

Strings

Strings and their methods

Produced Dr. Siobhán Drohan
by: Mr. Colm Dunphy
 Mr. Diarmuid O'Connor



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

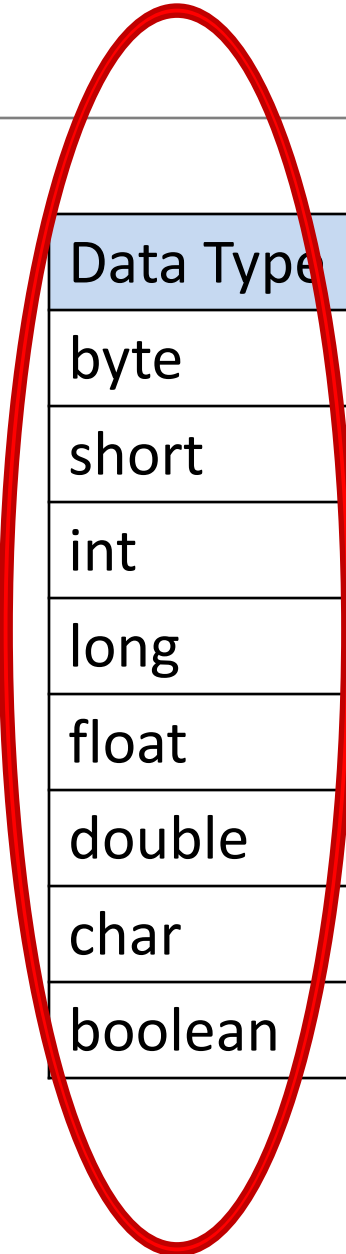
Department of Computing and Mathematics
<http://www.wit.ie/>

Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus** **Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
 - **Internal**
 - **External**
 - **Dot notation**
7. 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.

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'

Example 3.18 – Alphabet

Example_3_18 ▼

```
1 char letter = 'A';  
2  
3 for (int i = 0; i < 26; i++)  
4 {  
5     print(letter);  
6     letter++;  
7 }
```

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.

ABCDEFGHIJKLMNOPQRSTUVWXYZ



Console



Errors

Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus** **Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
 - **Internal**
 - **External**
 - **Dot notation**
7. Using String methods: some **examples**

Object types e.g. String

- Strings, which are widely used in Java programming, are a sequence of characters enclosed by double quotes ("").
e.g. **"seq of chars"**
- In Java, 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!";

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.



***Object Data Types start with a Capital Letter
to distinguish them from Primitive data types***

Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive Types versus Object Types**
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
 - **Internal**
 - **External**
 - **Dot notation**
7. Using String methods: some **examples**

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

Directly stored
in memory...

17

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

Directly stored
in memory...

17

Object type

```
String hi = "Hello";
```

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

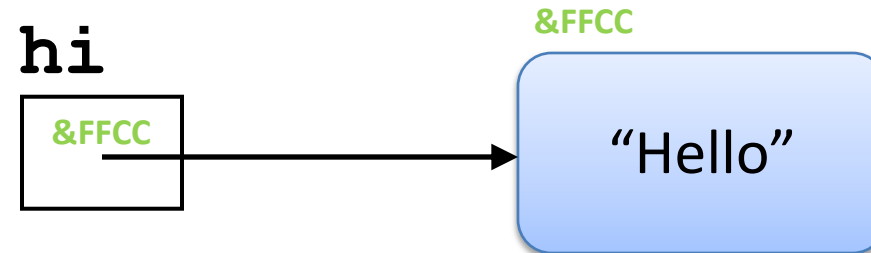
Directly stored
in memory...

17

Object type

```
String hi = "Hello";
```

hi variable
contains a reference (*address*)
to where the String is stored in
memory



Primitive types vs. Object types

Primitive type

```
int i = 17;
```

**Directly stored
in memory...**

17

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.

