

Data Types and Operators

Variables, Data Types & Arithmetic Operators

Produced
by:

Department of Computing and Mathematics



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Topics list

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- Arithmetic operators and Order of Evaluation.

Variables

In Programming, variables:

- are created (defined) in your programs.
- are used to store data (whose value can change over time).
- have a data type.
- have a name.
- are a VERY important programming concept.

Variable names...

- Are case-sensitive.
- Begin with either:
 - a **letter (preferable)**,
 - the dollar sign "\$", or
 - the underscore character "_".
- Can contain letters, digits, dollar signs, or underscore characters.
- Can be any length you choose.
- Must not be a **keyword or reserved word** e.g. int, while, etc.
- Cannot contain white spaces (i.e. space bar value).

Variable names should be carefully chosen

- Use full words instead of cryptic abbreviations e.g.
 - variables named **speed** and **gear** are much more intuitive than abbreviated versions, such as **s** and **g**.
- If the name consists of:
 - only one word, spell that word in all lowercase letters e.g. **ratio**.
 - more than one word, capitalise the first letter of each subsequent word e.g. **gearRatio** and **currentGear**.

Topics list

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- Arithmetic operators and Order of Evaluation.

Assignment Statement

- Values are stored in variables via assignment statements:

Syntax	<code>variable = expression;</code>
Example	<code>numberOfItems = 100;</code>

- A variable stores a single value, so any previous value is lost.
- Assignment statements work by taking the value of what appears on the right-hand side of the operator and copying that value into a variable on the left-hand side.

Topics list

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- Arithmetic operators and Order of Evaluation.

Data Types

- In Java, when we define a variable, we **have** to give it a data type.
- The data type defines the **kinds of values** (data) that can be stored in the variable e.g.
 - - 456
 - 2
 - 45.7897
 - I Love Programming
 - S
 - true
- The data type also determines the operations that may be performed on it.

Topics list

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- Arithmetic operators and Order of Evaluation.

Java's Primitive Data Types

- Java programming language has eight primitive data types.
- A primitive type is predefined by the language and is named by a reserved keyword.
- A primitive type is highlighted red when it is typed into the PDE e.g.

```
int a;  
boolean flag;  
float number;
```

