

An Introduction to Processing

Variables, Data Types & Arithmetic Operators

Lecturer: Caio Fonseca



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Introduction

- Data types are used in programs to create named locations in the computer's memory.
- These will contain values while a program is running.
- This process is known as **declaring**.
- These named locations are called **variables**.

Topics List

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

Topics List

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- 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.

Variable Names...

- Must not be a **keyword or reserved word** e.g. int, void, while, etc.
- Cannot contain white spaces.
- Variables must not have the same name as a variables already declared (must be unique).
- Variables must be declared before they can be used.

Variable Names should be Carefully Chosen

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

Topics List

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

Assignment Statement

- Values are stored in variables via assignment statements:

Syntax	variable = expression;
Example	yourAge = 25;

- 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

- Assignments allow values to be put into variables.
- They are written in Java with the use of the equality symbol (=).
- This symbol is known as **the assignment operator**.
- Simple assignments take the following form:
variableName = value;

```
age = 21;  
weeklySalary = 565.62;
```

Topics List

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

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.

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 in the next module.

Topics List

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- 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 age;  
boolean flag;  
float number;  
double number1;
```

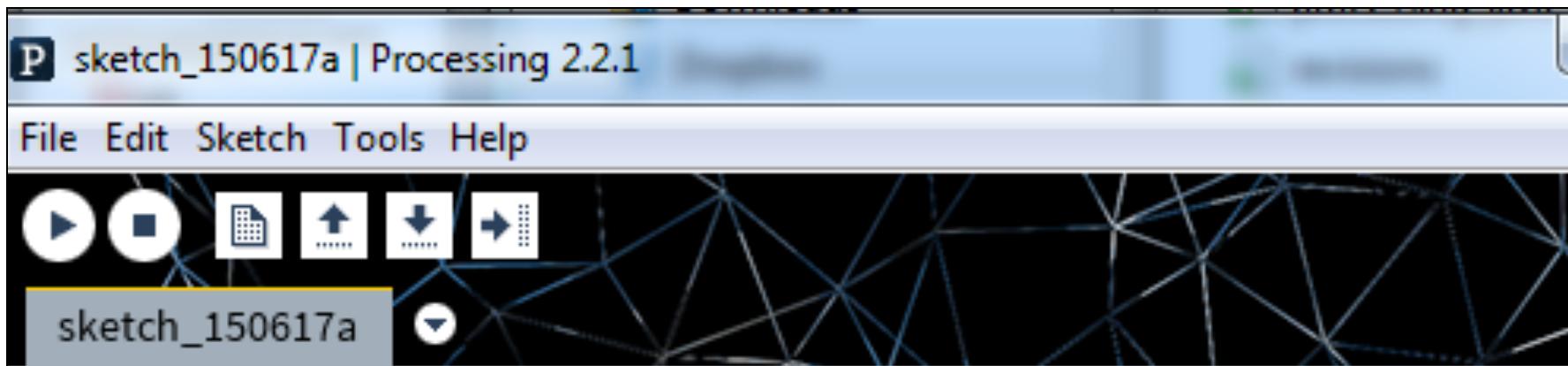
Topics List

- Variables.
- Assignment statement.
- Data Types.
- Java's Primitive Data Types
 - Whole numbers.
 - Decimal numbers.
 - Others.
- 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.

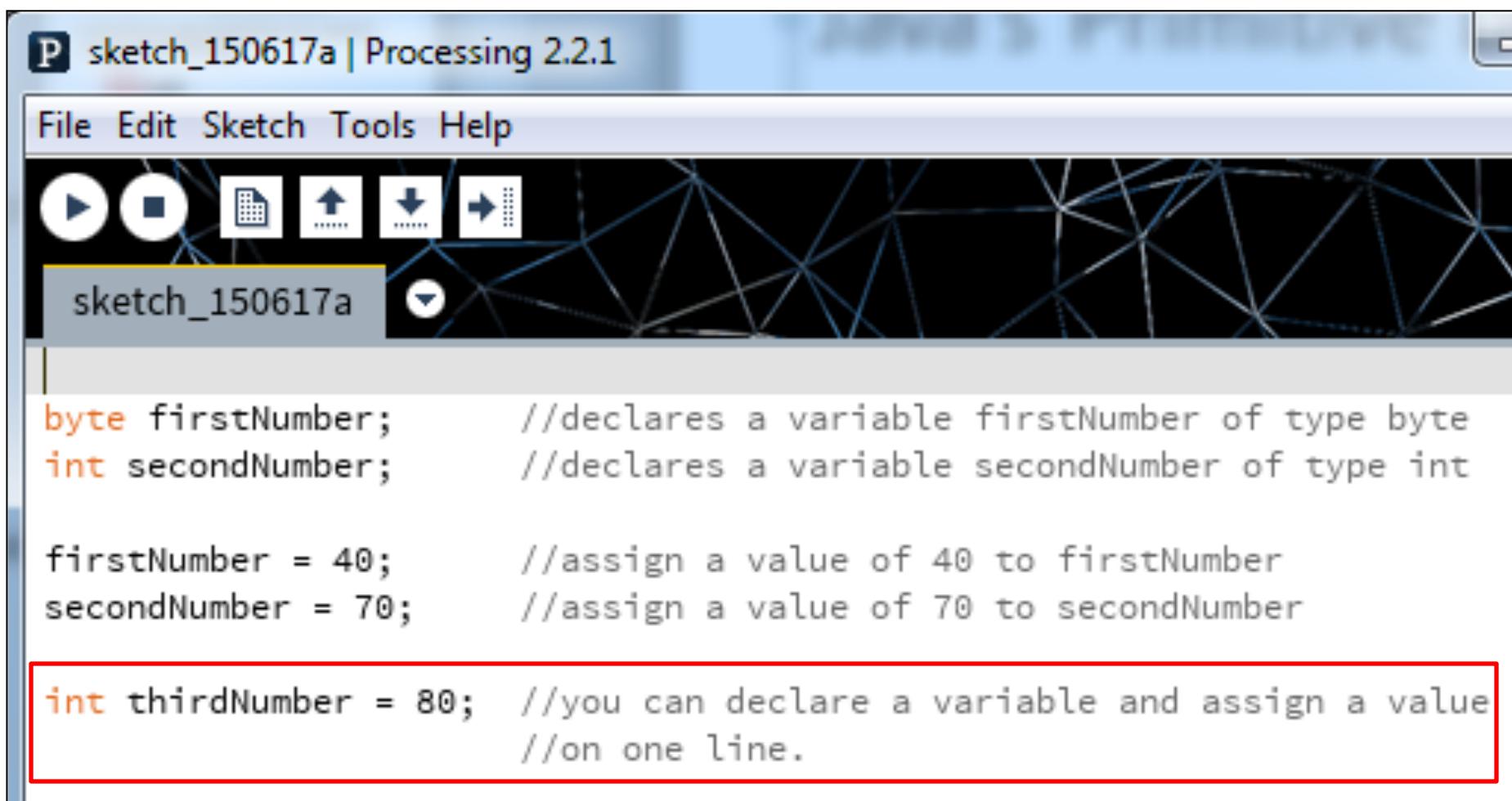
Declaring Variables of a Specific Type



```
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
```

Declaring Variables of a Specific Type



The screenshot shows the Processing 2.2.1 software interface. The title bar reads "P sketch_150617a | Processing 2.2.1". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu is a toolbar with various icons. The main workspace displays a sketch titled "sketch_150617a" which contains a complex geometric pattern of blue lines on a black background. The code area contains the following pseudocode:

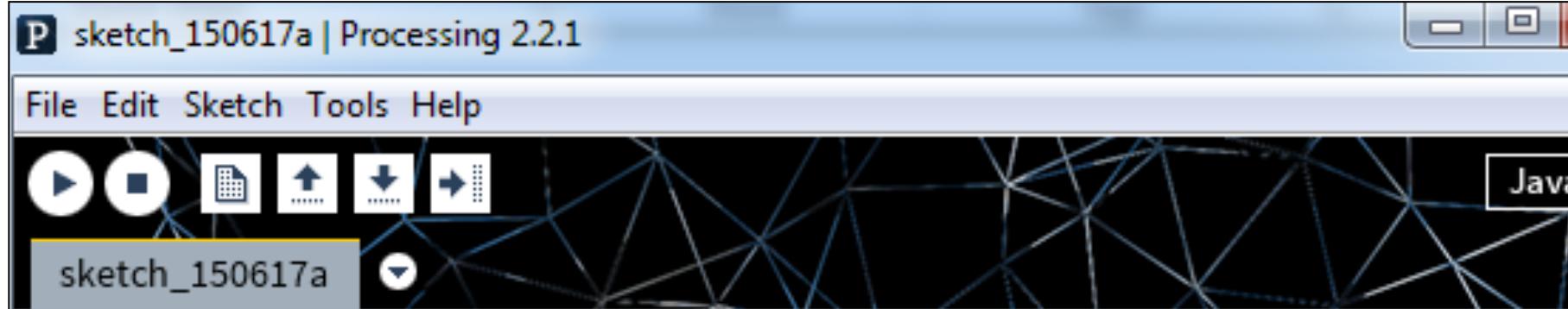
```
byte firstNumber;           //declares a variable firstNumber of type byte
int secondNumber;           //declares a variable secondNumber of type int

