An Introduction to Processing

Formatting shapes

Lecturer: Caio Fonseca

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Topics list

- Filling shapes with colour.
- Formatting the shape outline.
- Adding comments to your code.

fill() - Syntax

fill (r, g, b)

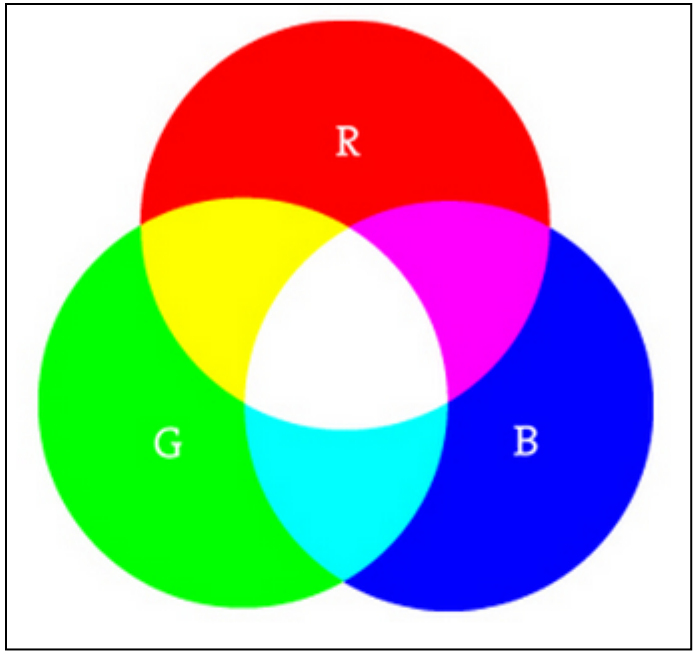
r = red colour (a number between 0 and 255 inclusive)

g = green colour (a number between 0 and 255 inclusive)

b = blue colour (a number between 0 and 255 inclusive)

- fills shapes with a chosen colour.
- can use the RGB colours to select a colour.
- all shapes drawn after the **fill** function is called, will be filled with the chosen colour.

A Recap of RGB Colours



"As with grayscale, the individual color elements are expressed as ranges from 0 (none of that color) to 255 (as much as possible), and they are listed in the order R, G, and B."

Digital colours are made by mixing the three primary colours of light (red, green, and blue).

https://www.processing.org/tutorials/color/

fill()

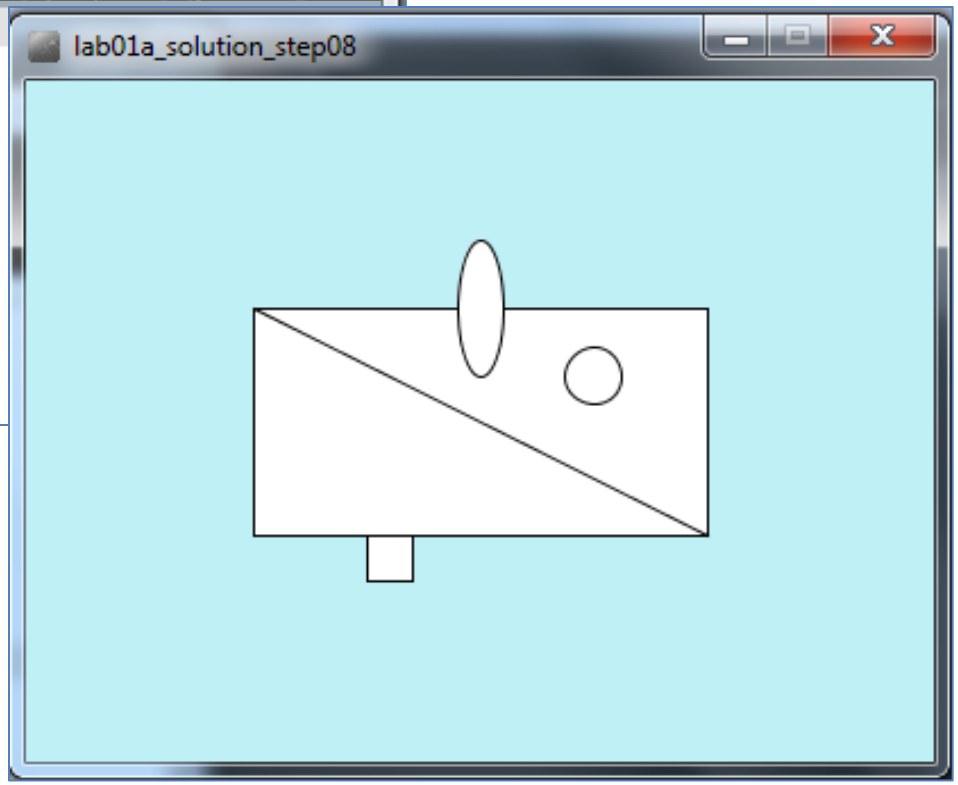
lab01a_solution_step08 | Processing 2.2.1

File Edit Sketch Tools Help

Java

lab01a_solution_step08

```
size(400,300);
background(190,240,245);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);
ellipse(200,100,20,60);
ellipse(250,130,25,25);
```



Starting code…

fill()