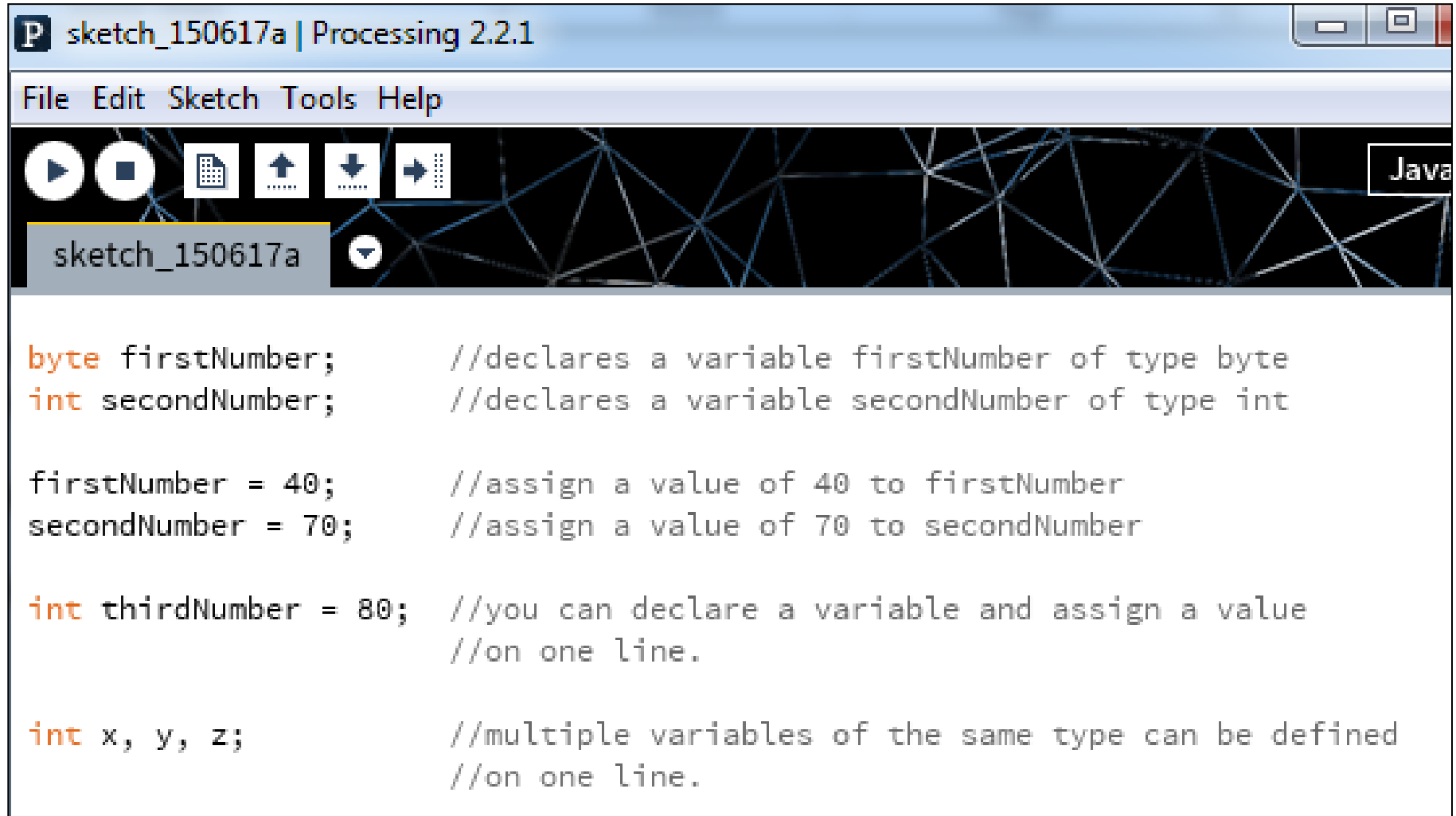
Topics list

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- Arithmetic operators and Order of Evaluation.

Java's Primitive Data Types (whole numbers)

Type	Byte-size	Minimum value (inclusive)	Maximum value (inclusive)	Default value	Typical Use
byte	8-bit	-128	127	0	Useful in applications where memory savings apply.
short	16-bit	-32,768	32,767	0	
int	32-bit	-2,147,483,648	2,147,483,647	0	Default choice.
long	64-bit	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	0L	Used when you need a data type with a range of values larger than that provided by int.

Declaring variables of a specific type



The screenshot shows the Processing 2.2.1 IDE interface. The title bar reads "P sketch_150617a | Processing 2.2.1". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for running, stopping, opening, saving, and other functions. The sketch name "sketch_150617a" is displayed in the top left of the editor area. The code editor contains the following Java code:

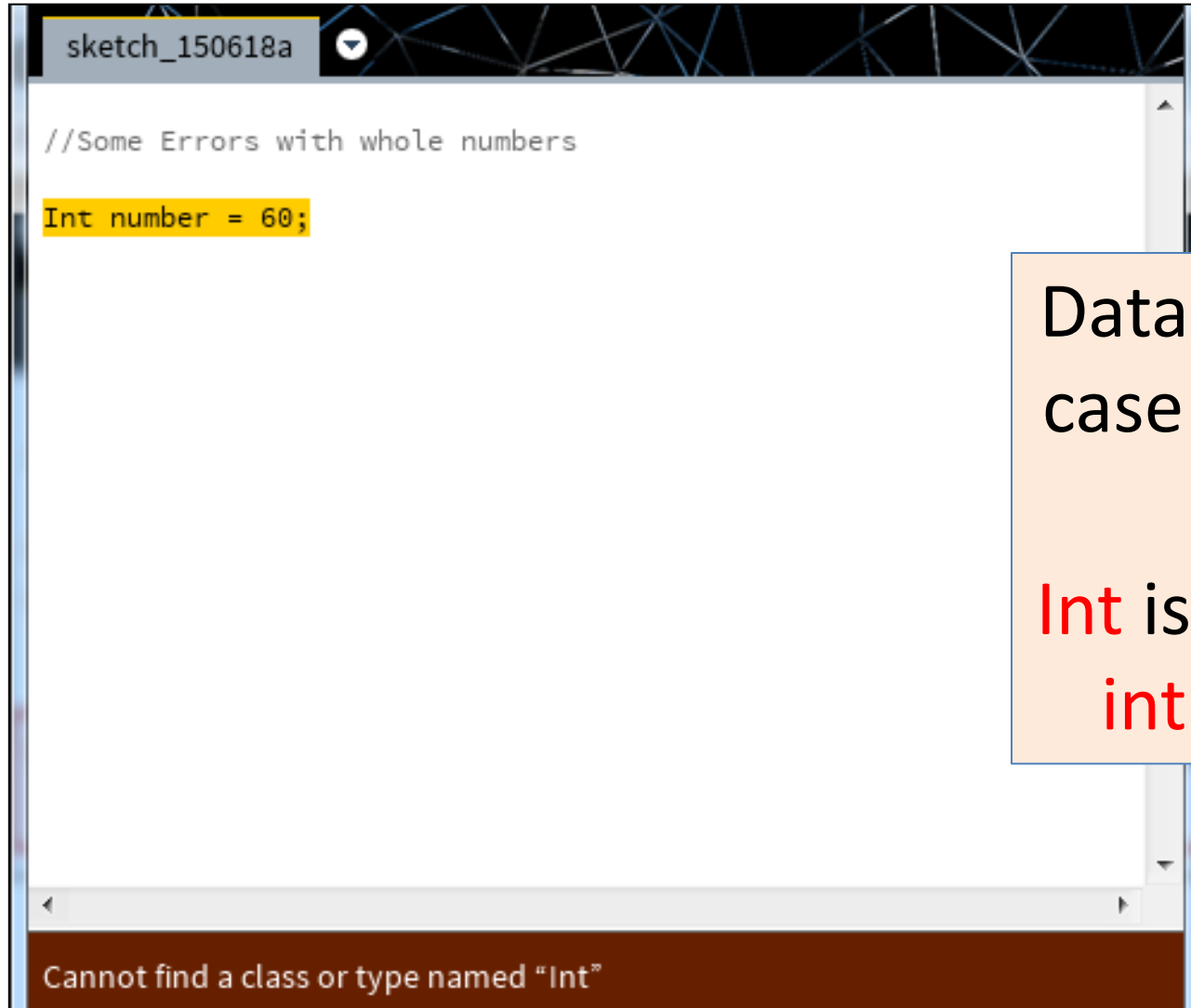
```
byte firstNumber;           //declares a variable firstNumber of type byte
int secondNumber;           //declares a variable secondNumber of type int

firstNumber = 40;           //assign a value of 40 to firstNumber
secondNumber = 70;          //assign a value of 70 to secondNumber

int thirdNumber = 80;        //you can declare a variable and assign a value
                             //on one line.

int x, y, z;                 //multiple variables of the same type can be defined
                             //on one line.
```

