

Conditional Events

Mouse events and Operators

Produced Dr. Siobhán Drohan
by: Mr. Colm Dunphy
 Mr. Diarmuid O'Connor



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Topics list

1. Mouse Events
2. Recap: Arithmetic Operators
3. Order of Evaluation

What is an event?

*“...an action such as
a key being pressed,
the mouse moving,
or a new piece of data
becoming available to read.”*

(Reas & Fry, 2014)

What happens when an event is “fired”?

*“An event **interrupts**
the normal flow
of a program
to run the code
within an **event block**”*

Mouse Events

Mouse Variables	Description
mousePressed	<p>true if any mouse button is pressed, false otherwise.</p> <p>Note: this variable reverts to false as soon as the button is released.</p>
mouseButton	<p>Can have the value LEFT, RIGHT and CENTER, depending on the mouse button most recently pressed.</p> <p>Note: this variable retains its value until a <u>different</u> mouse button is pressed.</p>

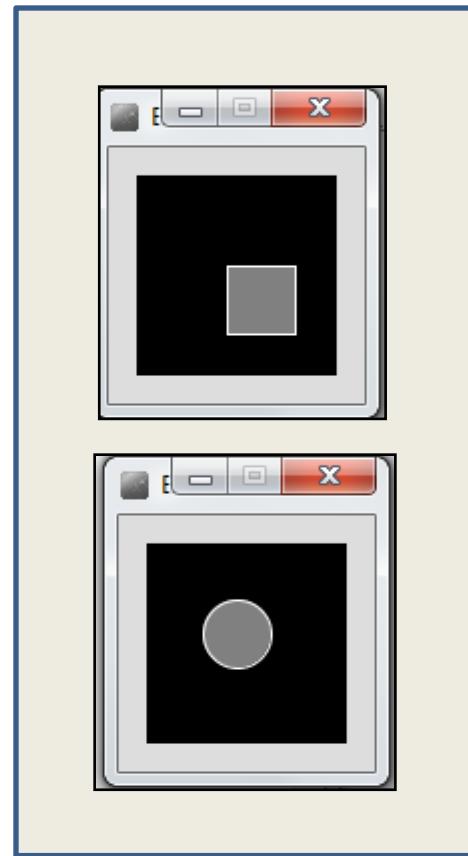
Mouse Events

- Mouse and keyboard events only work when a program has `draw()`.
- Without `draw()`, the code is only run once and then stops “listening” for events.

Processing Example 2.5

Functionality:

- If the mouse is pressed:
 - draw a grey square with a white outline.
 - otherwise draw a grey circle with a white outline.



Processing Example 2.5 - Code

The image shows the Processing IDE interface with the title bar "Example_2_5 | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. The sketch window titled "Example_2_5" contains the following Java code:

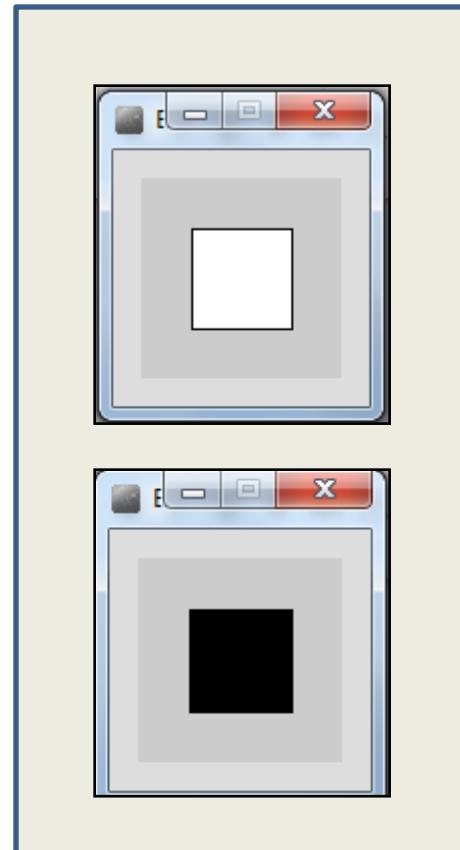
```
1 //Reas, C. & Fry, B. (2014) Processing - A First Look
2
3 void setup() {
4     size(100,100);
5 }
6
7 void draw() {
8     background(0);
9     stroke(255);
10    fill(128);
11    if (mousePressed){
12        rect(45,45,34,34);
13    } else{
14        ellipse(45,45,34,34);
15    }
16}
17}
```

A red rectangular box highlights the conditional statements from line 11 to line 16. Two red arrows point from this highlighted area to two separate windows on the right, each showing a black square canvas with a gray shape (either a rectangle or an ellipse) at its center.

