An Introduction to Processing

Variables, Data Types & Arithmetic 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

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
    1. Whole numbers.
    2. Decimal numbers.
    3. Others.
5. Arithmetic operators.

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.

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html

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.
    - This is called **camelCase**

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
    1. Whole numbers.
    2. Decimal numbers.
    3. Others.
5. Arithmetic operators.

Assignment Statement

- Values are stored in variables via assignment statements:

Syntax	`variable = expression;`
Example	`diameter = 100;`

- 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

1. Variables.

2. Assignment statement.

3. Data Types.

4. Java's Primitive Data Types
    1. Whole numbers.
    2. Decimal numbers.
    3. Others.

5. Arithmetic operators.

Data Types

- In Java, when we define a variable, we **<u>have</u>** to give it a data type.

- The data type defines the **<u>kinds of values</u>** (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 **<u>operations</u>** that may be performed on it.

Data Types

- Java uses two kinds of data types:
  - **Primitive** types
  - **Object** types

- We are only looking at **Primitive** types now; we will cover Object types later in the module.

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
    1. Whole numbers.
    2. Decimal numbers.
    3. Others.
5. Arithmetic operators.

Java's Primitive Data Types

- Java programming language supports 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 numberOfItems;
boolean bounceUp;
float lengthOfRectangle;
```

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
    1. Whole numbers.
    2. Decimal numbers.
    3. Others.
5. Arithmetic operators.

Java's Primitive Data Types (whole numbers)

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

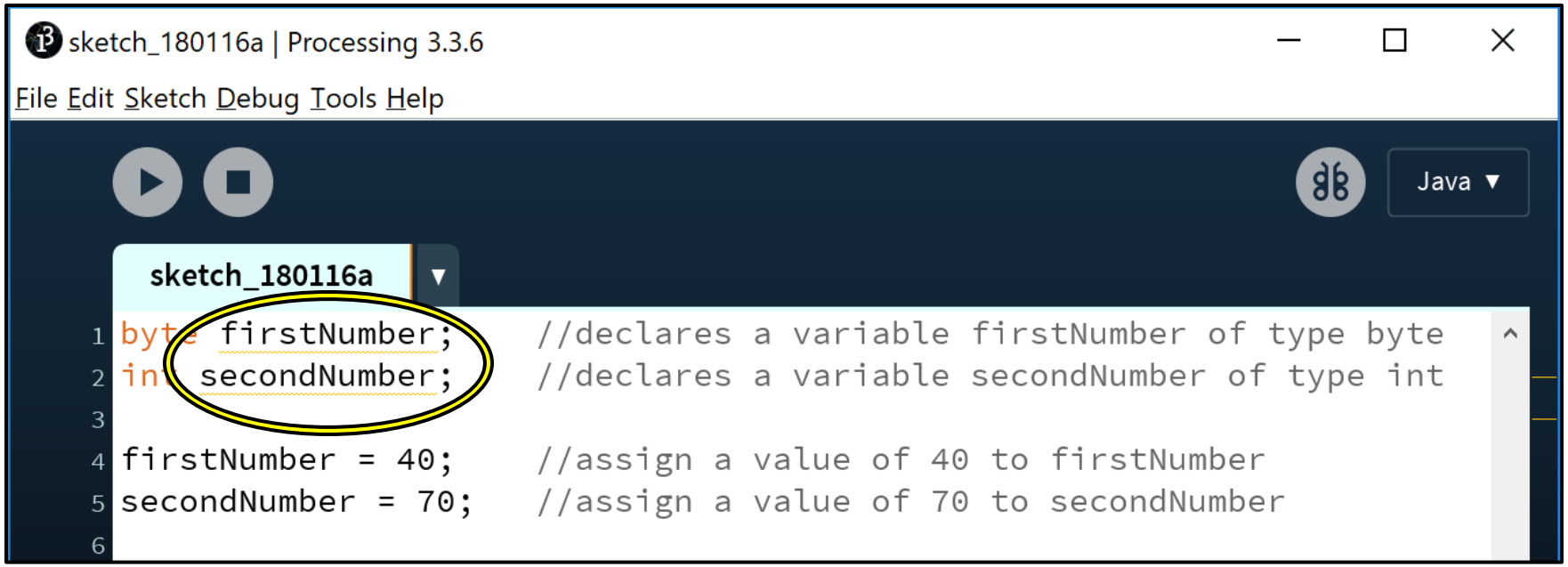
$2^{NumberOfBits}$	Range of values
2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256

If the eight bit is used for the sign, we get a range from -128 to +127

i.e. 256/2 = +-128 values,

but a value is required to store 0, so range is **-128 to +127**

Declaring variables of a specific type



YELLOW underline - indicates that the variable hasn't been used meaningfully

Declaring variables of a specific type

sketch_180116a | Processing 3.3.6

File Edit Sketch Debug Tools Help

Java

sketch_180116a