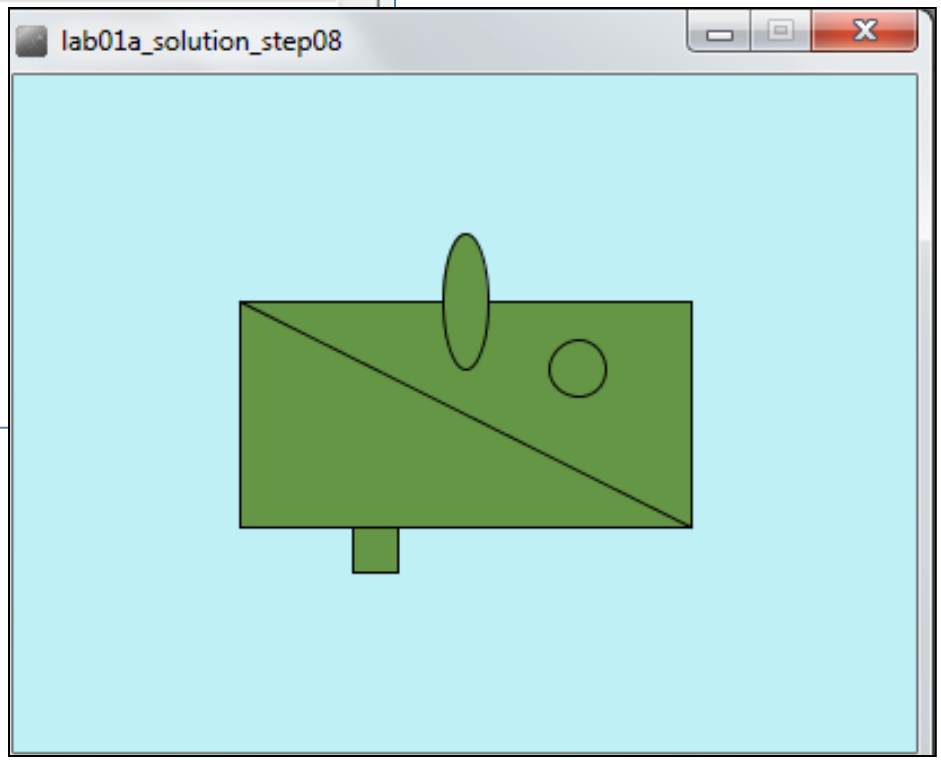
lab01a_solution_step08 | Processing 2.2.1

File Edit Sketch Tools Help

Java

lab01a_solution_step08

```
size(400,300);
background(190,240,245);

fill(100,150,70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);
ellipse(200,100,20,60);
ellipse(250,130,25,25);
```



All shapes filled with dark green…

fill()

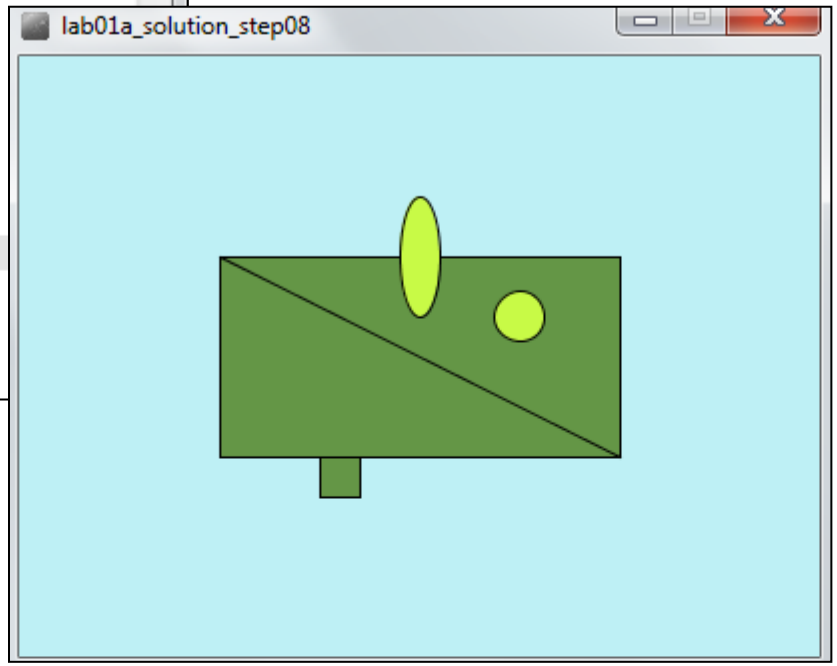
lab01a_solution_step08 | Processing 2.2.1

