Conditional Events

Mouse events and Operators

Produced by:
Dr. Siobhán Drohan
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

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”

(Reas & Fry, 2014)

Mouse Events

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

Mouse Events

- Mouse and keyboard events only work when a program has draw().

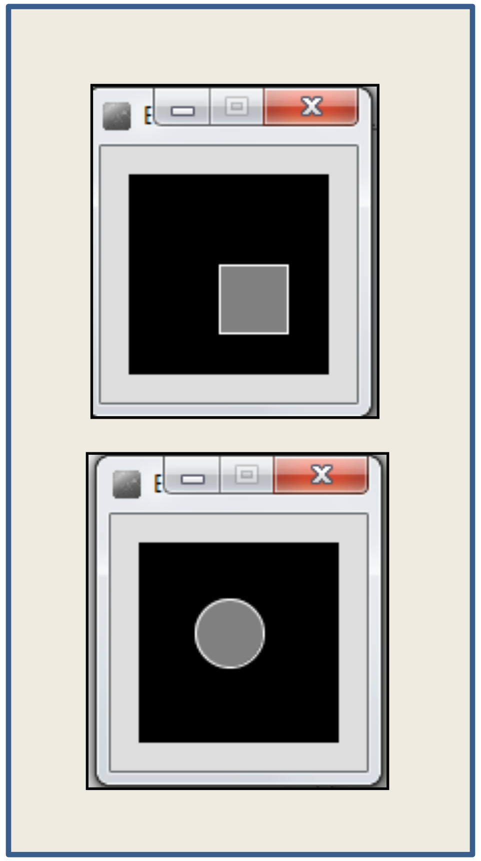
- Without draw(), the code is only run once and then stops “listening” for events.