```
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.
```

Declaring variables of a specific type

sketch_180116a | Processing 3.3.6

File Edit Sketch Debug Tools Help

Java

sketch_180116a

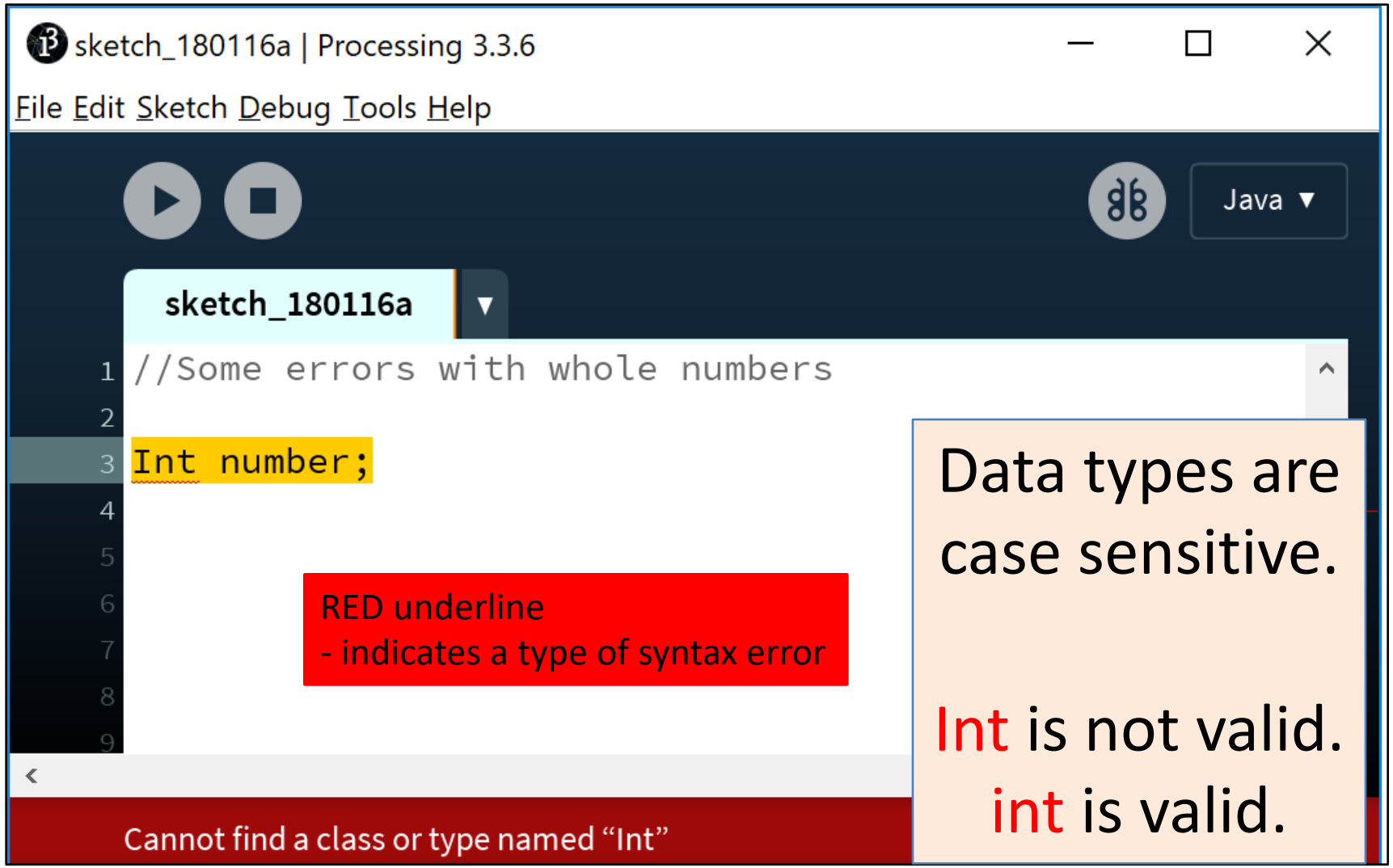
```
byte firstNumber;     //declares a variable firstNumber of type byte
int secondNumber;     //declares a variable secondNumber of type int