File Edit Sketch Tools Help

Java

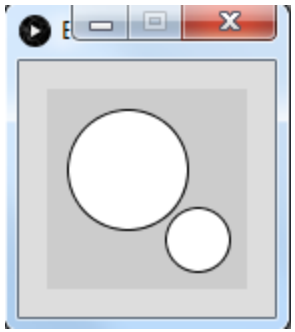
lab01a_solution_step08

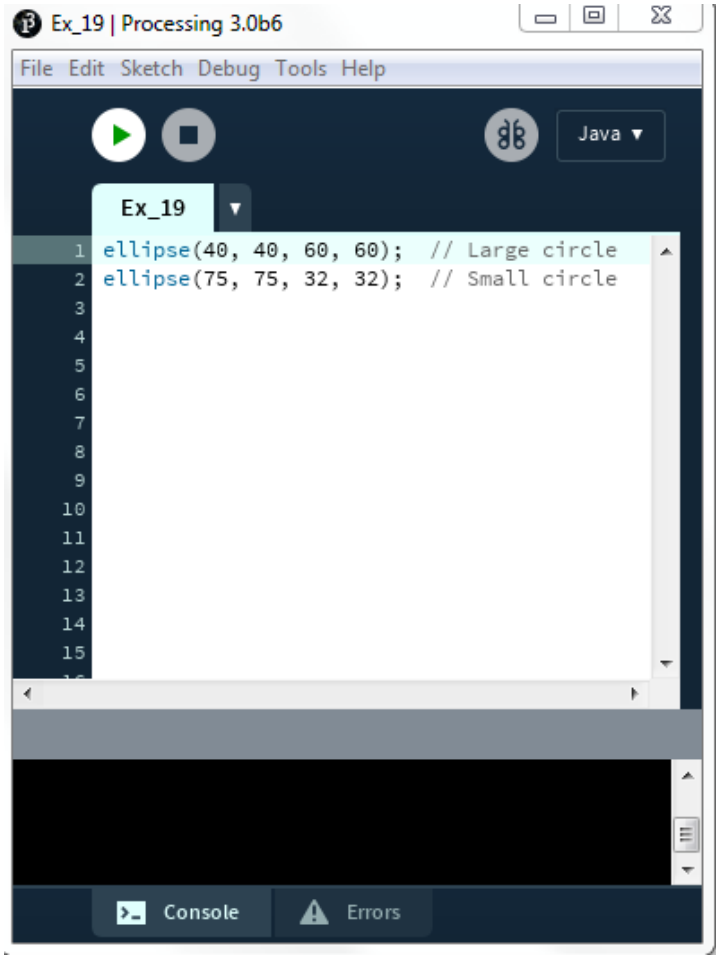
```
size(400,300);
background(190,240,245);

fill(100,150,70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);

fill(200,250,70);

ellipse(200,100,20,60);
ellipse(250,130,25,25);
```



Rectangles filled with dark green…
Ellipses filled with light green…
Order of statements matter!!!

fill() – More Examples

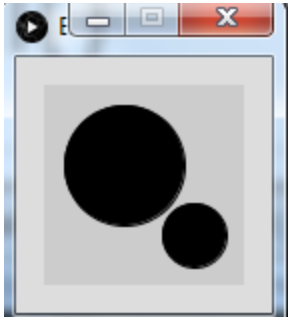


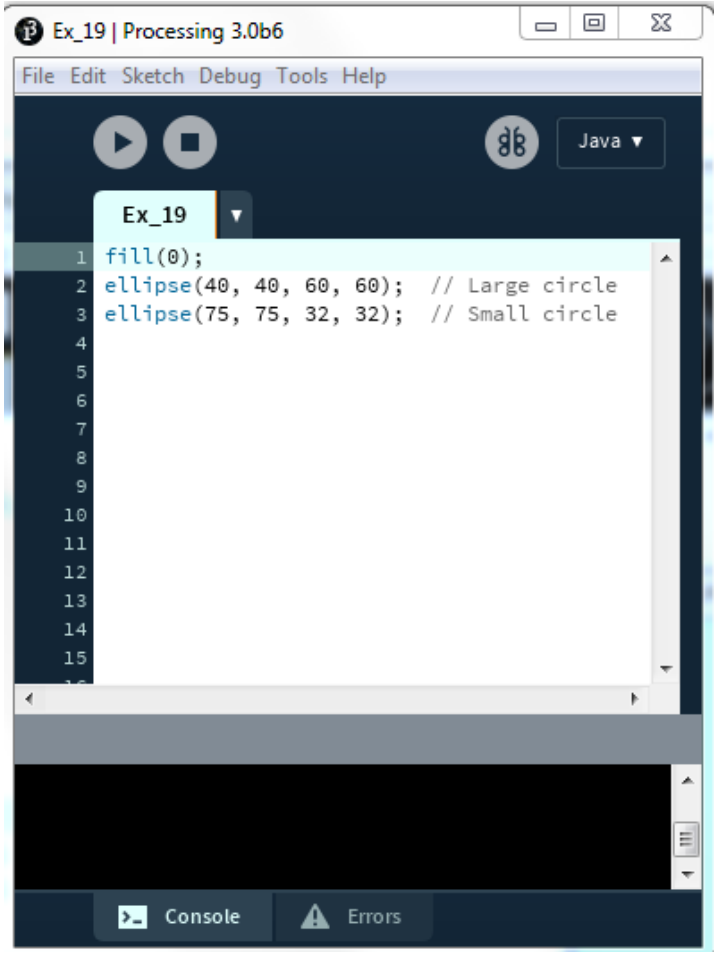


Starting code…

fill() – More Examples

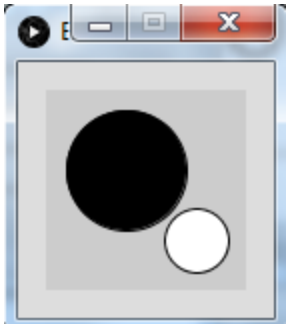
noFill()

