Strings

Strings and their methods

Produced by: Mairead Meagher
Dr. Siobhán Drohan

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Topics list

- Primitive Types: char
- Object Types: String
- Primitive Types versus Object Types
- Strings and Java API
- Strings and methods
- Method calls (internal, external, dot notation)
- Using String methods: some examples

Recap: Primitive Types

- Java programming language supports eight primitive data types.
- The char data type stores one single character which is delimited by single quotes(‘) e.g.

```
char letter = ‘a’;
```

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

Primitive Types: char

```
// VALID USE
char letter  = ‘n’;        //Assign ‘n’ to the letter variable
char letter = ‘N’;         //Assign ‘N’ to the letter variable


// INVALID USE
char letter = n;           //ERROR – no single quotes around n.
char letter = “n”;         //ERROR – double quotes around n.
char letter = “not”;       //ERROR – char can only hold one character.
```

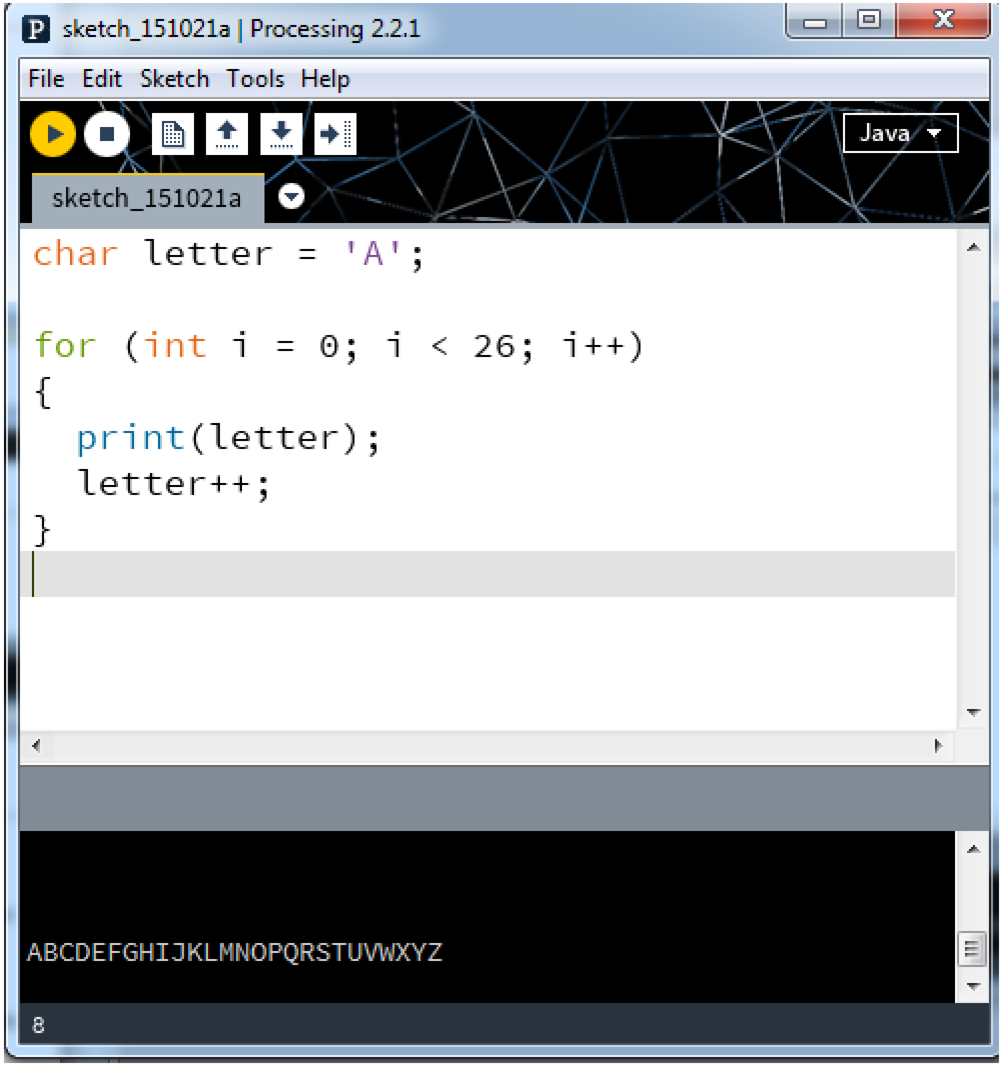
Source: Reas & Fry (2014)

Primitive Types: char

- char is a 16-bit Unicode character.
- It’s values range:
  - from '\u0000‘ (or 0)
  - to '\uffff' (or 65,535)
- For example:
  - ‘A’ is ‘\u0041’
  - ‘a’ is ‘\u0061’

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

Processing example 6.12 - Alphabet


sketch_151021a | Processing 2.2.1
File Edit Sketch Tools Help
Java
sketch_151021a

```
char letter = 'A';

for (int i = 0; i < 26; i++)
{
  print(letter);
  letter++;
}
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

This code uses the underlying Unicode value for ‘A’ (i.e. ‘\u0041’ and adds one to it each time the for loop is iterated.

As the for loop is iterated 26 times, and the starting value is ‘A’, our loop will print the alphabet to the console.

Source: Reas & Fry (2014)

Topics list

- Primitive Types: char
- Object Types: String
- Primitive Types versus Object Types
- Strings and Java API
- Strings and methods
- Method calls (internal, external, dot notation)
- Using String methods: some examples

Object types e.g. String

- Strings, which are widely used in Java programming, are a <u>sequence of characters</u> enclosed by double quotes (“).
- In the Java programming language, a String is an object type.
- The Java platform provides the String class to create and manipulate strings.
- The most direct way to create a String is to write:

```
String greeting = "Hello world!";
```

http://www.tutorialspoint.com/java/pdf/java_strings.pdf

Object types - String

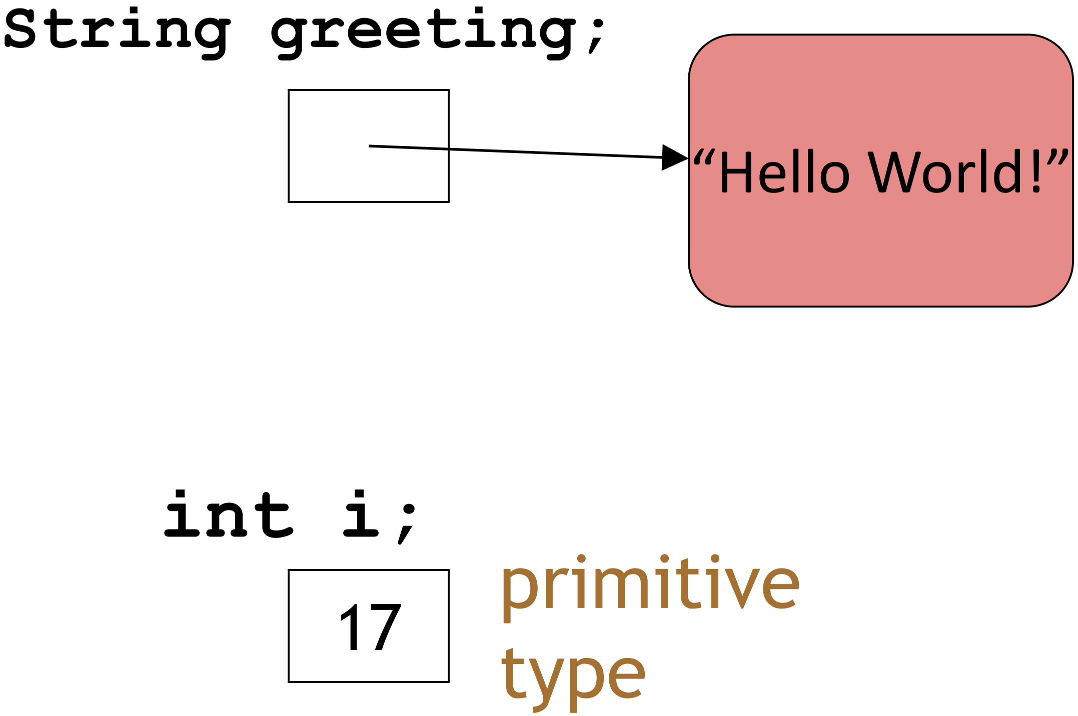
```
// VALID USE
String str = “I am a sentence”;  //Assigns the full sentence to str variable.
String word = “dog”;       //Assigns the word “dog” to the word variable.
String letter = “A”;       //Assigns the letter “A” to the letter variable.

// INVALID USE
String letter = n;         //ERROR – no double quotes around n.
String letter = ‘n’;       //ERROR – single quotes around n; use double.
string letter = “n”;       //ERROR – String should have a capital S.
```

Source: Reas & Fry (2014)

Topics list

- Primitive Types: char
- Object Types: String
- Primitive Types versus Object Types
- Strings and Java API
- Strings and methods
- Method calls (internal, external, dot notation)
- Using String methods: some examples

Primitive types vs. object types



