number in the x array is the x coordinate of the 2nd point The 2nd number in the y array is the y coordinate of the 2nd point IN THIS EXAMPLE:   the green dot has x coordinate of 200 & y coordinate of 100

## The 3rd number in the x array is the x coordinate of the 3rd point The 3rd number in the y array is the y coordinate of the 3rd point IN THIS EXAMPLE:   the green dot has x coordinate of 200 & y coordinate of 100

## 



## drawString(String str, int x, int y)     str:  the text that is to be displayed     x:     x coordinate of first character     y:     y coordinate of the baseline

## The code drawString("jog",100,100).

Begin photo  
The code drawString("jog",100,100).  
The "jog" is the string that is going to be drawn on the string.  
The "100,100" is the left edge of the text's base line.  
End photo



Begin photo  
Drawing showing what a base line is.  
When we learned how to write letters in grade school, we used ruled paper.  
Think of the base line as one of the lines on the ruled paper.  
The bottom of most of the letters touches the base line.  
The lower case letters g, j, p, q, and y are special cases.  The bottom of these letters are slightly below the base line.  
The left edge of the base line is where the first letter will be drawn; this edge has an (x,y) coordinate.  
End photo

#### 

## The code g.setFont(new Font(String, int, int) has three arguments.

Begin photo  
The code g.setFont(new Font(String, int, int) has three arguments.  
The String argument is the name of a typeface.  Some examples are "Dialog' or "Helvetica" or "SansSerif" or "TimesRoman" or "Serif" or "Courier" or "Monospace" or "Symbol."  
The int argument is the style.  A style can be either Font.PLAIN or Font.BOLD or Font.ITALIC.  
The 2nd int argument is the size.  Some examples of sizes are 9, 10, 12, 14, 16.  
End photo

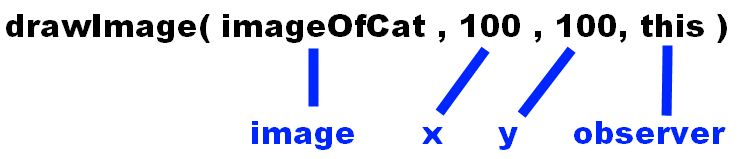
## Example: Suppose I have a graphics object named g. Write a small snippet of java code to display the string "hello" at the xy coordinates (123,321) in the following font:     "Courier" typeface     italic style     font size of 16 g.setFont(new Font("Courier", Font.ITALIC, 16)); g.drawString("hello",123, 321);

## Example: Suppose I have a graphics object named g. Write the java code to display this circle:     at x coordinate 123     at y coordinate 321     with a width of 100       in the color orange g.setColor(Color.ORANGE); g.drawOval(123, 321, 100, 100); NOTE:  make sure you call setColor( ) BEFORE you call drawOval( )

## javadocs Graphics class search for a method that can draw an image You will find several methods--I suggest you pick the simplest method.

## drawImage(Image image, int x, int y, ImageObserver observer)     Image is the image we want to render on screen     (x,y) is the location of the image's upper left corner     ImageObserver is the object that is notified     when the image is finished being copied from disk to memory

## 



Begin photo  
The code drawImage(imageOfCat, 100, 100, this)  
The "imageOfCat" is a variable that contains an image of a cat.  
The "100 100" are the (x,y) coordinates of the cat.  
The "this" is the observer.  
End photo

## Screenshot of the screen.  A dot is seen at x,y coordinates (100,100).   The dot is in the upper left corner of a cat. Begin photo Screenshot of the screen.  A dot is seen at x,y coordinates (100,100).   The dot is in the upper left corner of a cat. End photo

#### [back](http://walshe.faculty.mjc.edu/virtual/introtoprogramming/lectures/11/003.html) [next](http://walshe.faculty.mjc.edu/virtual/introtoprogramming/lectures/11/005.html)

## "this" refers to the object that contains this command the drawImage( ) command is in the paintFrame( ) method. The paintFrame( ) method is inside the [Game](http://walshe.faculty.mjc.edu/virtual/introtoprogramming/lectures/11/Game.java) class.

## drawImage(Image image, int x, int y, ImageObserver observer) How does a Java program render an image onto the screen? You give the Java program the name of a graphics file (e.g. cat.gif). The Java program copies the image from the file into memory. Because disk drives are slow, this takes an incredible amount of time. When the image has finally finished loading into memory, a notification is sent to the ImageObserver (e.g. "hey there, your image is ready"). In our simple game, we are going to ignore the notification.

## Game is a class. A class is a factory that can create objects. We used the Game class to create a Game object. The "this" refers to the Game object that we have created.