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.
```

The last line of code, "int thirdNumber = 80; //you can declare a variable and assign a value //on one line.", is highlighted with a red rectangular border.

Declaring Variables of a Specific Type



```
P sketch_150617a | Processing 2.2.1
File Edit Sketch Tools Help
sketch_150617a Java
```

```
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

The screenshot shows a Java code editor window titled "sketch_150618a". The code is as follows:

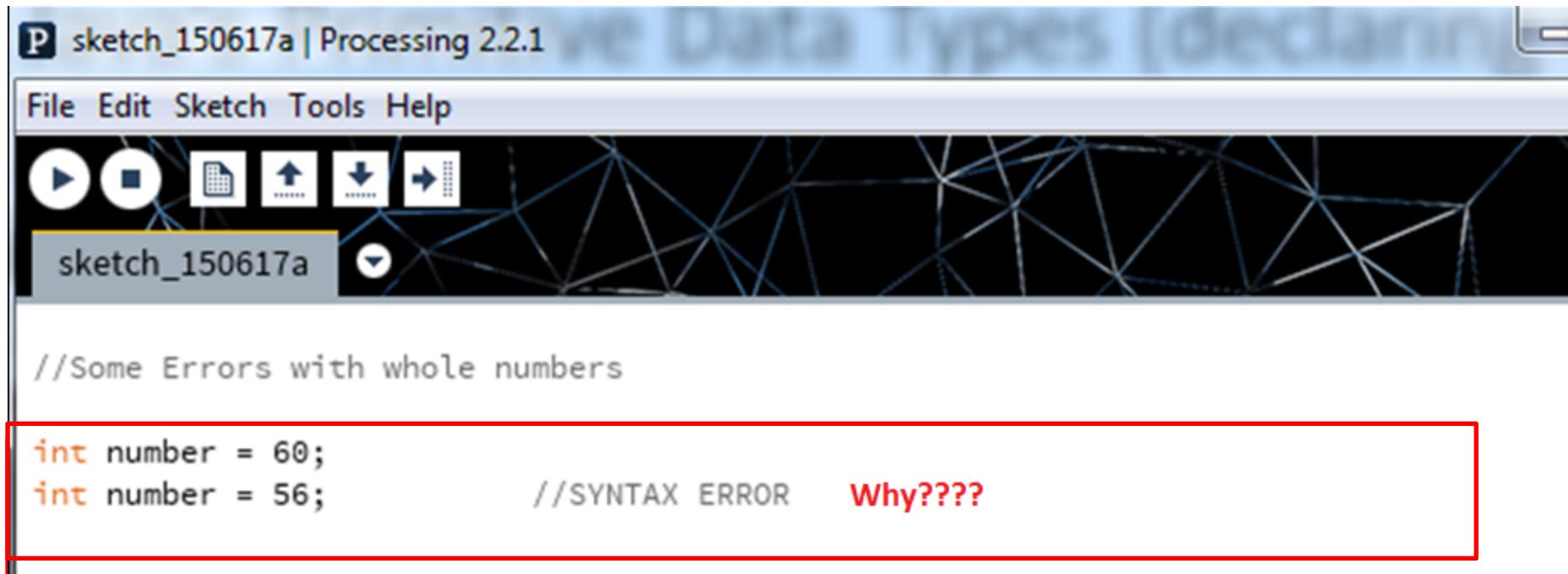
```
//Some Errors with whole numbers  
Int number = 60;
```

The word "Int" is highlighted in yellow, indicating a syntax error. A red error message at the bottom of the editor states: "Cannot find a class or type named 'Int'".

Data types are case sensitive.

Int is not valid.
int is valid.

Declaring Variables - Some Errors



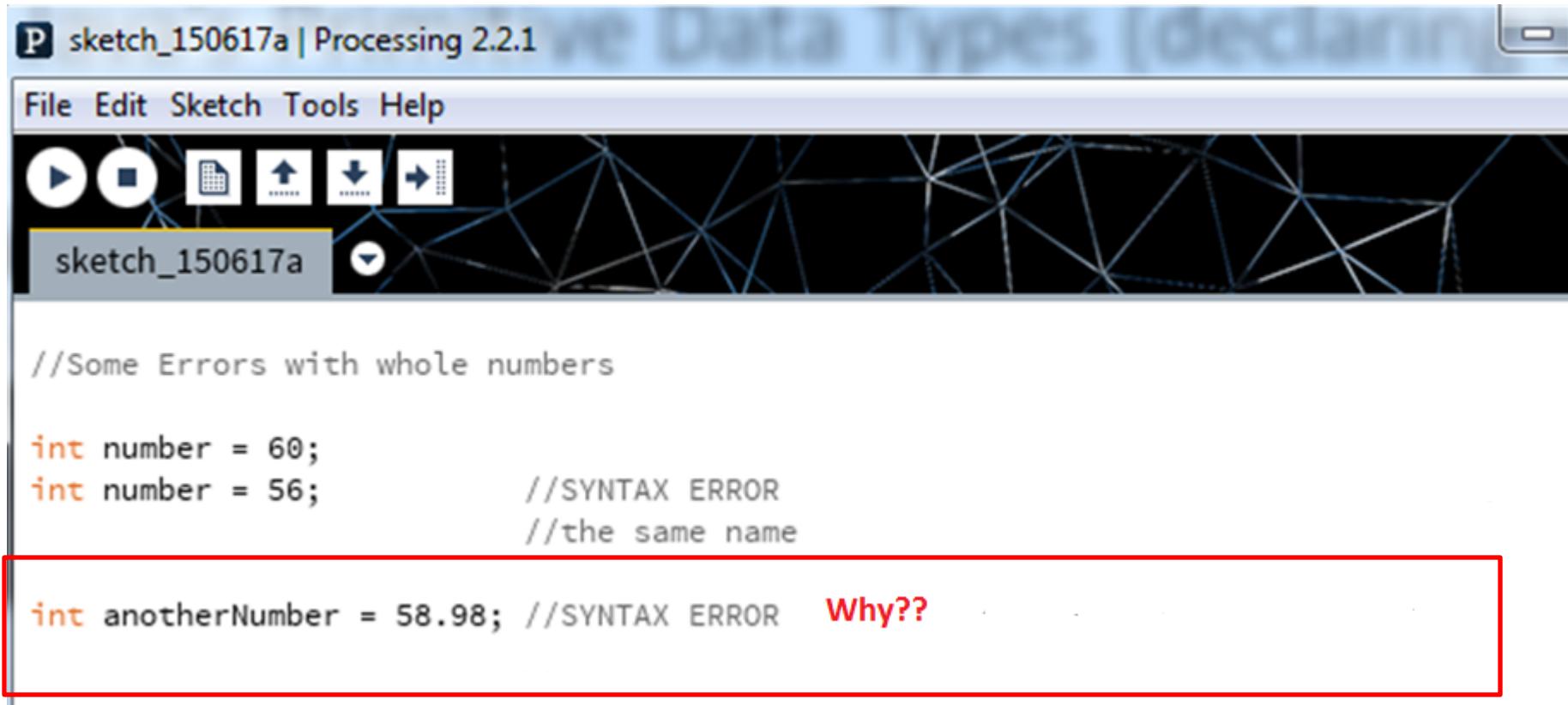
The screenshot shows the Processing 2.2.1 IDE interface. The title bar says "sketch_150617a | Processing 2.2.1". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with various icons. The main workspace contains the following code:

```
//Some Errors with whole numbers

int number = 60;
int number = 56;          //SYNTAX ERROR  Why????
```

A red rectangular box highlights the second line of code: "int number = 56;". To its right, the text "Why?????" is written in red, indicating a question about the syntax error.

