



Programming Fundamentals 1

Produced by Mr. Dave Drohan (david.drohan@setu.ie)
Dr. Siobhán Drohan
Ms. Mairead Meagher

Department of Computing & Mathematics
South East Technological University
Waterford, Ireland

setu.ie

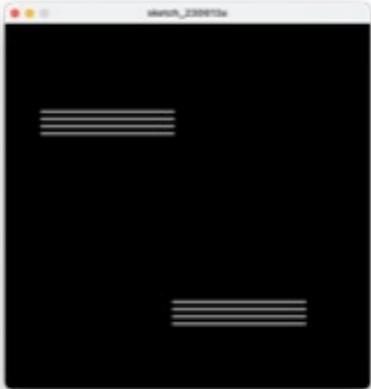




Introduction to Processing

Variables, Data Types &
Arithmetic Operators

Data Types



primitive data types · vars · ops (Vid 17mins)

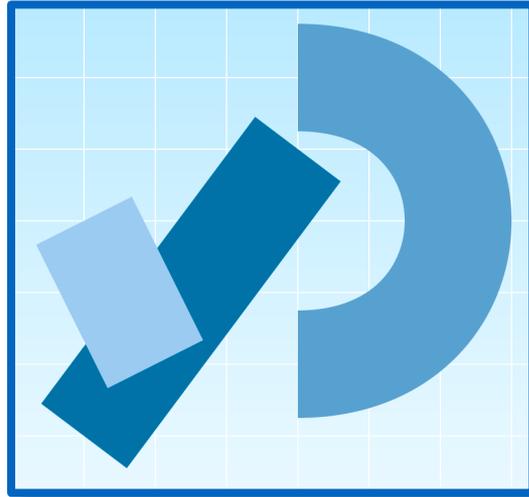


Agenda

- Variables
- Assignment statement
- Data Types
- Java's Primitive Data Types
 - ◆ Whole numbers
 - ◆ Decimal numbers
 - ◆ Others
- Arithmetic operators



Variables





Variables

□ In Programming, **variables**:

- are **defined** (created) 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, "\$", or "_" characters
- ❑ **Can** be any length you choose
- ❑ **Cannot** be a **keyword** or **reserved word** e.g. **int**, **while**, etc.
- ❑ **Cannot** contain white spaces.



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>



Assignment Statement





Assignment Statement

- ❑ Values are stored in variables via **assignment statements**:

Syntax	<code>variable = expression;</code>
Example	<code>diameter = 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



Assignment Statement

- ❑ Values are stored in variables via **assignment statements**:

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

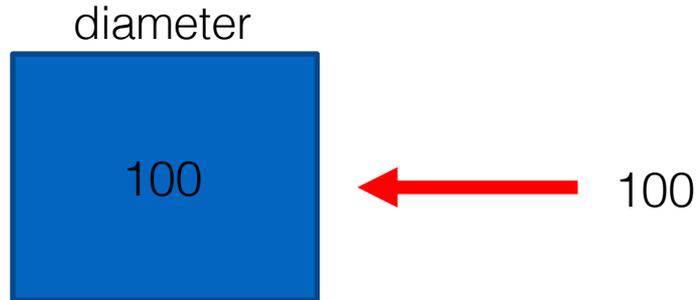
- ❑ A variable stores a **single** value, so any previous value is lost
- ❑ Assignment statements work by taking the value of what appears on the right-hand side of the **operator** and **copying** that value into a variable on the left-hand side



Assignment Statement

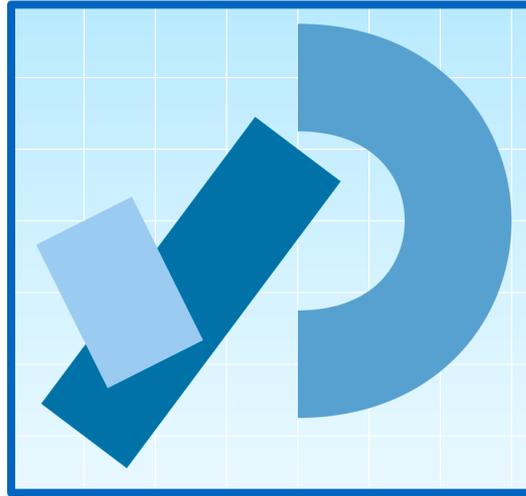
- Values are stored in variables via **assignment statements**:

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





Data Types





Data Types

- ❑ In Java, when we **define** (create) 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.



