Using Methods

Methods that handle events

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/

Caveat

- The term function is used in Processing e.g. line(), fill(), etc.
- The term method is used in Java.
- As this course is primarily about learning the Java language, we are planning on using the word method instead of function from here on in.

Topics list

1. **Method** terminology:
   - Return type
   - Method names
   - Parameter list
2. Using methods to handle **mouse events**.

Recap: Methods in Processing

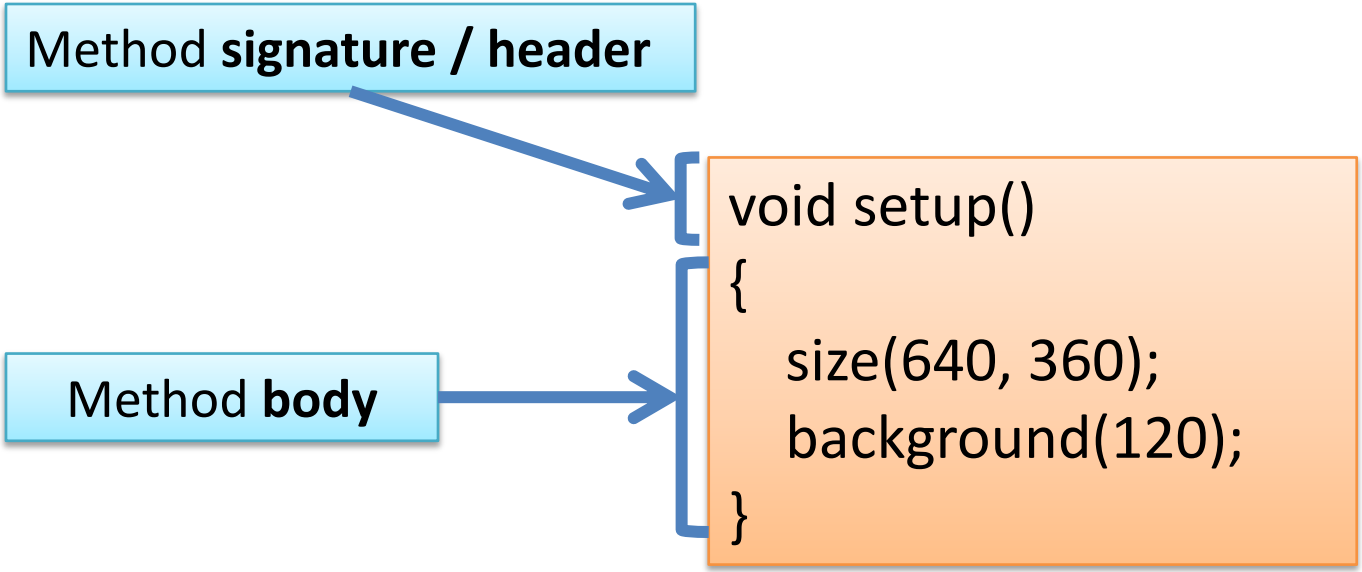
- Processing comes with several **pre-written methods** that we can use.
- A method comprises a **set of instructions** that performs some task.
- When we invoke the method, it performs the task.
- Some methods we have used are:

  rect(), ellipse(), line(), stroke(), fill(), background(), random(), etc.