Processing Example 2.6

Functionality:

- If the mouse is pressed:
 - set the fill to white and draw a square.
 - otherwise set the fill to black and draw a square.



Processing Example 2.6

The image shows the Processing IDE interface with the title bar "Example_2_6 | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. Below the menu is a toolbar with play, stop, and step buttons, and a Java dropdown. The sketch window titled "Example_2_6" contains the following code:

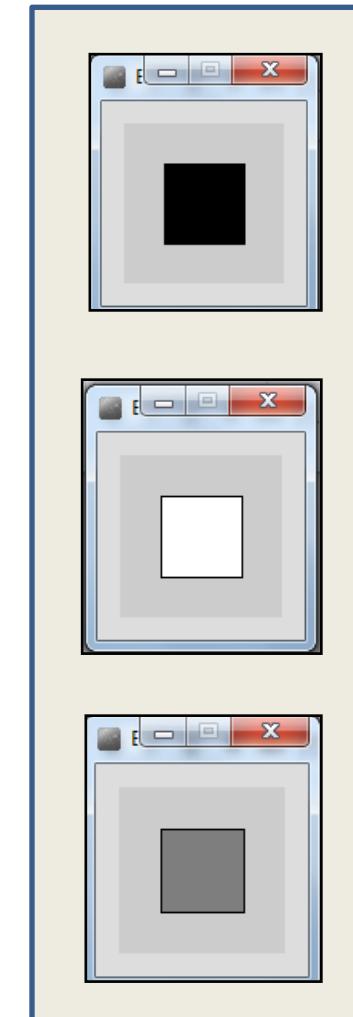
```
1 //Reas, C. & Fry, B. (2014) Processing - A F ▾
2
3 void setup() {
4     size(100, 100);
5 }
6
7 void draw() {
8     background(204);
9     if (mousePressed == true) {
10         fill(255); // White
11     } else {
12         fill(0); // Black
13     }
14     rect(25, 25, 50, 50);
15 }
```

Two output windows are shown on the right. The top window displays a white square on a gray background. The bottom window displays a black square on a gray background. Red arrows point from the corresponding code lines (fill(255) and fill(0)) to their respective outputs.

Processing Example 2.7

Functionality:

- If the **LEFT** button on the mouse is pressed, set the fill to **black** and draw a square. As soon as the LEFT button is **released**, **grey** fill the square.
- If the **RIGHT** button on the mouse is pressed, set the fill to **white** and draw a square. As soon as the **RIGHT** button is released, **grey** fill the square.
- If no mouse button is pressed, set the fill to **grey** and draw a square.



Processing Example 2.7

Nested if

The image shows the Processing IDE window titled "Example_2_7 | Processing 3.3.6". The code editor contains the following Java code:

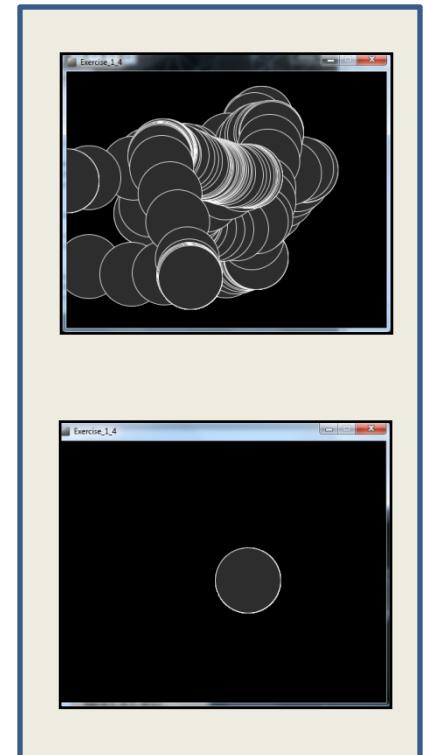
```
1 //Reas, C. & Fry, B. (2014) Processing - A First Look and How To Think Like a Computer Scientist
2
3 void setup() {
4     size(100, 100);
5 }
6
7 void draw() {
8     if (mousePressed){
9         if (mouseButton == LEFT)
10             fill(0);      // black
11         else if (mouseButton == RIGHT)
12             fill(255);   // white
13     }
14     else {
15         fill(126);    // gray
16     }
17     rect(25, 25, 50, 50);
18 }
```

A red box highlights the nested if statements from line 8 to line 16. Three red arrows point from this highlighted area to three separate preview windows on the right, each showing a different colored square (black, white, and gray) on a gray background.