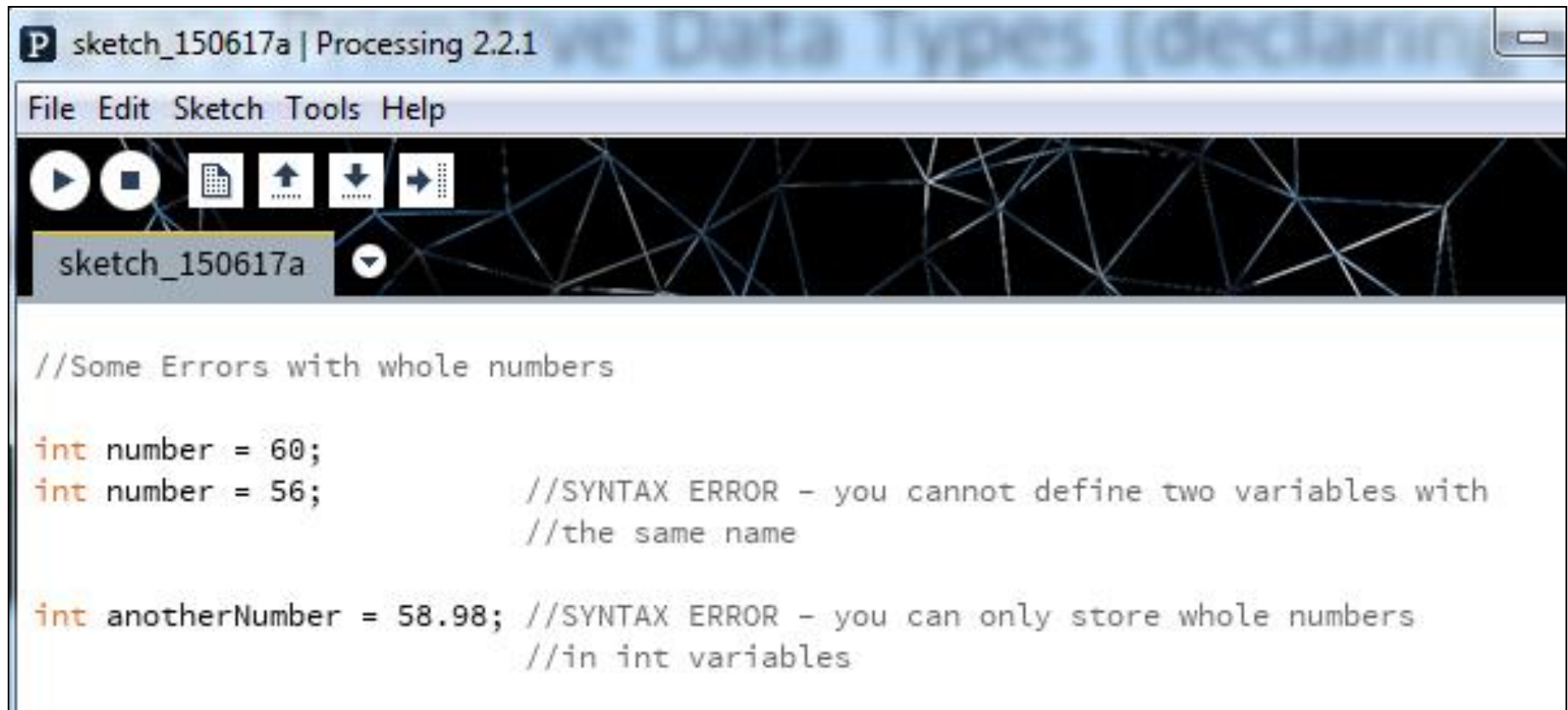
Declaring variables - some errors



Data types are
case sensitive.