Primitive types vs. **Object** types

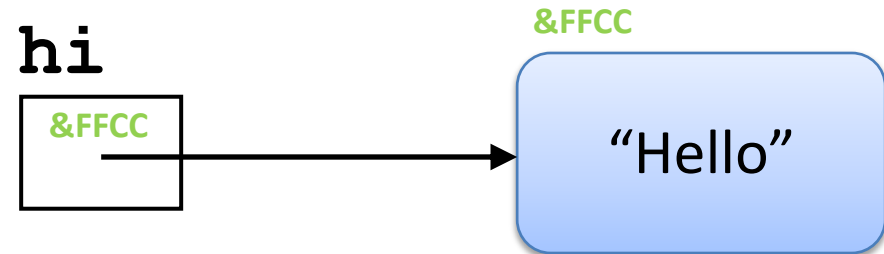
With object types,
the variable holds the **memory
address**
of where the object is located
– not the values inside the object.

This memory address is called
a **reference** to the object.

Object type

```
String hi = "Hello";
```

hi variable
contains a reference (*address*)
to where the String is stored in
memory



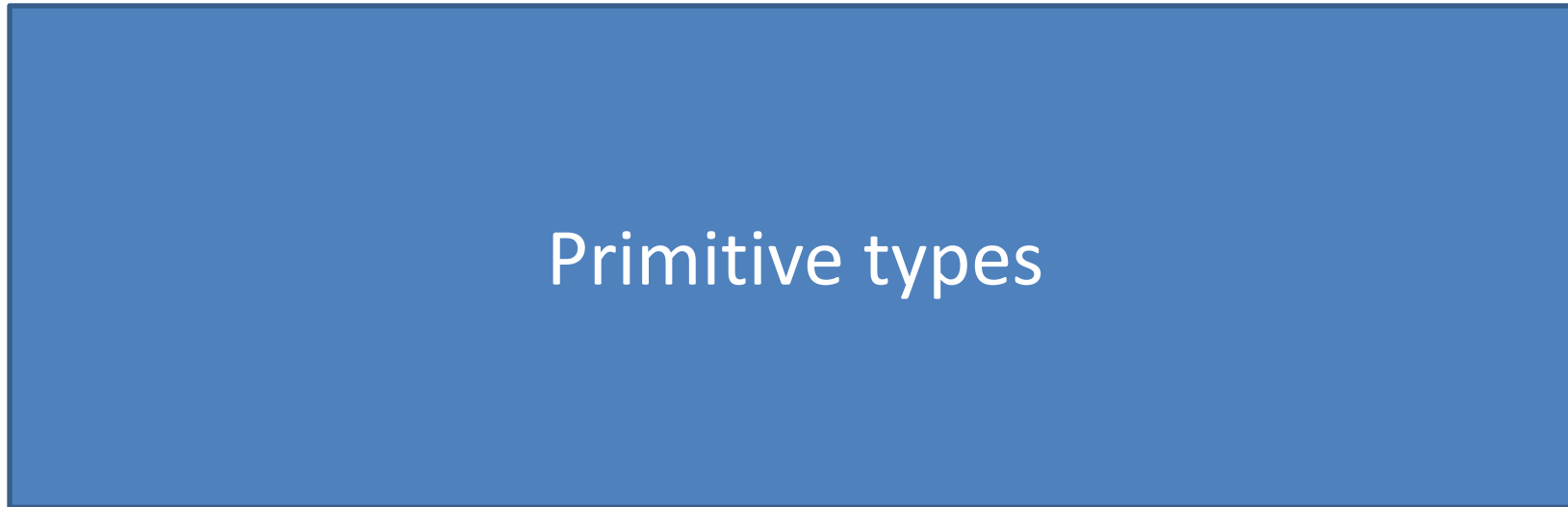
Primitive types vs. Object types

Now that we know how primitive types and object types store data,

we will look at this statement (b=a)
in the context of primitive and object types.

b = a;