Source: https://processing.org/reference/

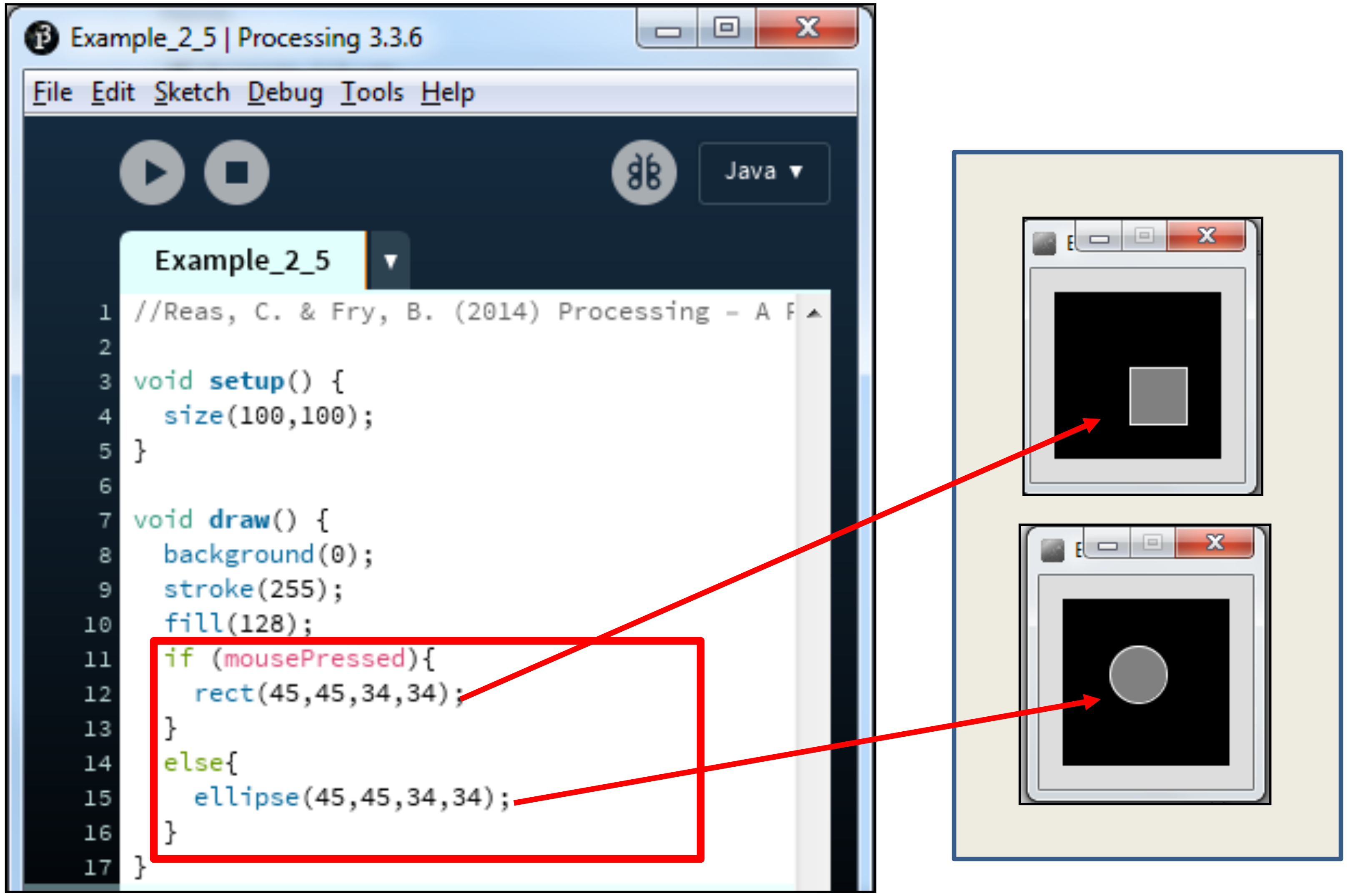
Processing Example 2.5

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

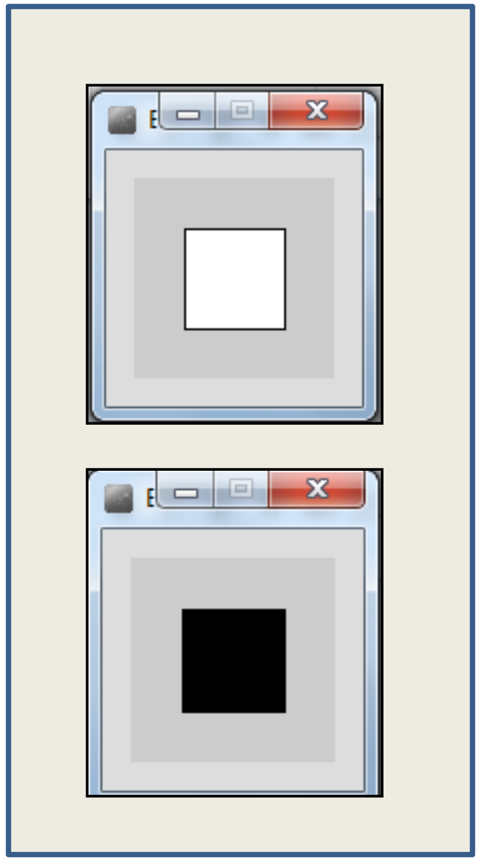

Example_2_5 | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
Example_2_5

```
//Reas, C. & Fry, B. (2014) Processing – A F

void setup() {
  size(100,100);
}

void draw() {
  background(0);
  stroke(255);
  fill(128);
  if (mousePressed){
    rect(45,45,34,34);
  }
  else{
    ellipse(45,45,34,34);
  }
}
```

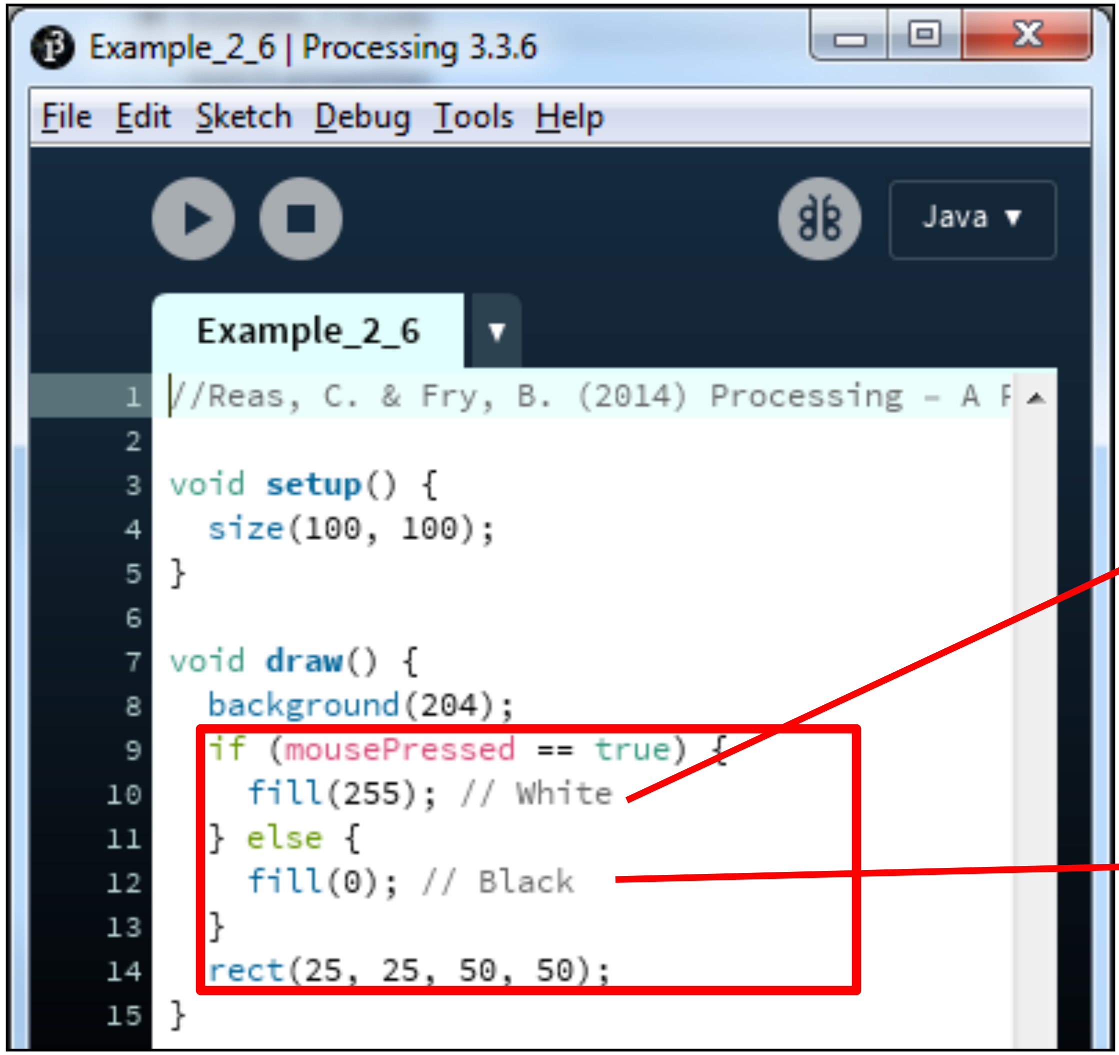
Processing Example 2.6

Functionality:

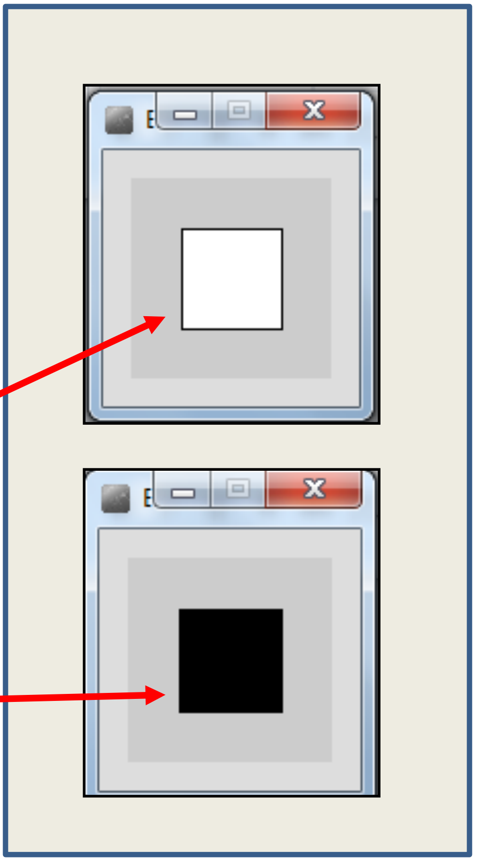
- If the mouse is pressed:
  - set the fill to white and draw a square.
  - otherwise set the fill to black and draw a square.