```
String greeting;
```

String is an object type. The **greeting** variable contains a reference to where the String is stored in memory.

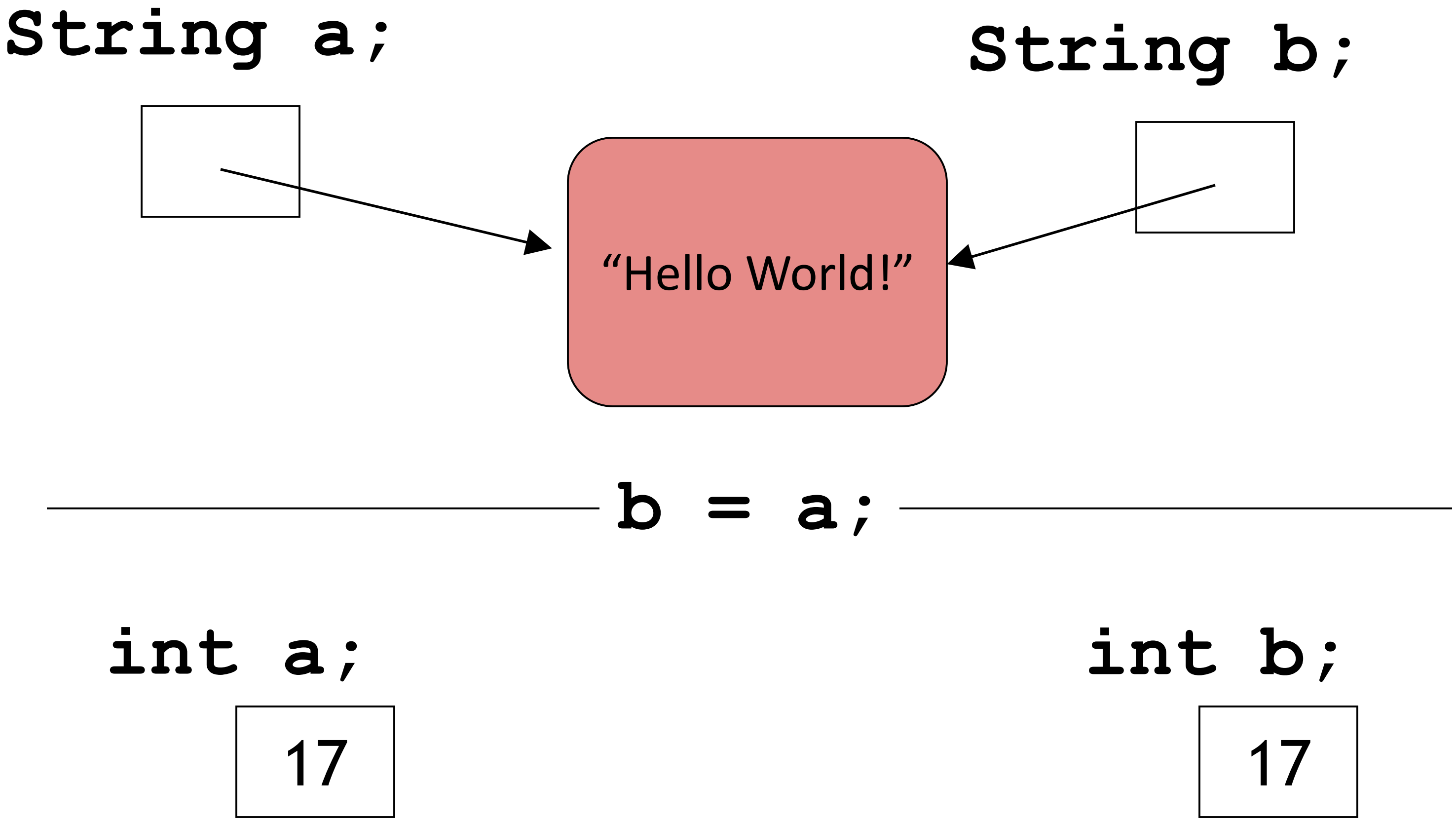
```
int i;
```

primitive type

Primitive types vs. object types

- With primitive type variables (e.g. int, float, char, etc) the value of the variable is stored in the memory location assigned to the variable.
- With object types, the variable holds the memory address of where the object is located – not the values inside the object. This memory address is called a **reference** to the object.

Primitive types vs. object types



String a;

String b;

“Hello World!”

b = a;

int a;

int b;

17

17

Topics list

- Primitive Types: char
- Object Types: String
- Primitive Types versus Object Types
- Strings and Java API
- Strings and methods
- Method calls (internal, external, dot notation)
- Using String methods: some examples

Strings are objects

- Variables created with the String data type are called objects.
- Objects are software structures that combine variables with methods that operate on those variables e.g.
  - every String object has a built-in method that can capitalise its letters.

Strings and Java’s API

- This link is to Java’s Application Programming Interface (API).

  https://docs.oracle.com/javase/8/docs/api/index.html?overview-summary.html

- At the moment, we are interested in finding out more information on String, particularly its methods:

  https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

Topics list

- Primitive Types: char
- Object Types: String
- Primitive Types versus Object Types
- Strings and Java API
- Strings and methods
- Method calls (internal, external, dot notation)
- Using String methods: some examples

Strings and some API methods

Return Type	Method Name	Description
int	length()	Returns the length of this string.
String	toLowerCase()	Converts all of the characters in this String to lower case.
String	toUpperCase()	Converts all of the characters in this String to upper case.
String	trim()	Returns a string whose value is this string, with any leading and trailing whitespace removed.
String	substring(int beginIndex, int endIndex)	Returns a string that is a substring of this string.
Int	compareTo(String anotherString)	Compares two strings lexicographically.
char	charAt(int index)	Returns the char value at the specified index.

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html

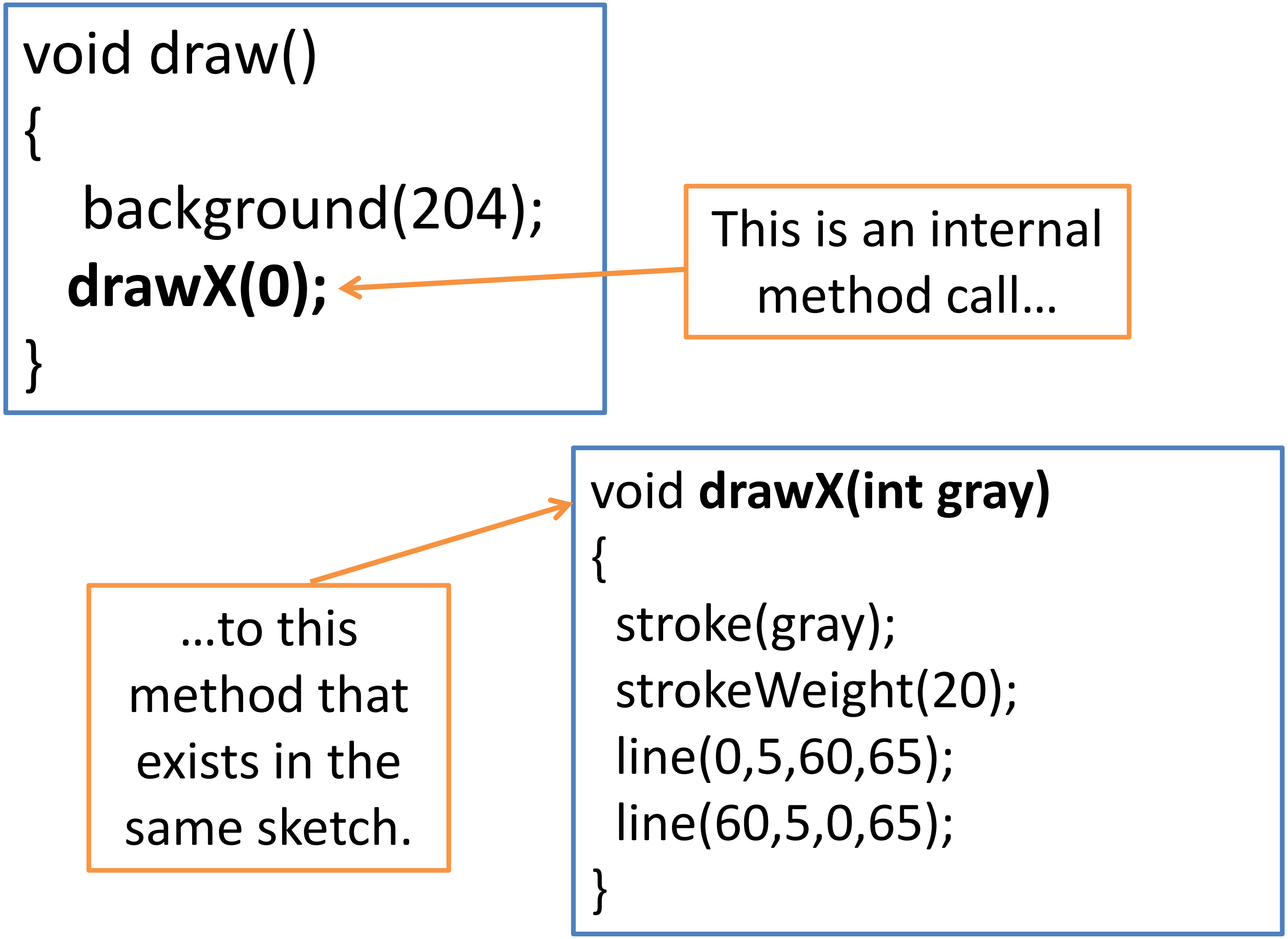
Topics list

- Primitive Types: char
- Object Types: String
- Primitive Types versus Object Types
- Strings and Java API
- Strings and methods
- Method calls (internal, external, dot notation)
- Using String methods: some examples

Strings and methods

- To use these built-in methods, we must first understand the difference between:
  - Internal method calls and
  - External method calls

Internal method calls