Primitive types vs. Object types

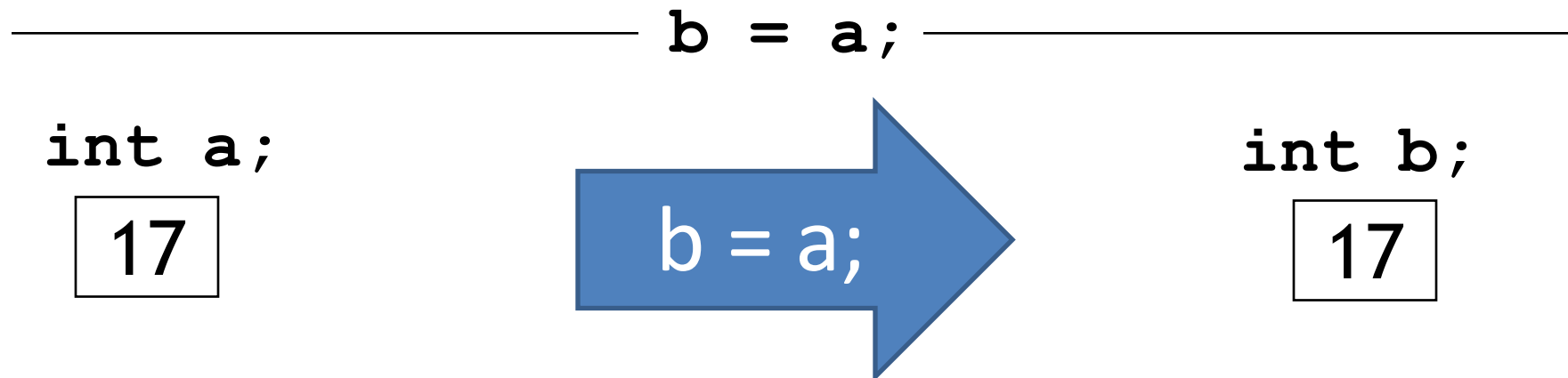
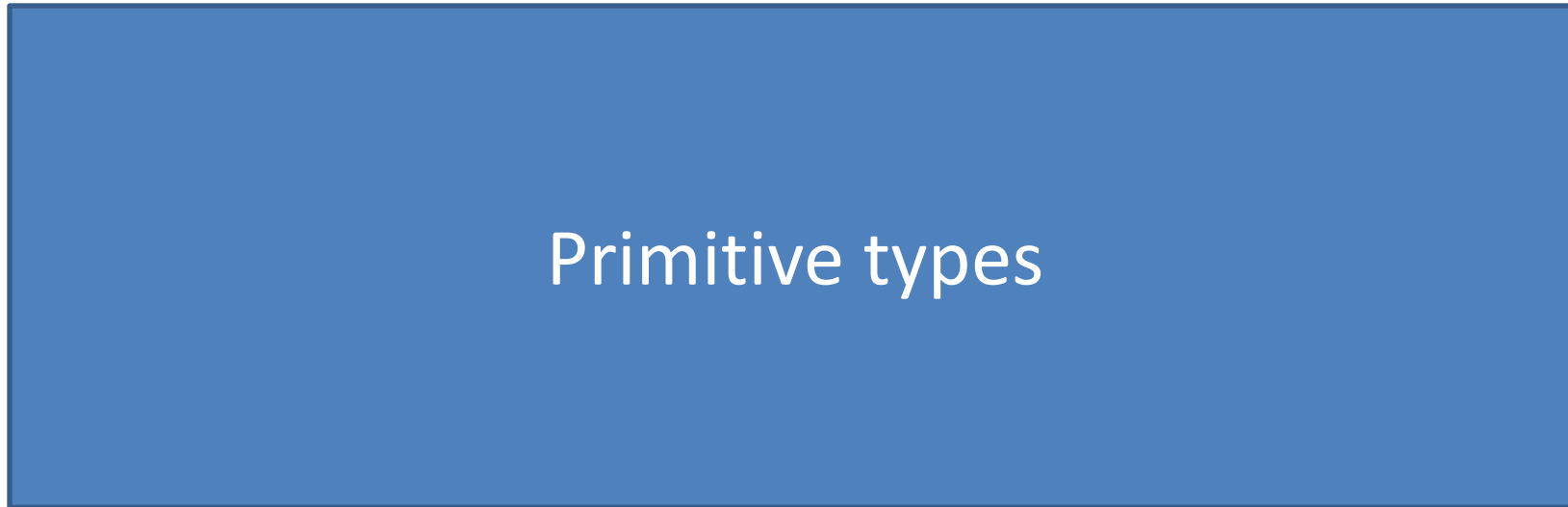


b = a;

int a;

