

The background is a dark blue gradient with faint, light blue circular patterns. On the left side, there are several concentric circles with degree markings ranging from 40 to 260. Some of these circles have arrows indicating a clockwise direction. The main title is centered on the right side in a large, white, sans-serif font.

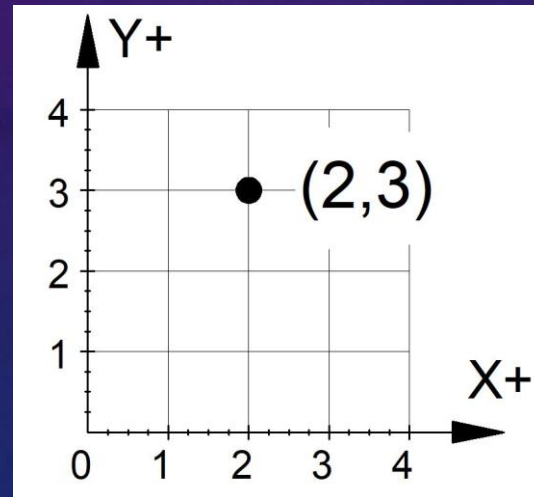
# PROCESSING: THE COORDINATE SYSTEM, FUNCTIONS, COLORS, AND COMMENTS

UNIVERSITY OF MOUNT UNION

FALL 2022

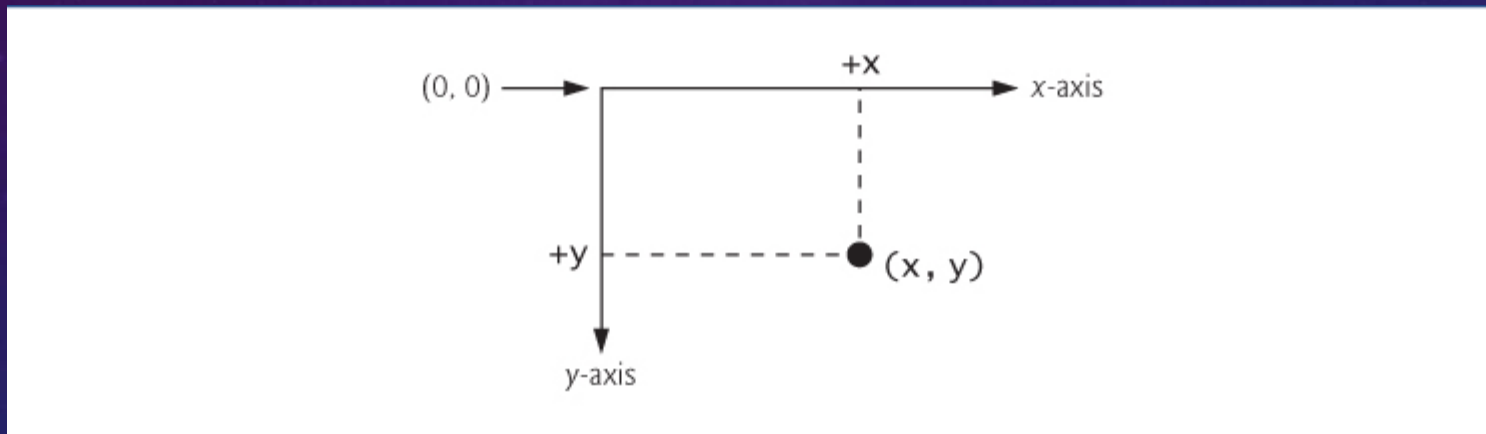
# CARTESIAN COORDINATE SYSTEM

- Positive x & y are in the top right quadrant (“Quadrant I”), so x-values (horizontal) increase as they go to the right and y-values (vertical) increase as they go up.