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.
```

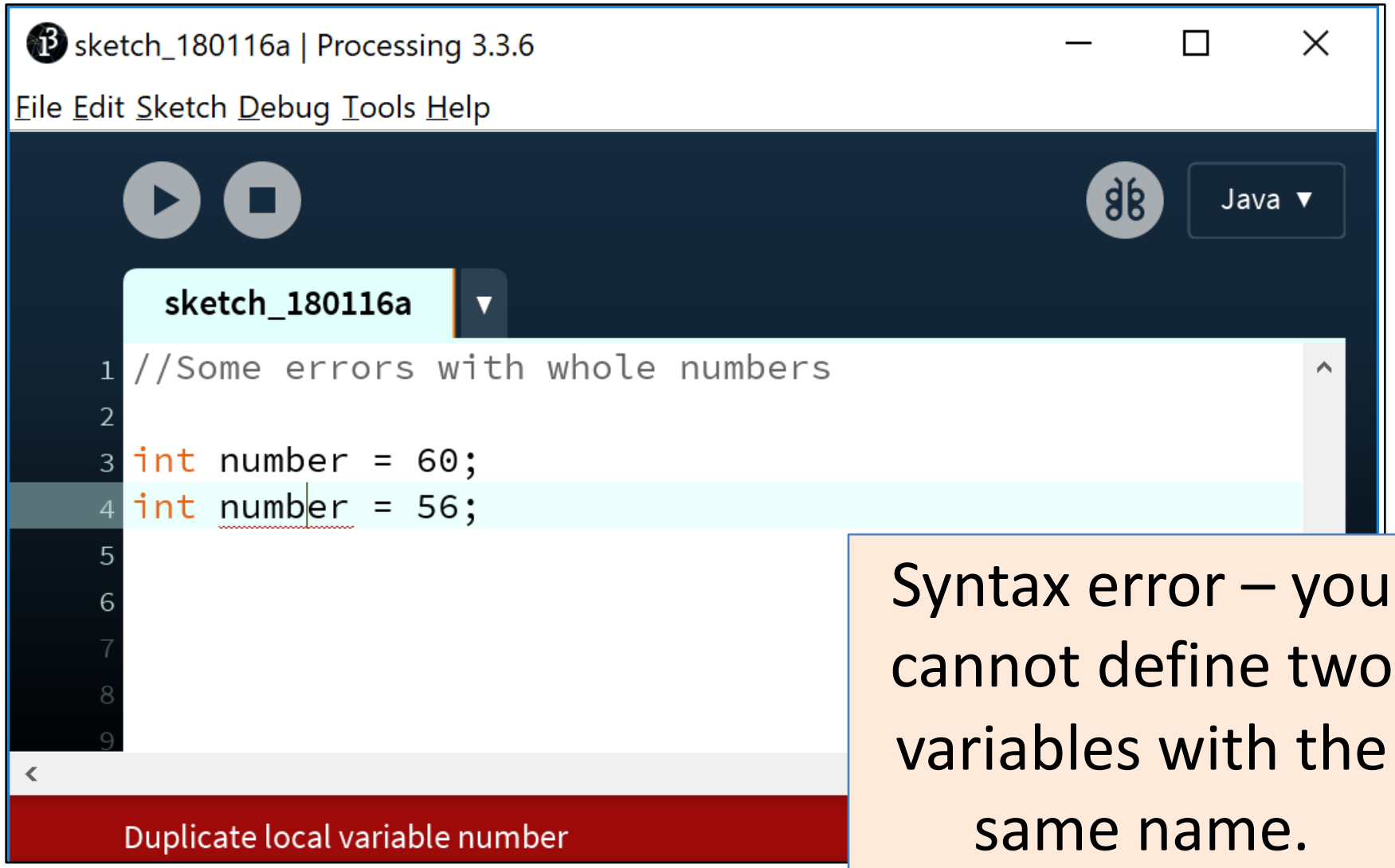
Declaring variables - some errors



Data types are case sensitive.

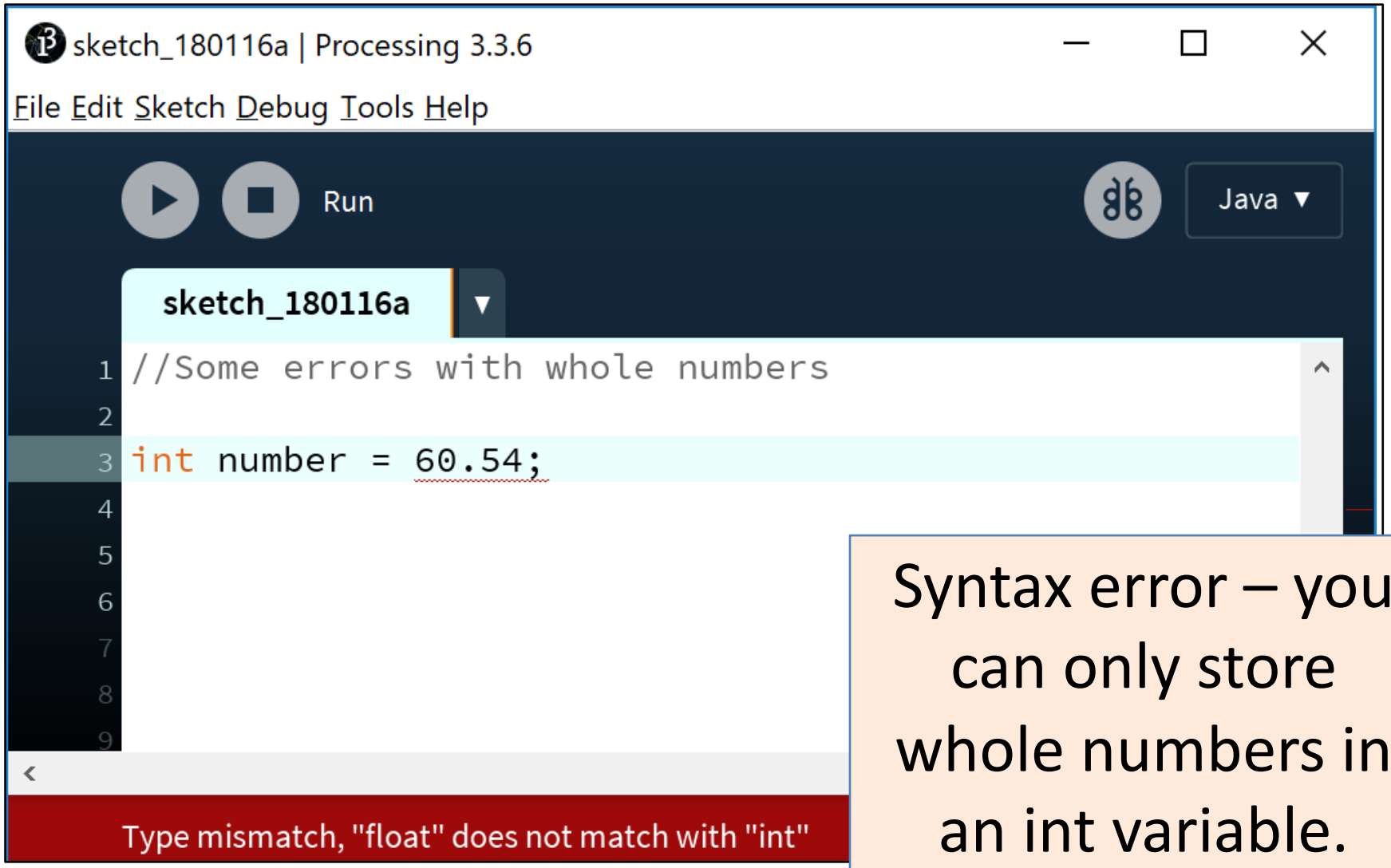
Int is not valid.
int is valid.

Declaring variables - some errors



Syntax error – you cannot define two variables with the same name.

Declaring variables - some errors



Syntax error – you can only store whole numbers in an int variable.

Java's Primitive Data Types: int example

sketch_180116a | Processing 3.3.6

File Edit Sketch Debug Tools Help

sketch_180116a