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



Java's Primitive Data Types





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;



Java's Primitive Data Types

(Whole Numbers)





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



Declaring variables of a specific type

A screenshot of the Processing IDE window titled "sketch_180116a | Processing 3.3.6". The window has a menu bar with "File", "Edit", "Sketch", "Debug", "Tools", and "Help". Below the menu bar are control buttons for play, stop, and a language dropdown menu set to "Java". The main area shows a code editor with the following code:

```
1 byte firstNumber; //declares a variable firstNumber of type byte
2 int secondNumber; //declares a variable secondNumber of type int
3
4 firstNumber = 40; //assign a value of 40 to firstNumber
5 secondNumber = 70; //assign a value of 70 to secondNumber
6
```



Declaring variables of a specific type

```
sketch_180116a | Processing 3.3.6
File Edit Sketch Debug Tools Help

sketch_180116a
1 byte firstNumber; //declares a variable firstNumber of type byte
2 int secondNumber; //declares a variable secondNumber of type int
3
4 firstNumber = 40; //assign a value of 40 to firstNumber
5 secondNumber = 70; //assign a value of 70 to secondNumber
6
```

YELLOW underline – a warning message that 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

sketch_180116a
1 byte firstNumber; //declares a variable firstNumber of type byte
2 int secondNumber; //declares a variable secondNumber of type int
3
4 firstNumber = 40; //assign a value of 40 to firstNumber
5 secondNumber = 70; //assign a value of 70 to secondNumber
6
7 int thirdNumber = 80; //you can declare a variable and assign a
8 //value on one line.
9
```



Declaring variables of a specific type

```
sketch_180116a | Processing 3.3.6
File Edit Sketch Debug Tools Help

sketch_180116a
1 byte firstNumber; //declares a variable firstNumber of type byte
2 int secondNumber; //declares a variable secondNumber of type int
3
4 firstNumber = 40; //assign a value of 40 to firstNumber
5 secondNumber = 70; //assign a value of 70 to secondNumber
6
7 int thirdNumber = 80; //you can declare a variable and assign a
8 //value on one line.
9
10 int x, y, z; //multiple variables of the same type can
11 //be defined on one line.
```



Declaring variables - some errors

The screenshot shows the Processing IDE interface. The title bar reads "sketch_180116a | Processing 3.3.6". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". The toolbar contains a play button, a stop button, a palette icon, and a language dropdown set to "Java". The sketch name "sketch_180116a" is shown in a dropdown menu. The code editor contains the following text:

```
1 //Some errors with whole numbers
2
3 Int number;
4
5
6
7
8
9
```

A red arrow points from the text "Data types are case sensitive." to the word "Int" in the code. Another red arrow points from the text "Int is not valid. int is valid." to the word "Int" in the code. A third red arrow points from the text "Cannot find a class or type named 'Int'" to the error message at the bottom of the IDE.

Cannot find a class or type named "Int"

Data types are case sensitive.

Int is not valid.
int is valid.



Declaring variables - some errors

The screenshot shows the Processing IDE window titled "sketch_180116a | Processing 3.3.6". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". The toolbar contains a play button, a stop button, a palette icon, and a language dropdown set to "Java". The code editor shows the following code:

```
1 //Some errors with whole numbers
2
3 Int number;
4
5
6
7
8
9
```

A red underline is present under the word "Int" on line 3. A red arrow points from a text box to this underline. The text box contains the text: "RED underline - indicates a type of syntax error".

At the bottom of the IDE, a red error message reads: "Cannot find a class or type named 'Int'".

Data types are case sensitive.

Int is not valid.
int is valid.



Declaring variables - some errors

```
sketch_180116a | Processing 3.3.6
File Edit Sketch Debug Tools Help

sketch_180116a
1 //Some errors with whole numbers
2
3 int number = 60;
4 int number = 56;
5
6
7
8
9
```

Duplicate local variable number

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



Declaring variables - some errors

sketch_180116a | Processing 3.3.6

File Edit Sketch Debug Tools Help

Run Java

```
1 //Some errors with whole numbers
2
3 int number = 60.54;
4
5
6
7
8
9
```

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

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
1 size(600, 400);
2 background(0); //black
3 stroke(153); //medium gray
4 strokeWeight(4);
5
6 int a = 50;
7 int b = 120;
8 int c = 180;
9
```

In this example, we have defined three variables (a, b and c) that:

- can hold whole numbers (`int`).
- are set with a starting value.