Processing Example 2.8

Functionality:

- Draw a **circle** on the **mouse (x,y)** coordinates.
 - `mouseX, mouseY`
- Each time you **move** the mouse, draw a new circle.
 - `ellipse()` in `draw()`
- All the circles remain in the sketch until you press a mouse button.
- When you **press a mouse button**, the sketch is **cleared** and a single circle is drawn at the mouse (x,y) coordinates.
 - `background()` in `mousePressed()`



Processing Example 2.8



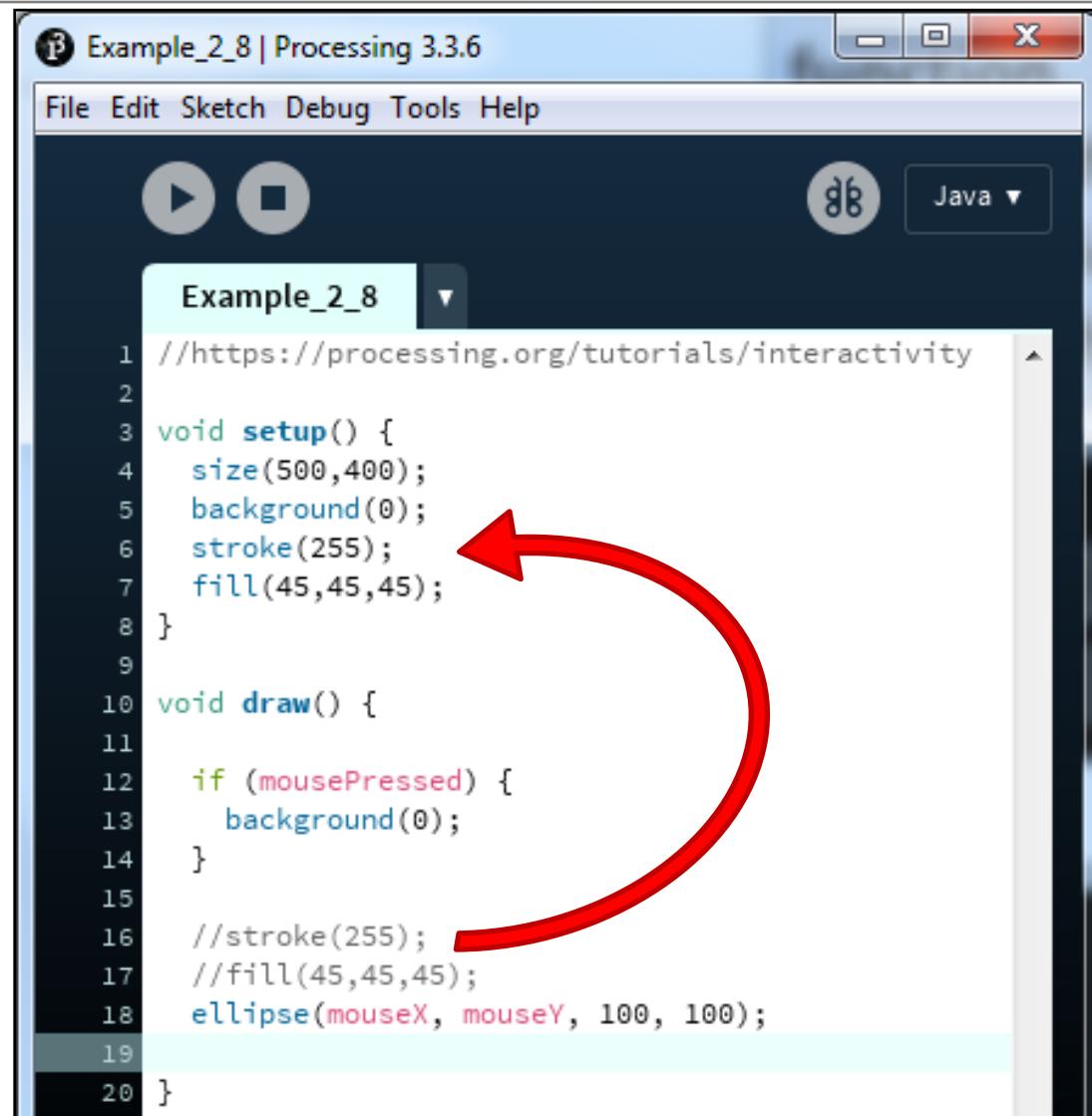
The screenshot shows the Processing 3.3.6 IDE interface. The title bar reads "Example_2_8 | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. Below the menu is a toolbar with play, stop, and other buttons. A circular icon with the number 88 is visible. The sketch window title is "Example_2_8". The code editor contains the following Java code:

```
1 //https://processing.org/tutorials/interactivity
2
3 void setup() {
4     size(500,400);
5     background(0);
6 }
7
8 void draw() {
9
10    if (mousePressed) {
11        background(0);
12    }
13
14    stroke(255);
15    fill(45,45,45);
16    ellipse(mouseX, mouseY, 100, 100);
17
18 }
```