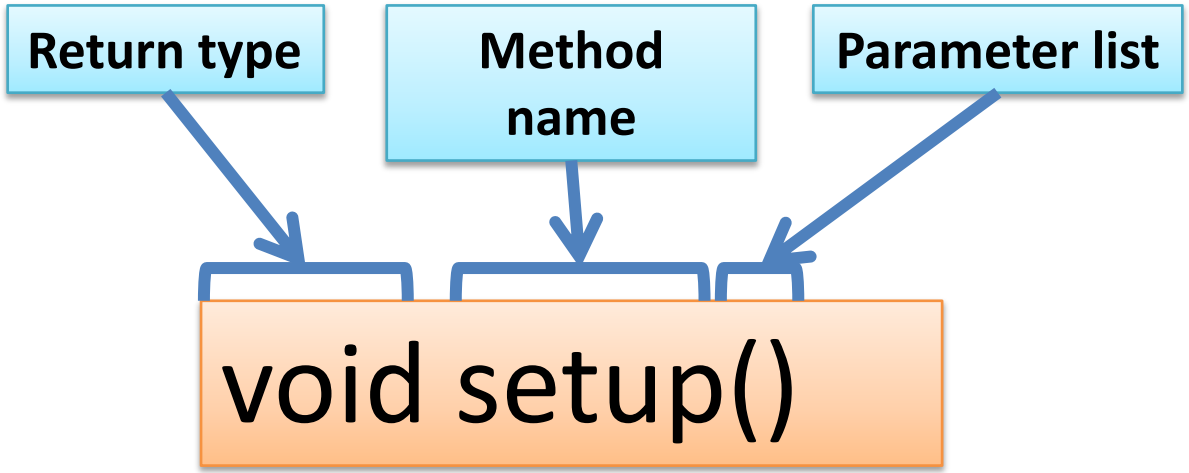
Recap: Methods in Processing

- We have also **written** two methods to animate our drawings:
  - void **setup**()
    - automatically called once when the program starts and should not be called again.
    - It typically sets up your display window e.g. screen size, background colour.
  - void **draw**()
    - automatically called straight after the setup() call.
    - It continuously executes the code contained inside it.

Method terminology



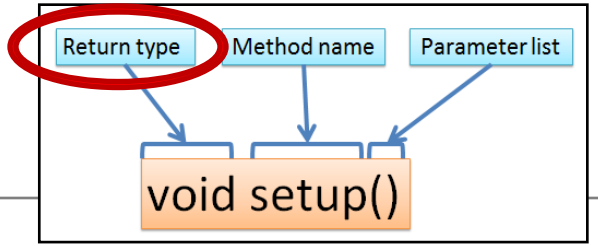
Method signature



Topics list

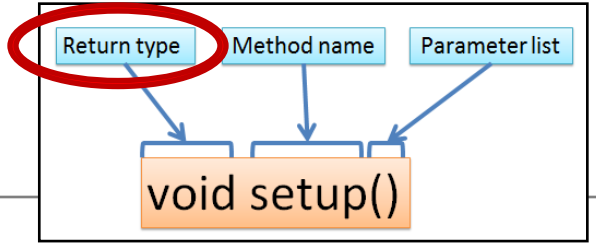
1. Method terminology:
   - Return type
   - Method names
   - Parameter list
2. Using methods to handle mouse events.

Return Type: **void**



- Methods can return information.
- The void keyword just before the method name means that **nothing is returned** from the method.
- void is a return type and must be included in the method signature, if your method returns no information.

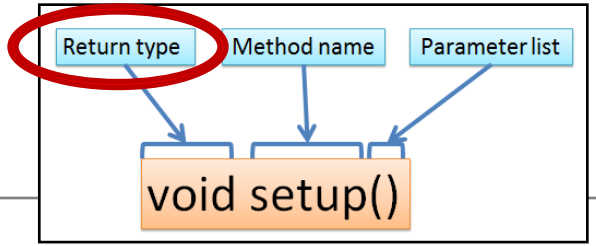
Return Type: **int**



- When a data type (e.g. int) appears before the method name, this means that **something is returned** from the method.

- Within the body of the method,
  you use the **return** statement to **return the value**.

Return Type: **int**



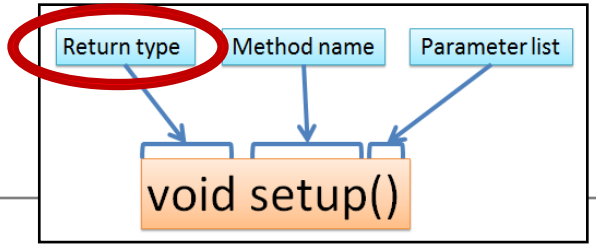

```
int val = 30;


void draw()
{
    int result = timestwo(val);
    println(result);
}
```

```
int timestwo(int number)
{
    number = number * 2;
    return number;
}
```

// The red **int** in the function declaration
// specifies the type of data to be returned.

https://processing.org/reference/return.html

Return Types

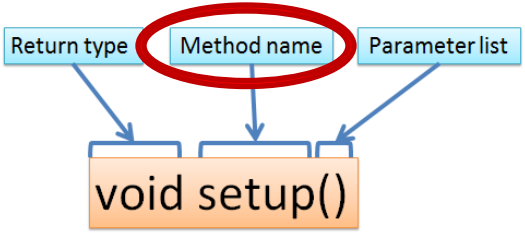


- Methods can return any **type of data** e.g.
  - boolean
  - byte
  - char
  - int
  - float
  - String
  - etc.
- You can only have **one return type per method**.