# PROCESSING COORDINATE SYSTEM

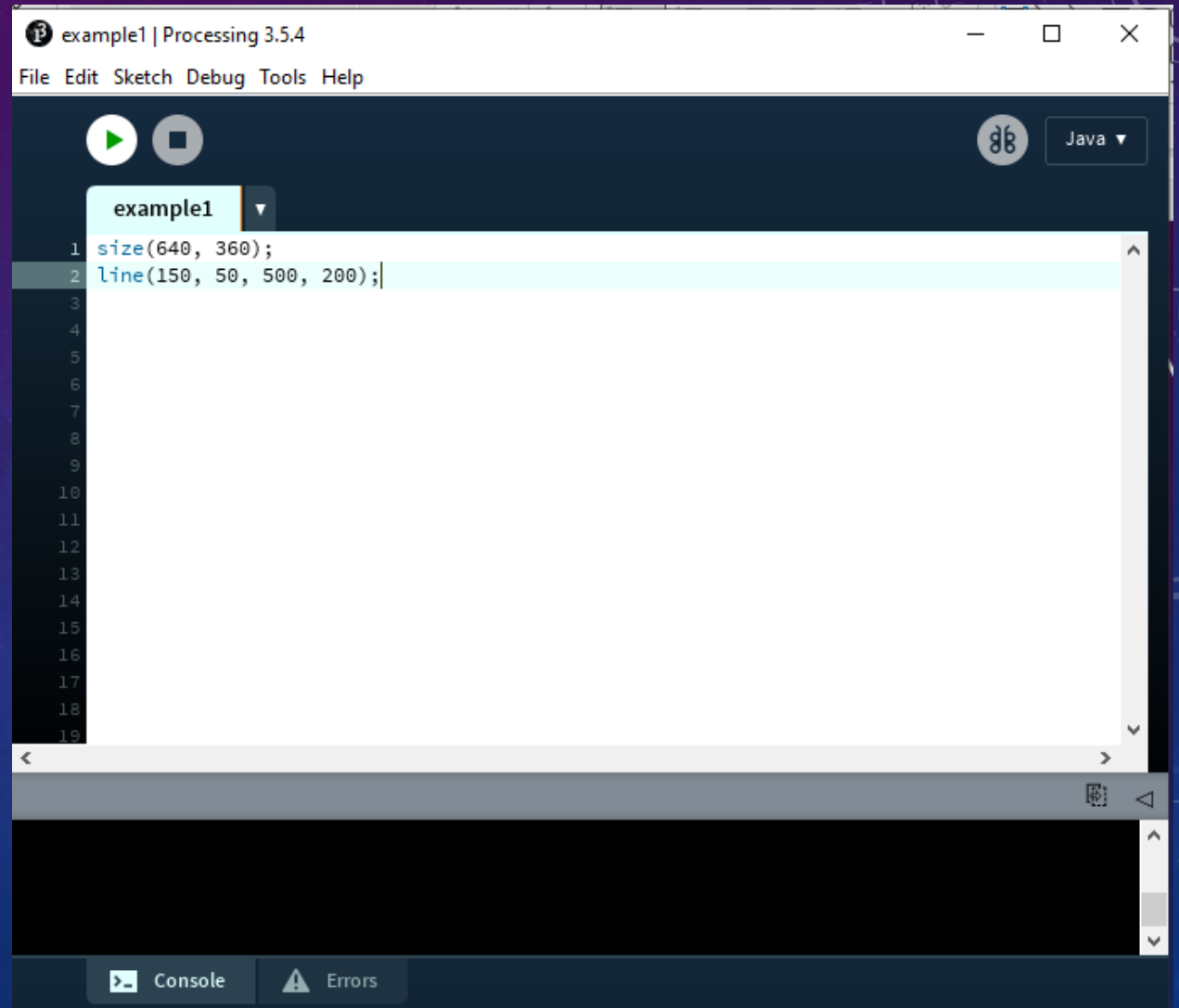
- The origin is at the TOP LEFT of the sketch, so x-values (horizontal) still increase as they go to the right...but y-values (vertical) increase as they go DOWN!