Declaring Variables - Some Errors



The screenshot shows the Processing 2.2.1 software interface. The title bar reads "sketch_150617a | Processing 2.2.1". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with various icons. The main area displays a complex geometric pattern of blue lines on a black background. The code editor window contains the following code:

```
//Some Errors with whole numbers

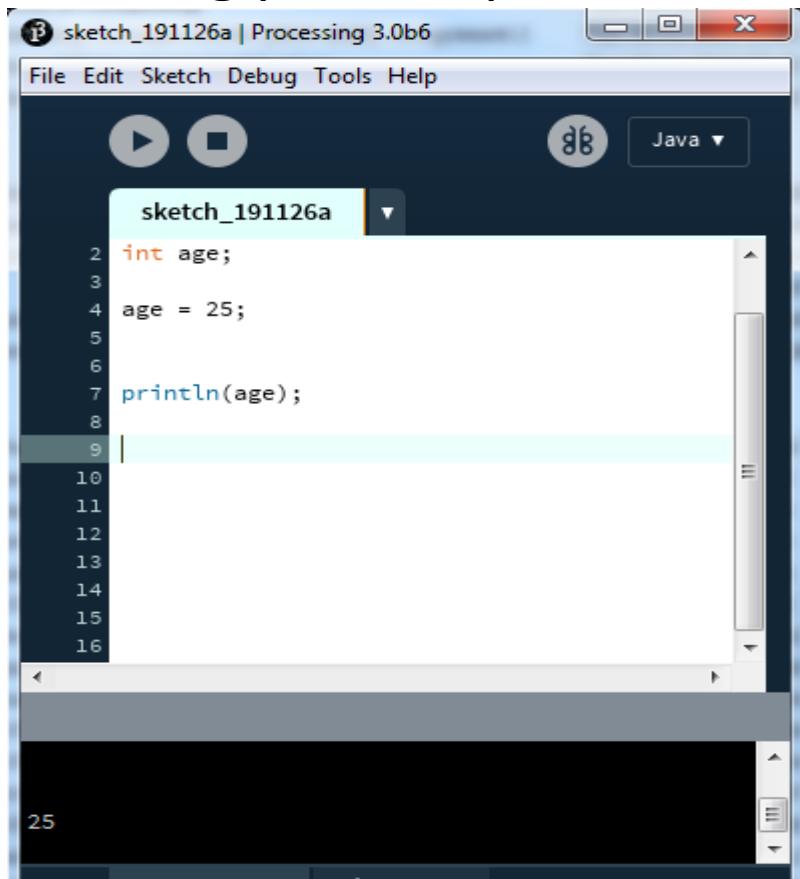
int number = 60;
int number = 56;          //SYNTAX ERROR
                        //the same name

int anotherNumber = 58.98; //SYNTAX ERROR Why??
```

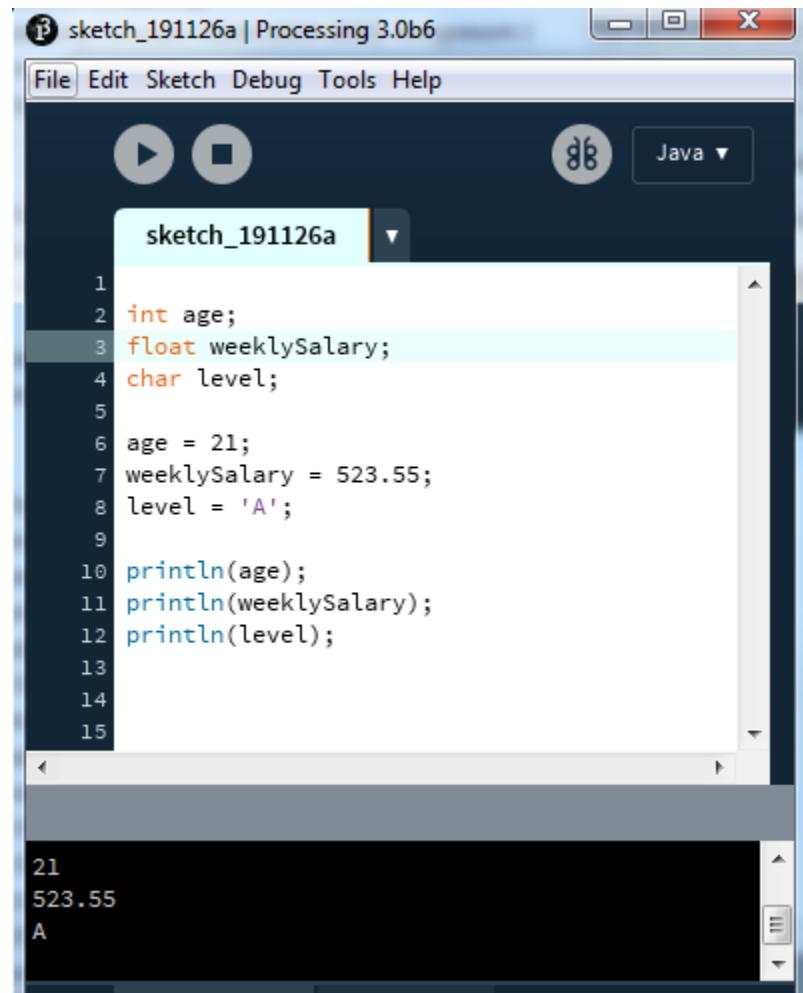
A red rectangular box highlights the last line of code: `int anotherNumber = 58.98; //SYNTAX ERROR Why??`. The word "Why??" is written in red text to the right of the error message.

Printing Variables

- You can print out the value of a variable to the screen using `print` or `println`.



```
sketch_191126a | Processing 3.0b6
File Edit Sketch Debug Tools Help
Java ▾
sketch_191126a
1
2 int age;
3
4 age = 25;
5
6
7 println(age);
8
9
10
11
12
13
14
15
16
25
```

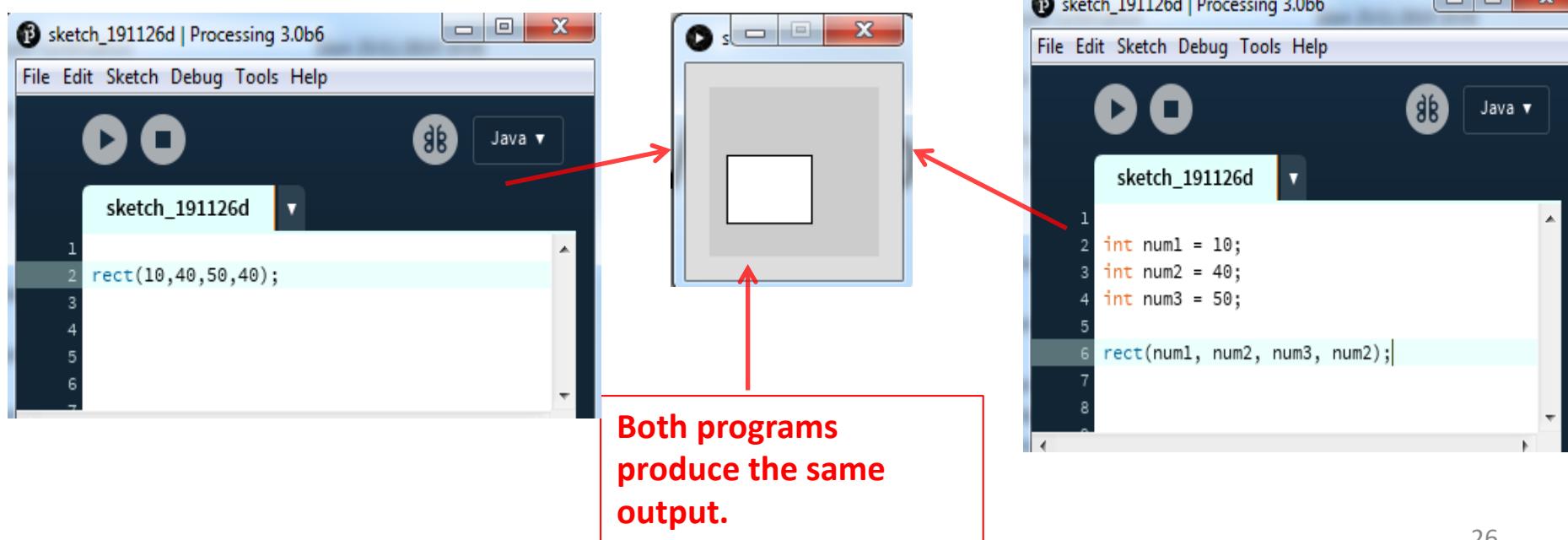


```
sketch_191126a | Processing 3.0b6
File Edit Sketch Debug Tools Help
Java ▾
sketch_191126a
1
2 int age;
3 float weeklySalary;
4 char level;
5
6 age = 21;
7 weeklySalary = 523.55;
8 level = 'A';
9
10 println(age);
11 println(weeklySalary);
12 println(level);
13
14
15
21
523.55
A
```