Topics list

1. Method terminology:
   - Return type
   - Method names
   - Parameter list
2. Using methods to handle mouse events.

Method name

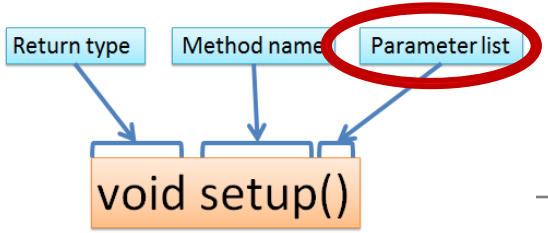


- Method names should:
  - Use **verbs** (i.e. actions) to describe what the method does e.g.
    - calculateTax
    - printResults
  - Be **mixed case** with the first letter lowercase and the first letter of each internal word capitalised. i.e. **camelCase**

Topics list

1. Method terminology:
   - Return type
   - Method names
   - Parameter list
2. Using methods to handle mouse events.

Parameter list



- Methods **take in data** via their **parameters**.

- Methods do not have to pass parameters
  e.g. setup() has **no parameters**.

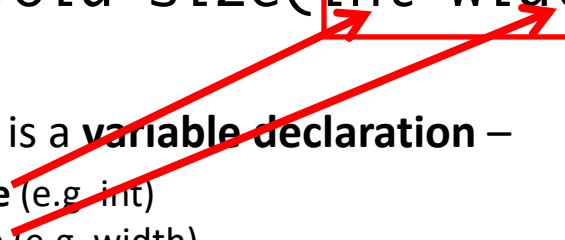
Methods **with NO parameters**

```
void noStroke()
void setup()
void noCursor()
```

- Methods do not have to pass parameters.
- These methods have **no parameters**;
  note how no variable is passed in the parenthesis i.e. ().
- These methods don’t need any additional information to do its tasks.

Methods **with Parameters**

```
void strokeWeight(float weight)

void size(int width, int height)
```



- A **parameter** is a **variable declaration** –
  - it has a **type** (e.g. int)
    and a **name** (e.g. width).
- If a method needs additional information to execute,
  we provide a parameter, so that the information can be passed into it.
- The first method, *strokeWeight*, above has **one parameter**.
- A second method *size* can have any number of parameters

  e.g. the second method, has two

Topics list

1. **Method** terminology:
   - Return type
   - Method names
   - Parameter list

2. Using methods to handle **mouse events**.

Mouse actions and their methods

Action	Description	Method
Clicked	Mouse button is pressed and then released	mouseClicked()
Pressed	Mouse button is pressed and held down	mousePressed()
Released	Mouse button was pressed, but now released	mouseReleased()
Moved	Mouse is moved without any buttons being pressed	mouseMoved()
Dragged	Mouse is moved with a button pressed	mouseDragged()

Mouse methods

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

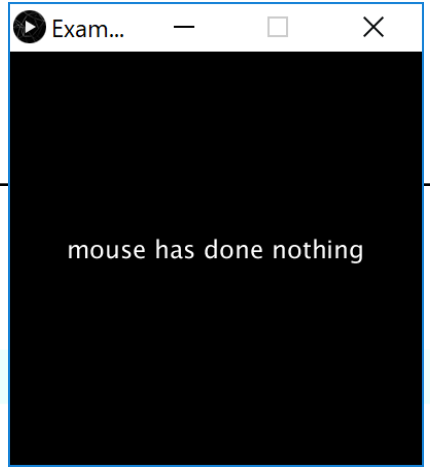
- Without draw(), the code is only run once and then stops listening for events.