- Not too hard to understand, but a little counter-intuitive, so we're bound to make things go floppy-floppy sometimes. No problem! Just try again 😊

# OK, SO WE HAVE A COORDINATE SYSTEM! LET'S...DRAW A LINE!

- “Draw a line from (150, 50) to (500, 200).”
- We understand that perfectly well...  
Processing doesn't.
- Let's try `line(150, 50, 500, 200);`
- Hmm! Too small...let's try this:



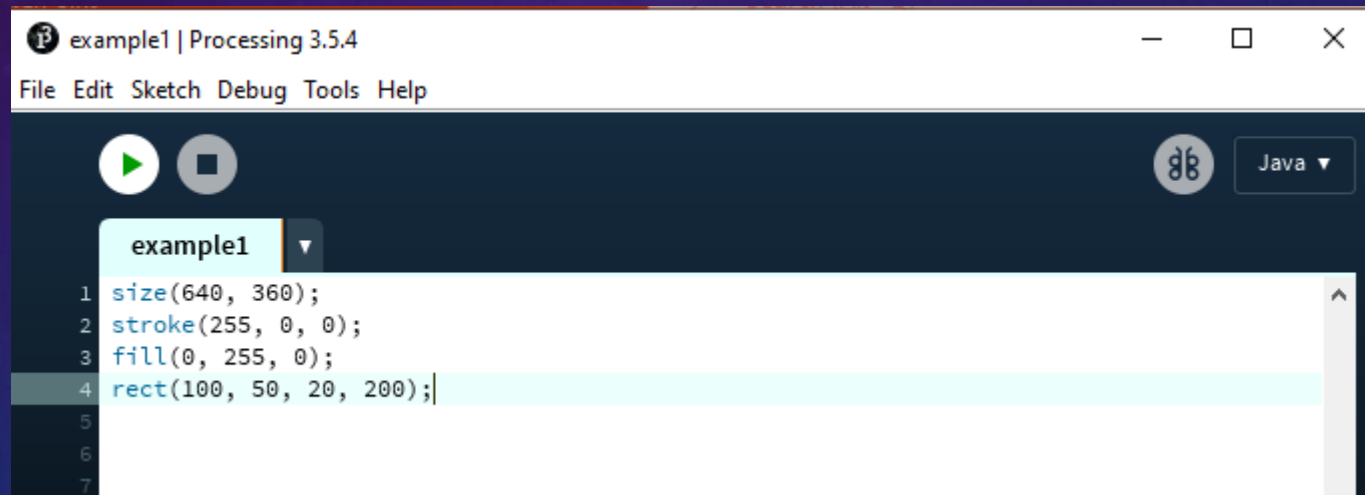
# BOOK EXAMPLES

- Try examples 2-1 and 2-2! *Make sure to read the text, too!* 😊



# COLOR

- Digital color mixing uses the properties of light, which mixes red, green, and blue (RGB).
- Red, green, and blue can have values from 0-255 (hexadecimal codes, just like in HTML)
- Try this:

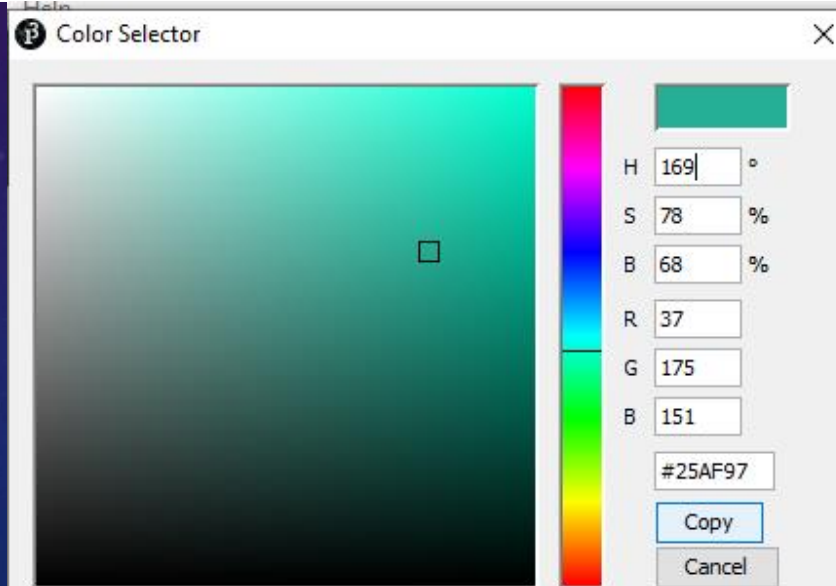
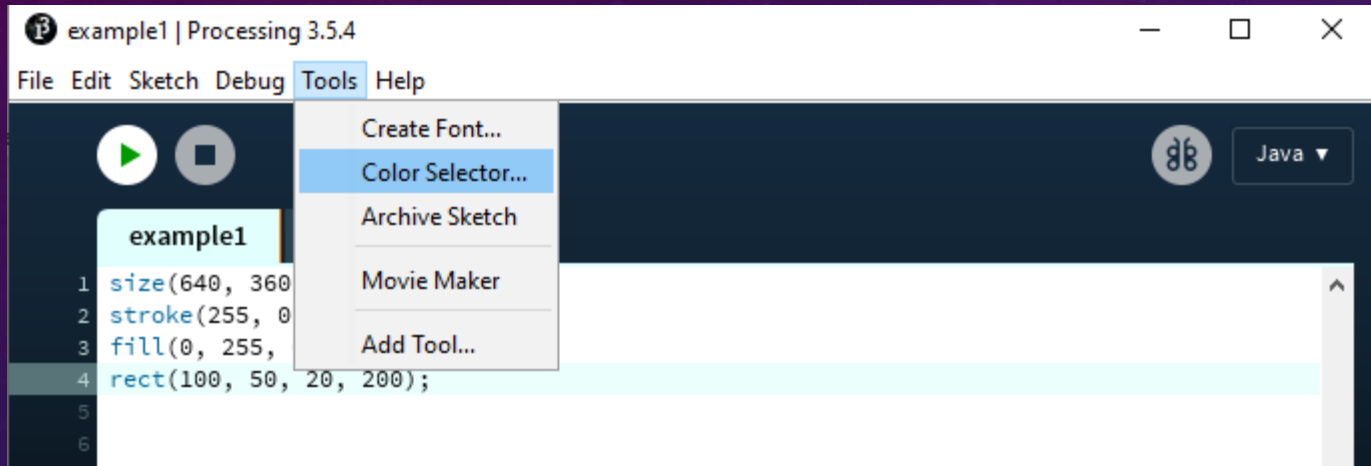


```
example1 | Processing 3.5.4
File Edit Sketch Debug Tools Help

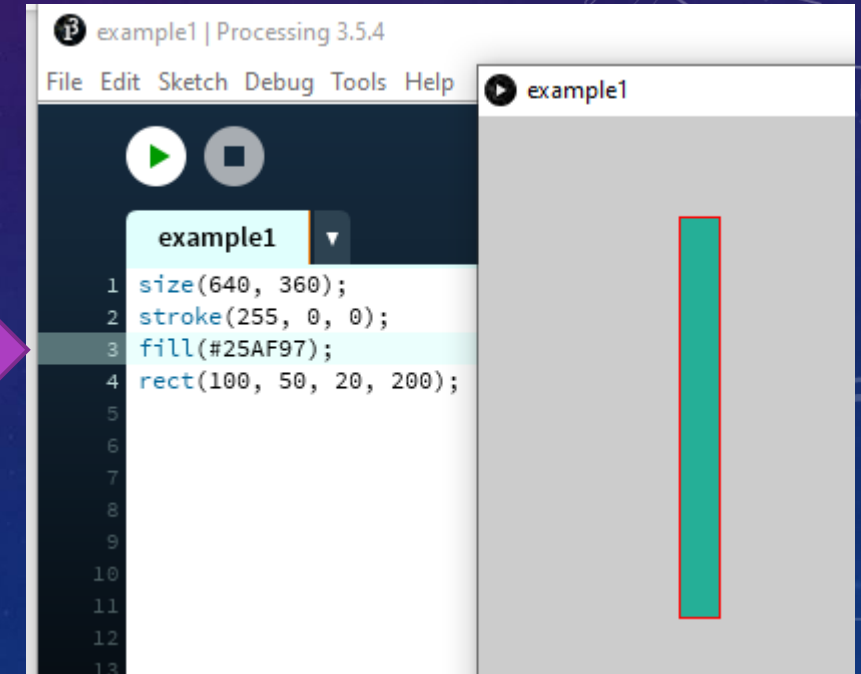
▶ ◼ Java ▼

example1 ▼
1 size(640, 360);
2 stroke(255, 0, 0);
3 fill(0, 255, 0);
4 rect(100, 50, 20, 200);|
5
6
7
```