```
void draw()
{
    background(204);
    drawX(0);
}
```

This is an internal method call…

…to this method that exists in the same sketch.

```
void drawX(int gray)
{
  stroke(gray);
  strokeWeight(20);
  line(0,5,60,65);
  line(60,5,0,65);
}
```

Internal method calls

- **drawX(0)** is a method call.
- The sketch has a method with the following signature:

  void drawX(int gray)

- The method call invokes this method.
- As the method is in the same sketch as the call of the method, we call it an internal method call.
- Internal method calls have the syntax:

  methodname (parameter-list)

External method calls

- We want to check the length of this String:

  String name = “Joe Soap”;

- Looking at the String API, we can see this method:

ReturnType	Method	Description
int	length()	Returns the length of this string.

- A call to a method of another object is called an external method call.

External method calls

- External method calls have the syntax:

  object.methodname (parameter-list)

- To find out the length of this String:

```
String name = “Joe Soap”;
```

- We make the following external method call:

```
name.length();
```

Dot Notation

- Methods can call methods of other objects using dot notation.
- This syntax is known as dot notation:

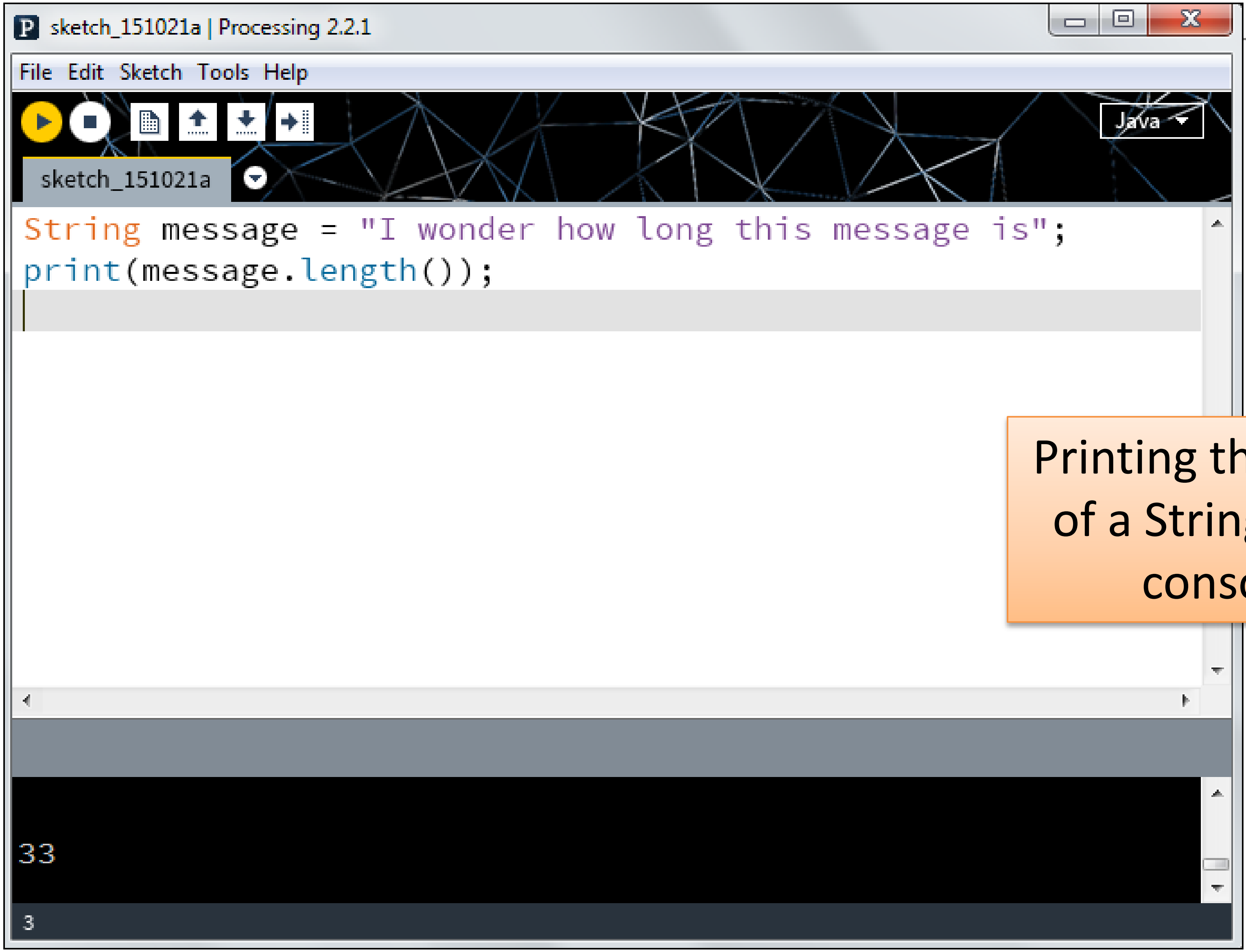
  object.methodname (parameter-list)

- It consists of:
  - An object
  - A dot
  - A method name
  - The parameters for the method

Topics list

- Primitive Types: char
- Object Types: String
- Primitive Types versus Object Types
- Strings and Java API
- Strings and methods
- Method calls (internal, external, dot notation)
- Using String methods: some examples

Processing example 6.12, version 1

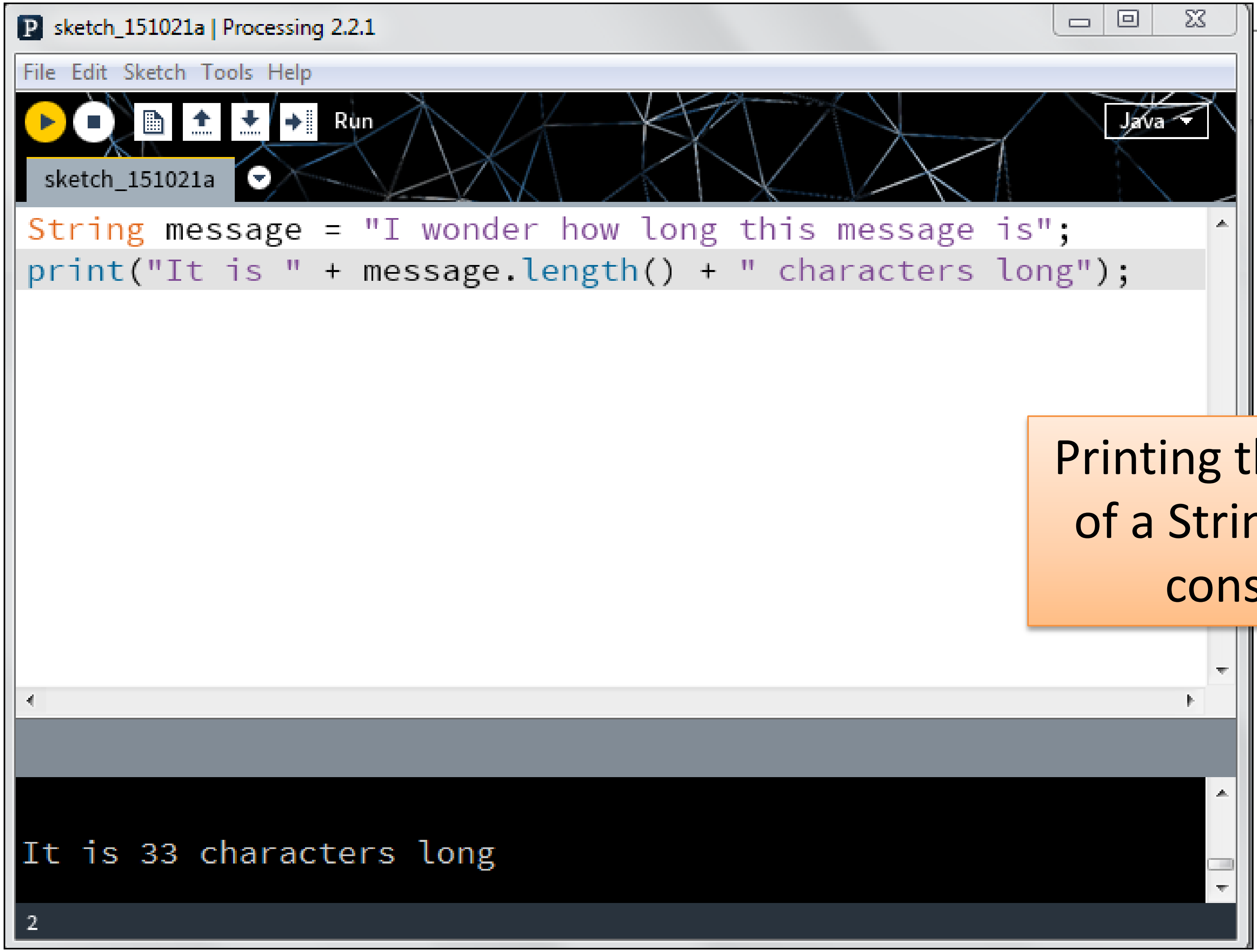

sketch_151021a | Processing 2.2.1
File Edit Sketch Tools Help
Java
sketch_151021a