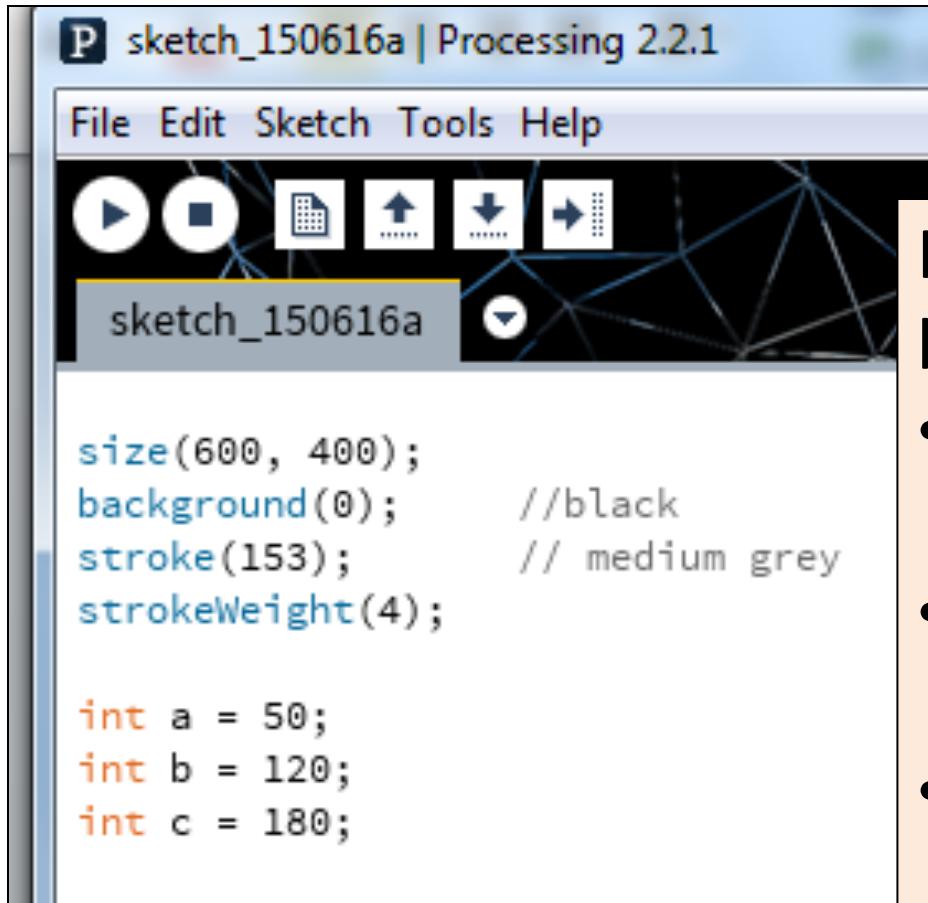
Java's Primitive Data Types: int

Example 1

- We can now use variables instead of hard values into our drawing.
- For example when drawing a rectangle we have used `rect(10,40,50,40)` we can now use variables.



Java's Primitive Data Types: int Example 2



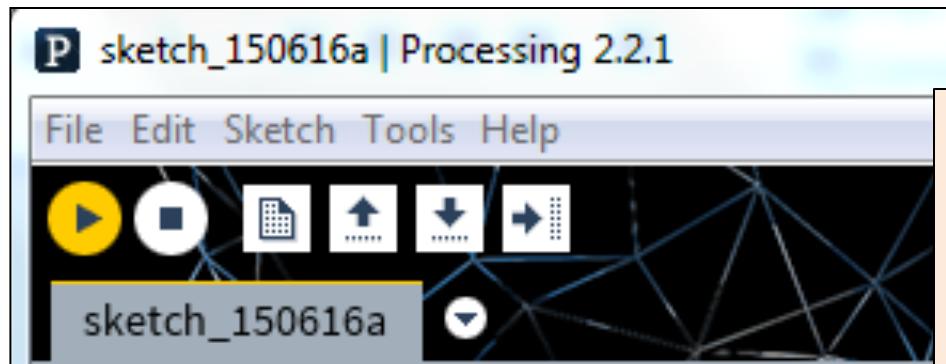
```
P sketch_150616a | Processing 2.2.1
File Edit Sketch Tools Help
sketch_150616a
size(600, 400);
background(0);      //black
stroke(153);        // medium grey
strokeWeight(4);

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.

Java's Primitive Data Types: int Example 2

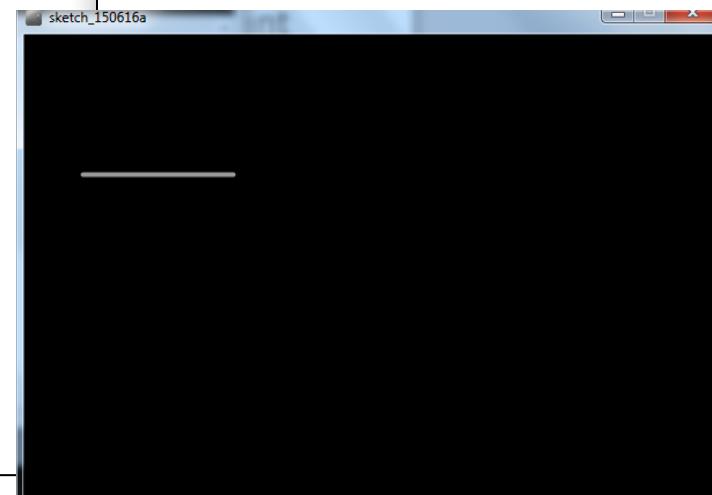


We can pass the defined variables as values to functions.

```
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);
```



Java's Primitive Data Types: int Example 2