Processing Example 2.6


Example_2_6 | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
Example_2_6

```
//Reas, C. & Fry, B. (2014) Processing – A F

void setup() {
  size(100, 100);
}

void draw() {
  background(204);
  if (mousePressed == true) {
    fill(255); // White
  } else {
    fill(0); // Black
  }
  rect(25, 25, 50, 50);
}
```



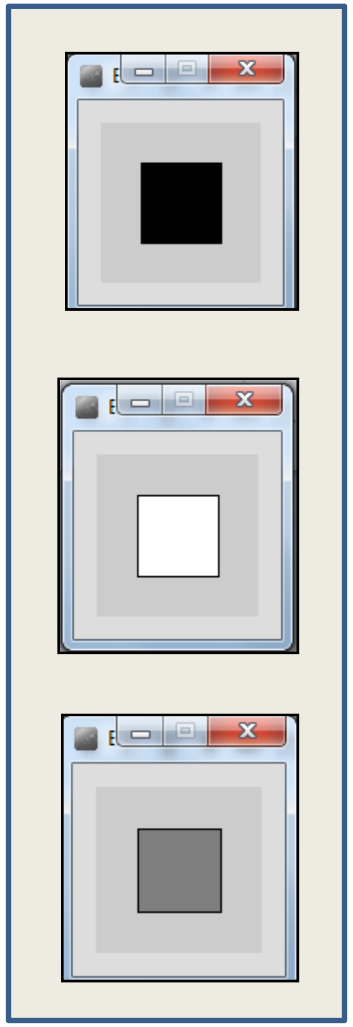
Processing Example 2.7

Functionality:

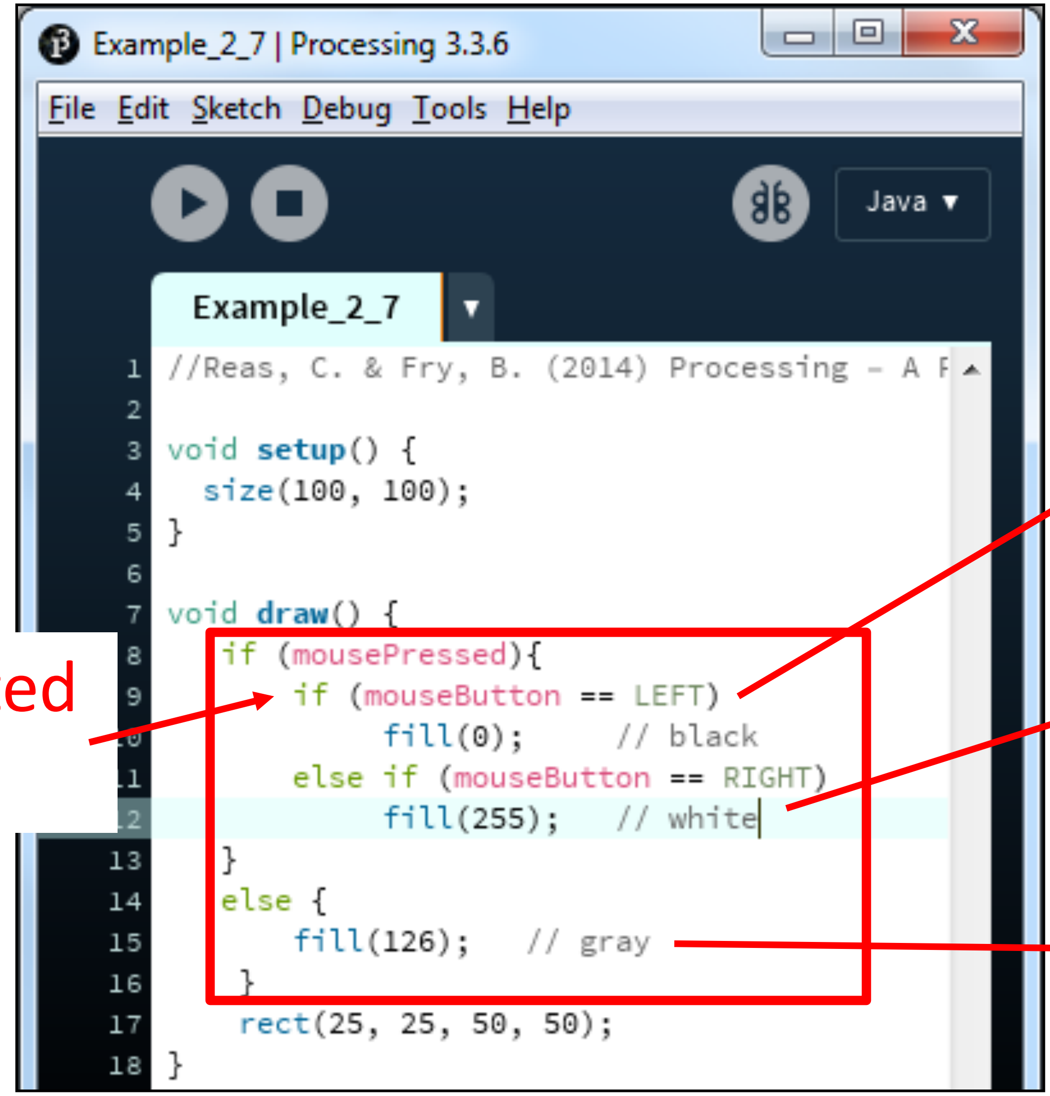
- If the **LEFT** button on the mouse is pressed, set the fill to **black** and draw a square. As soon as the LEFT button is **released**, **grey** fill the square.

- 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.