https://processing.org/reference/mousePressed_.html

EXAMPLE 2.5

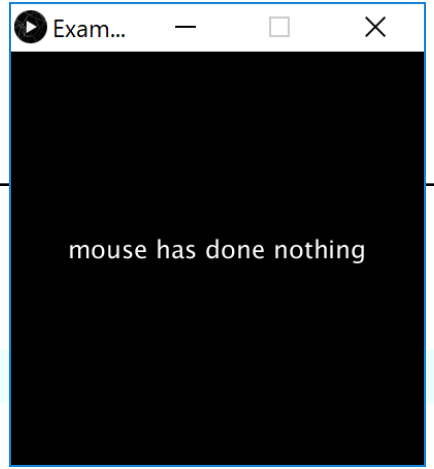
Processing Example 3.1 – setup()

```
void setup() {
  size(400, 400);
  background(0);
  textAlign(CENTER);
  textSize(24);
  fill(255);
  text("mouse has done nothing", width/2, height/2);
}
```



Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

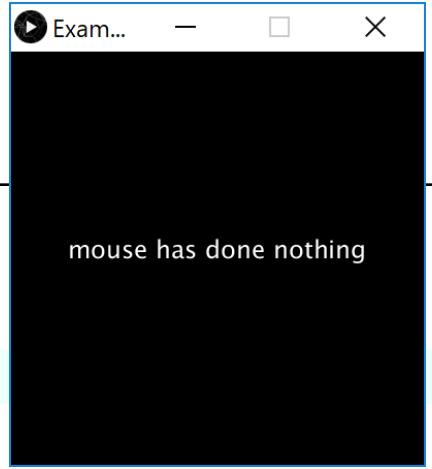
Processing Example 3.1 – draw()




```
void setup() {
  size(400, 400);
  background(0);
  textAlign(CENTER);
  textSize(24);
  fill(255);
  text("mouse has done nothing", width/2, height/2);
}

void draw() {
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Processing Example 3.1 – draw()

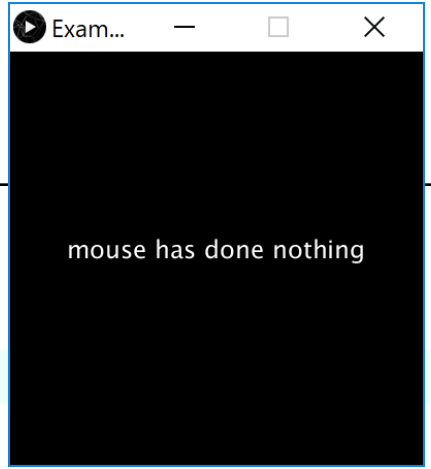



```
void setup() {
  size(400, 400);
  background(0);
  textAlign(CENTER);
  textSize(24);
  fill(255);
  text("mouse has done nothing", width/2, height/2);
}

void draw() {
}
```

Q: Why did we include the draw() method, particularly as it is empty?

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Processing Example 3.1 – draw()



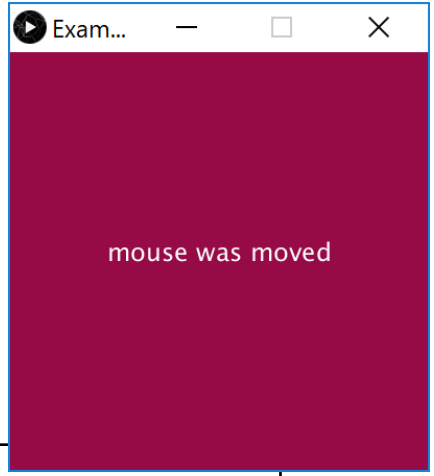

```
void setup() {
  size(400, 400);
  background(0);
  textAlign(CENTER);
  textSize(24);
  fill(255);
  text("mouse has done nothing", width/2, height/2);
}

void draw() {
}
```

A: draw() is required because
mouse events only work when a program has it.

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Processing Example 3.1 – mouseMoved()