```
size(600, 400);
background(0);    //black
stroke(153);      //medium gray
strokeWeight(4);

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

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.

Based on the Processing Example: Basics → Data → Variables

Java's Primitive Data Types: int example

sketch_180116a | Processing 3.3.6

File Edit Sketch Debug Tools Help

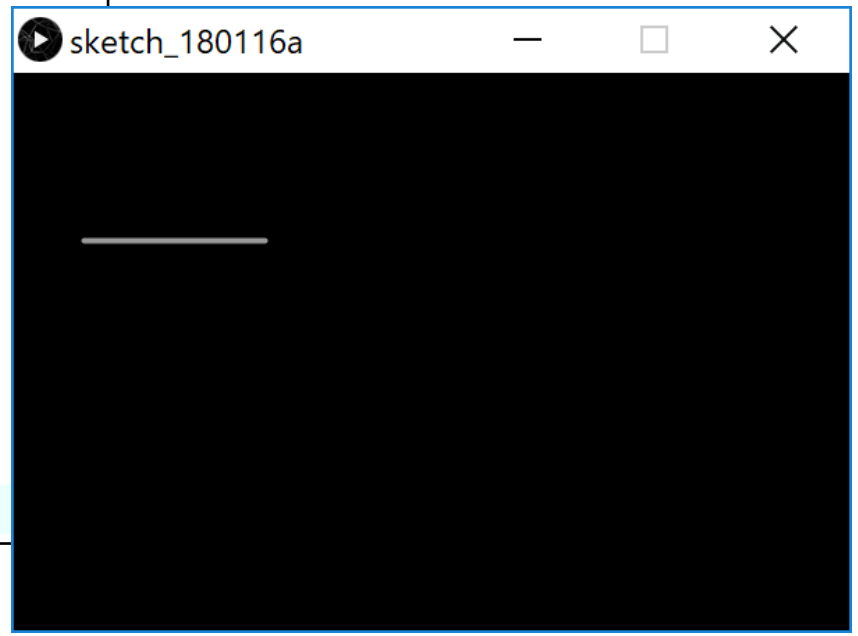
sketch_180116a

```
size(600, 400);
background(0);    //black
stroke(153);      //medium gray
strokeWeight(4);

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

line (a, b, c, b);
```

We can pass the defined variables as values to functions.