```
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

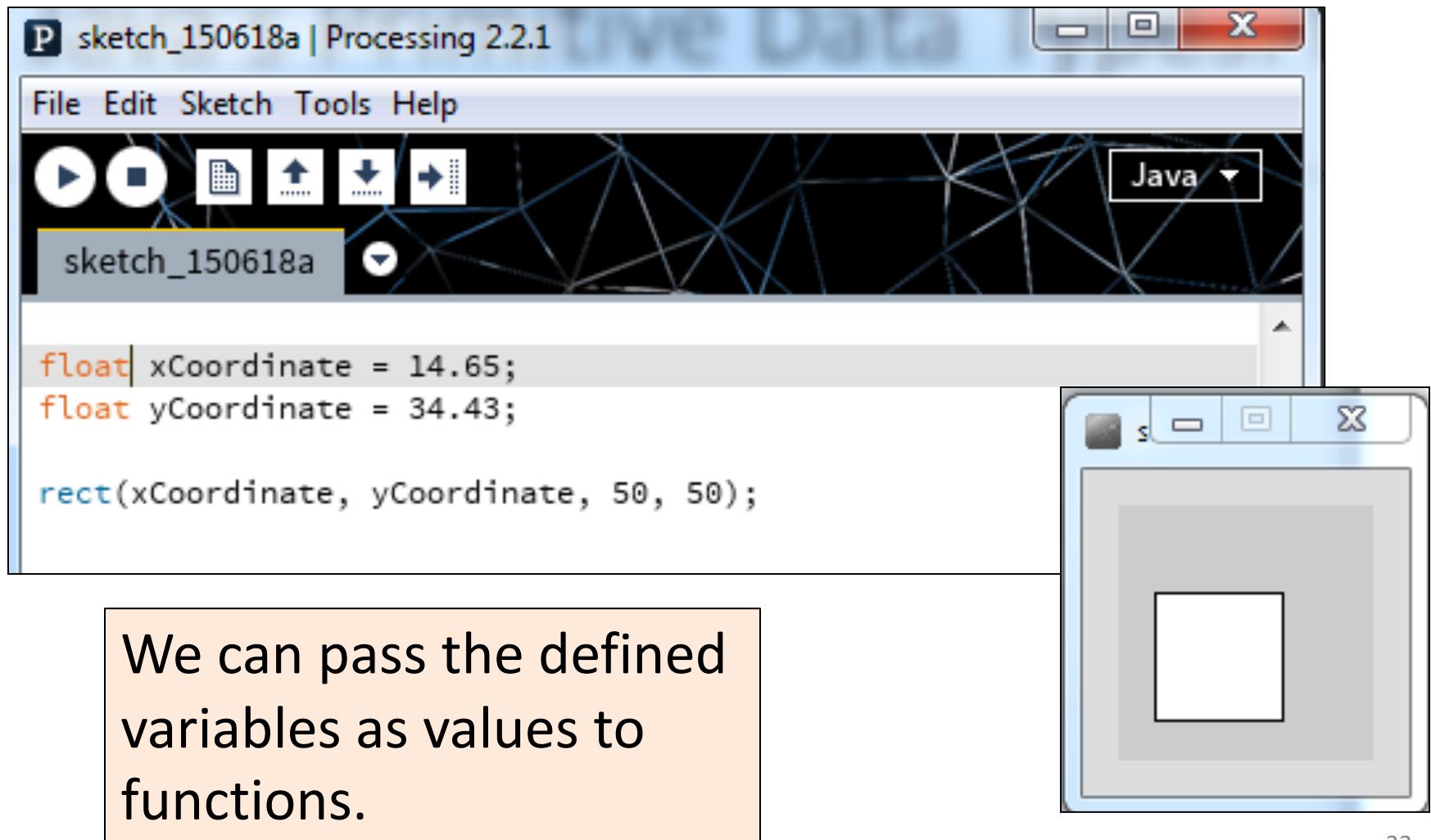
Topics List

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

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: float example



The image shows the Processing 2.2.1 IDE interface. The title bar says "sketch_150618a | Processing 2.2.1". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for play, stop, save, and zoom. A dropdown menu shows "Java". The sketch name "sketch_150618a" is highlighted. The code editor contains the following Java code:

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

To the right, a preview window shows a gray background with a single white square centered at the coordinates (14.65, 34.43).

We can pass the defined variables as values to functions.

Java's Primitive Data Types: float example

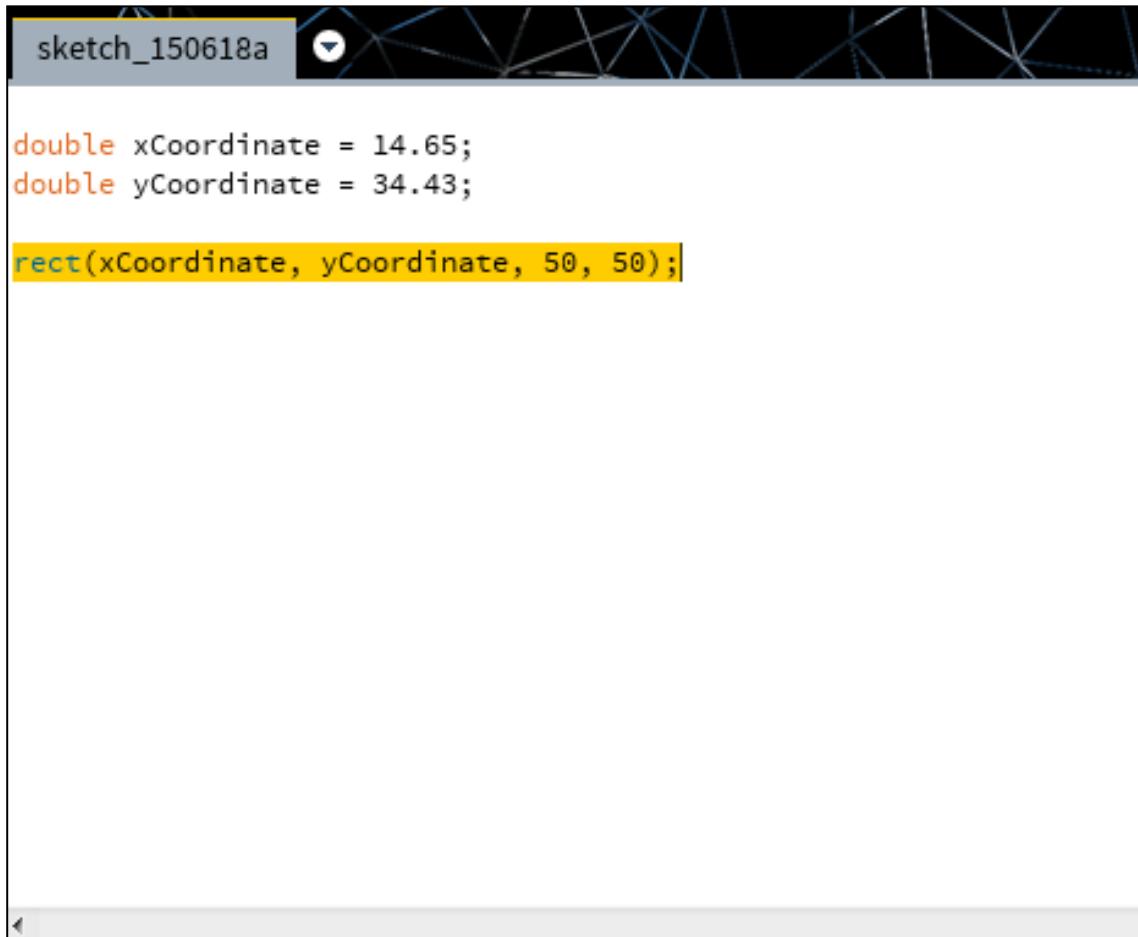


```
float xCoordinate = 14;  
float yCoordinate = 34;  
  
rect(xCoordinate, yCoordinate, 50, 50);
```

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

Q: Why?

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

```
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 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.

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.

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

Java's Primitive Data Types (others)

- char is the data type to hold any single character from the keyboard
 - i.e. a or B or € or * or 1 or 2
- We declare this type as follows:
 - char grade;
- We assign a value to a char using single quotes
 - grade = 'A';
 - grade = 'a';
 - grade = '1';

Java's Primitive Data Types (others)

- boolean is used to represent a value of true or false.
- We declare this type as follows:
 - boolean `isTrue`;
- We assign a value to a boolean as follows:
 - `isTrue = true;`
 - `isTrue = false;`
- We often use boolean values to test certain conditions (we will look at this in greater detail in later sections).

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

Topics List

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

Variables and Arithmetic Operators

The arithmetic operators of Java	
Operation	Java operator
addition	+
subtraction	-
multiplication	*
division	/
remainder	%



Where did we see this??

We will look at this in more detail later.

Variables and Arithmetic Operators

- These are used when you want to do some calculation to number or to your variables. These are very similar to how you calculate in maths.