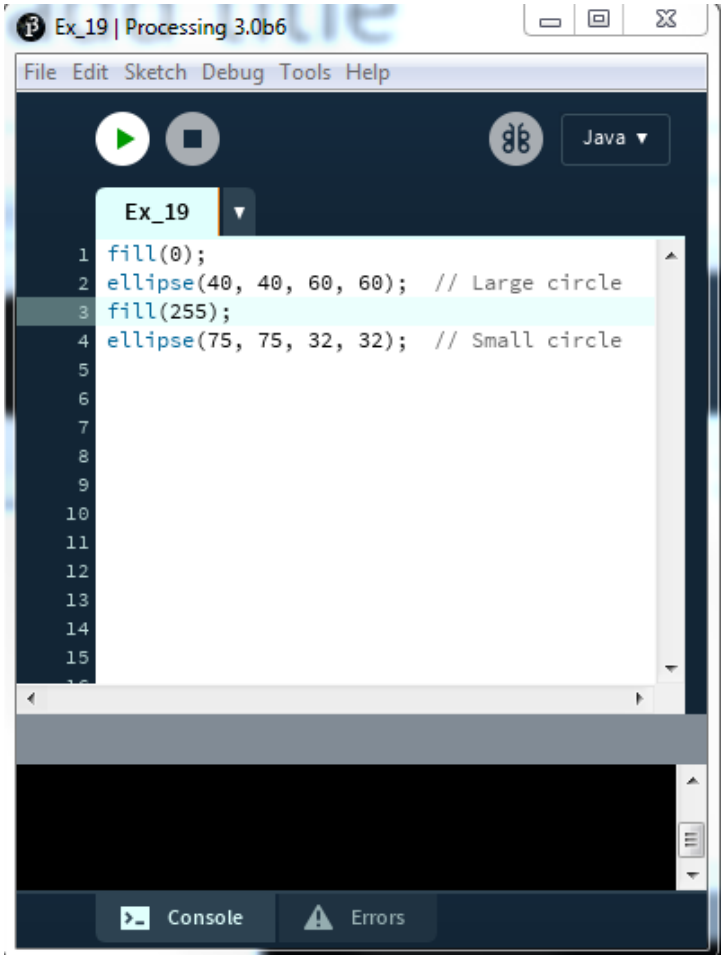




Circles filled with black

fill() – More Examples





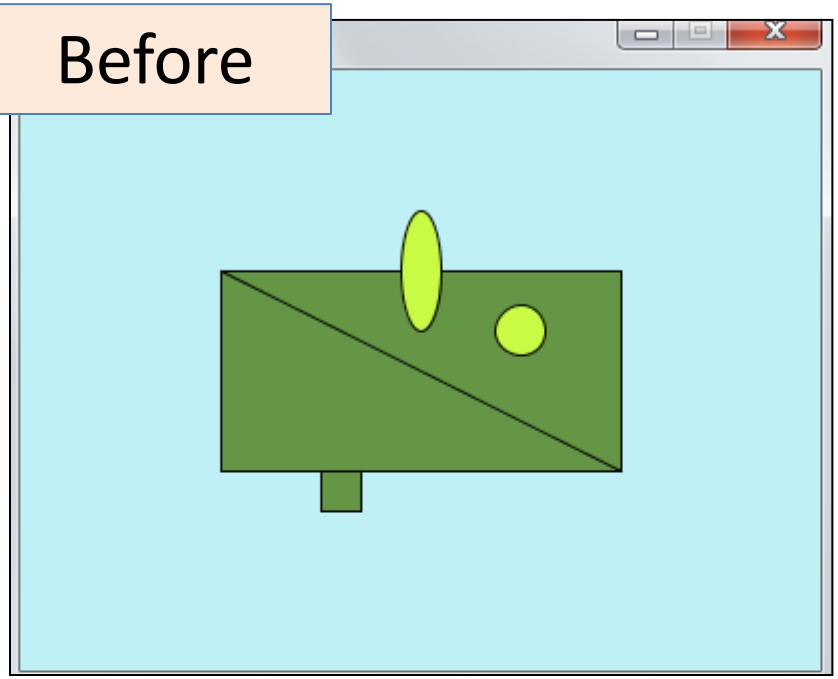
First Circle filled with black…
Second Circle filled with white…
Order of statements matter!!!

Topics List

- Filling shapes with colour.
- Formatting the shape outline.
- Adding comments to your code.

Changing the outline (i.e. stroke)

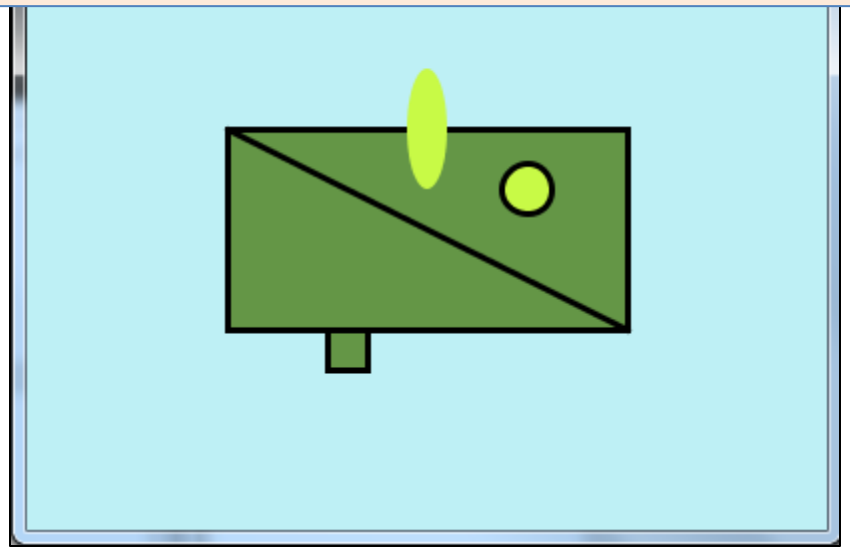
Before



After (changes):

- The oval has no border; all other shapes do.
- The outline is heavier.