Based on the Processing Example: Basics → Data → Variables

Java's Primitive Data Types: int example

sketch_180116a

```
size(600, 400);
background(0);    //black
stroke(153);      //medium gray
strokeWeight(4);

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

line (a, b, c, b);
```

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

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

Based on the Processing Example: Basics → Data → Variables

Java's Primitive Data Types: int example

sketch_180116a | Processing 3.3.6

File Edit Sketch Debug Tools Help

sketch_180116a

```
size(600, 400);
background(0);    //black
stroke(153);      //medium gray
strokeWeight(4);

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

line (a, b, c, b);
```

Type mismatch, "int" does not match with "byte"

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

A: *For **a** and **b** we could have; 50 and 120 fall below the max value of 127. But **c** produces a syntax error; 180 cannot fit into a 127 capacity variable.*

Type	Min value	Max value
byte	-128	127
short	-32,768	32,767

Based on the Processing Example: Basics → Data → Variables

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
    1. Whole numbers.
    2. Decimal numbers.
    3. Others.
5. Arithmetic operators.

Java's Primitive Data Types (decimal numbers)

Type	Byte-size	Minimum value (inclusive)	Maximum value (inclusive)	Typical Use
float	32-bit	*Beyond the scope of this lecture .* *There is also a loss of precision in this data-type that we will cover in later lectures.*		Useful in applications where memory savings apply. Default choice when using **Processing**.
double	64-bit			Default choice when programming **Java apps.**

Java's Primitive Data Types: float example

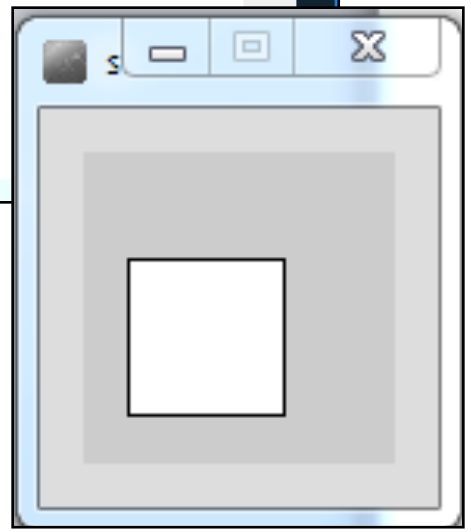
sketch_180116a | Processing 3.3.6

File Edit Sketch Debug Tools Help

Java

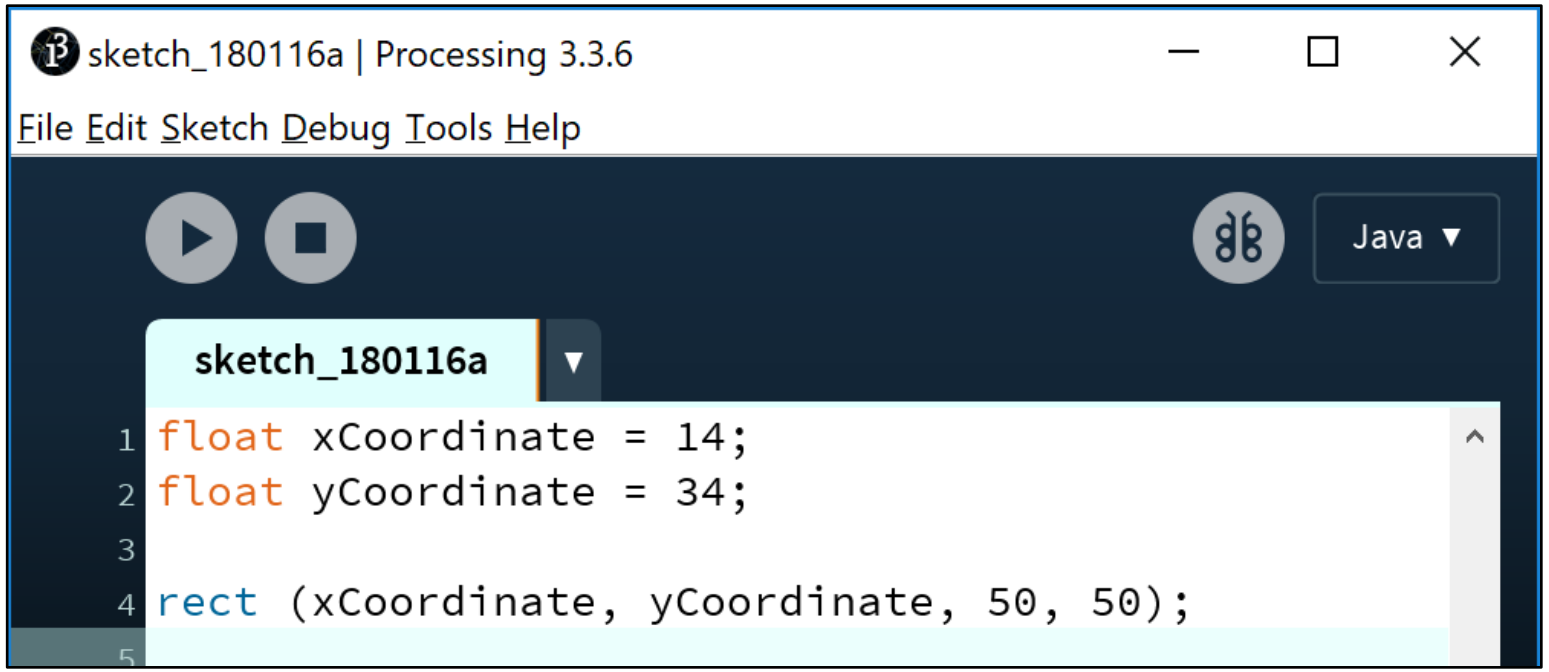
sketch_180116a

```
1 float xCoordinate = 14.65;
2 float yCoordinate = 34.43;
3
4 rect (xCoordinate, yCoordinate, 50, 50);
```