```
String message = "I wonder how long this message is";
print(message.length());
```

```
33
```

Printing the length of a String to the console.

Processing example 6.12, version 2



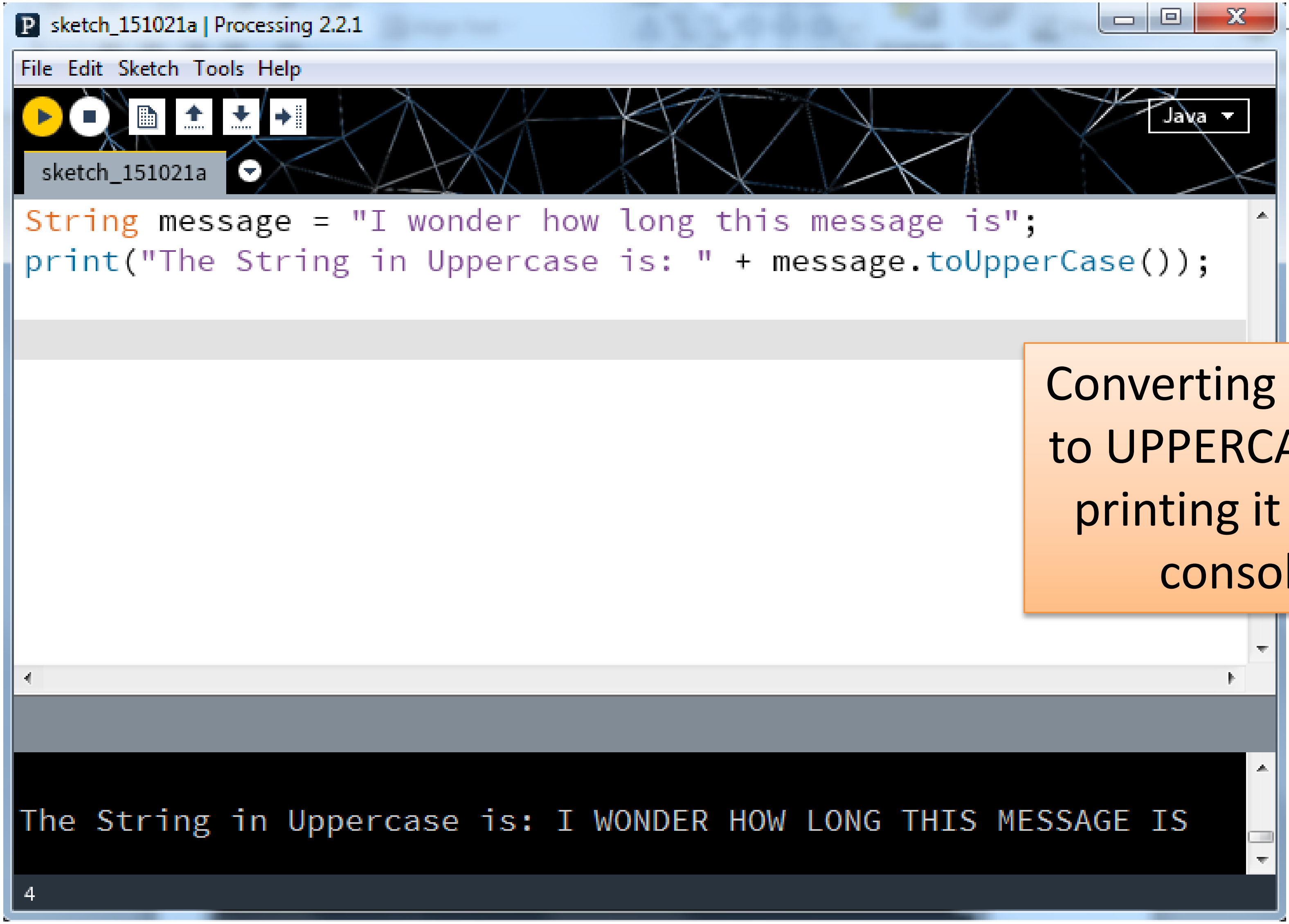
sketch_151021a | Processing 2.2.1
File Edit Sketch Tools Help
Run
Java
sketch_151021a