We will now make those changes

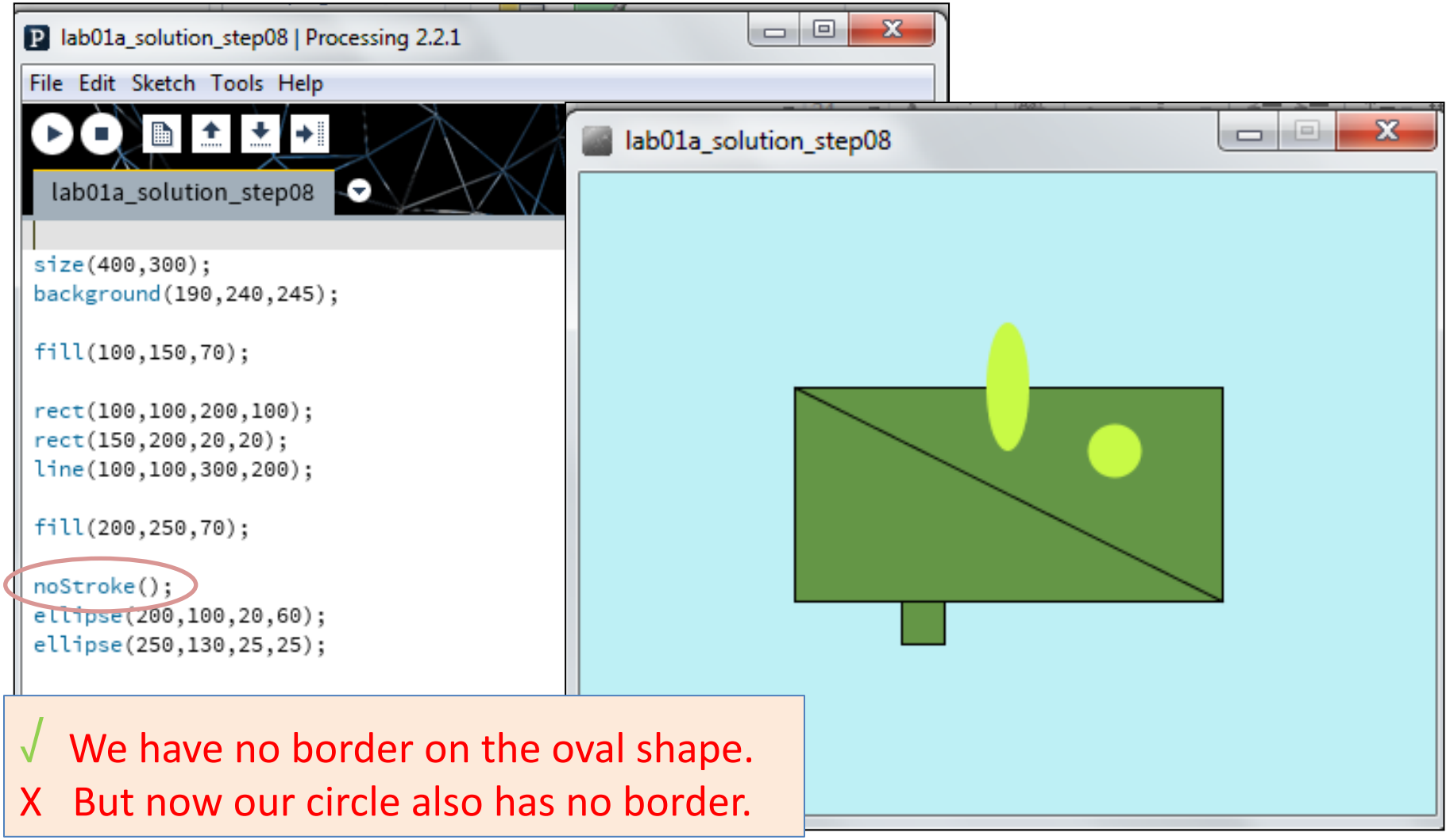


noStroke() - syntax

```
noStroke();
    //no parameters defined for this function.
```

- A **stroke** is the outline of a shape.
- The noStroke() function disables the outline on shapes that are drawn after the function is called.
- All shapes drawn after the **noStroke** function is called, will have no outline.

noStroke()



√ We have no border on the oval shape.

X But now our circle also has no border.

stroke() - syntax

stroke (r, g, b)

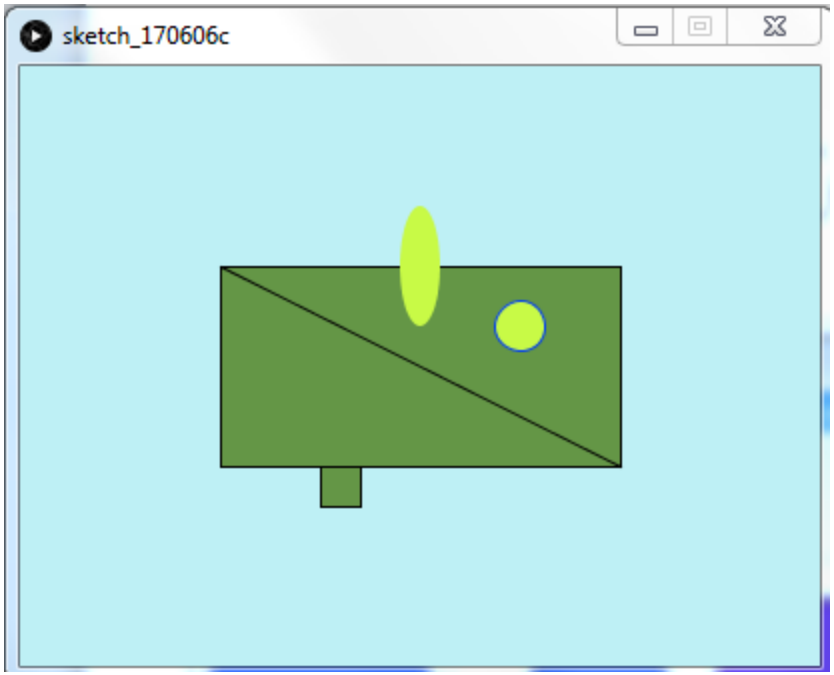
r = red colour (a number between 0 and 255 inclusive)