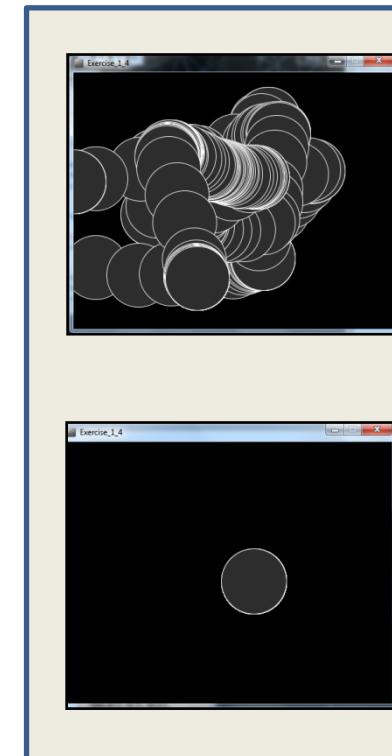
A red rectangular box highlights the code block from line 10 to line 13.



Processing Example 2.8



```
Example_2_8 | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java ▾
Example_2_8
1 //https://processing.org/tutorials/interactivity
2
3 void setup() {
4     size(500,400);
5     background(0);
6     stroke(255);
7     fill(45,45,45);
8 }
9
10 void draw() {
11
12     if (mousePressed) {
13         background(0);
14     }
15
16     //stroke(255);
17     //fill(45,45,45);
18     ellipse(mouseX, mouseY, 100, 100);
19
20 }
```



We moved the stroke and fill function calls to the setup() function.

Q: Does this change the functionality of our sketch?

A: No...
it just calls them once,
in setup();