```
int num1;  
num1 = 25 + 25;  
println(num1);
```

```
sketch_191126d | Processing 3.0b6  
File Edit Sketch Debug Tools Help  
sketch_191126d  
1  
2 int num1 = 10;  
3  
4 num1 = 25 + 25;  
5  
6 println(num1);  
7  
8
```

- Terms on the right-hand side of assignment operators (like $25 + 25$) are referred to as **expressions**.

Variables and Arithmetic Operators

- When doing any calculations in Java you must follow the order of evaluation.
 - Brackets
 - Multiplication
 - Division
 - Addition
 - Subtraction
- **(Brackets and My Dear Aunt Sally)(BOMDAS)**

Variables and Arithmetic Operators

- Example 1

```
int answer;
```

```
answer = 20 + 20 / 10 + 5;
```

- Example 2

```
int solution ;
```

```
solution = (20 + 20 )/ 10 + 5;
```

- Example 3

```
int solution1 ;
```

```
solution1 = (20 + 20 )/ (10 + 5);
```

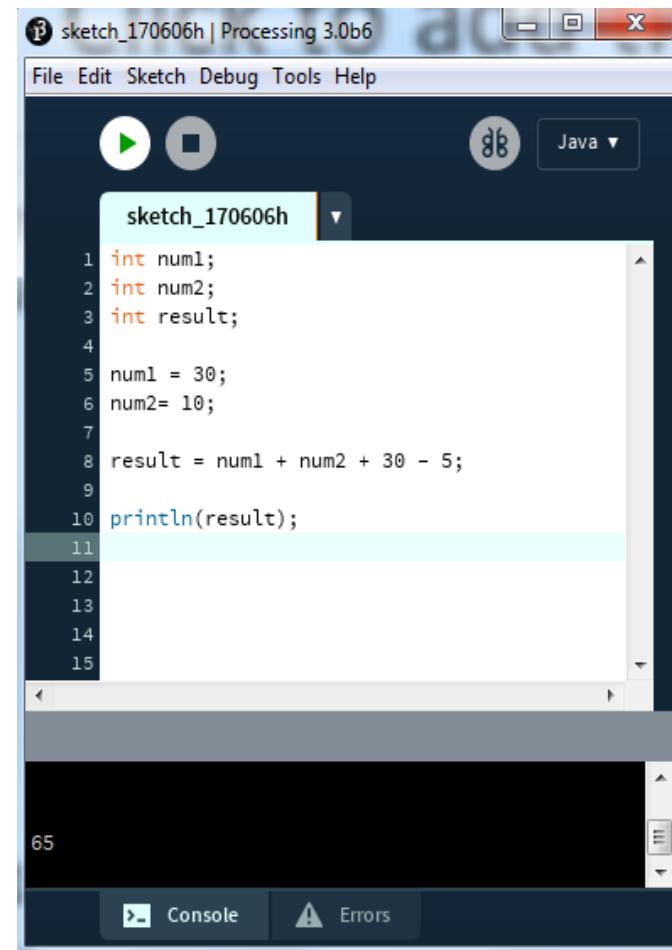
```
sketch_170606g | Processing 3.0b6
File Edit Sketch Debug Tools Help
Java ▾

sketch_170606g
1 int answer;
2 answer = 20 + 20 / 10 + 5;
3 println(answer);
4
5 int solution ;
6 solution = (20 + 20 )/ 10 + 5;
7 println(solution);
8
9 int solution1 ;
10 solution1 = (20 + 20 )/ (10 + 5);
11 println(solution1);
12
13
14
15
```

Variables and Arithmetic Operators

- The expression on the right-hand side of an assignment statement can itself contain variable names;

```
int num1;  
int num2;  
int result;  
num1 = 30;  
num2= 10;  
result = num1 + num2 + 30 - 5;  
println(result);
```

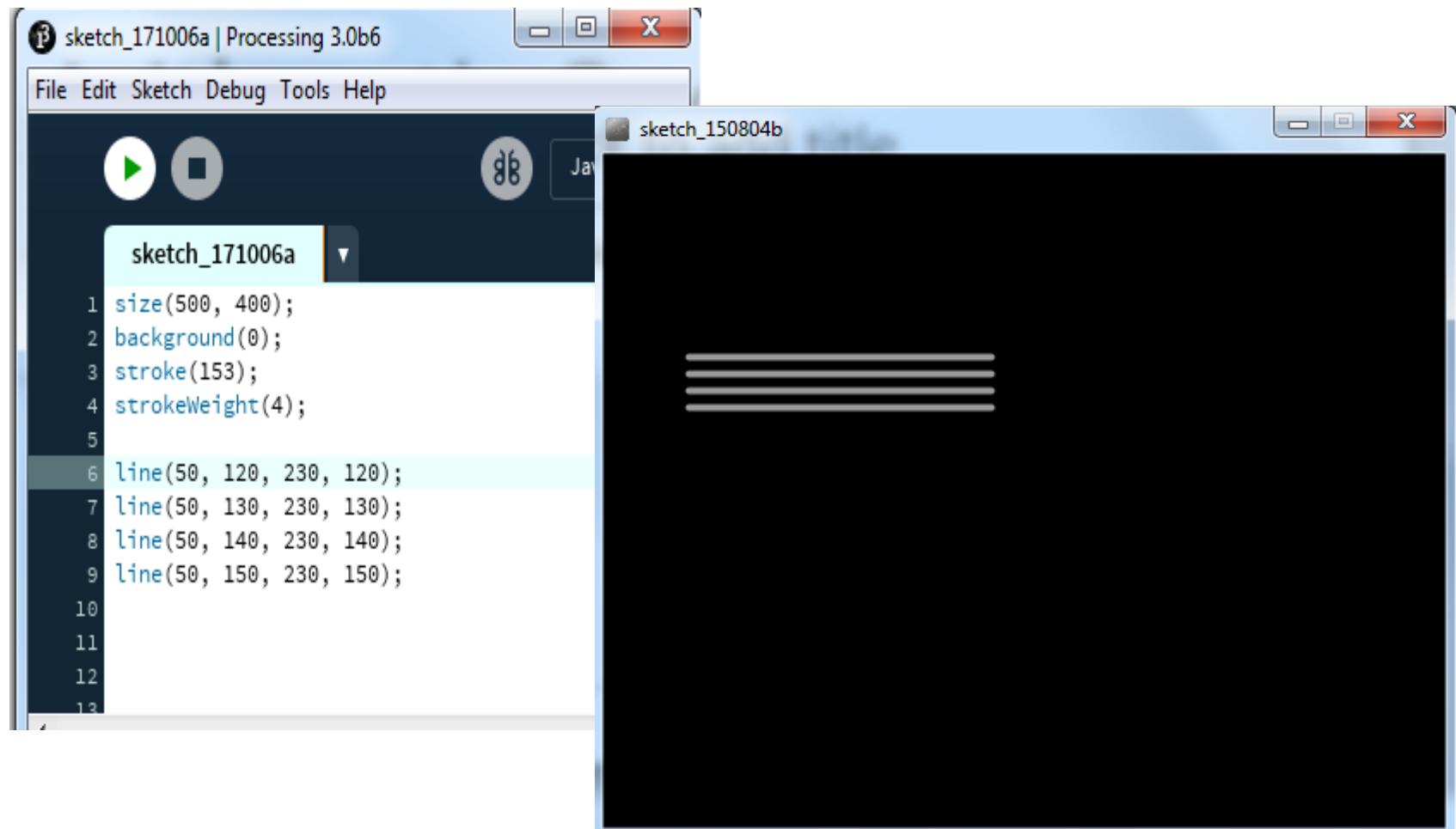


A screenshot of the Processing 3.0b6 IDE. The window title is "sketch_170606h | Processing 3.0b6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. The toolbar has a play button, a square button, a magnifying glass icon, and a Java dropdown. The code editor shows the following code:

```
1 int num1;  
2 int num2;  
3 int result;  
4  
5 num1 = 30;  
6 num2= 10;  
7  
8 result = num1 + num2 + 30 - 5;  
9  
10 println(result);  
11  
12  
13  
14  
15
```

The code editor has a light blue background with line numbers on the left. The code is highlighted in different colors: int, num1, num2, result, and println are in blue; = and + are in orange; and the numerical values 30, 10, and 5 are in black. The bottom of the screen shows the Processing interface with tabs for Console and Errors, and a status bar with the number 65.

Arithmetic Operators: Example 1

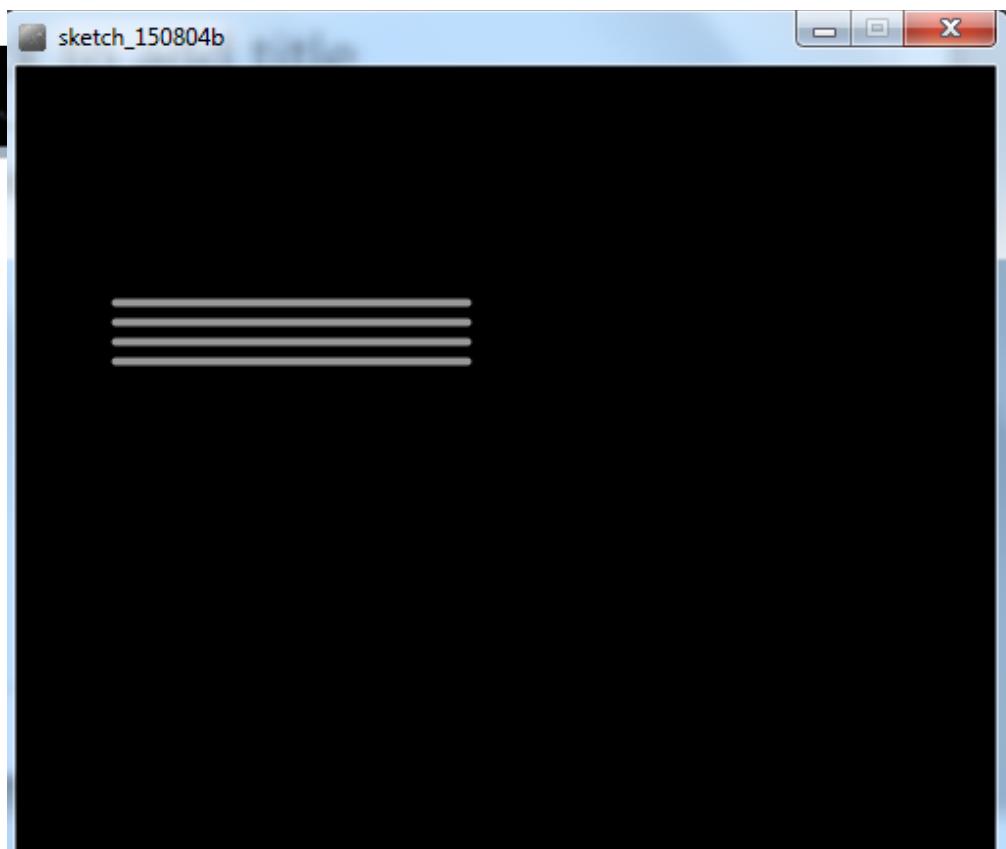


Arithmetic Operators: Example 1

```
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);
```



Arithmetic Operators: Example 2

```
sketch_150804b

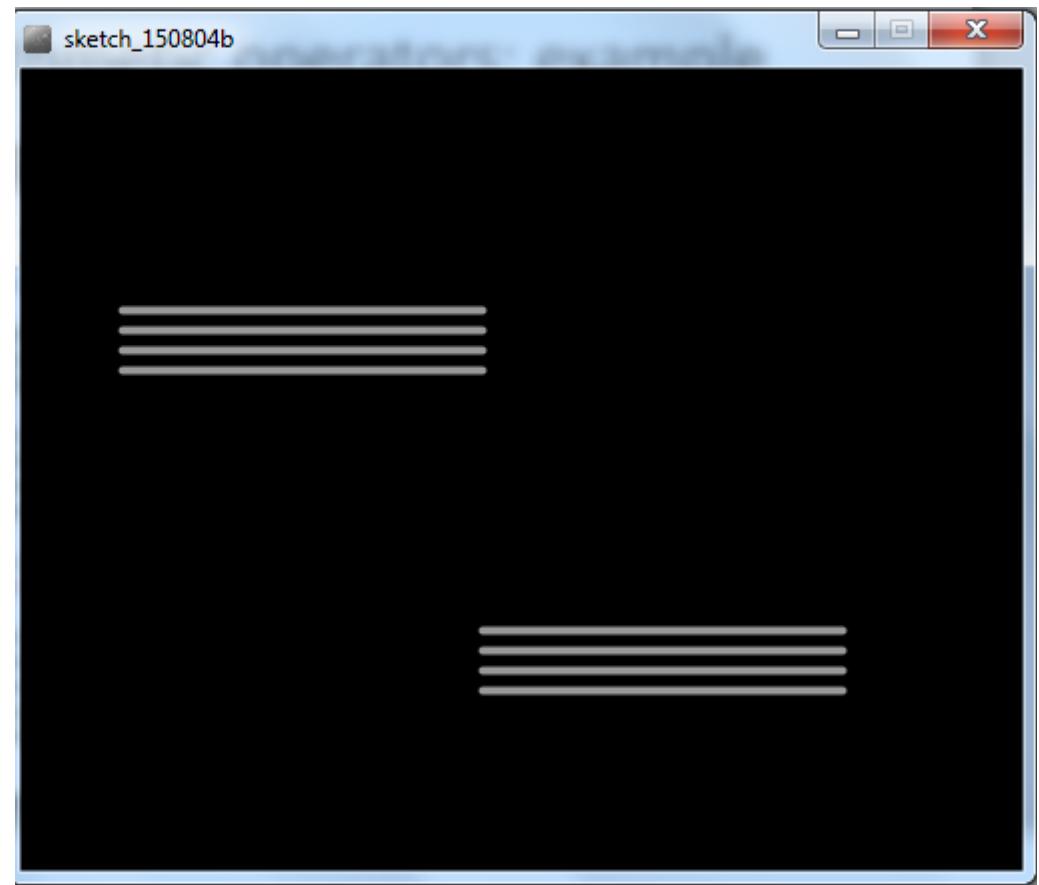
size(500, 400);
background(0);
stroke(153);
strokeWeight(4);

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

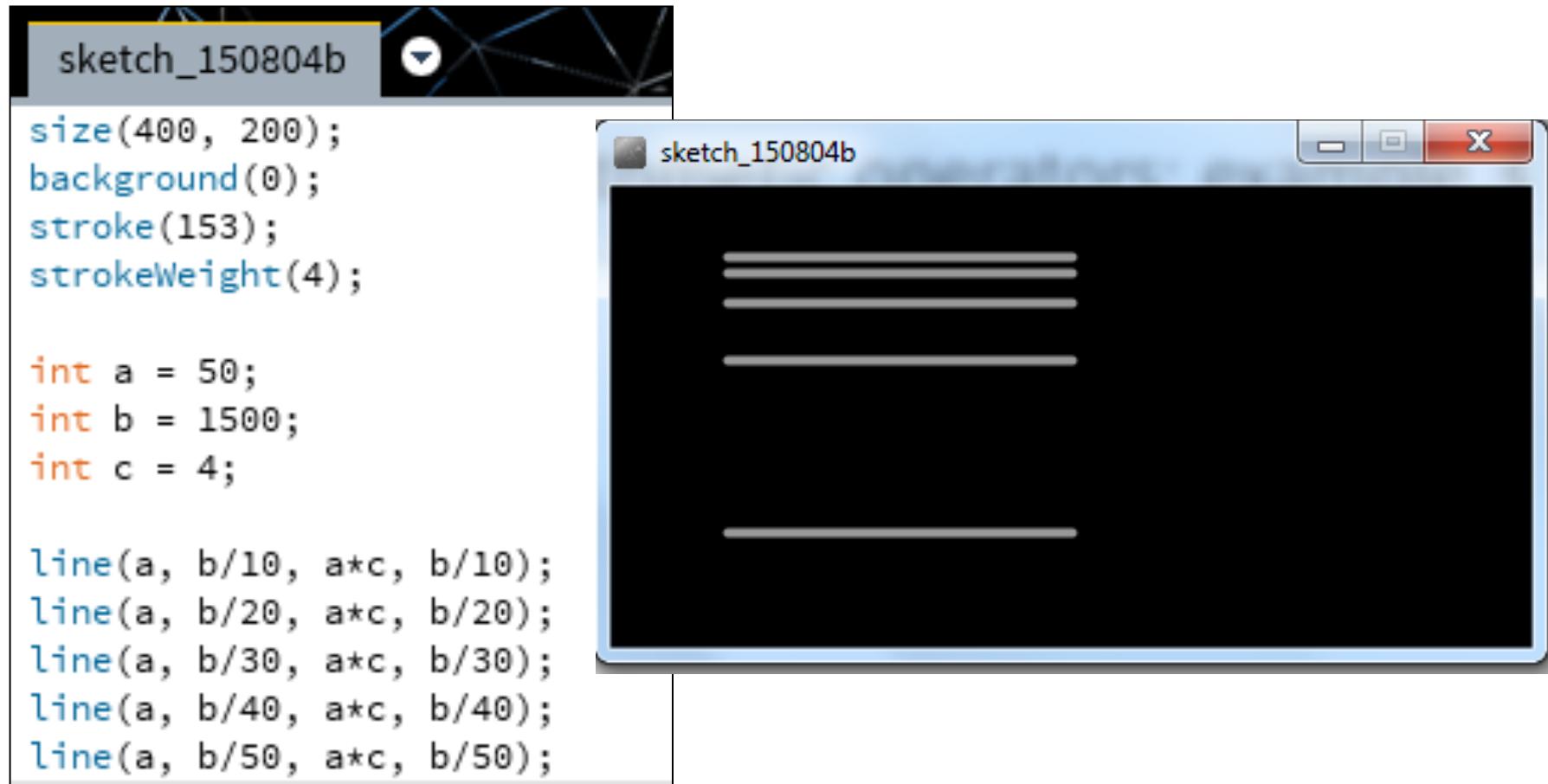
line(a, b, a+c, b);
line(a, b+10, a+c, b+10);
line(a, b+20, a+c, b+20);
line(a, b+30, a+c, b+30);