We can pass the defined variables as values to functions.

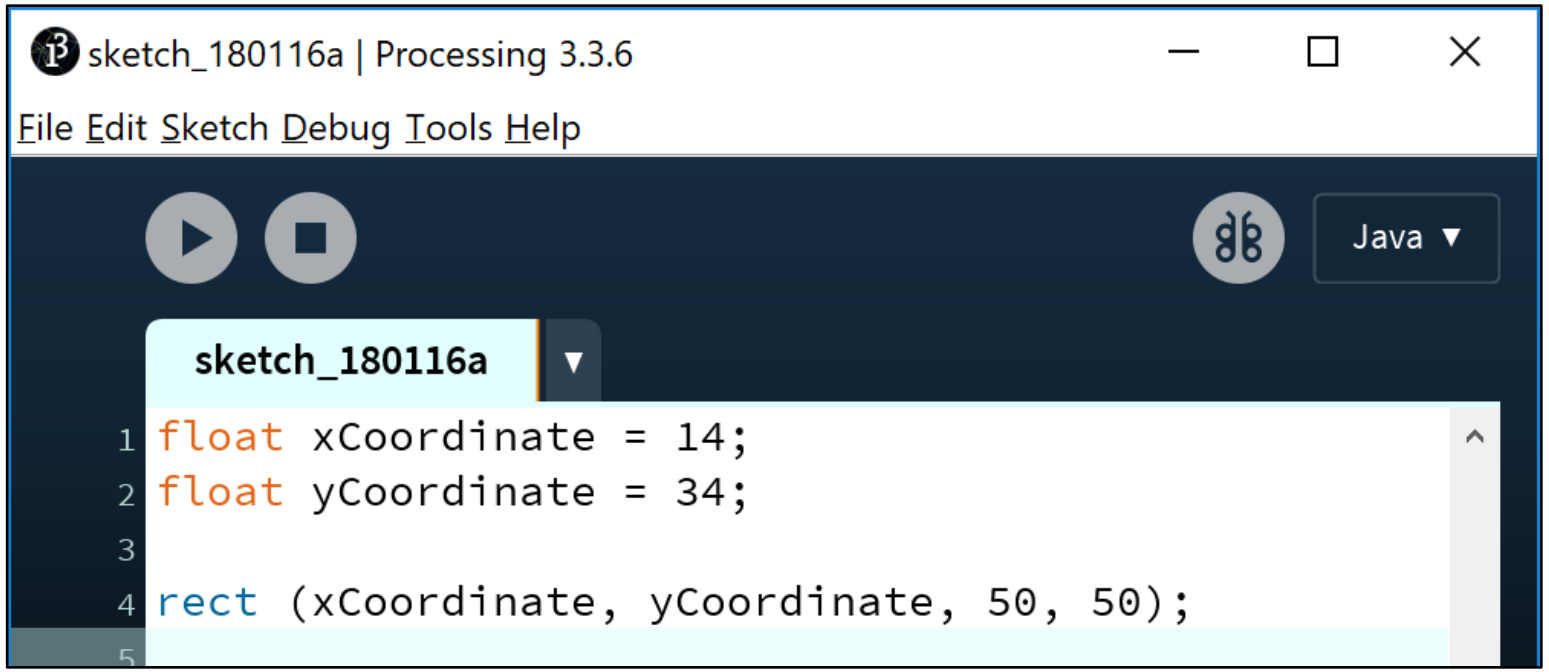
Java's Primitive Data Types: float example



Whole numbers can be placed into a float variable.

Q: Why?

Java's Primitive Data Types: float example



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_180116a | Processing 3.3.6

File Edit Sketch Debug Tools Help

sketch_180116a

```
1 double xCoordinate = 14.65;
2 double yCoordinate = 34.43;
3
4 rect (xCoordinate, yCoordinate, 50, 50);
5
6
```

The function "rect()" expects parameters like: "rect(float, float, float, float)"

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

Q: Why are we getting this syntax error?