# YOU CAN ALSO USE HEXADECIMAL CODES

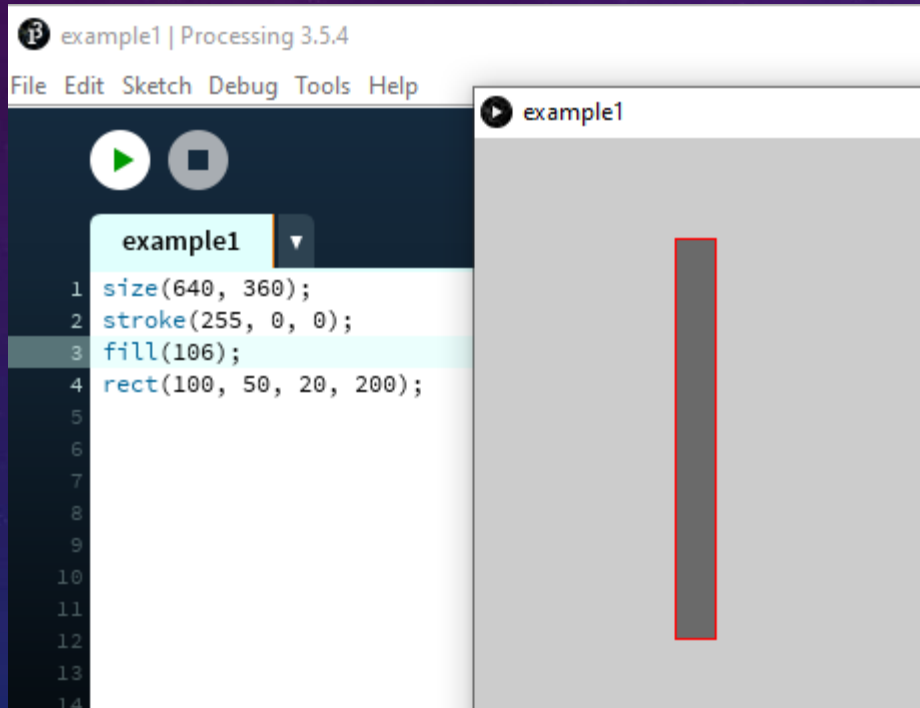


After  
copying the  
hex code,  
paste it  
inside the  
"fill"  
parentheses

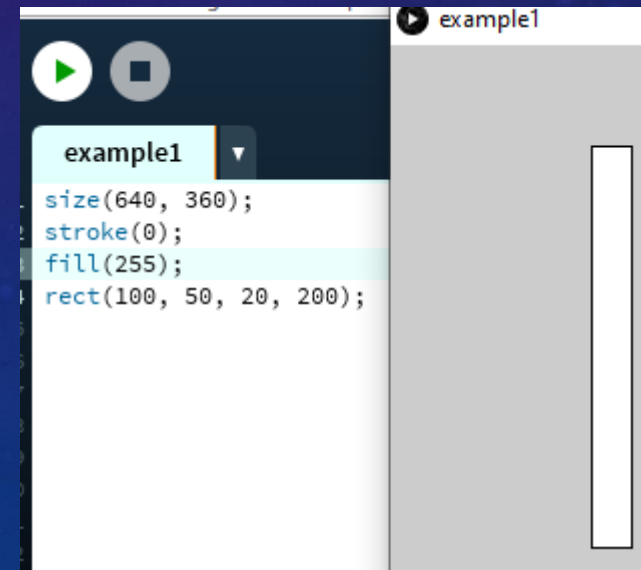


# AND LET'S NOT FORGET GRAYSCALE

- If just one number between 0-255 is inside the parentheses, Processing will assume that this same value should be applied to R, G, and B...which results in a shade between black and white!