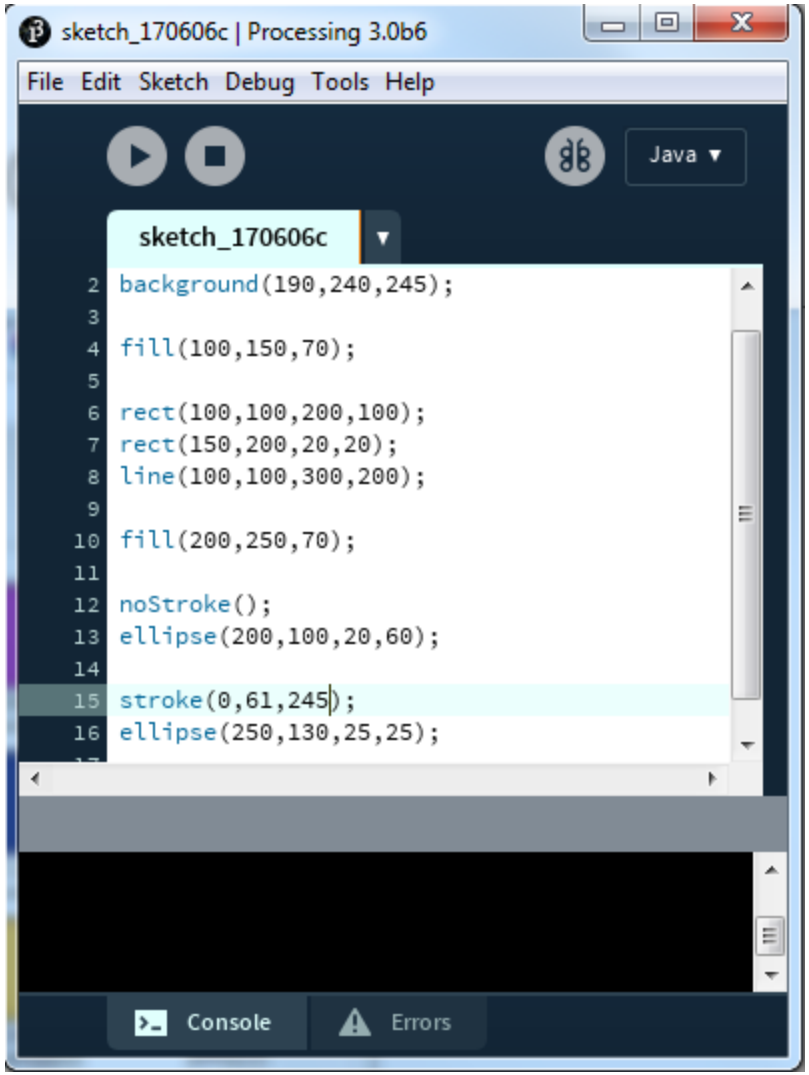
g = green colour (a number between 0 and 255 inclusive)

b = blue colour (a number between 0 and 255 inclusive)

- The stroke() function enables the outline on all shapes that are drawn after the function is called.
- When you call stroke(), you need to specify a colour.

stroke()





✓ Our circle now has a blue border.

stroke()

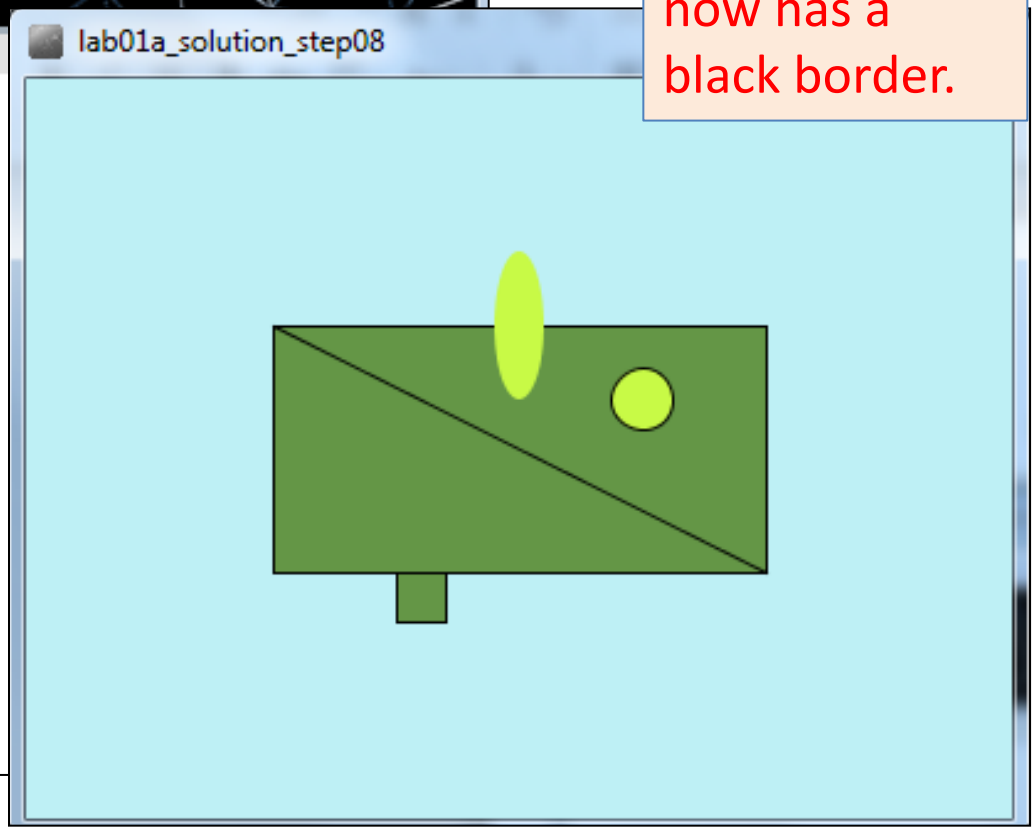
lab01a_solution_step08 | Processing 2.2.1

File Edit Sketch Tools Help

Java

lab01a_solution_step08

```
size(400,300);
background(190,240,245);

fill(100,150,70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);

fill(200,250,70);

noStroke();
ellipse(200,100,20,60);

stroke(0,0,0);
ellipse(250,130,25,25);
```



√ Our circle now has a black border.

strokeWeight() - Syntax

strokeWeight (pixels)

pixels = thickness of the outline measures in pixels.