```
void setup() {
  size(400, 400);
  background(0);
  textAlign(CENTER);
  textSize(24);
  fill(255);
  text("mouse has done nothing", width/2, height/2);
}

void draw() {
}
```

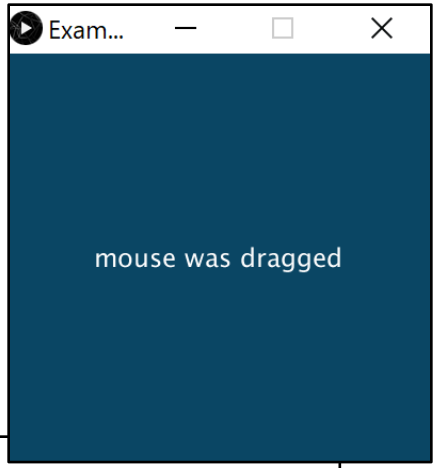



```
void mouseMoved() {
  background(150, 10, 70);
  text("mouse was moved", width/2, height/2);
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Processing Example 3.1 – mouseDragged()

```
void setup() {
  size(400, 400);
  background(0);
  textAlign(CENTER);
  textSize(24);
  fill(255);
  text("mouse has done nothing", width/2, height/2);
}

void draw() {
}
```

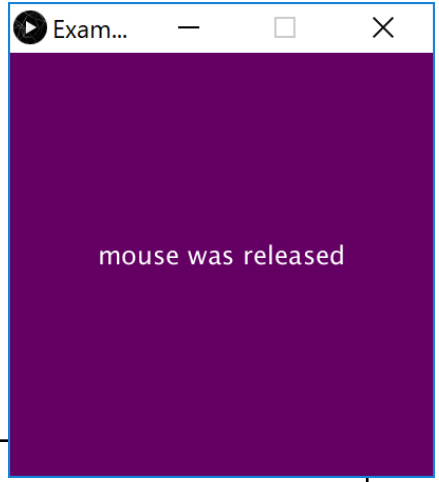



```
void mouseDragged() {
  background(10, 70, 100);
  text("mouse was dragged", width/2, height/2);
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Processing Example 3.1 – mouseReleased()

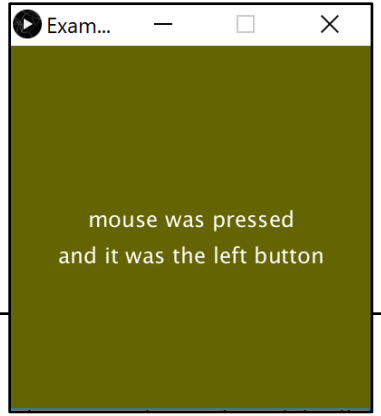
```
void setup() {
  size(400, 400);
  background(0);
  textAlign(CENTER);
  textSize(24);
  fill(255);
  text("mouse has done nothing", width/2, height/2);
}

void draw() {
}
```




```
void mouseReleased() {
  background(100, 0, 100);
  text("mouse was released", width/2, height/2);
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Processing Example 3.1 – mousePressed ()




```
void mousePressed() {
  background(100, 100, 0);
  text("mouse was pressed", width/2, height/2);
  if ( mouseButton == LEFT) {
    text("and it was the left button", width/2, height/2 + 40);
  }
  if (mouseButton == RIGHT) {
    text("and it was the right button", width/2, height/2 + 40);
  }
}
```

Based on: http://learning.codasign.com/index.php?title=Mouse_Events_in_Processing

Some previous exercises

- We will now re-work the following examples that we covered previously:
  - Example 2.5
  - Example 2.6
  - Example 2.7
  - Example 2.8

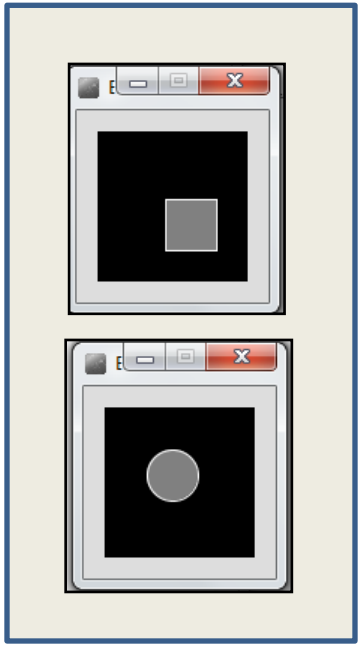
- Each of these exercises tested the *mousePressed variable*.
  - Now we want them to use the ***mousePressed()* method** instead.