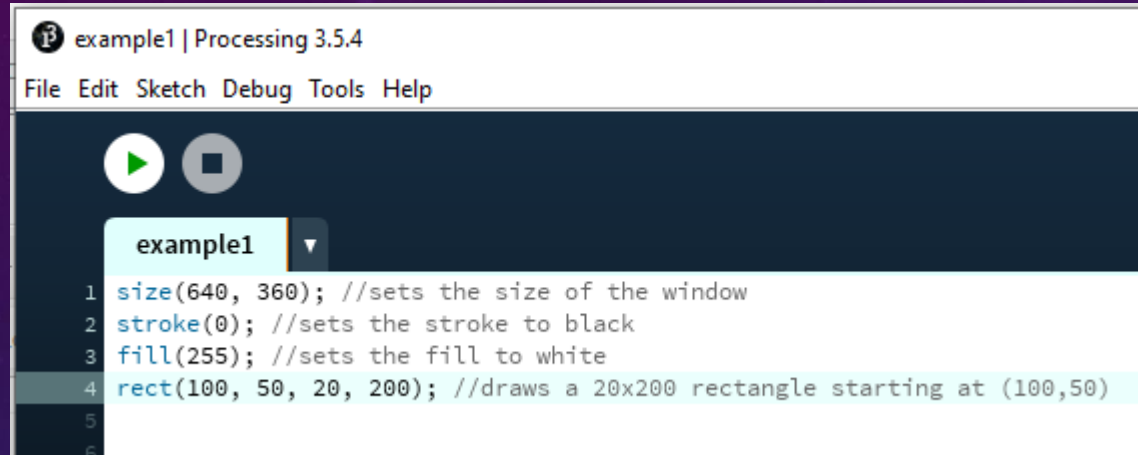


- Want black or white? Black is 0 (shorthand for #000000 or 0, 0, 0) and white is 255 (shorthand for #FFFFFF or 255, 255, 255)



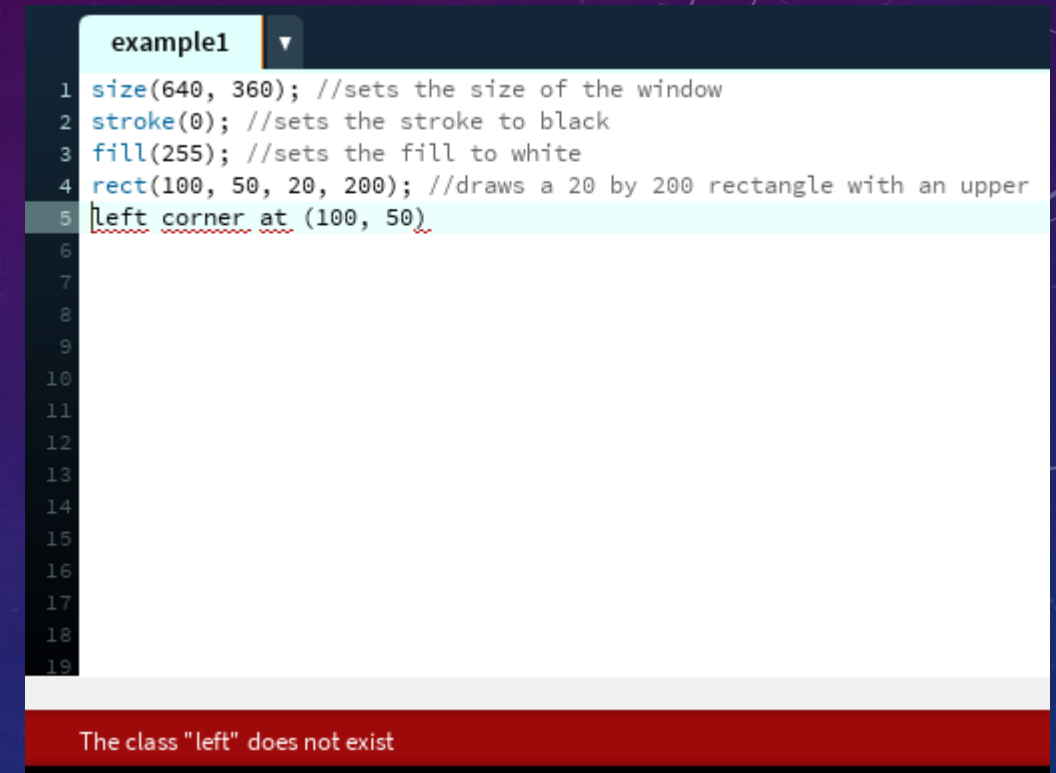


# COMMENTING IN CODE



A screenshot of the Processing IDE window titled "example1 | Processing 3.5.4". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. Below the menu bar are play and stop buttons. A code editor shows the following code:

```
1 size(640, 360); //sets the size of the window
2 stroke(0); //sets the stroke to black
3 fill(255); //sets the fill to white
4 rect(100, 50, 20, 200); //draws a 20x200 rectangle starting at (100,50)
```

