

Using Methods

Writing your own methods

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. Recap of method **terminology**:

- Return type
- Method names
- Parameter list

2. **Writing your own methods**:

- With no parameters
- With parameters
- That return data

Recap: Methods in Processing

- A method comprises a **set of instructions that performs some task.**
- When we **invoke** the method, it performs the task.
- Some methods that we have used are:
 - rect(), ellipse(), stroke(), line(), fill(), etc.
 - void mousePressed()
 - void setup(), void draw()

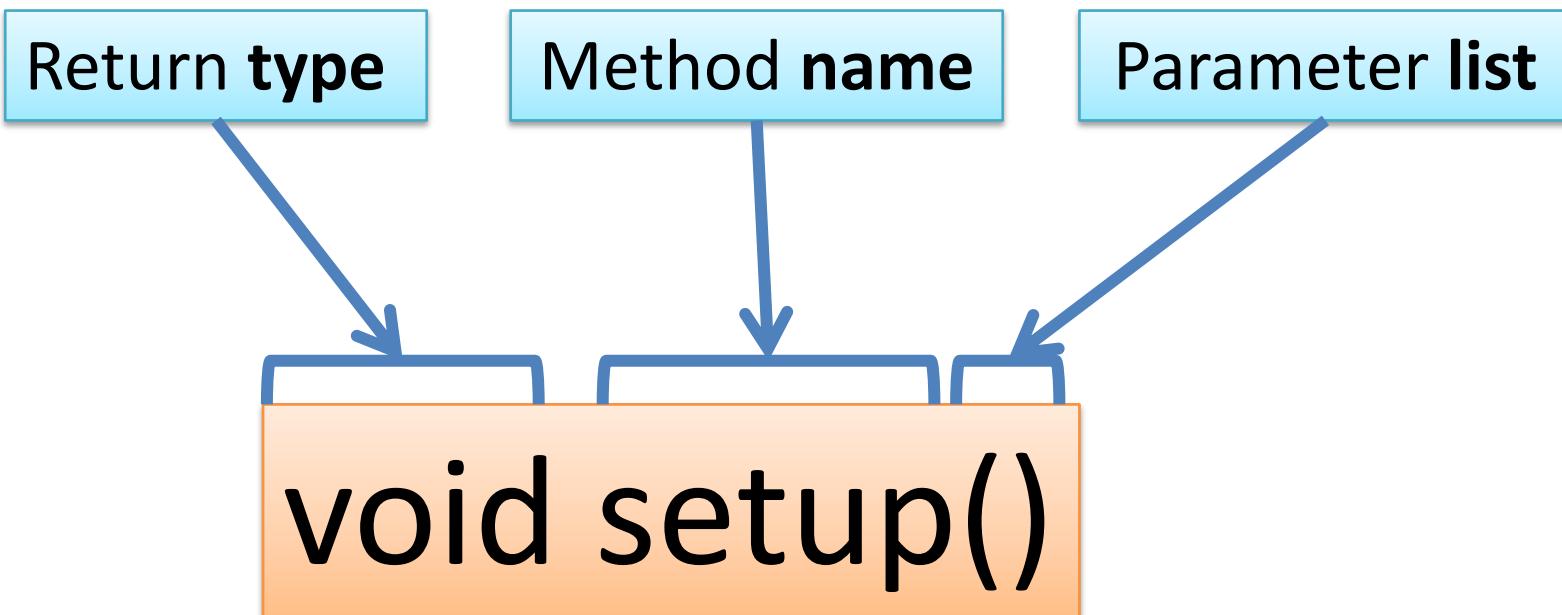
Recap: Method terminology

Method **signature / header**

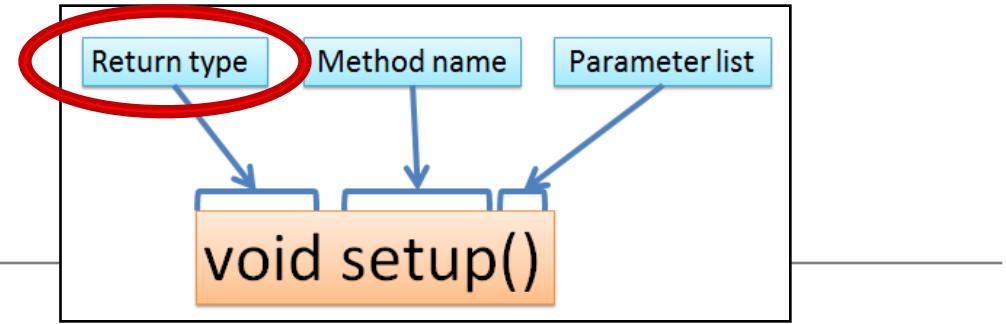
Method **body**

```
void setup()
{
    size(640, 360);
    background(120);
}
```

Recap: Method signature



Recap: Return Types



- Methods can **return information**.
- The **void** keyword means that **nothing** is returned from the method.
- 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.
- You can **only have one return type per method**.
- Methods can return any type of data
e.g. boolean, byte, char, int, float, String, etc.