Passing variables as arguments: some errors

sketch_180116a | Processing 3.3.6

File Edit Sketch Debug Tools Help

sketch_180116a

```
1 double xCoordinate = 14.65;
2 double yCoordinate = 34.43;
3
4 rect (xCoordinate, yCoordinate, 50, 50);
5
6
```

The function "rect()" expects parameters like: "rect(float, float, float, float)"

A: a double variable has a larger capacity than a float. A float is required in the rect() method. The value stored in the double may not fit into the float.

Passing variables as arguments: some errors

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

Syntax	`rect(a, b, c, d)`	
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 function "rect()" expects parameters like: "rect(float, float, float, float)"

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
    1. Whole numbers.
    2. Decimal numbers.
    3. Others.
5. Arithmetic operators.

Java's Primitive Data Types (others)

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

- We will go into more detail on these two data types in later lectures.

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

Java's Primitive Data Types (default values)

Data Type	Default Value
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
boolean	false

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
    1. Whole numbers.
    2. Decimal numbers.
    3. Others.
5. Arithmetic operators.

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

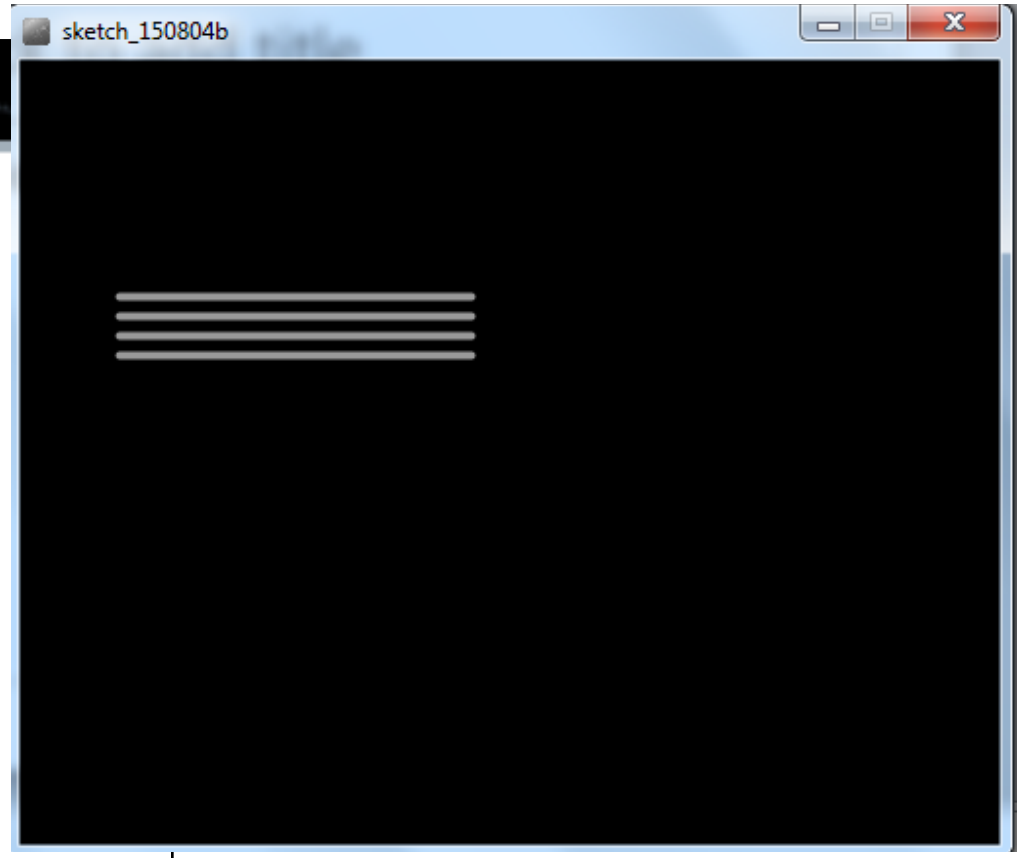
Arithmetic operators: example 1

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);
```



Based on the Processing Example: Basics → Data → Variables

Arithmetic operators: example 2

sketch_150804b

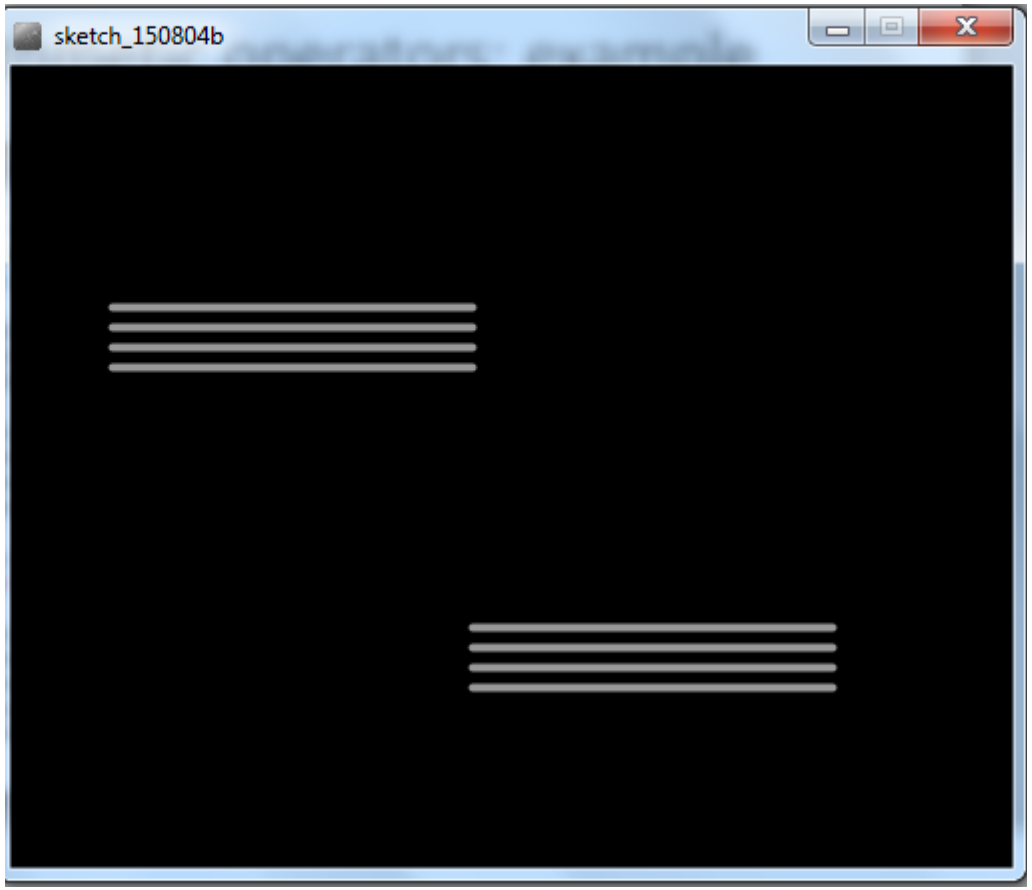
```
size(500, 400);
background(0);
stroke(153);
strokeWeight(4);

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

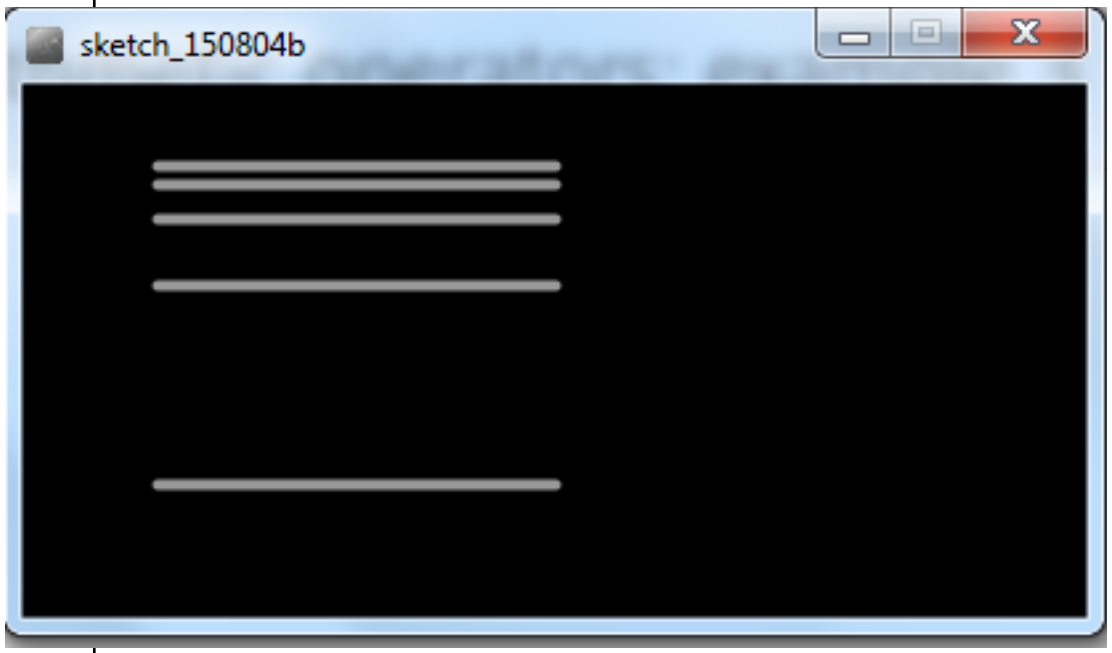
Arithmetic operators: example 3

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);
```



Based on the Processing Example: Basics → Data → Variables

Questions?