- The strokeWeight() function allows you to choose the thickness of a line/outline on shapes.
- The chosen thickness will apply to all lines/shapes that are drawn after the function is called.
- The thickness is specified in pixels.
- The default thickness is 1 pixel.

strokeWeight()

lab01a_solution_step08 | Processing 2.2.1

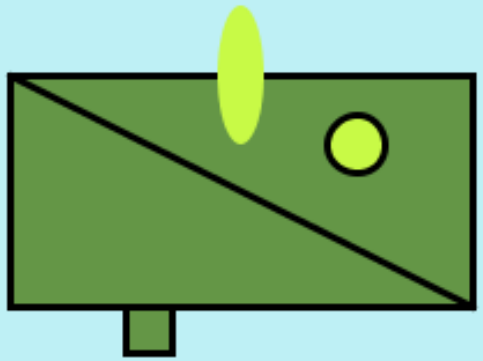
File Edit Sketch Tools Help

Stop Java

lab01a_solution_step08

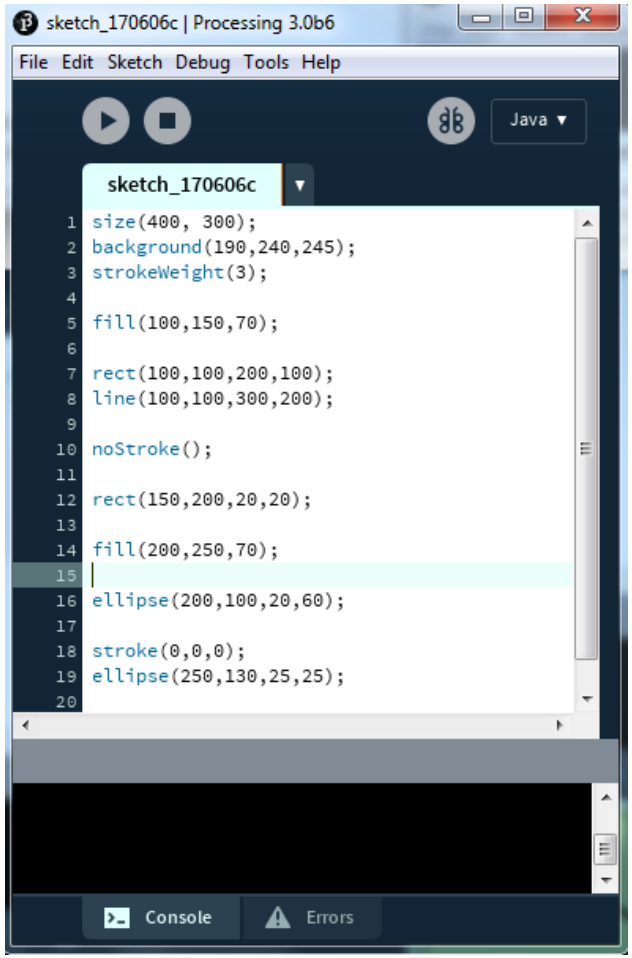
```
size(400,300);
background(190,240,245);
strokeWeight(3);

fill(100,150,70);

rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);

fill(200,250,70);

noStroke();
ellipse(200,100,20,60);

stroke(0,0,0);
ellipse(250,130,25,25);
```

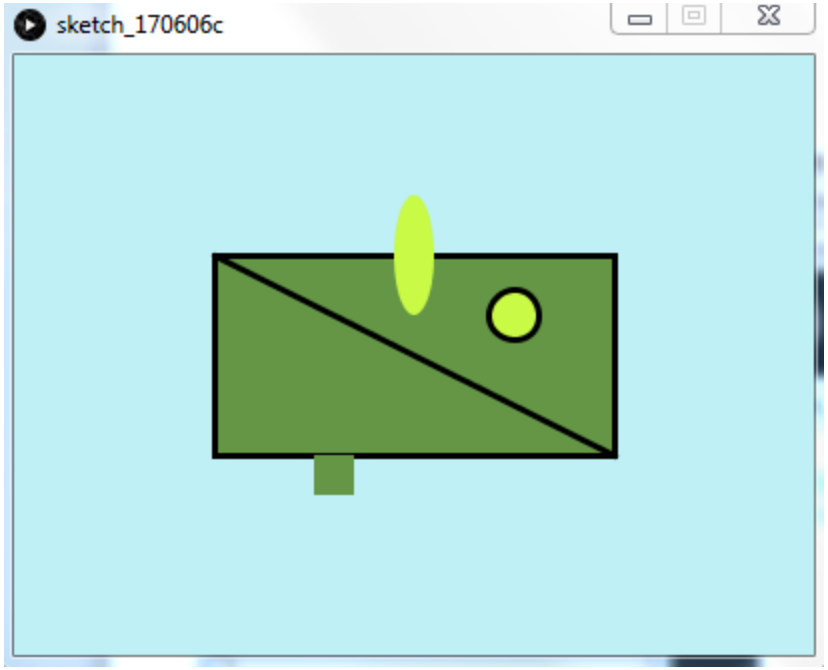
lab01a_solution_step08



√ Our outline is now heavier

strokeWeight()