```
String message = "I wonder how long this message is";
print("It is " + message.length() + " characters long");
```

```
It is 33 characters long
```

Printing the length of a String to the console.

Processing example 6.13



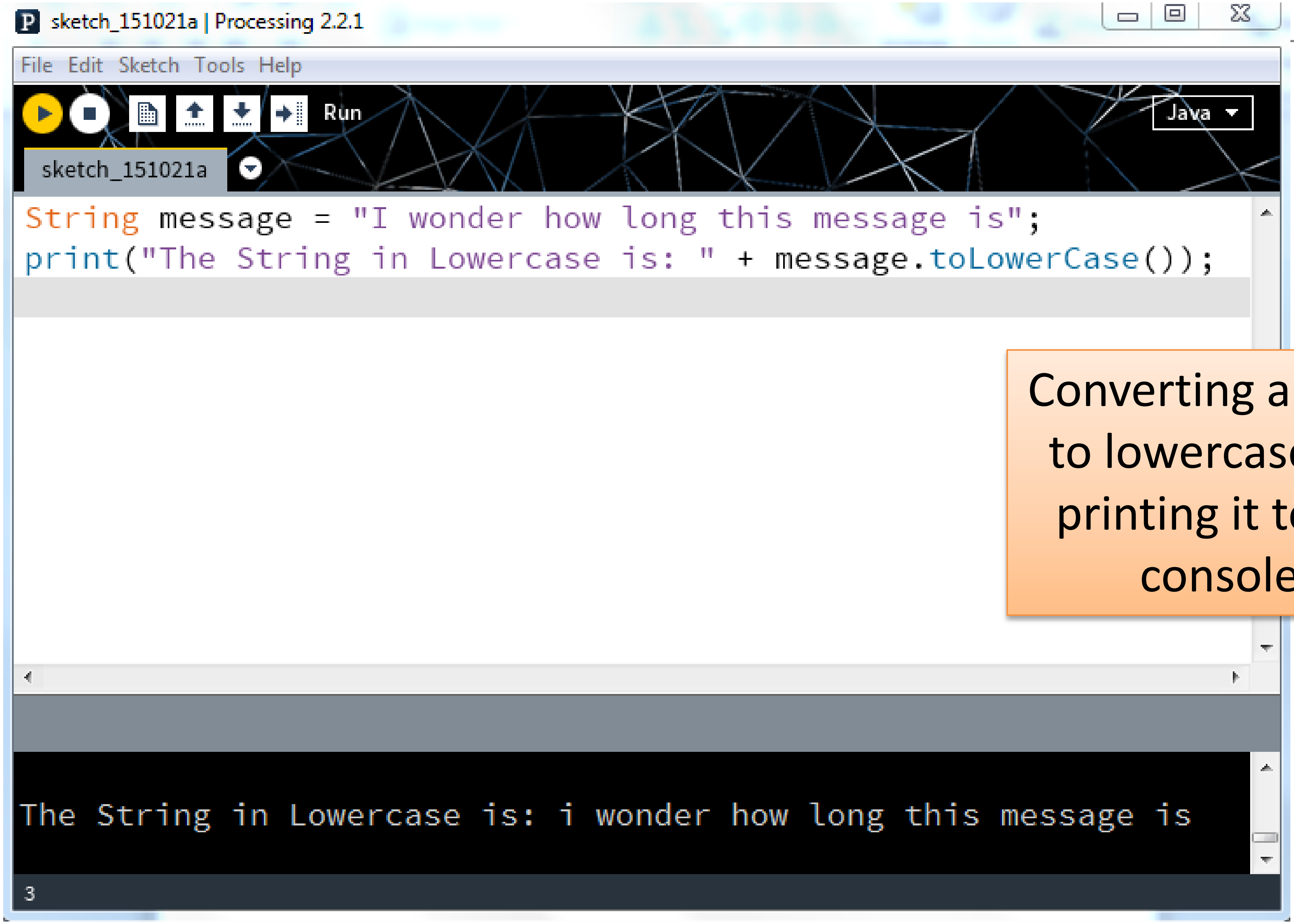
sketch_151021a | Processing 2.2.1
File Edit Sketch Tools Help
Java
sketch_151021a

```
String message = "I wonder how long this message is";
print("The String in Uppercase is: " + message.toUpperCase());
```