EXAMPLE 2.5

Recap: Processing **Example 2.5**

Functionality:

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



Recap: Processing Example 2.5

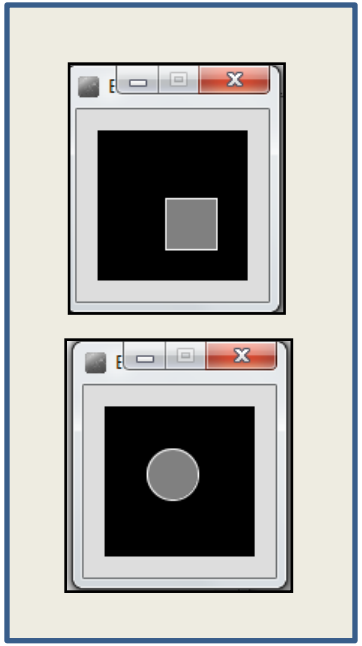
Example_2_5 | Processing 3.3.6

File Edit Sketch Debug Tools Help

Java ▾

Example_2_5

```
1  //Reas, C. & Fry, B. (2014) Processing - A F
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   }
14   else{
15     ellipse(45,45,34,34);
16   }
17 }
```



Example 2.5 (v2) – using mouse methods instead

```
void setup()
{
  size(100,100);
  stroke(255);
  fill(150);
  background(0);
  ellipse(45,45,34,34);
}

void draw(){
}
```

```
void mousePressed(){
  background(0);
  rect(45,45,34,34);
}

void mouseReleased(){
  background(0);
  ellipse(45,45,34,34);
}
```

Before VS After

```
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);
  }
}
```

```
void setup() {
  size(100,100);
  stroke(255);
  fill(150);
  background(0);
  ellipse(45,45,34,34);
}

void draw(){
}
```

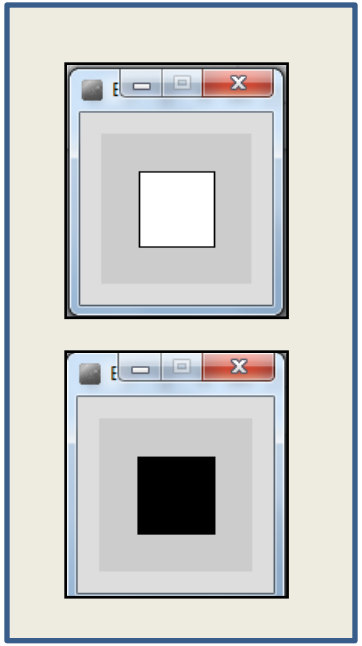
```
void mousePressed(){
  background(0);
  rect(45,45,34,34);
}

void mouseReleased(){
  background(0);
  ellipse(45,45,34,34);
}
```