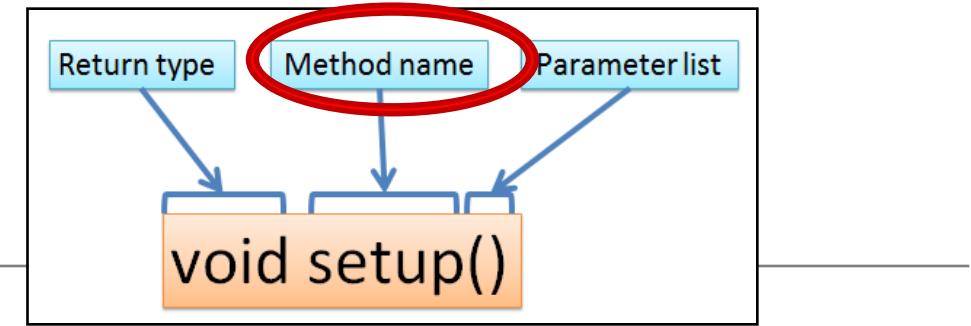
Recap: Return Types

```
int val = 30;  
  
void draw()  
{  
    int result = timestwo(val);  
    println(result);  
}
```

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

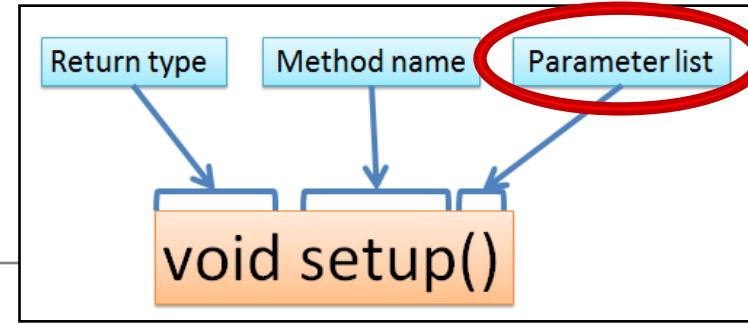
// The red **int** in the function declaration
// specifies the type of data to be returned.

Recap: Method name



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

Recap: Parameter list



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

Methods do not have to pass parameters.

These methods don't need any additional information to do their tasks.

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

If a method needs additional information to execute, we provide a parameter so that the information can be passed into it.

A method can have any number of parameters.

`void strokeWeight (float weight)`
`void size (int width, int height)`

Topics list

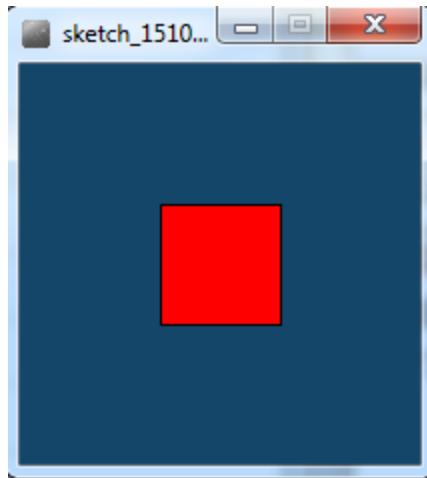
1. Recap of method **terminology**:

- Return type
- Method names
- Parameter list

2. Writing your own methods:

- 
- With no parameters
 - With parameters
 - That return data

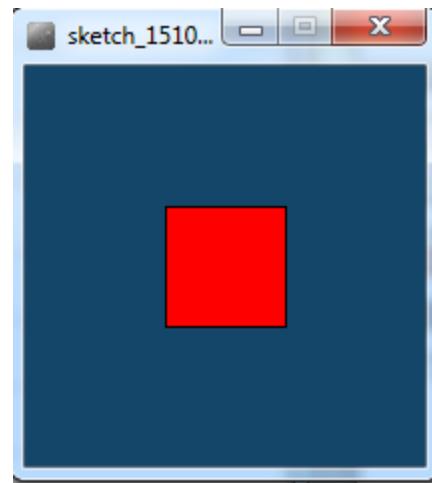
Writing methods with NO parameters



- Draw a red square at certain (x, y) coordinates.