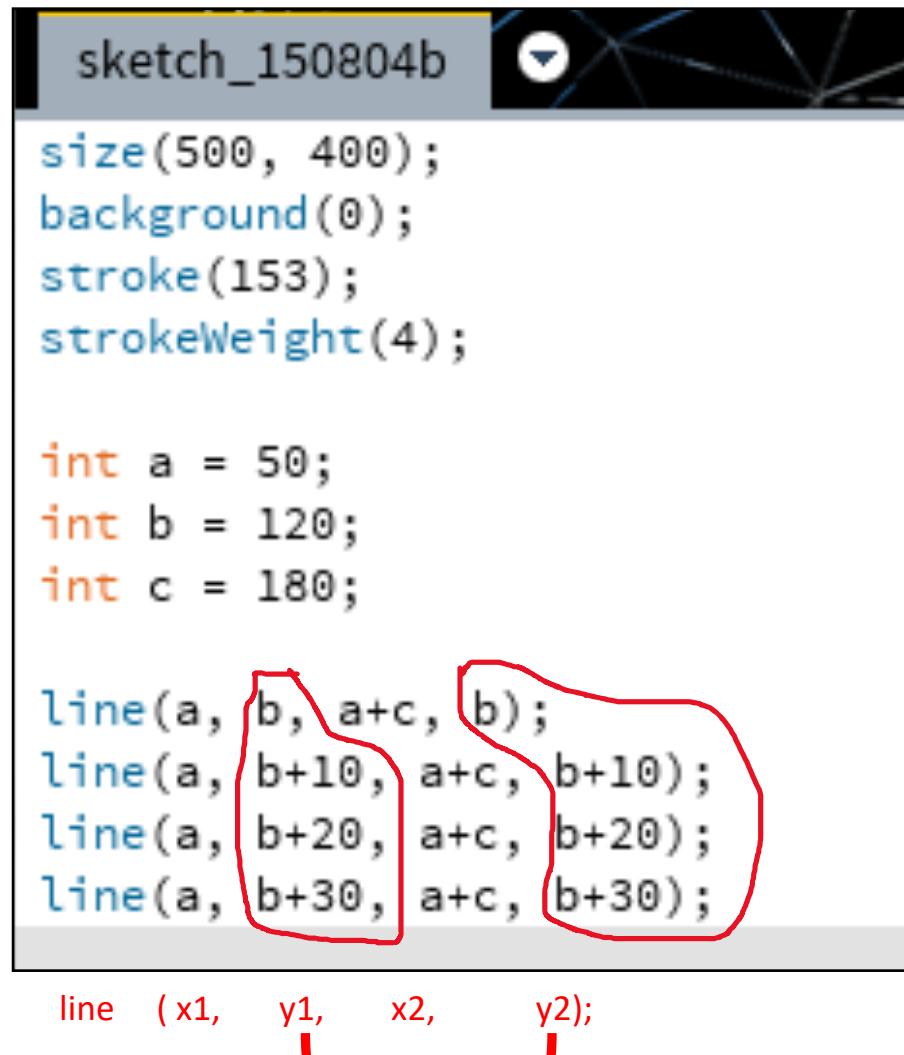
Topics list

1. Mouse Events
2. Recap: Arithmetic Operators
3. Order of Evaluation

Recap: Arithmetic Operators

Arithmetic Operator	Explanation	Example(s)
+	Addition	$6 + 2$ <code>amountOwed + 10</code>
-	Subtraction	$6 - 2$ <code>amountOwed - 10</code>
*	Multiplication	$6 * 2$ <code>amountOwed * 10</code>
/	Division	$6 / 2$ <code>amountOwed / 10</code>