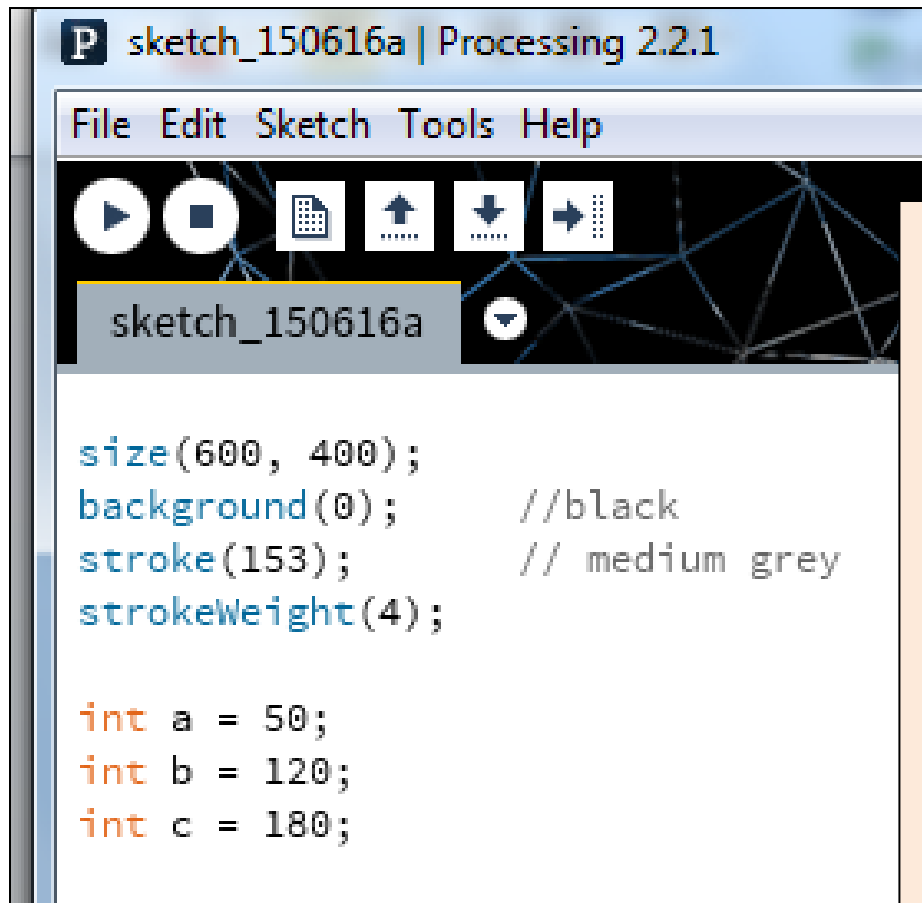
Int is not valid.
int is valid.

Declaring variables - some errors

A screenshot of the Processing 2.2.1 IDE interface. The title bar shows 'P sketch_150617a | Processing 2.2.1'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for running, stopping, opening, saving, and other functions. The main text area contains the following code:

```
//Some Errors with whole numbers  
  
int number = 60;  
int number = 56;           //SYNTAX ERROR - you cannot define two variables with  
                           //the same name  
  
int anotherNumber = 58.98; //SYNTAX ERROR - you can only store whole numbers  
                           //in int variables
```