EXAMPLE 2.6

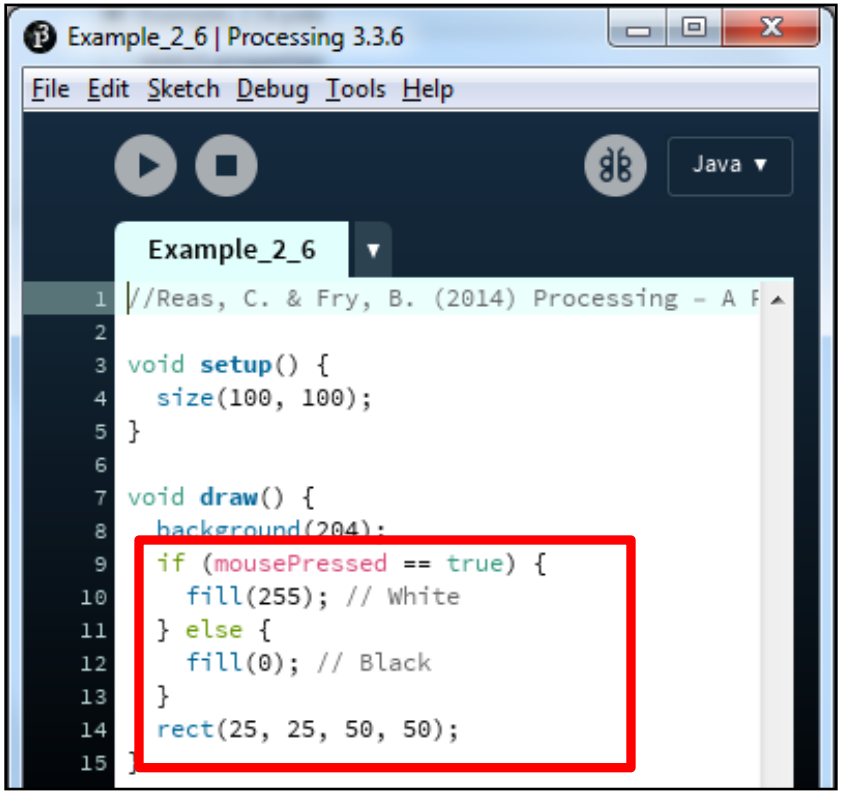
Recap: Processing **Example 2.6**

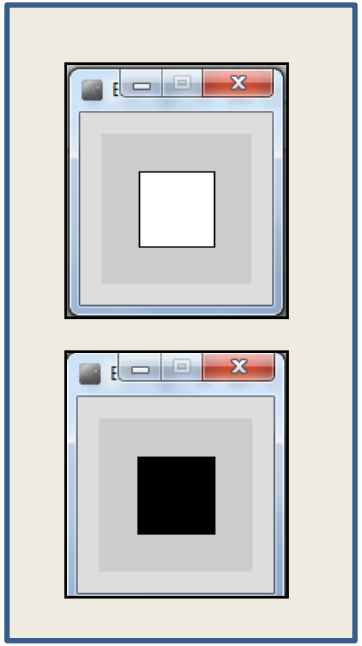
Functionality:

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



Recap: Processing Example 2.6





Example 2.6 (v2) – using mouse methods instead

```
void setup()
{
  size(100,100);
  background(204);
  fill(0);
}

void draw(){
  rect(25, 25, 50, 50);
}
```

```
void mousePressed(){
  fill(255);
}

void mouseReleased(){
  fill(0);
}
```

Before VS After

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

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

```
void setup() {
  size(100,100);
  background(204);
  fill(0);
}

void draw(){
  rect(25, 25, 50, 50);
}

void mousePressed(){
  fill(255);
}

void mouseReleased(){
  fill(0);
}
```

EXAMPLE 2.7

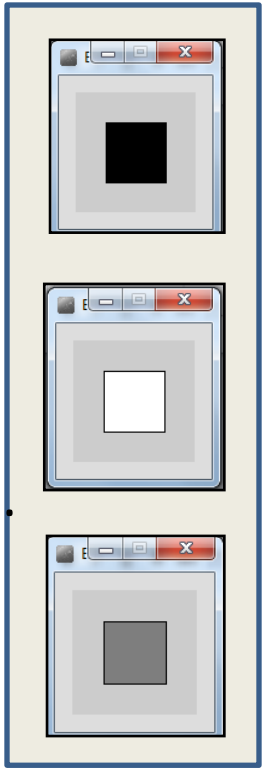
Recap: Processing **Example 2.7**

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, gray 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, gray fill the square.

- If **no mouse button** is pressed,
  set the fill to **gray** and draw a square.