Recap: Arithmetic operators

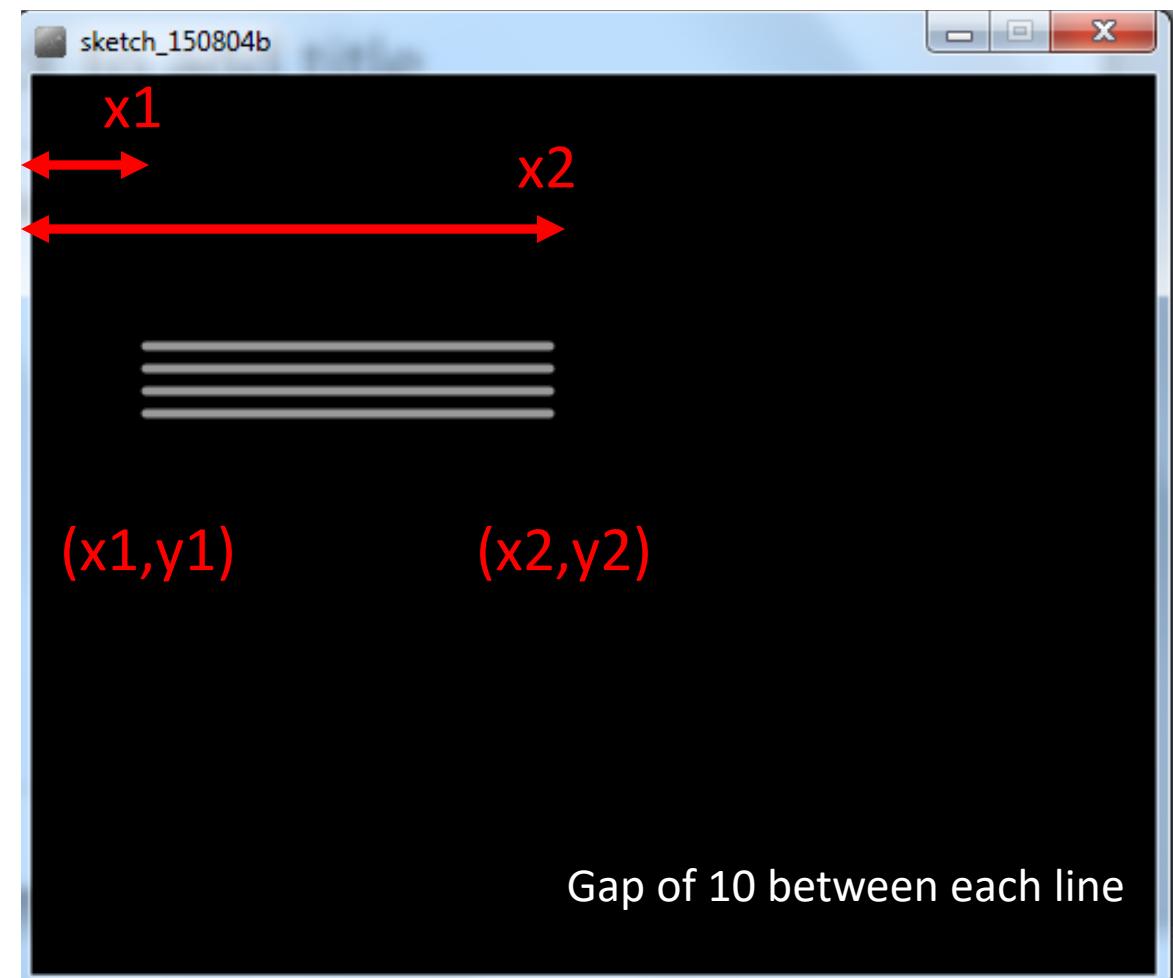


```
sketch_150804b
size(500, 400);
background(0);
stroke(153);
strokeWeight(4);

int a = 50;
int b = 120;
int c = 180;

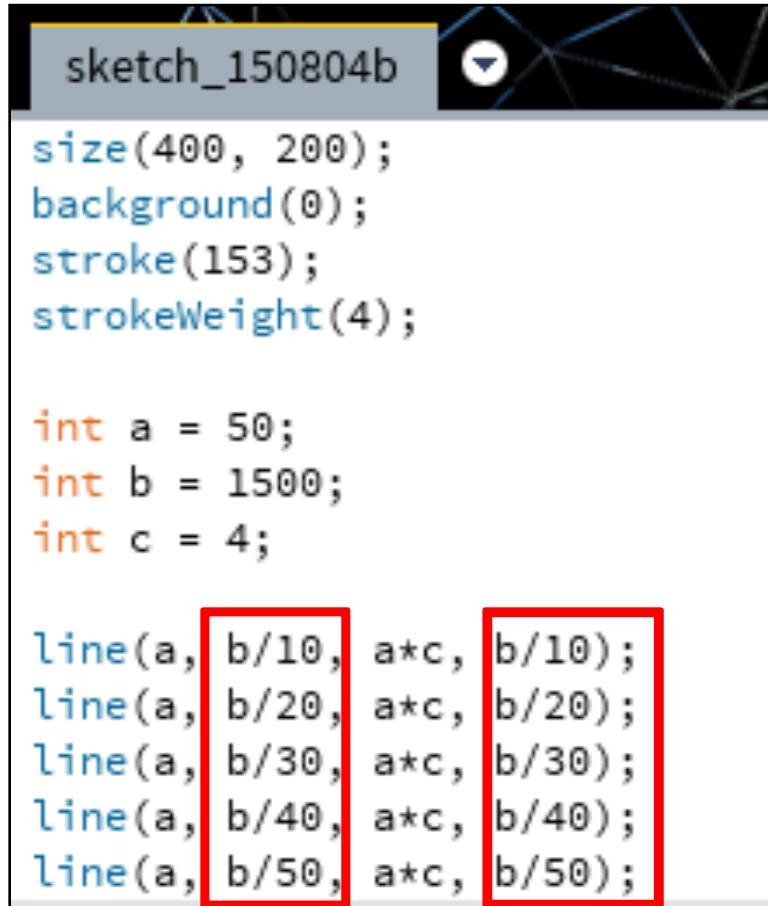
line(a, b, a+c, b);
line(a, b+10, a+c, b+10);
line(a, b+20, a+c, b+20);
line(a, b+30, a+c, b+30);
```

line (x1, y1, x2, y2);