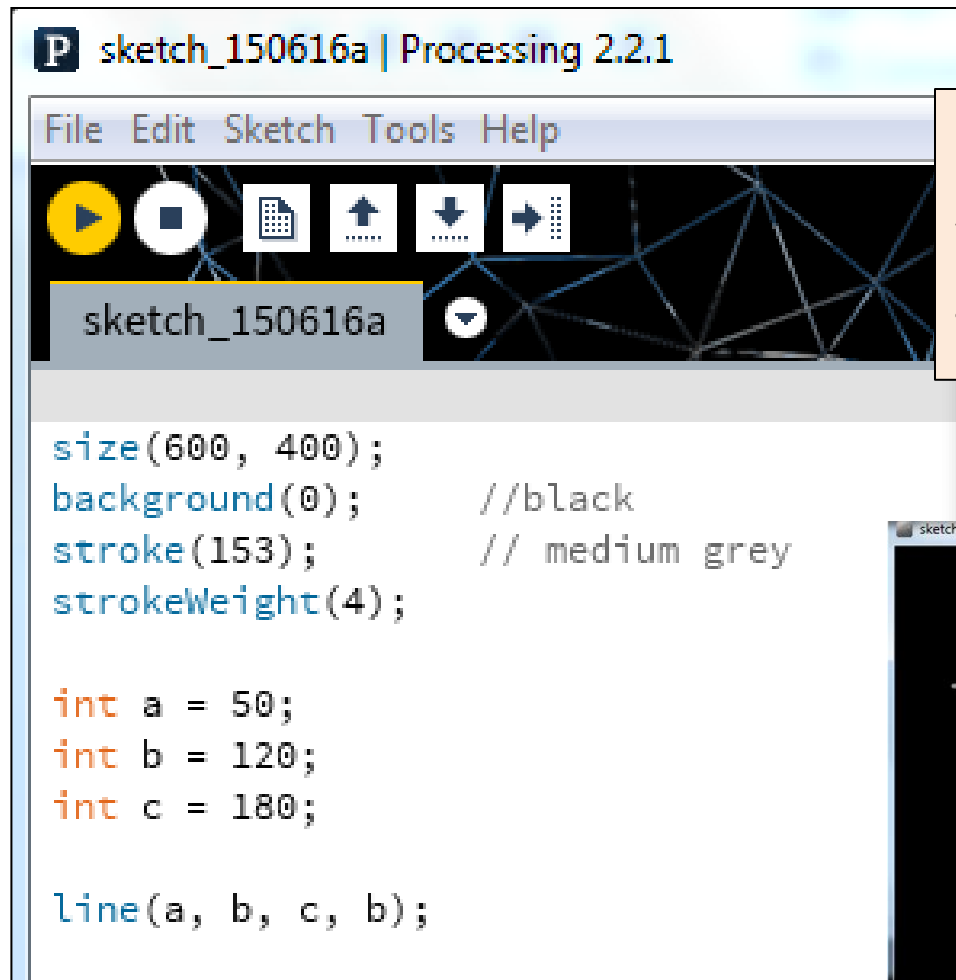

Java's Primitive Data Types: int Example 3.1



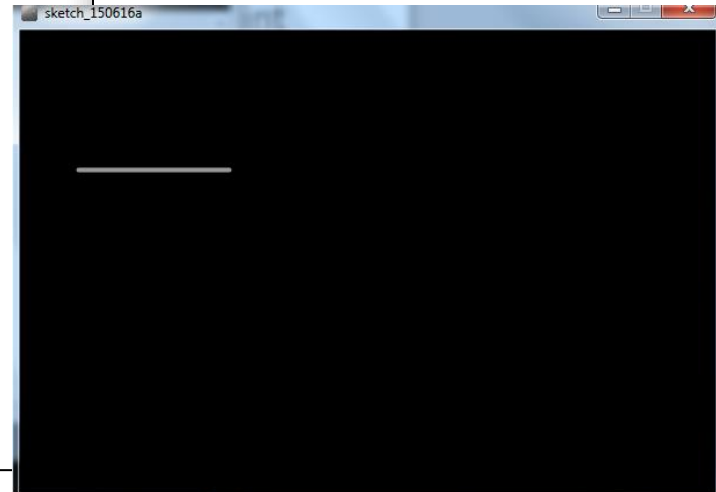
In this example, we have:

- defined three variables (a, b and c)
- that can hold whole numbers (int).
- and are set with a starting value.

Java's Primitive Data Types: int Example 3.2



We can pass the defined variables as values to functions.



Java's Primitive Data Types: int Example 3.3

```
size(600, 400);  
background(0);    //black  
stroke(153);      // medium grey  
strokeWeight(4);  
  
int a = 50;  
int b = 120;  
int c = 180;  
  
line(a, b, c, b);
```

Could we have used the **byte** data type instead of **int**? Why?

Type	Minimum value (inclusive)	Maximum value (inclusive)
byte	-128	127
short	-32,768	32,767
int	-2,147,483,648	2,147,483,647
long	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

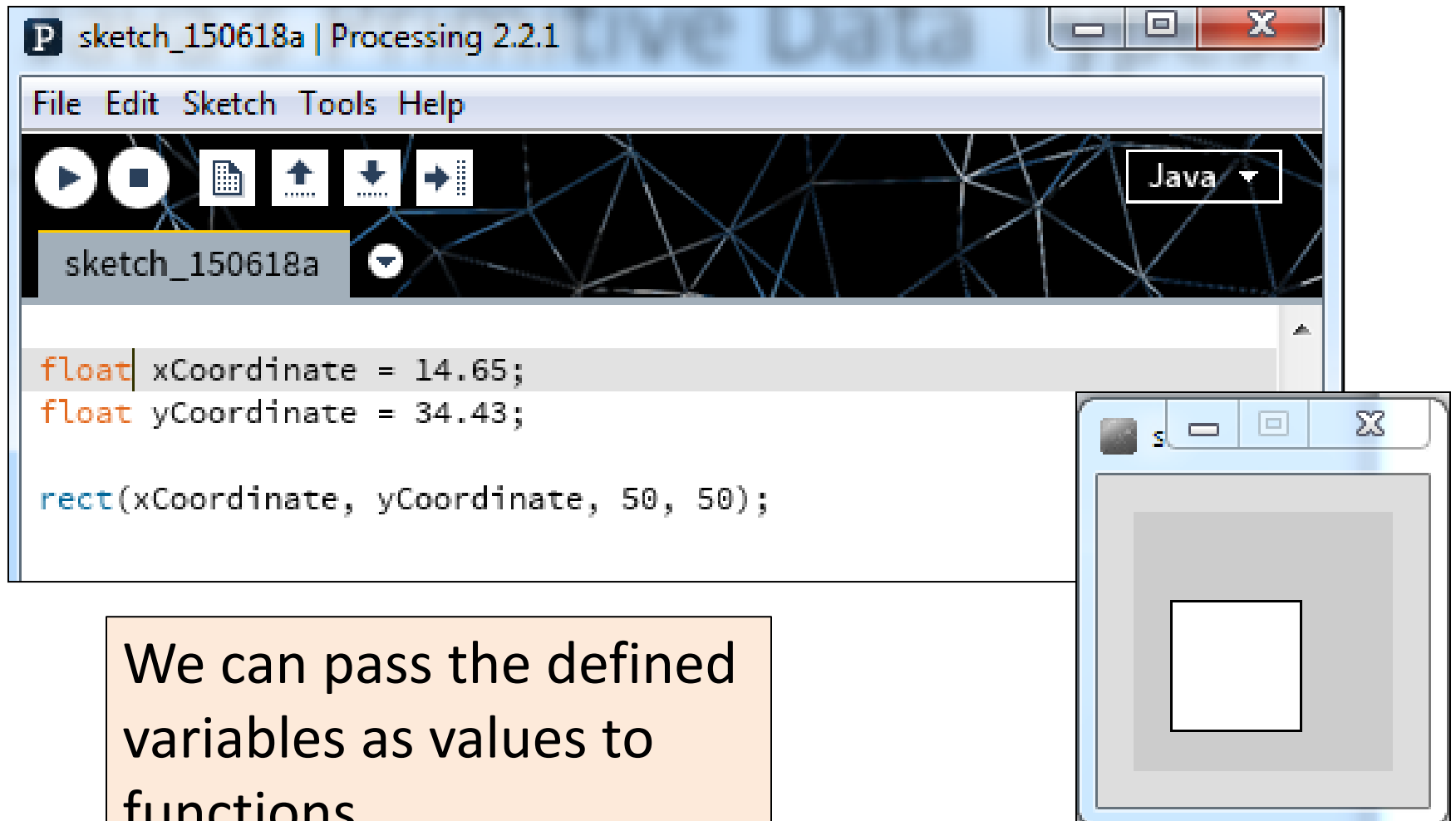
Topics list

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- Arithmetic operators and Order of Evaluation.