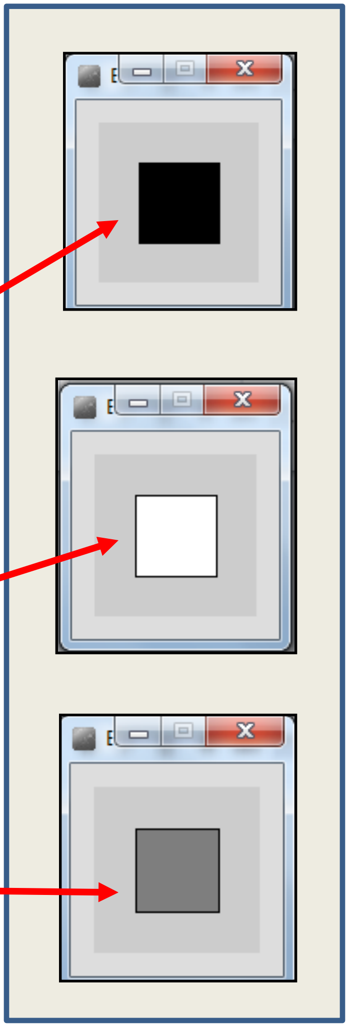


Processing Example 2.7


Example_2_7 | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
Example_2_7

```
//Reas, C. & Fry, B. (2014) Processing – A P

void setup() {
  size(100, 100);
}

void draw() {
    if (mousePressed){
      if (mouseButton == LEFT)
            fill(0);      // black
      else if (mouseButton == RIGHT)
            fill(255);    // white
    }
    else {
      fill(126);    // gray
     }
    rect(25, 25, 50, 50);
}
```

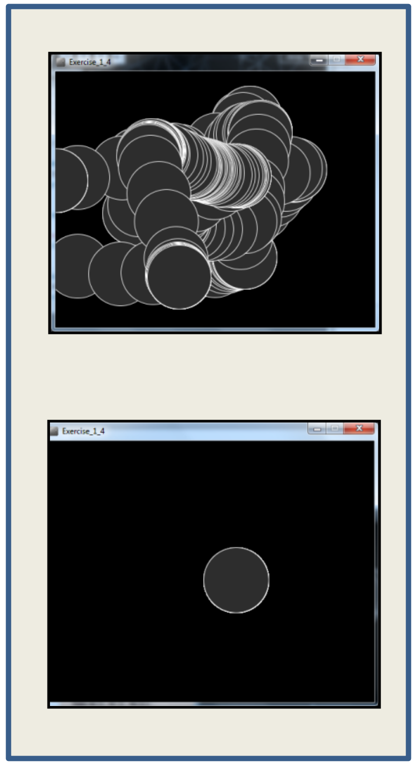
Nested if



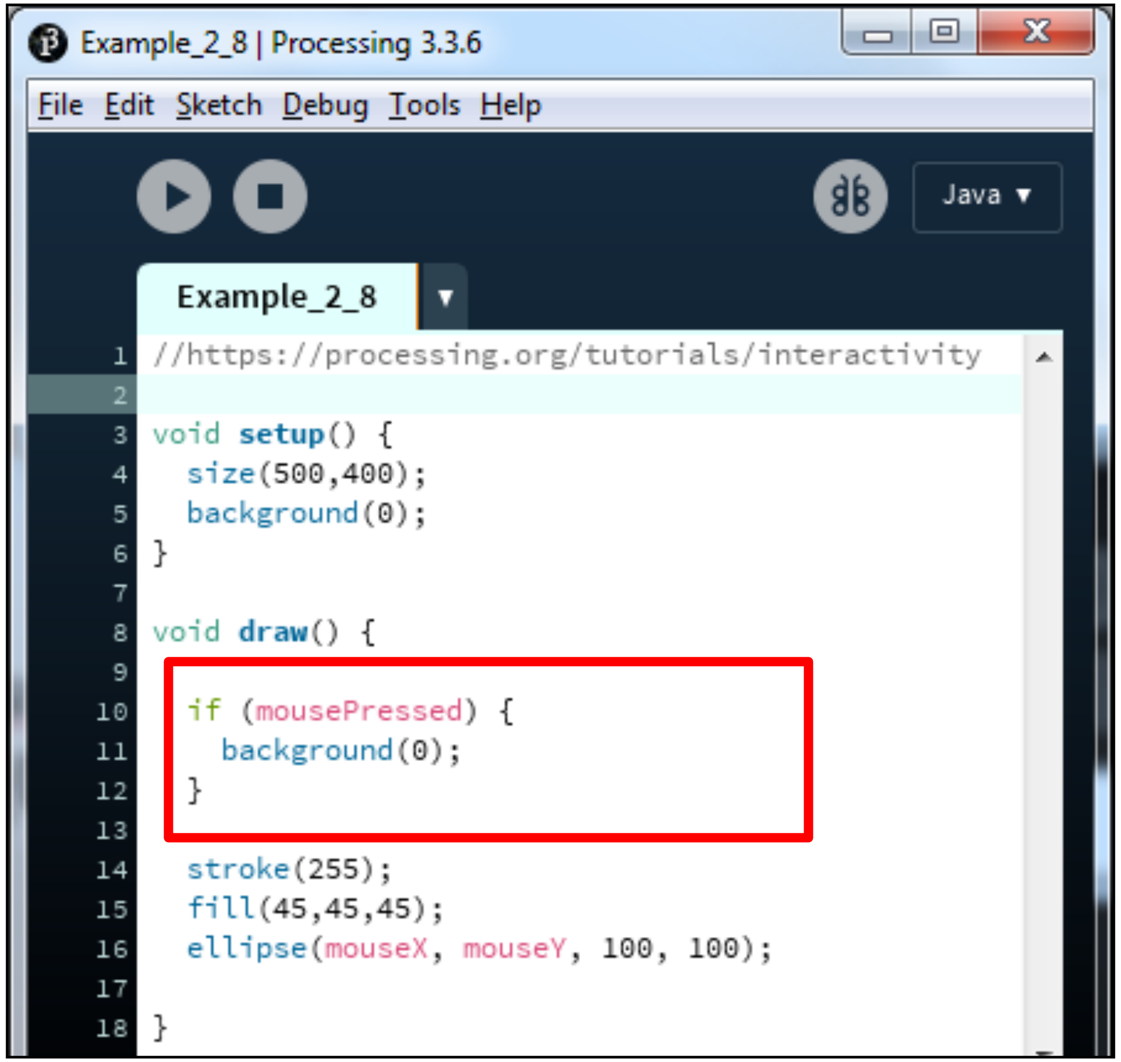
Processing Example 2.8

Functionality:

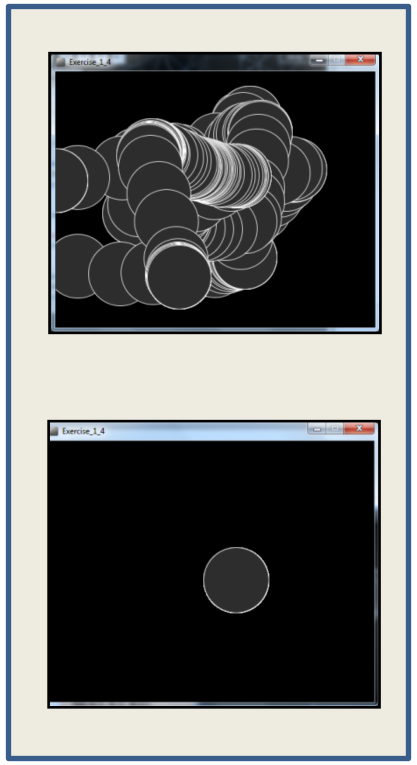
- Draw a **circle** on the **mouse (x,y)** coordinates.
  - mouseX, mouseY
- Each time you **move** the mouse, draw a new circle.
  - ellipse() in draw()
- All the circles remain in the sketch until you press a mouse button.
- When you **press a mouse button**, the sketch is **cleared** and a single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



Processing Example 2.8

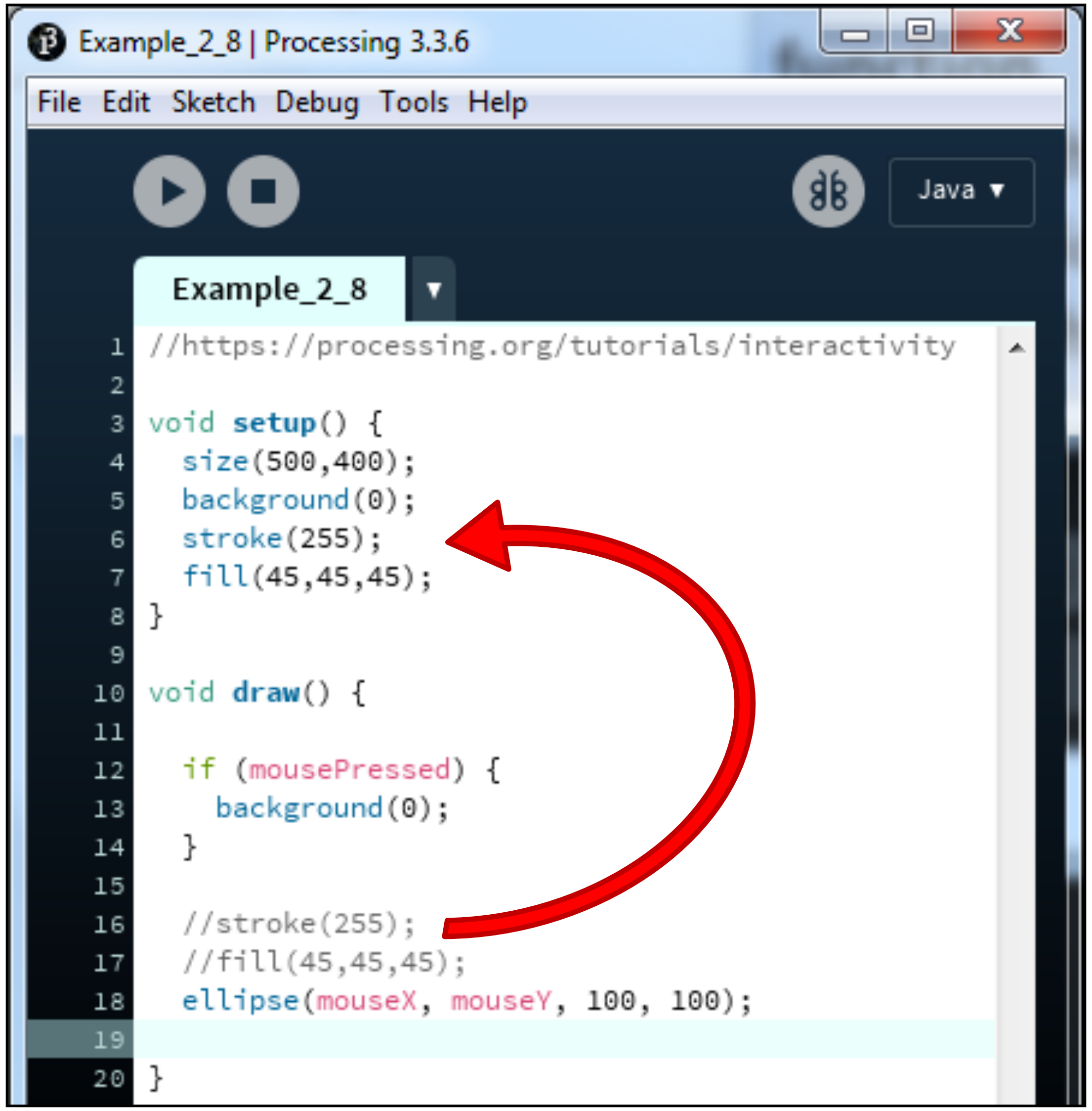

Example_2_8 | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
Example_2_8

```
//https://processing.org/tutorials/interactivity

void setup() {
  size(500,400);
  background(0);
}

void draw() {

  if (mousePressed) {
    background(0);
  }

  stroke(255);
  fill(45,45,45);
  ellipse(mouseX, mouseY, 100, 100);

}
```