```
The String in Uppercase is: I WONDER HOW LONG THIS MESSAGE IS
```

Converting a String to UPPERCASE and printing it to the console.

Processing example 6.14



sketch_151021a | Processing 2.2.1
File Edit Sketch Tools Help
Run
Java
sketch_151021a

```
String message = "I wonder how long this message is";
print("The String in Lowercase is: " + message.toLowerCase());
```

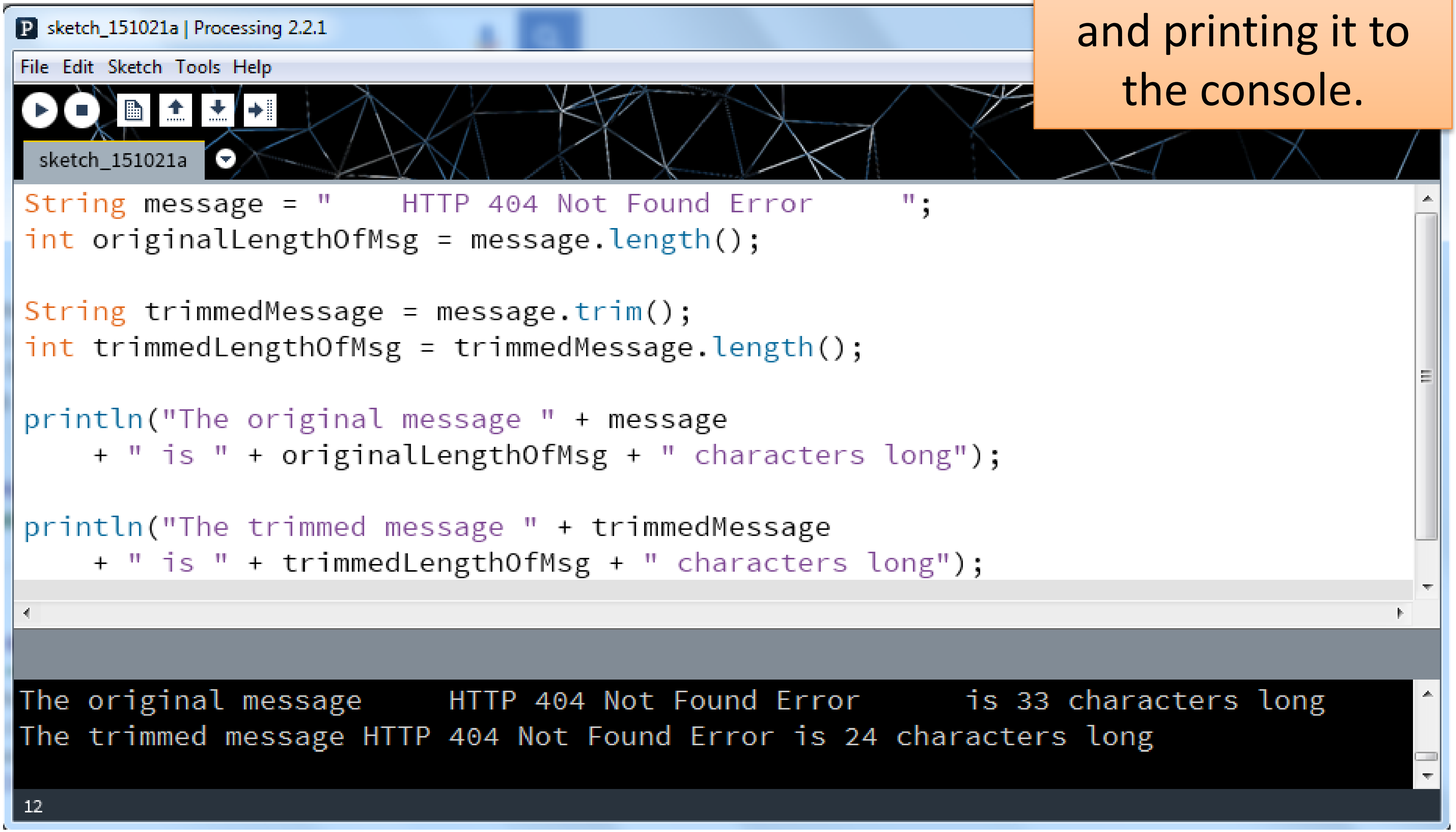
```
The String in Lowercase is: i wonder how long this message is
```

3

Converting a String to lowercase and printing it to the console.

Processing example 6.15

Removing all the leading and trailing spaces in a String and printing it to the console.


sketch_151021a | Processing 2.2.1
File Edit Sketch Tools Help
sketch_151021a
12

```
String message = "    HTTP 404 Not Found Error     ";
int originalLengthOfMsg = message.length();

String trimmedMessage = message.trim();
int trimmedLengthOfMsg = trimmedMessage.length();

println("The original message " + message
    + " is " + originalLengthOfMsg + " characters long");

println("The trimmed message " + trimmedMessage
    + " is " + trimmedLengthOfMsg + " characters long");
```

```
The original message     HTTP 404 Not Found Error      is 33 characters long
The trimmed message HTTP 404 Not Found Error is 24 characters long
```

Questions?

References

- Reas, C. & Fry, B. (2014) Processing – A Programming Handbook for Visual Designers and Artists, 2nd Edition, MIT Press, London.