## The word "this" refers to the Game object that is being created.

Begin photo  
The word "this" refers to the Game object that is being created.  
This is a screenshot of the following code:  
    class Game extends JFrame {  
        public void paintFrame(Graphics g) {  
            /\* Type your code here \*/  
            g.drawImage(imageOfCat, 100, 100, this);  
        }  
        //lots of additional code  
    }  
An arrow points from the word "this" that is in the fourth line to the word "Game" that is in the first line.  
Write a snippet of code to perform the following tasks:

Create an int variable named x

Insert the value 123 into the variable

Extract the value out of the variable

Use System.out.println to print the extracted value.

int x = 123;

System.out.println(x);

Write a snippet of code to perform the following tasks:

Create a double variable named salary

Insert the value 1234.56 into the variable

Extract the value out of the variable

Use System.out.println to print the extracted value.

double salary = 1234.56;

System.out.println(salary);

Write a snippet of code to perform the following tasks:

Create a String variable named nameOfPlayer

Insert the value "John Doe" into the variable

Extract the value out of the variable

Use System.out.println to print the extracted value.

String nameOfPlayer = "John Doe";

System.out.println(nameOfPlayer);

## /\* Type your variable declarations here \*/  Sprite cat = new Sprite(new ImageIcon("cat.gif").getImage(), 267, 167); Sprite bananas1 = new Sprite(new ImageIcon("bananas.png").getImage(), 67, 92); /\*\*  \* paint the current frame of the game  \* @param g handle to the screen \*/ public void paintFrame(Graphics g) {      if (this.isLeftArrowPressed()) {         cat.x = cat.x - 10;     }     if (this.isRightArrowPressed()) {         cat.x = cat.x + 10;     }     g.drawImage(cat.image, cat.x, cat.y, this);     if (bananas1.visible) {             bananas1.paintComponent(g);     } }

## Concentric Circles:

## int x = 100; int y = 100; int width = 100; int height = 100; do {     g.drawOval(x, y, width, height);     x = x + 5;     y = y + 5;     width = width - 10;     height = height - 10; } while (width>0 && height>0);

## Draw Steps:

## int x = 100;     int y = 200;     int width = 100;     int height = 10;     do {         g.drawRect(x, y, width, height);         x = x + 5;         y = y - 10;         width = width - 10;     } while (width>0);

Draw a Circle of Circles:

## int angle = 360; do {     int x = 200+(int)Math.round(100.0\*Math.cos(Math.toRadians(angle)));     int y = 200+(int)Math.round(100.0\*Math.sin(Math.toRadians(angle)));     g.drawOval(x, y, 30, 30);     angle = angle-10; } while (angle>=0);

## This code does this: display 1 wait 1 second display 2 wait 1 second display 3 wait 2 second display 4 etc.

## int timeDelay = 15; int number = 1; public void paintFrame(Graphics g) {     g.setFont(new Font("Dialog",Font.BOLD,72));     g.drawString(""+number, 100, 100);     if (this.timeDelay>0) {         this.timeDelay--;          return;     }     number++;     timeDelay = 15;  }

## Draw a circle when the mouse is clicked, and draw it on the cursor:

## int x = -100; int y = -100; int timeDelay=0; public void paintFrame(Graphics g) {      if (isMouseClicked()) {         x = getMouseX();         y = getMouseY();         timeDelay = 30;     }     g.drawOval(x, y, 10, 10);     if (timeDelay>0) {         timeDelay--;         return;     }     x = -100;     y = -100; }

## This object has no properties(there is nothing in the middle box)

## The methods are in the bottom of the class diagram.

## Concentration(Sprite sprites[]) is the constructor.

|  |
| --- |
| Concentration |
|  |
| Concentration(Sprite sprites[]) void ifMouseIsClickedOnCardThenFlipCardFaceUp() int numberOfFaceUpCards() boolean doAllFaceUpCardsMatch() void flipAllCardsFaceDown() void makeAllFaceUpCardsInvisible() void resetAllCards() int numberOfFaceDownCards() int numberOfVisibleCards() void paintComponent(Graphics g) |

## 

|  |
| --- |
| Sprite |
| Image image int x int y int width int height boolean visible |
| Sprite(Image image, int x, int y) boolean intersects(Sprite sprite) void paintComponent(Graphics g) |

|  |
| --- |
| Game |
| GameBoard gameboard |
| void initializeSprites() boolean isUpArrowPressed() boolean isDownArrowPressed() boolean isLeftArrowPressed() boolean isRightArrowPressed() boolean isSpaceKeyPressed() int getMouseX() int getMouseY() boolean isMouseClicked() int randomNumber(int minNumber, int maxNumber) |

|  |
| --- |
| GameBoard |
|  |
| int getWidth() int getHeight() |