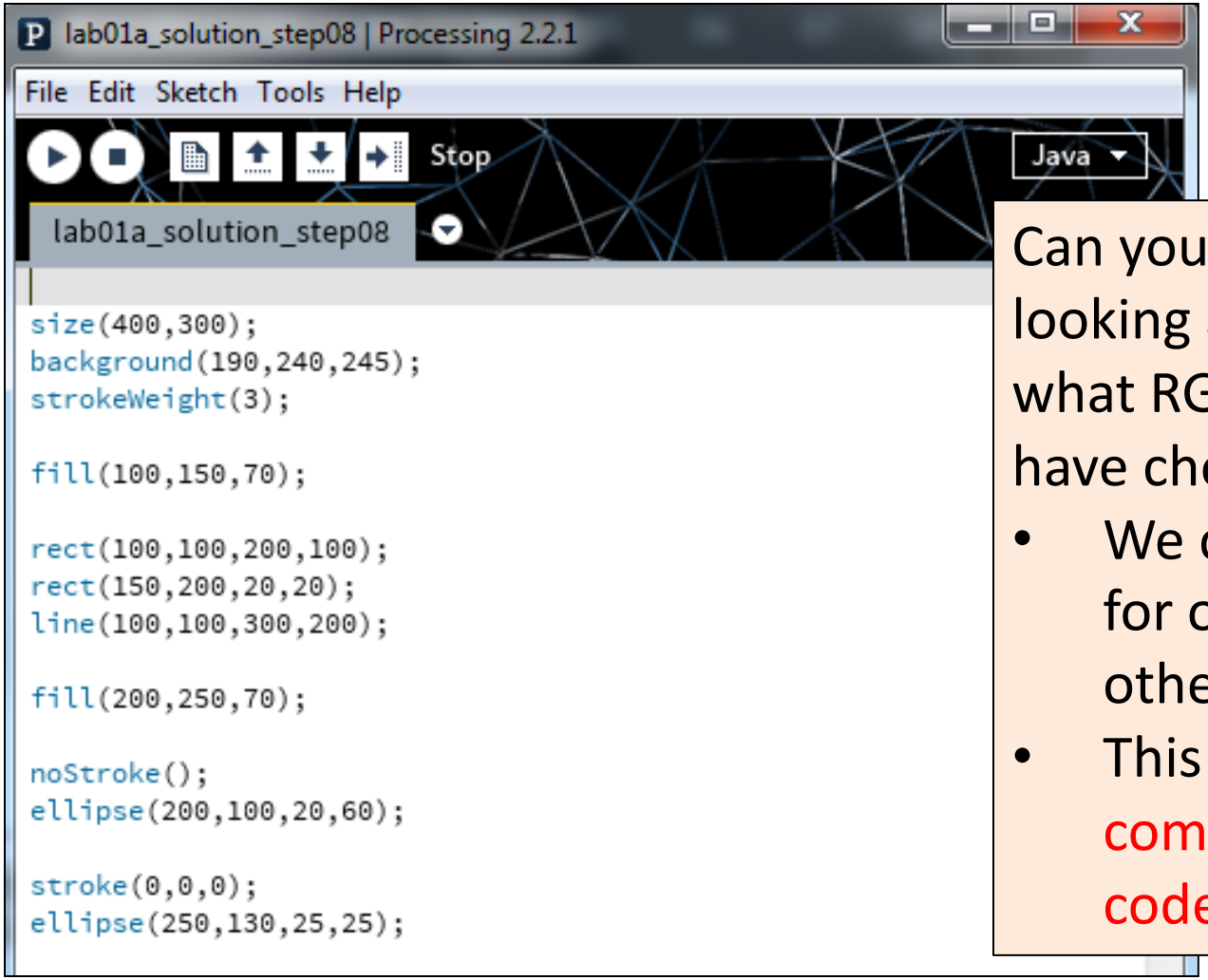


√ Our outline is heavier but noStroke on 2 shapes

Topics list

- Filling shapes with colour.
- Formatting the shape outline.
- Adding comments to your code.

Code so far…



Can you tell, from looking at the code, what RGB colours you have chosen?

- We can leave notes for ourselves and others in our code.
- This is called commenting your code.

Commenting your code…

```
//  This is a comment.
// Anything typed after the two slashes
// up to the end of the line, is ignored by Java.

/*  This is a longer comment.  As you can span
more than one line with this comment style, it
can be quite handy. */
```

Code so far…with commenting


```
//Setting up the display window and strokeWeight
size(400,300);
background(190,240,245);
strokeWeight(3);

//fill the rectangles with dark green
fill(100,150,70);

/* Drawing a rectangle, followed by a
   square and finally, a line */
rect(100,100,200,100);
rect(150,200,20,20);
line(100,100,300,200);

//fill both ellipses with light green
fill(200,250,70);

//Drawing an ellipse with no outline
noStroke();
ellipse(200,100,20,60);

//Drawing a circle with a black outline
stroke(0,0,0);
ellipse(250,130,25,25);
```

We have commented our code with explanations of what is happening.

This makes our code easier to read, understand and maintain.

It is considered best practice to comment your code.

Comments do not affect your code at all.

Questions?

Produced by:

Dr. Siobhán Drohan
Mairead Meagher
Sinéad Walsh

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/