Processing

Example 3.2



Method call →

Method definition →

13 Example_3_2 | Processing 3.3.6

File Edit Sketch Debug Tools Help

Example_3_2

```
1 void setup()
2 {
3     size(200,200);
4     background(20,70,105);
5 }
6
7 void draw()
8 {
9     drawRedSquare();
10 }
11
12 void drawRedSquare()
13 {
14     fill(255,0,0);
15     rect(70,70,60,60);
16 }
```

Topics list

1. Recap of method **terminology**:

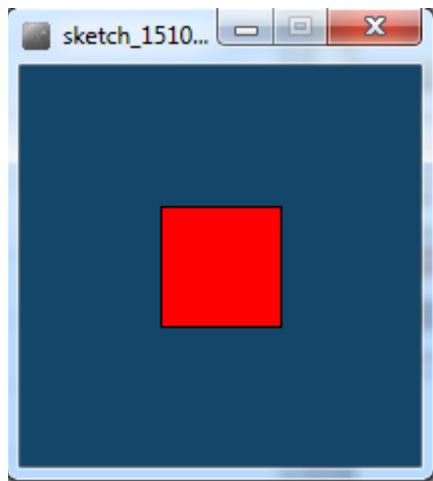
- Return type
- Method names
- Parameter list

2. Writing your own methods:

- With no parameters
- With parameters
- That return data



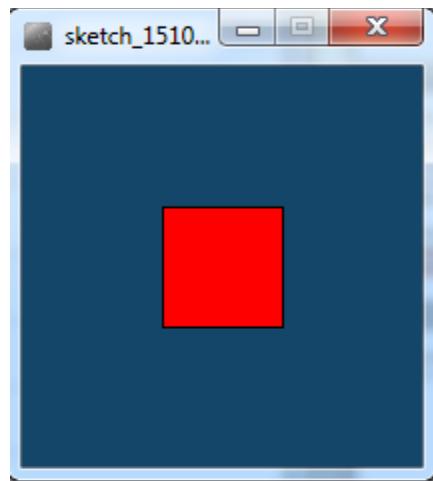
Writing methods with parameters



- Now update the code so that you can:
pass in the length of the square into the method
drawRedSquare().

Processing

Example 3.3



P Example_3_3 | Processing 3.3.6

File Edit Sketch Debug Tools Help

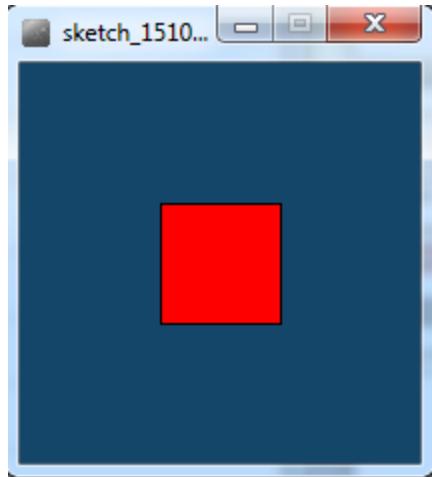
Example_3_3

```
1 void setup()
2 {
3     size(200,200);
4     background(20,70,105);
5 }
6
7 void draw()
8 {
9     drawRedSquare(60);
10 }
11
12 void drawRedSquare(int length)
13 {
14     fill(255,0,0);
15     rect(70,70,length, length);
16 }
```

The code in the Processing editor shows a sketch named "Example_3_3". It starts with a setup function that sets the size to 200x200 and the background color to (20, 70, 105). The draw function contains a call to "drawRedSquare(60)". Below it, a red box highlights the "drawRedSquare" function definition, which takes an integer parameter "length" and fills a rectangle at coordinates (70, 70) with a width and height equal to "length". A red arrow points from the highlighted "drawRedSquare(60)" call in the draw() function down to the definition of the drawRedSquare function.

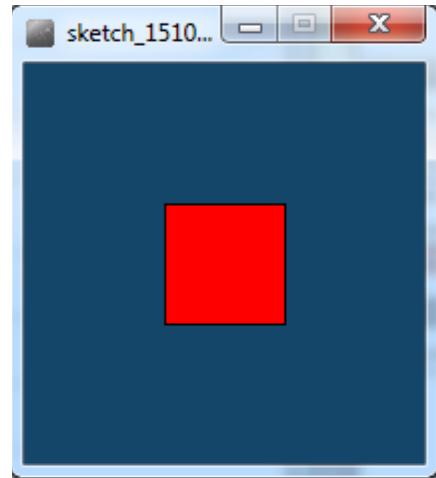
Writing methods with parameters

- Now update the code so that you can pass in the:
 - **length** of the square
 - **xCoordinate** of the square
 - **yCoordinate** of the square
- into the method, *drawRedSquare()*.