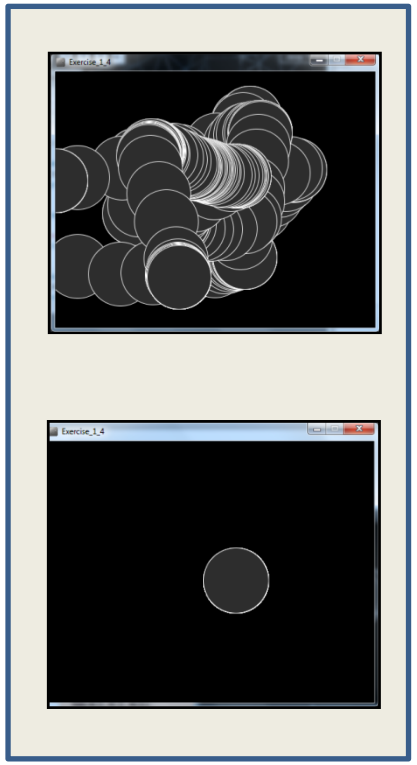


https://processing.org/tutorials/interactivity/

Processing Example 2.8


Example_2_8 | Processing 3.3.6
File Edit Sketch Debug Tools Help
Java
Example_2_8

```
//https://processing.org/tutorials/interactivity

void setup() {
  size(500,400);
  background(0);
  stroke(255);
  fill(45,45,45);
}

void draw() {

  if (mousePressed) {
    background(0);
  }

  //stroke(255);
  //fill(45,45,45);
  ellipse(mouseX, mouseY, 100, 100);

}
```



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();**

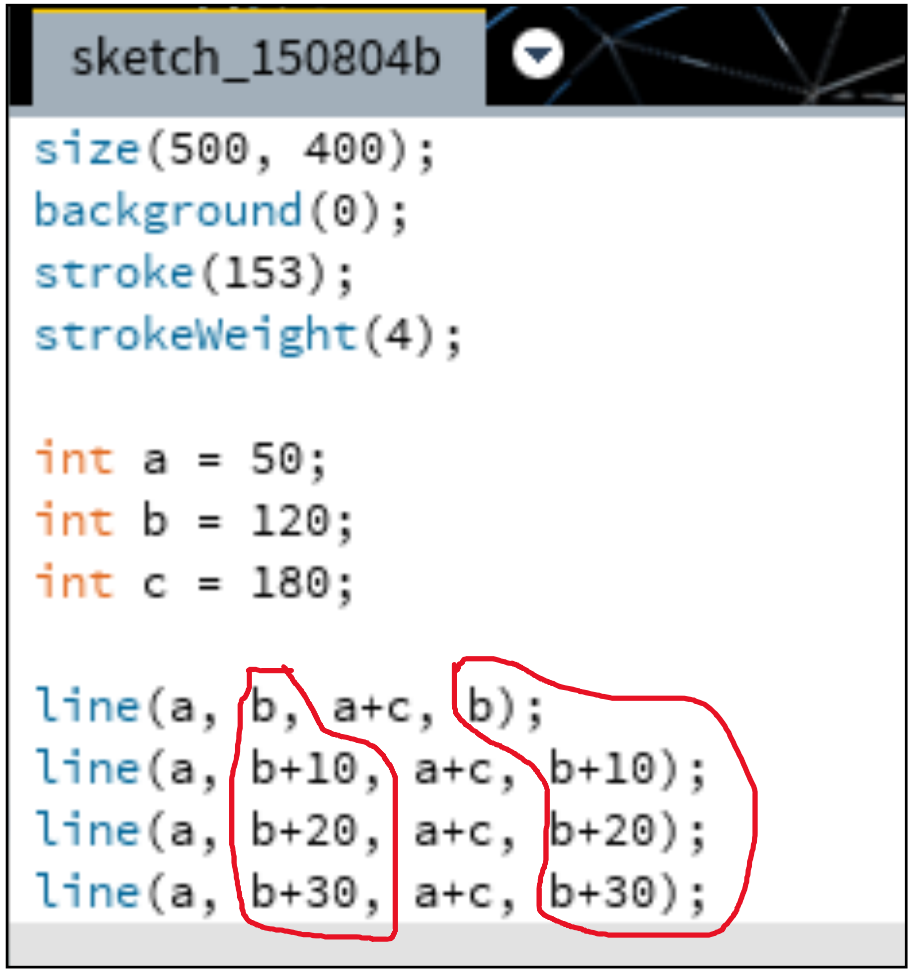
https://processing.org/tutorials/interactivity/

Topics list

Recap: Arithmetic Operators

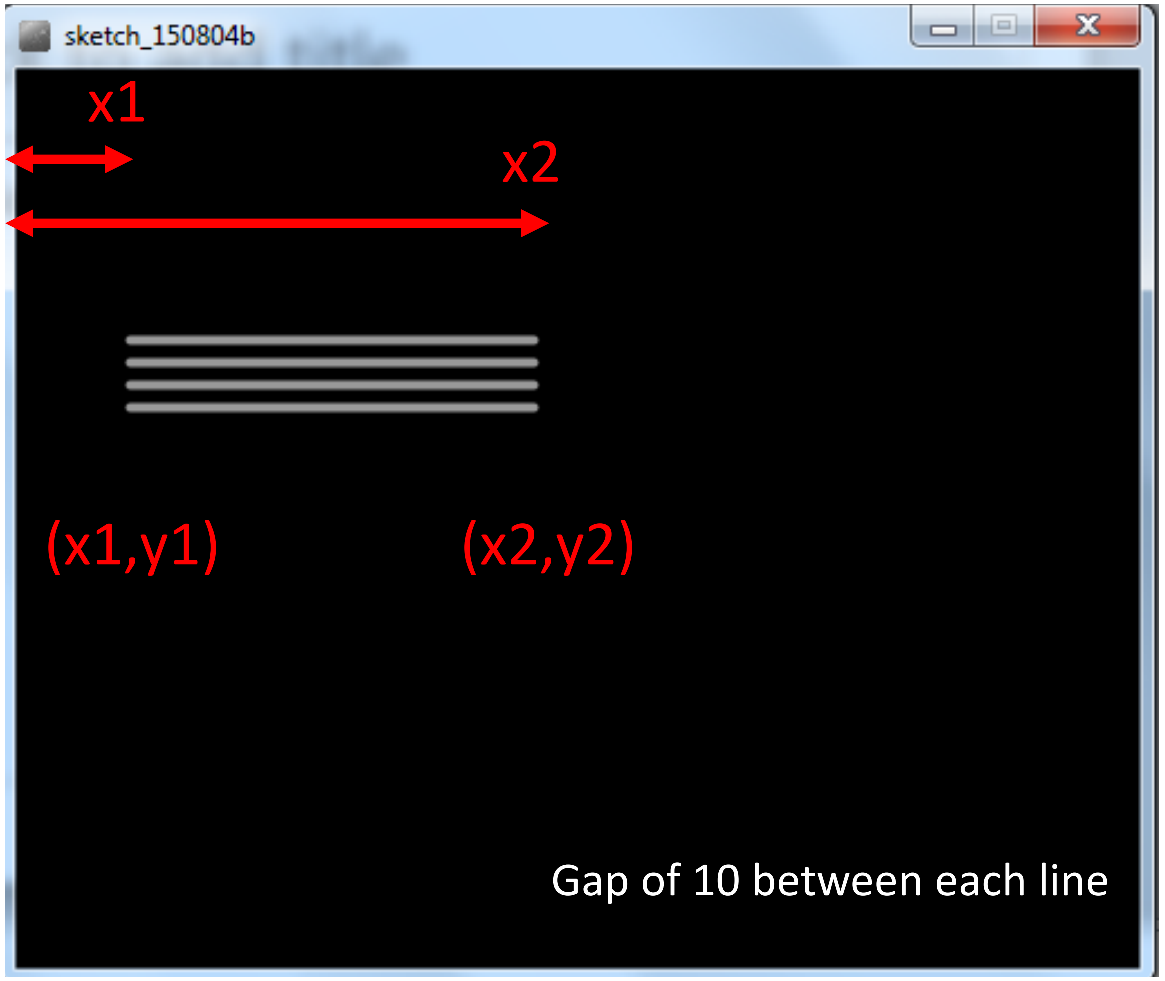
Arithmetic Operator	Explanation	Example(s)
+	Addition	6 + 2 amountOwed + 10
-	Subtraction	6 – 2 amountOwed – 10
*	Multiplication	6 * 2 amountOwed * 10
/	Division	6 / 2 amountOwed / 10