y1=y2 => horizontal line. Equal gaps => parallel lines

Recap: Arithmetic operators

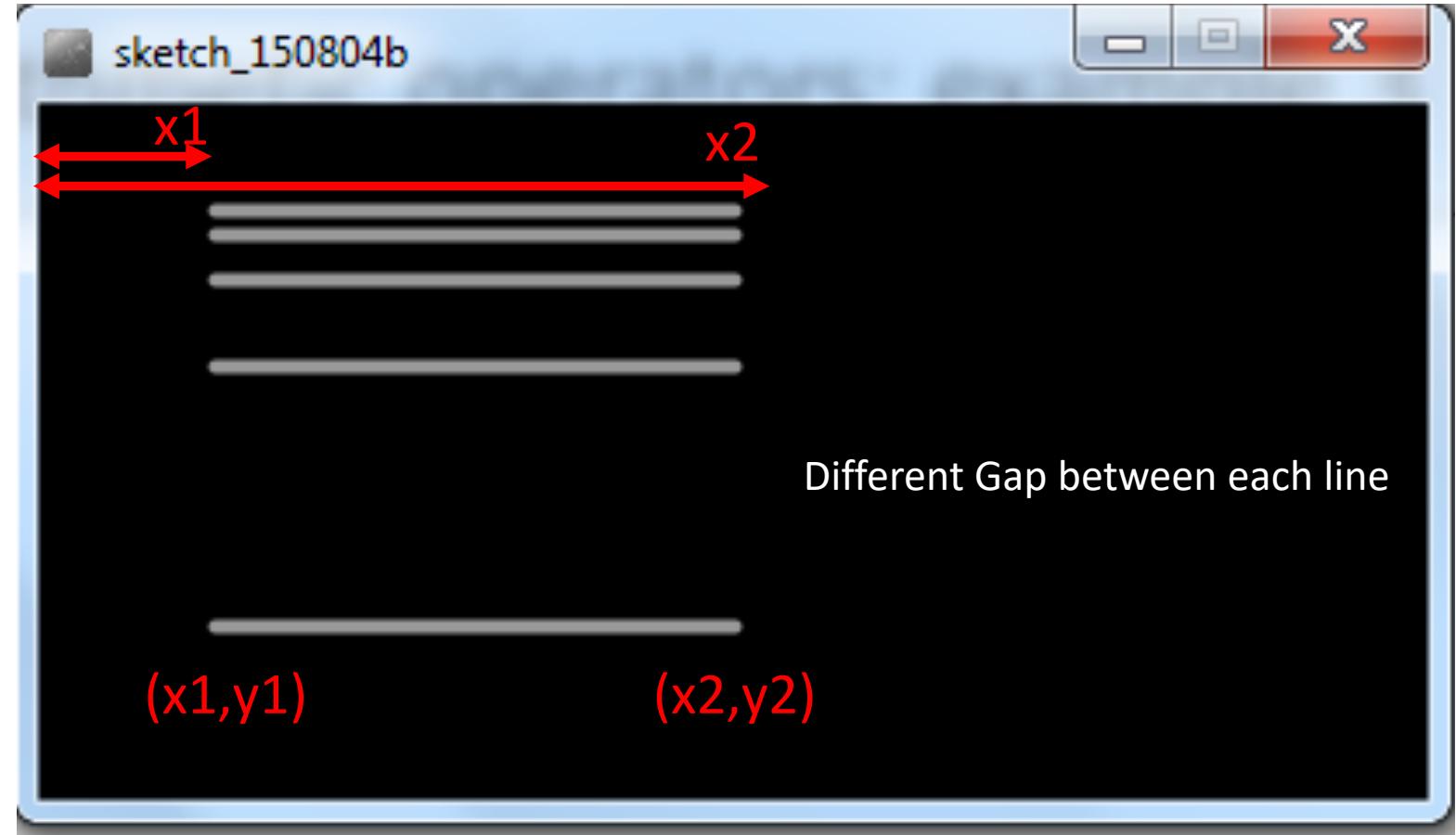


```
sketch_150804b
size(400, 200);
background(0);
stroke(153);
strokeWeight(4);

int a = 50;
int b = 1500;
int c = 4;

line(a, b/10, a*c, b/10);
line(a, b/20, a*c, b/20);
line(a, b/30, a*c, b/30);
line(a, b/40, a*c, b/40);
line(a, b/50, a*c, b/50);
```

line (x1, y1, x2, y2);



y1=y2 => horizontal line.

Arithmetic Operators

- If you want to keep track of how many times something happens, you are keeping a **running total** e.g.
 - The number of times you drew a line on the computer screen.
 - As each line is drawn, you add one to your counter variable.