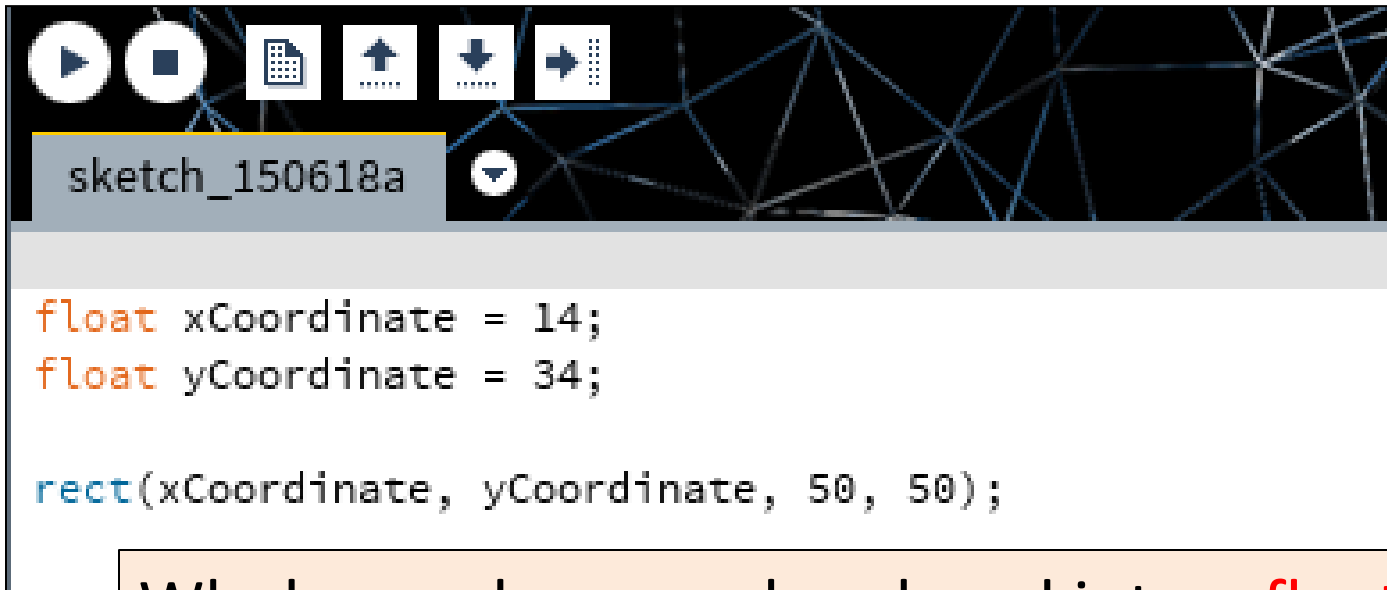
Java's Primitive Data Types (decimal numbers)

Type	Byte-size	Minimum value (inclusive)	Maximum value (inclusive)	Default value	Typical Use
float	32-bit	<i>Beyond the scope of this course.</i>		0.0f	Useful in applications where memory savings apply. Default choice when using Processing .
double	64-bit	<i>Beyond the scope of this course.</i>		0.0d	Default choice when programming Java apps .

Java's Primitive Data Types: float Example 3.4



Java's Primitive Data Types: float Example 3.5

A screenshot of a Java IDE window. The title bar shows a play button, a square icon, and a document icon. Below the title bar, the file name 'sketch_150618a' is displayed. The main editing area contains the following code:

```
float xCoordinate = 14;  
float yCoordinate = 34;  
  
rect(xCoordinate, yCoordinate, 50, 50);
```

Whole numbers can be placed into a **float** variable.

Q: Why?

A: There is no loss of precision. We are not losing any data.

Passing variables as arguments: some errors

```
sketch_150618a  
  
double xCoordinate = 14.65;  
double yCoordinate = 34.43;  
  
rect(xCoordinate, yCoordinate, 50, 50);
```

We changed the data type of our variables from **float** to **double**.

When we try to run the code, we get this syntax error.

What's wrong?

The method rect(float, float, float, float) in the type PApplet is not applicable for the arguments (double, double, int, int)

Passing variables as arguments: some errors

From: https://processing.org/reference/rect_.html

Syntax	<code>rect(a, b, c, d)</code>	
Parameters	a	float: x-coordinate of the rectangle by default
	b	float: y-coordinate of the rectangle by default
	c	float: width of the rectangle by default
	d	float: height of the rectangle by default

```
double xCoordinate = 14.65;  
double yCoordinate = 34.43;  
  
rect(xCoordinate, yCoordinate, 50, 50);
```

The method `rect(float, float, float, float)` in the type `PApplet` is not applicable for the arguments `(double, double, int, int)`

Topics list

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- Arithmetic operators and Order of Evaluation.