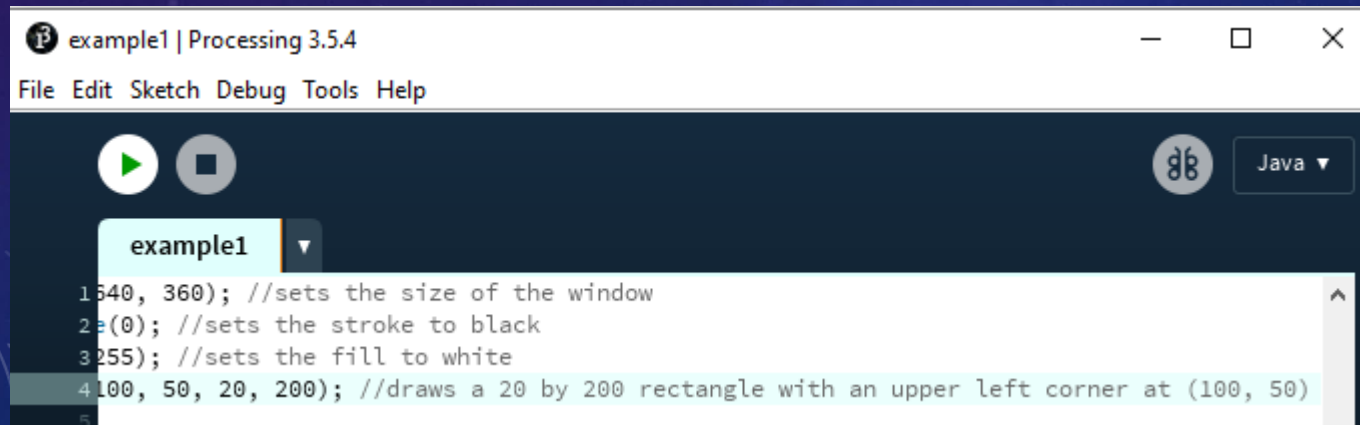


A screenshot of the Processing IDE window titled "example1". The code editor shows the same code as the previous image, but the comment for the `rect` function is split across two lines:

```
4 rect(100, 50, 20, 200); //draws a 20 by 200 rectangle with an upper
5 left corner at (100, 50)
```

At the bottom of the IDE, a red error bar displays the message: "The class 'left' does not exist".

What if the comment can't fit on one line?



A screenshot of the Processing IDE window titled "example1 | Processing 3.5.4". The code editor shows the same code as the previous images, but the comment for the `rect` function is split across two lines:

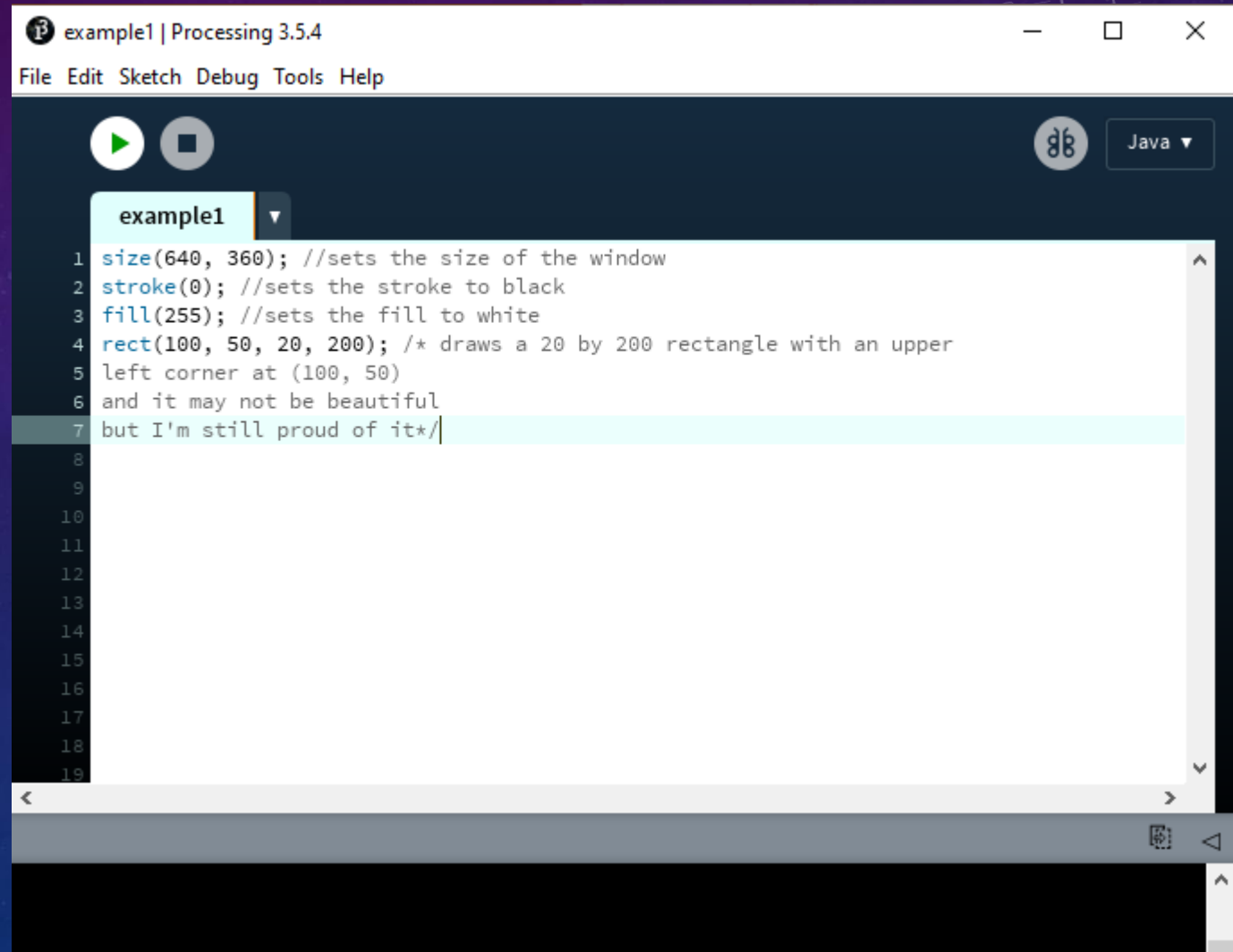
```
4 100, 50, 20, 200); //draws a 20 by 200 rectangle with an upper left corner at (100, 50)
```



Uh oh...that doesn't look good

# MULTI-LINE COMMENTS

- Whew! That's better.



The screenshot shows the Processing IDE window titled "example1 | Processing 3.5.4". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". Below the menu bar are icons for running (a green play button) and stopping (a grey square button), and a language dropdown menu set to "Java". The code editor shows a sketch named "example1" with the following code:

```
1 size(640, 360); //sets the size of the window
2 stroke(0); //sets the stroke to black
3 fill(255); //sets the fill to white
4 rect(100, 50, 20, 200); /* draws a 20 by 200 rectangle with an upper
5 left corner at (100, 50)
6 and it may not be beautiful
7 but I'm still proud of it*/
8
9
10
11
12
13
14
15
16
17
18
19
```

The multi-line comment on lines 4 through 7 is highlighted with a light blue background. The code editor has a scrollbar on the right side.

# PRINT() AND PRINTLN()

- Here is an example with both...what do you think the difference is?

Code:

```
1 println("hello world");  
2 print("it's nice");  
3 print(" to meet you");  
4  
5
```

Console Output:

```
hello world  
it's nice to meet you
```

# PRINT() AND PRINTLN()

- Here is an example with both...what do you think the difference is?
- If your thought was that println() forces a carriage return (new line) after whatever it prints, and print() does not (it doesn't even insert a space)...you're right!

Code:

```
1 println("hello world");  
2 print("it's nice");  
3 print(" to meet you");  
4  
5
```

Console Output:

```
hello world  
it's nice to meet you
```