Based on the Processing Example: Basics → Data → Variables



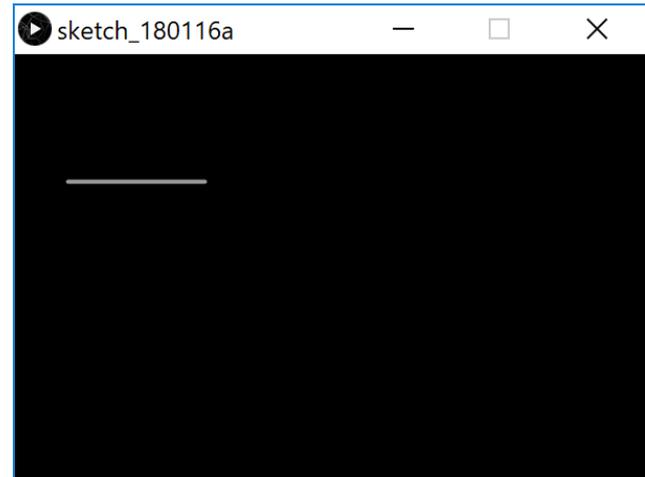
Java's Primitive Data Types: `int` example

```
sketch_180116a | Processing 3.3.6
File Edit Sketch Debug Tools Help

▶ ◻

sketch_180116a ▼
1 size(600, 400);
2 background(0); //black
3 stroke(153); //medium gray
4 strokeWeight(4);
5
6 int a = 50;
7 int b = 120;
8 int c = 180;
9
10 line (a, b, c, b);
11
```

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);
```

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

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

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
1 size(600, 400);
2 background(0); //black
3 stroke(153); //medium gray
4 strokeWeight(4);
5
6 byte a = 50;
7 byte b = 120;
8 byte c = 180;
9
10 line (a, b, c, b);
11

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

Type	Min value	Max value
<code>byte</code>	-128	127
<code>short</code>	-32,768	32,767

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.

Based on the Processing Example: Basics → Data → Variables



Java's Primitive Data Types

(Decimal Numbers)



Java's Primitive Data Types (decimal numbers)



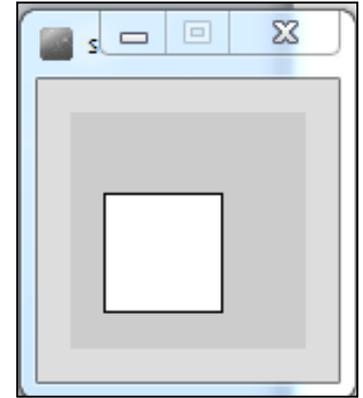
Type	Byte-size	Minimum value (inclusive)	Maximum value (inclusive)	Typical Use
float	32-bit	<i>Beyond the scope of this lecture .</i> <i>There is also a loss of precision in this data-type that we will cover in later lectures.</i>		Useful in applications where memory savings apply. Default choice when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

Java's Primitive Data Types (decimal numbers)



```
sketch_180116a | Processing 3.3.6
File Edit Sketch Debug Tools Help

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

```
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);
```

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

Q: Why?