Recap: Processing Example 2.7

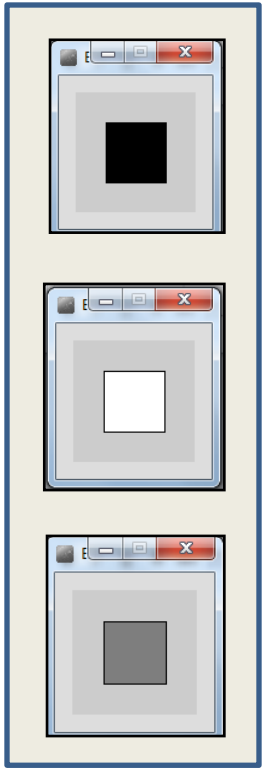
Example_2_7 | Processing 3.3.6

File Edit Sketch Debug Tools Help

Java ▾

Example_2_7

```
1  //Reas, C. & Fry, B. (2014) Processing - A F
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 }
```



Example 2.7 (v2) – using mouse methods instead

```
void setup()
{
  size(100,100);
  background(204);
  fill(126);
}

void draw(){
  rect(25, 25, 50, 50);
}
```

```
void mousePressed(){
  if (mouseButton == LEFT)
    fill(0);          // black
  else if (mouseButton == RIGHT)
    fill(255);        // white
}

void mouseReleased(){
  fill(126);
}
```

Before VS After

```
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);
}
```

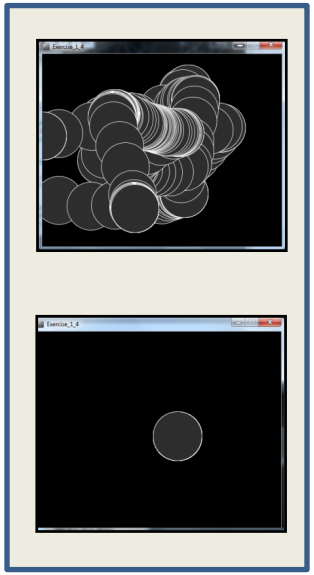
```
void setup() {
  size(100,100);
  background(204);
  fill(126);
}

void draw(){
  rect(25, 25, 50, 50);
}

void mousePressed(){
  if (mouseButton == LEFT)
    fill(0);      // black
  else if (mouseButton == RIGHT)
    fill(255);    // white
}

void mouseReleased(){
  fill(126);
}
```

EXAMPLE 2.8

Recap: Processing **Example 2.8**

Functionality:

- Draw a circle on the mouse (x,y) coordinates.
- Each time you **move the mouse**, **draw a new circle**.
- 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.



Recap: Processing Example 2.8

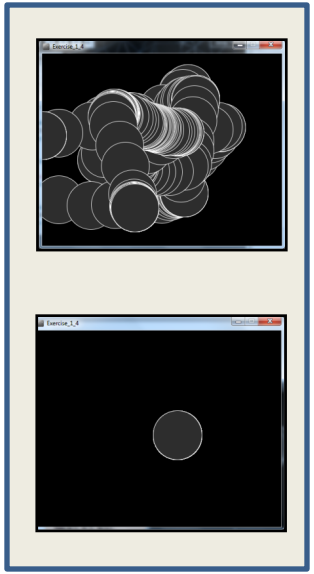
Example_2_8 | Processing 3.3.6

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  }
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 }
```



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

Example 2.5 (v2) – using mouse methods instead

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

void draw(){
   ellipse(mouseX, mouseY, 100, 100);
}
```

```
void mousePressed(){
   background(0);
}
```

Before VS After

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

void draw() {

  if (mousePressed) {
    background(0);
  }

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

}
```

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

void draw(){
   ellipse(mouseX, mouseY, 100, 100);
}

void mousePressed(){
   background(0);
}
```

EXAMPLE 2.5 (V3)

Example 2.5 (v3) – using mouse methods instead

```
void setup()
{
  size(500,400);

background(0);
  stroke(255);
  fill(45,45,45);
}

void draw(){
}
```

```
void mouseMoved(){
   ellipse(mouseX, mouseY, 100, 100);
}

void mouseClicked(){
  background(0);
  ellipse(mouseX, mouseY, 100, 100);
}
```

Before VS After

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

void draw() {

  if (mousePressed) {
    background(0);
  }

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

}
```

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

void draw(){
}

void mouseMoved(){
  ellipse(mouseX, mouseY, 100, 100);
}

void mouseClicked(){
  background(0);
  ellipse(mouseX, mouseY, 100, 100);
}
```

NOTE: draw is empty. Why do we include it?

Questions?