Java's Primitive Data Types (others)

Type	Byte-size	Minimum value (inclusive)	Maximum value (inclusive)	Default value	Typical Use
char	16-bit	'\u0000' (or 0)	'\uffff' (or 65,535)	'\u0000'	Represents a Unicode character.
boolean	1-bit	n/a		false	Holds either true or false and is typically used as a flag.

http://en.wikipedia.org/wiki/List_of_Unicode_characters

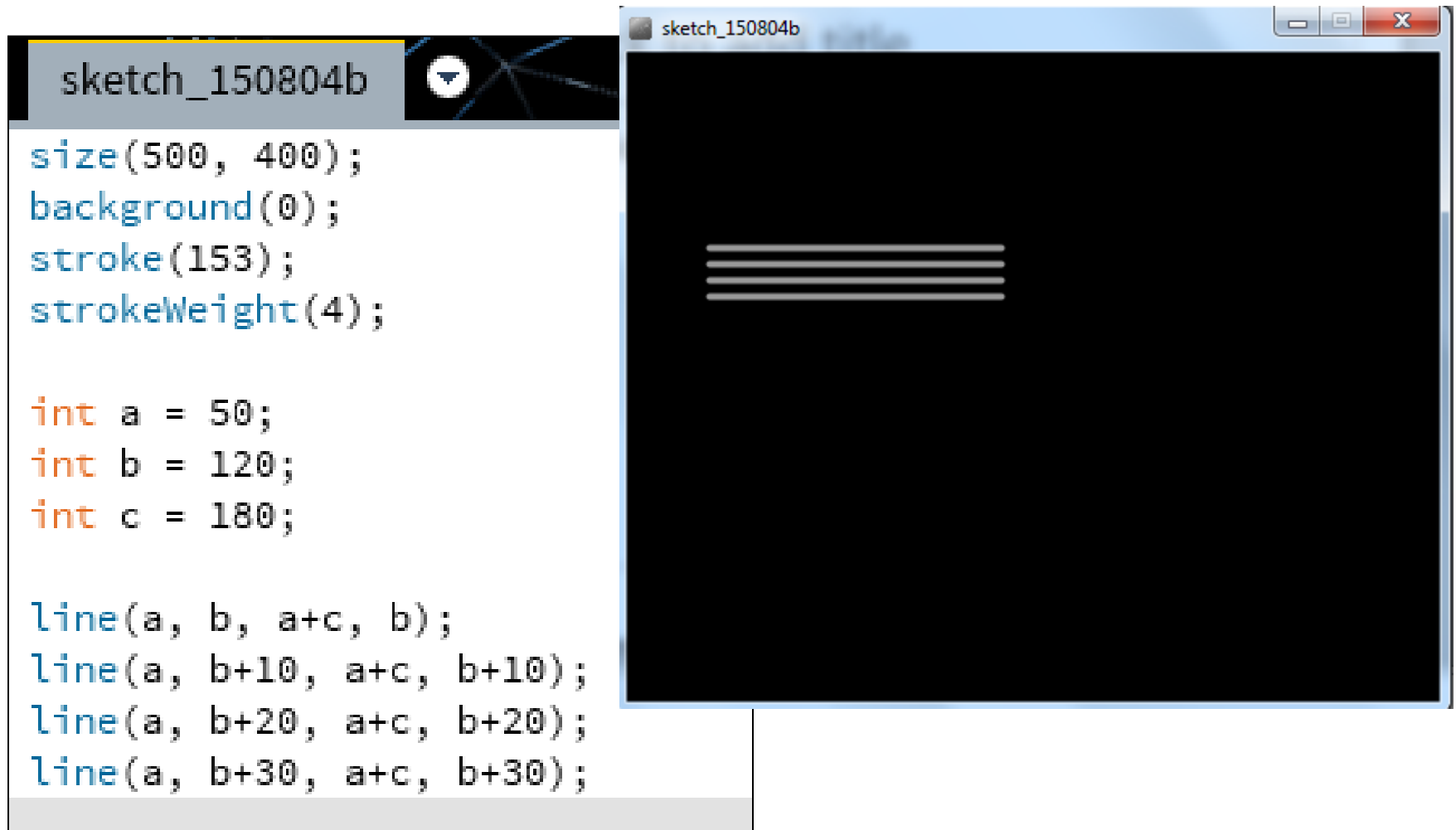
Topics list

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- Arithmetic operators and Order of Evaluation.