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);
```

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
7
<
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
7
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?

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



Java's Primitive Data Types

(Others)





Java's Primitive Data Types (**others**)

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

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

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

Java's Primitive Data Types (default values)

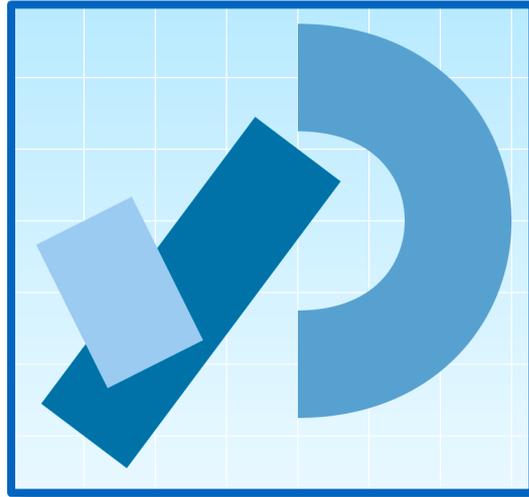


Data Type	Default Value
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
boolean	false

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



Arithmetic operators





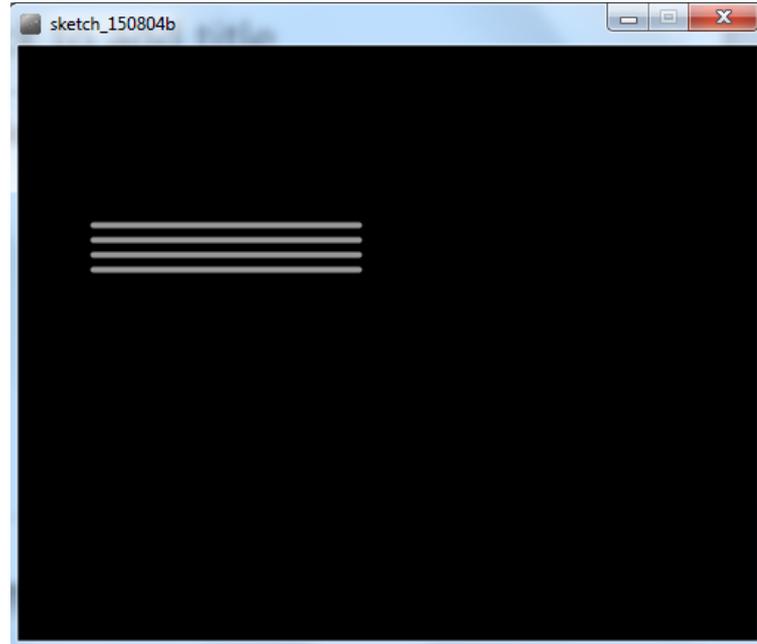
Arithmetic Operators

Arithmetic Operator	Explanation	Example(s)
$+$	Addition	$6 + 2$ <code>amountOwed + 10</code>
$-$	Subtraction	$6 - 2$ <code>amountOwed - 10</code>
$*$	Multiplication	$6 * 2$ <code>amountOwed * 10</code>
$/$	Division	$6 / 2$ <code>amountOwed / 10</code>



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

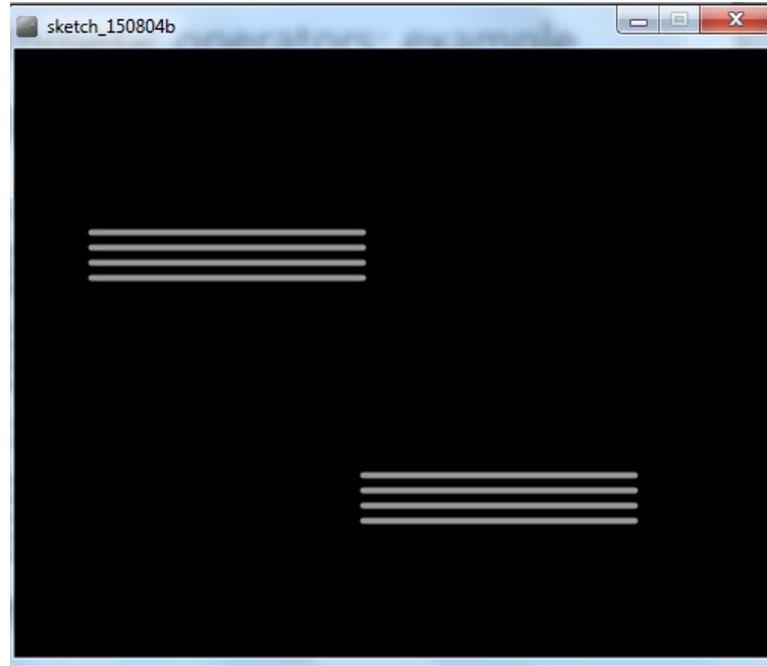
```
sketch_150804b
size(500, 400);
background(0);
stroke(153);
strokeWeight(4);

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

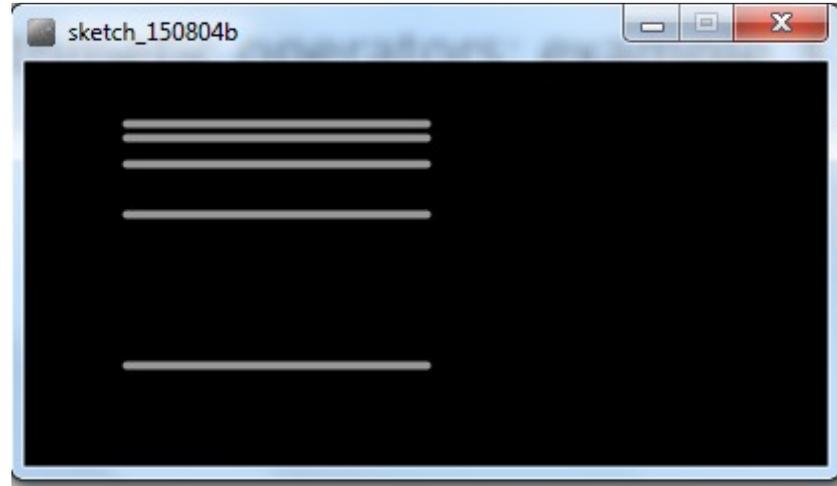


Arithmetic operators: example 3

```
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?