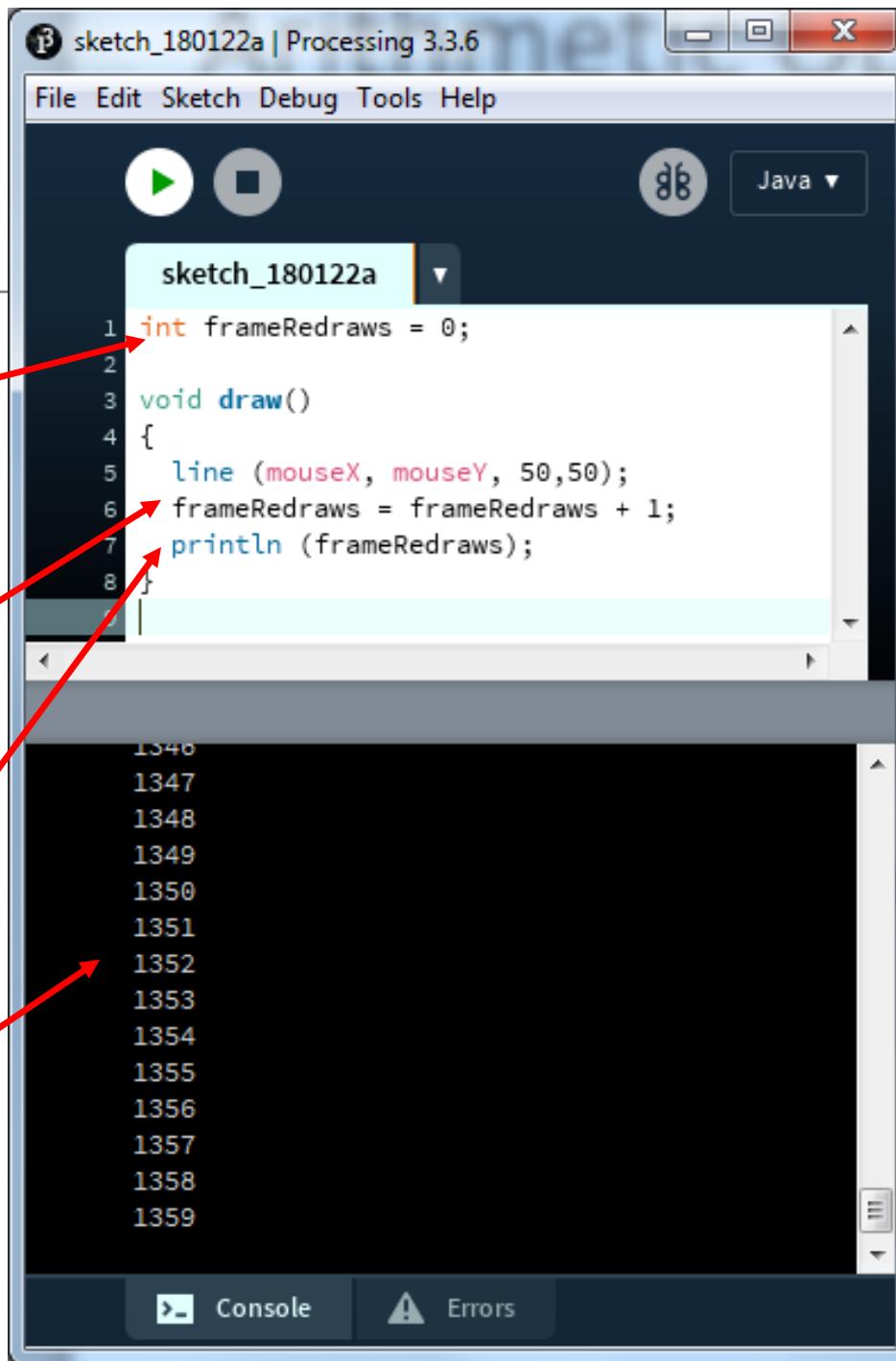
Arithmetic Operators

This code declares a new variable of type integer called `frameRedraws` and initialises it to 0.

One is added to the `frameRedraws` variable each time the `draw()` method is called.

The value of `frameRedraws` is then printed to the console.

`frameRedraws` is a “running total” of the number of frame redraws.



```
sketch_180122a | Processing 3.3.6
File Edit Sketch Debug Tools Help
sketch_180122a
1 int frameRedraws = 0;
2
3 void draw()
4 {
5   line (mouseX, mouseY, 50,50);
6   frameRedraws = frameRedraws + 1;
7   println (frameRedraws);
8 }

1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
```

Console Errors

Arithmetic Operators

- These examples are straightforward uses of the arithmetic operators.
- However, we typically want to do more complex calculations involving many arithmetic operators.
- To do this, we need to understand the **Order of Evaluation**.

Topics list

1. Mouse Events
2. Recap: Arithmetic Operators
3. Order of Evaluation

Order of Evaluation

- Brackets ()
- Multiplication (*)
- Division (/)
- Addition (+)
- Subtraction (-)

BoMDAS

Beware My Dear Aunt Sally

Order of Evaluation - Quiz

What are the results of these calculations?

- Q1: $3+6*5-2$
- Q2: $3+6*(5-2)$
- Q3: $(3+6)*5-2$

Questions?



References

- Reas, C. & Fry, B. (2014) Processing – A Programming Handbook for Visual Designers and Artists, 2nd Edition, MIT Press, London.