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

Arithmetic operators: Example 3.6



Based on the Processing Example: Basics → Data → Variables

Arithmetic operators: Example 3.7

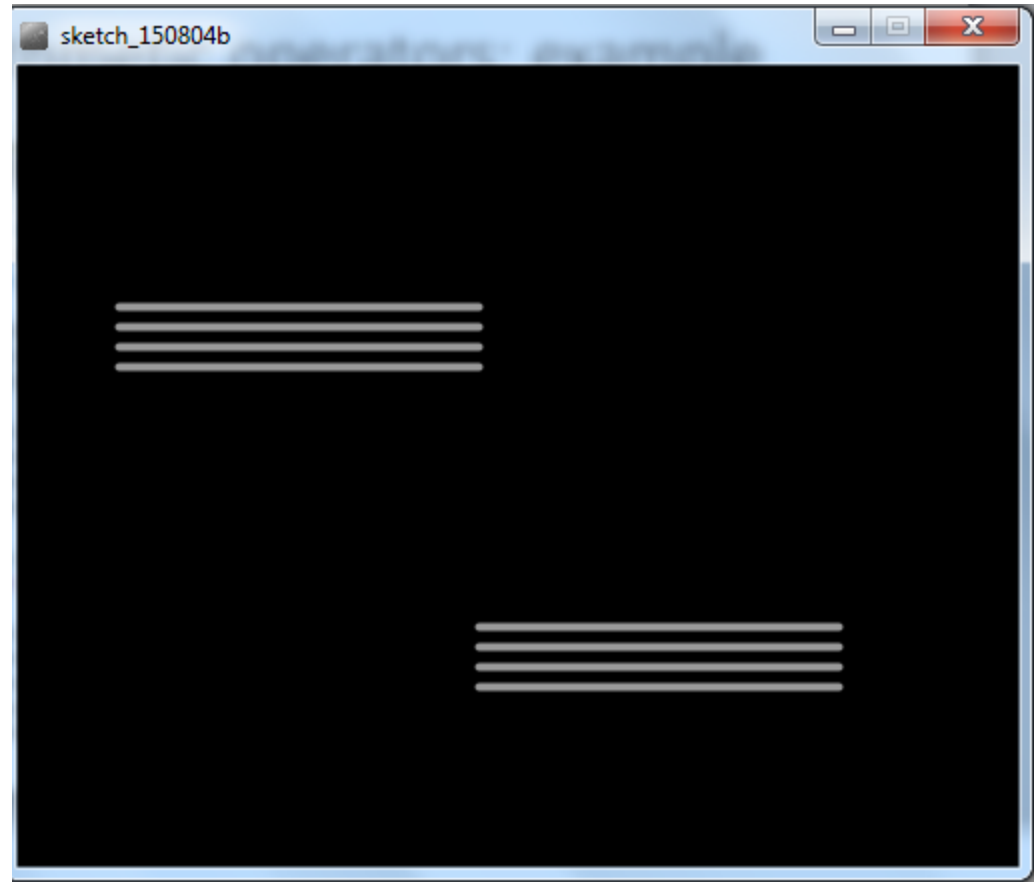
```
sketch_150804b
size(500, 400);
background(0);
stroke(153);
strokeWeight(4);

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);

a = a + c;
b = height-b;

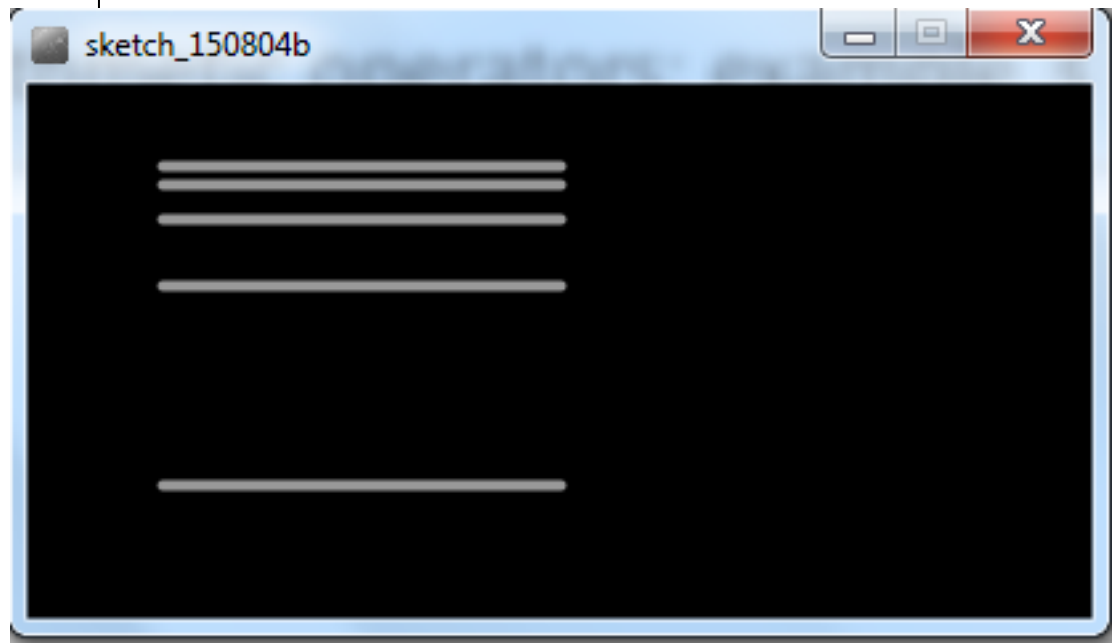
line(a, b, a+c, b);
line(a, b+10, a+c, b+10);
line(a, b+20, a+c, b+20);
line(a, b+30, a+c, b+30);
```



Based on the Processing Example: Basics → Data → Variables

Arithmetic operators: Example 3.8

```
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);
```



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

```
int counter = 0;
```

```
void draw()
```

```
{
```

```
  line (mouseX, mouseY, 50,50);
```

```
  counter = counter + 1;
```

```
  println (counter);
```

```
}
```

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**.

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?





Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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