Recap: Arithmetic operators



```
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);
```

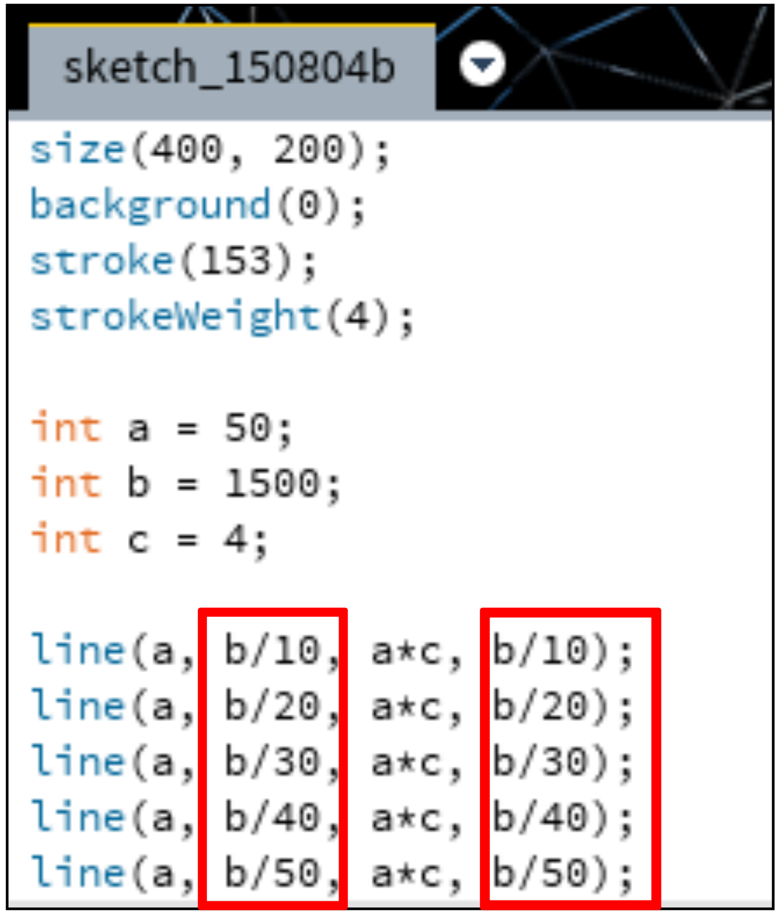
line (x1, $\underbrace{\text{y1, x2, y2}}$);



Gap of 10 between each line

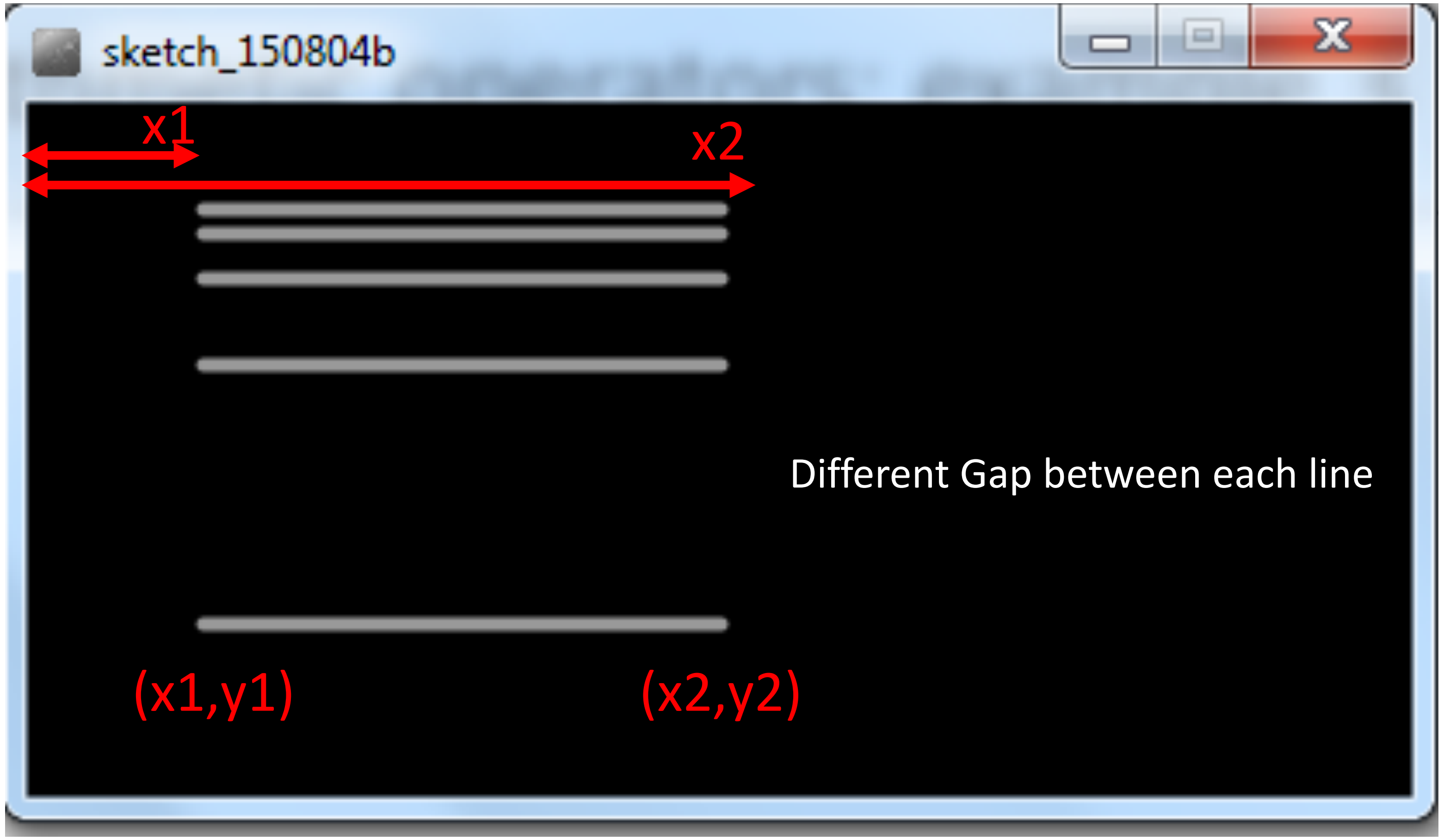
y1=y2 => horizontal line. Equal gaps => parallel lines