Processing

Example 3.4



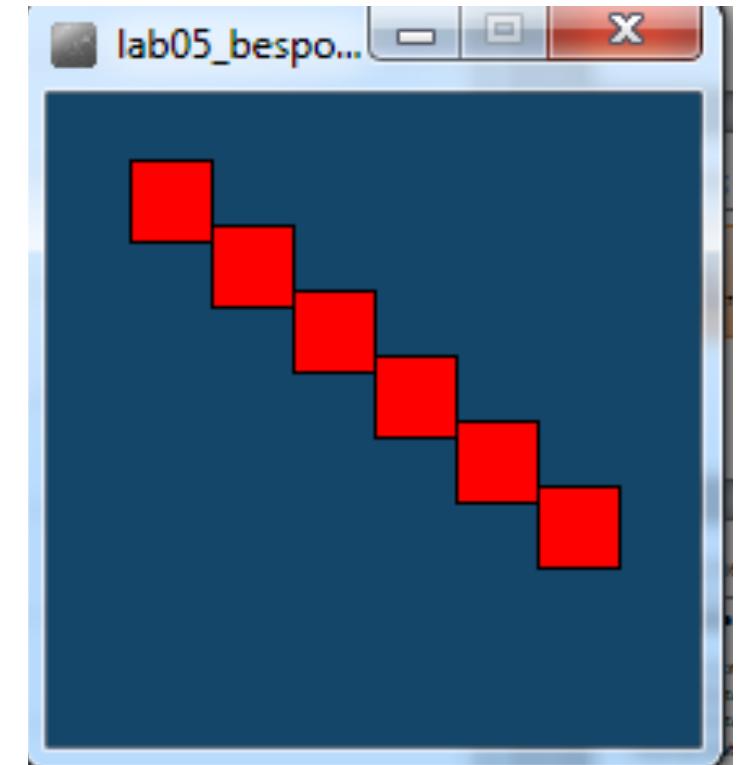
```
Example_3_4 ▾

1 void setup()
2 {
3     size(200,200);
4     background(20,70,105);
5 }
6
7 void draw()
8 {
9     drawRedSquare(60, 70, 40);
10}
11
12 void drawRedSquare(int length, int xCoord, int yCoord)
13{
14    fill(255,0,0);
15    rect(xCoord,yCoord, length, length);
16}
```

The code example illustrates the use of a custom function to draw a red square. The `drawRedSquare` function takes three parameters: `length`, `xCoord`, and `yCoord`. It uses the `fill` and `rect` functions to draw a red square at the specified coordinates with a given side length. The `draw` function calls this custom function with specific arguments (60, 70, 40) to draw a red square at the center of the canvas.

Writing methods with parameters

- Now update the code so that you can call the *drawRedSquare()* multiple times (using a **loop**).



Processing Example 3.5

Example_3_5

```
1 void setup()
2 {
3     size(200,200);
4     background(20,70,105);
5 }
6
7 void draw()
8 {
9     for (int i = 1; i < 7; i++)
10    {
11        drawRedSquare(25, i*25, i*20);
12    }
13 }
14
15 void drawRedSquare(int length, int xCoord, int yCoord)
16 {
17     fill(255,0,0);
18     rect(xCoord,yCoord, length, length);
19 }
```

Topics list

1. Recap of method **terminology**:

- Return type
- Method names
- Parameter list

2. Writing your own methods:

- With no parameters
- With parameters
- That return data



Writing methods that return data

- Write a method called **timesTwo**.
- This method should
 - take in one **int** parameter.
 - multiply this **int** by 2 and
 - return it back to where the **timesTwo** method was called from.
 - The returned value should be **printed to the console**.

Processing Example 3.6

```
Example_3_6 ▾  
1 //source: https://processing.org/reference/return.html  
2  
3 int value = 30;  
4  
5 void setup() {  
6     int result = timestwo(value);  
7     println(result);  
8 }  
9  
10 int timestwo(int val) {  
11     val = val * 2;  
12     return val;  
13 }  
14
```



Summary

1. Recap of method **terminology**:

- Return type
- Method names
- Parameter list

2. Writing your own methods:

- With no parameters
- With parameters
- That return data

Questions?