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);
```



Arithmetic Operators: Example 3



The image shows the Processing IDE interface. On the left is the code editor window titled "sketch_150804b" containing the following Pseudocode:

```
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);
```

On the right is the "sketch_150804b" window showing the output of the code. It displays five horizontal white lines of increasing length from top to bottom, corresponding to the values of $b/10$, $b/20$, $b/30$, $b/40$, and $b/50$.

Built in Variables

- Processing has built in variables for storing commonly used data.
- Width and height of the display window are stored in variable called **width** and **height**.
- If the program does not include size () width and height both are automatically set to 100.

Built in Variables

The image shows two side-by-side Processing IDE windows. Both windows have a title bar "sketch_170608a | Processing 3.0b6" and a menu bar "File Edit Sketch Debug Tools Help". The left window has the following code:

```
1 println(width);
2 println(height);
```

The right window has the following code:

```
1 size(500, 200);
2
3 println(width);
4 println(height);
```

Below each window is its corresponding output window. The left output window shows:

100
100

The right output window shows:

500
200

A red callout box with a blue border and red text points from the left output window to the left code window. It contains the text: "100 is printed for the variable width and height". Another red callout box with a blue border and red text points from the right output window to the right code window. It contains the text: "500 is printed for the variable width 200 for height". A third red callout box with a blue border and red text points from the right output window to the right code window. It contains the text: "The size of the display window is clearly different in each program run". Red arrows also point from the text in the callout boxes to their respective windows.

A Special Shorthand

- If a variable `x` has been declared as an `int`, then the instruction for incrementing `x` (adding 1) would be:

`x = x + 1;`

- This is so common that there is a special shorthand for this instruction:

`x++;`

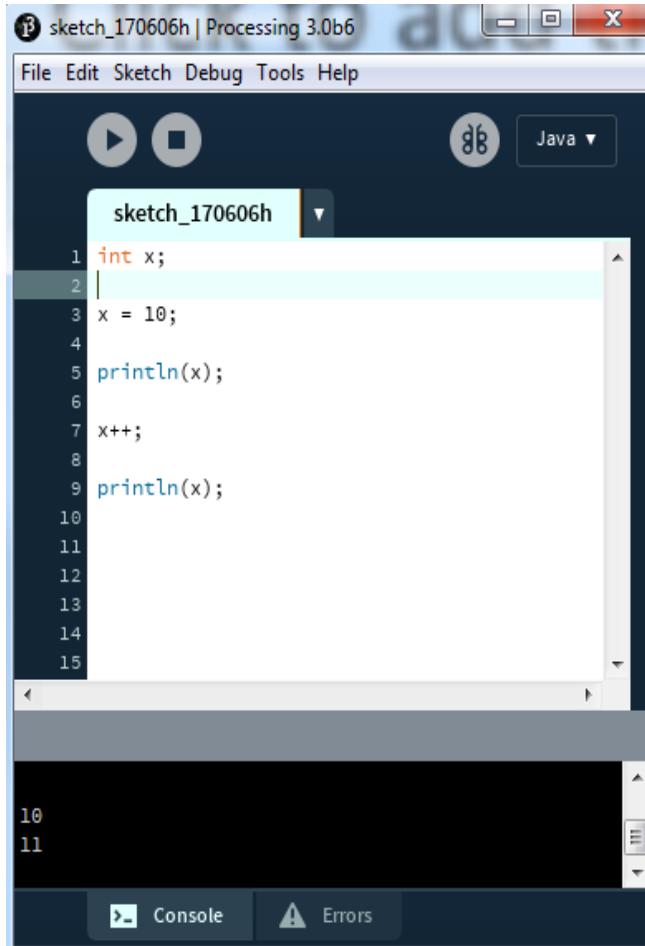
- The '`++`' is therefore known as the increment operator;
- Similarly there exists a decrement operator, '`--`':

`x--;`

is shorthand for:

`x = x - 1;`

A Special Shorthand



sketch_170606h | Processing 3.0b6

File Edit Sketch Debug Tools Help

Java

sketch_170606h

```
1 int x;
2
3 x = 10;
4
5 println(x);
6
7 x++;
8
9 println(x);
10
11
12
13
14
15
```

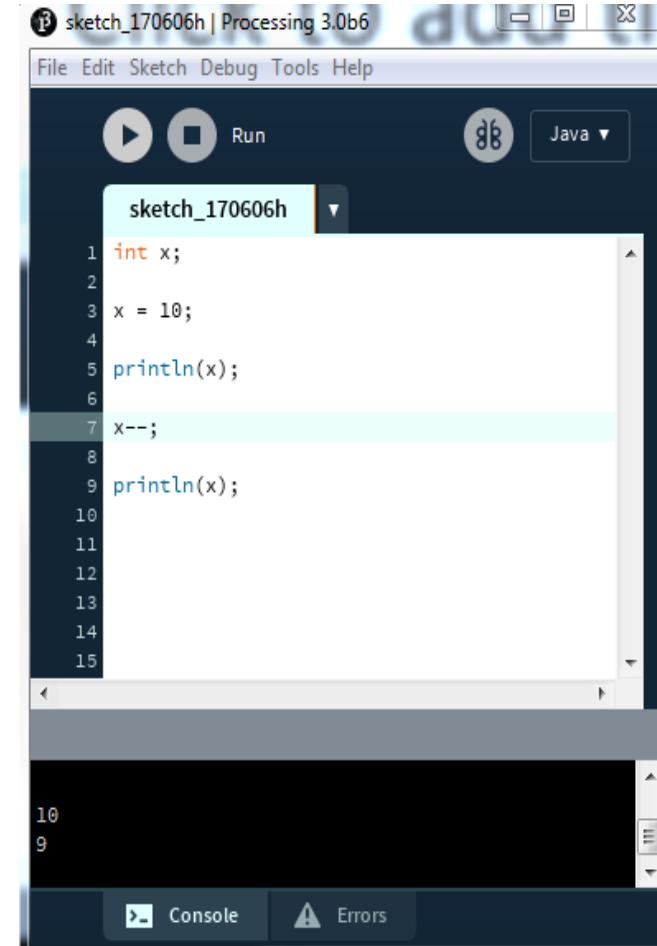
10
11

Console Errors

This screenshot shows the Processing IDE interface with a Java sketch titled "sketch_170606h". The code editor contains the following Java code:

```
int x;
x = 10;
println(x);
x++;
println(x);
```

The code is numbered from 1 to 15. Lines 7 and 9 are highlighted in light blue, indicating they are part of a loop body. The output window at the bottom shows the numbers 10 and 9, representing the values of variable x before and after the increment operation.



sketch_170606h | Processing 3.0b6

File Edit Sketch Debug Tools Help

Run

Java

sketch_170606h

```
1 int x;
2
3 x = 10;
4
5 println(x);
6
7 x--;
8
9 println(x);
10
11
12
13
14
15
```

10
9

Console Errors

This screenshot shows the Processing IDE interface with a Java sketch titled "sketch_170606h". The code editor contains the following Java code:

```
int x;
x = 10;
println(x);
x--;
println(x);
```

The code is numbered from 1 to 15. Lines 7 and 9 are highlighted in light blue, indicating they are part of a loop body. The output window at the bottom shows the numbers 10 and 9, representing the values of variable x before and after the decrement operation.

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/>

Produced
by:

Dr. Siobhán Drohan
Mairead Meagher
Sinéad Walsh



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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