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.

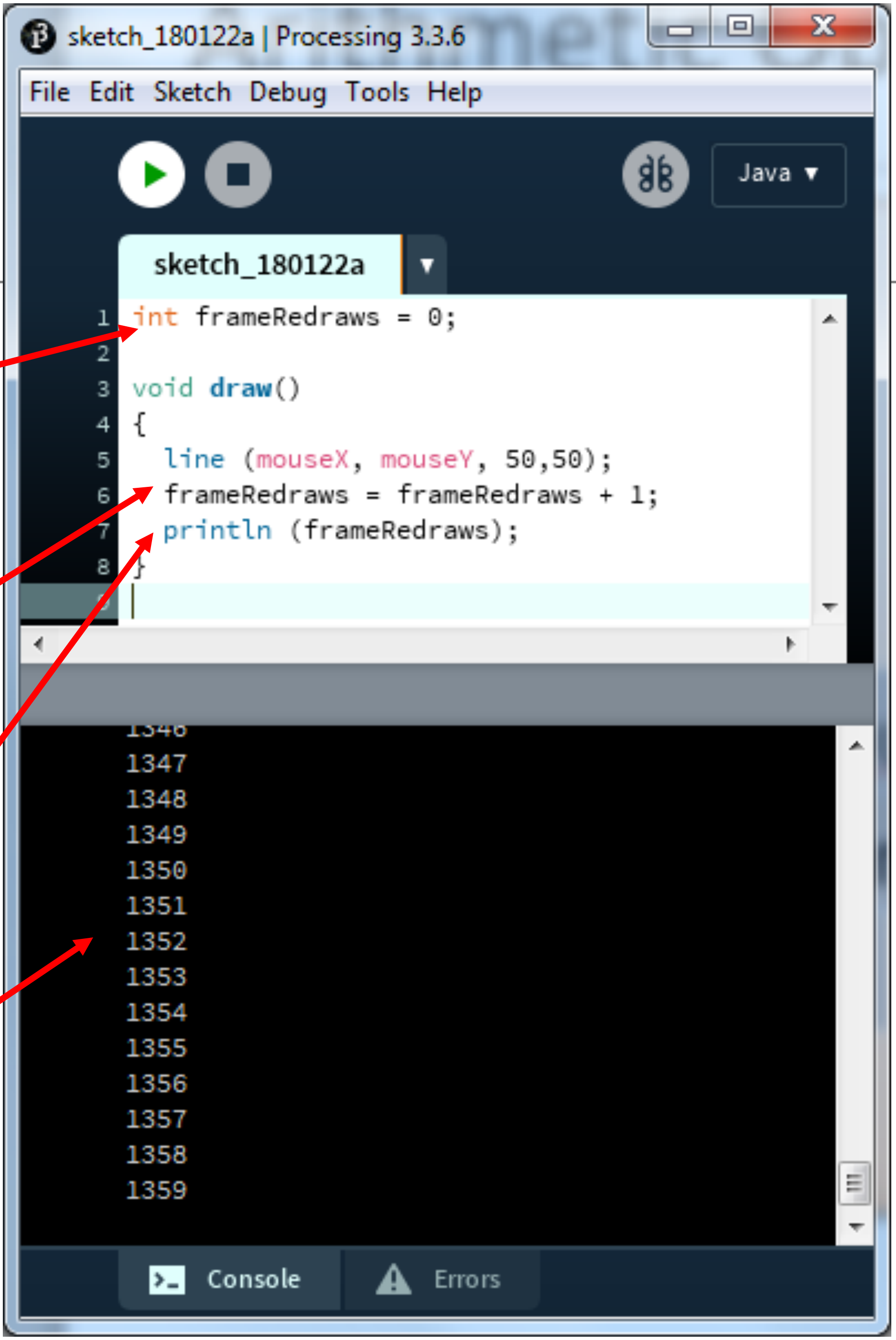
Arithmetic Operators

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
Java
sketch_180122a
Console
Errors

```
int frameRedraws = 0;

void draw()
{
  line (mouseX, mouseY, 50,50);
  frameRedraws = frameRedraws + 1;
  println (frameRedraws);
}
```

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

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

- **B**rackets **()**
- **M**ultiplication (*****)
- **D**ivision (**/**)
- **A**ddition (**+**)
- **S**ubtraction (**-**)

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.