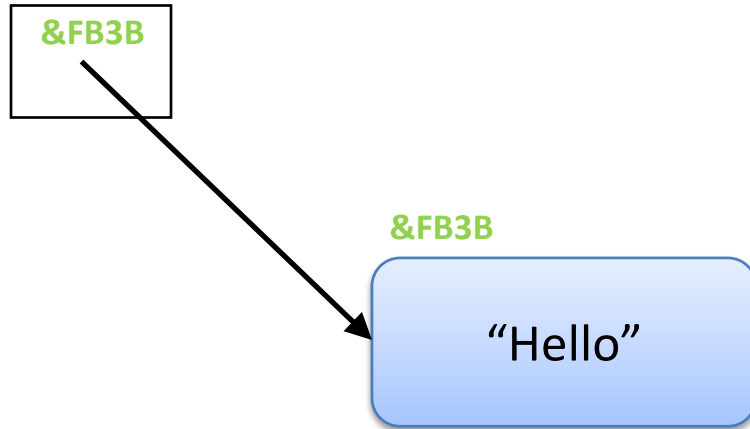
17

Primitive types vs. Object types



Primitive types vs. **Object** types

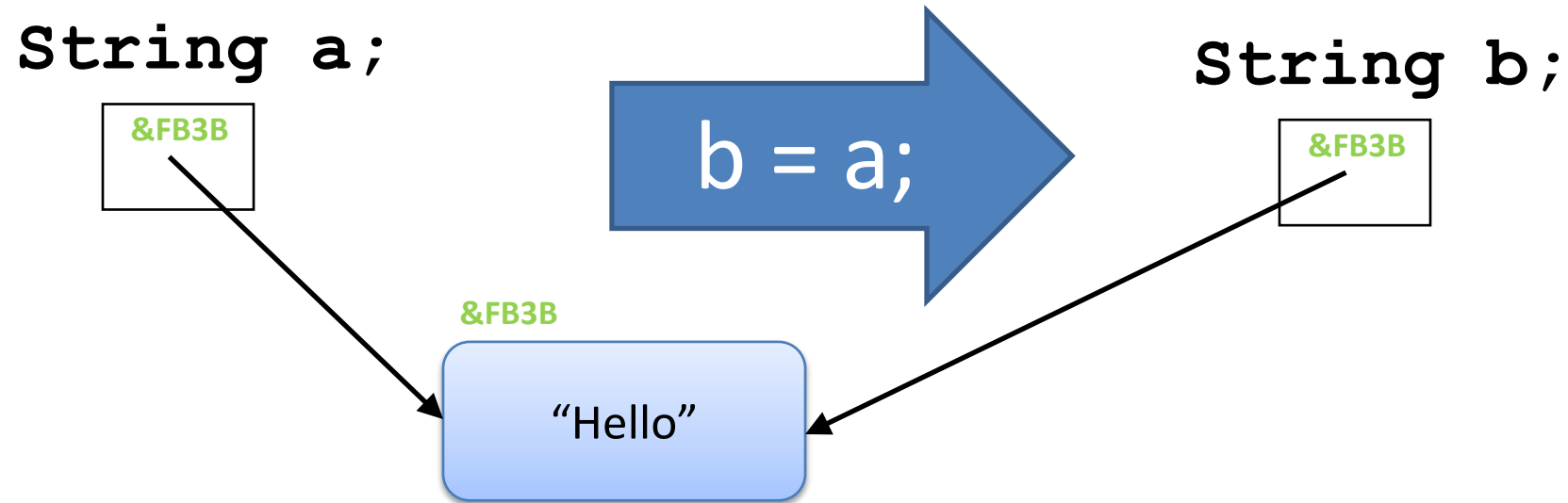
String a;



b = a;

Object types

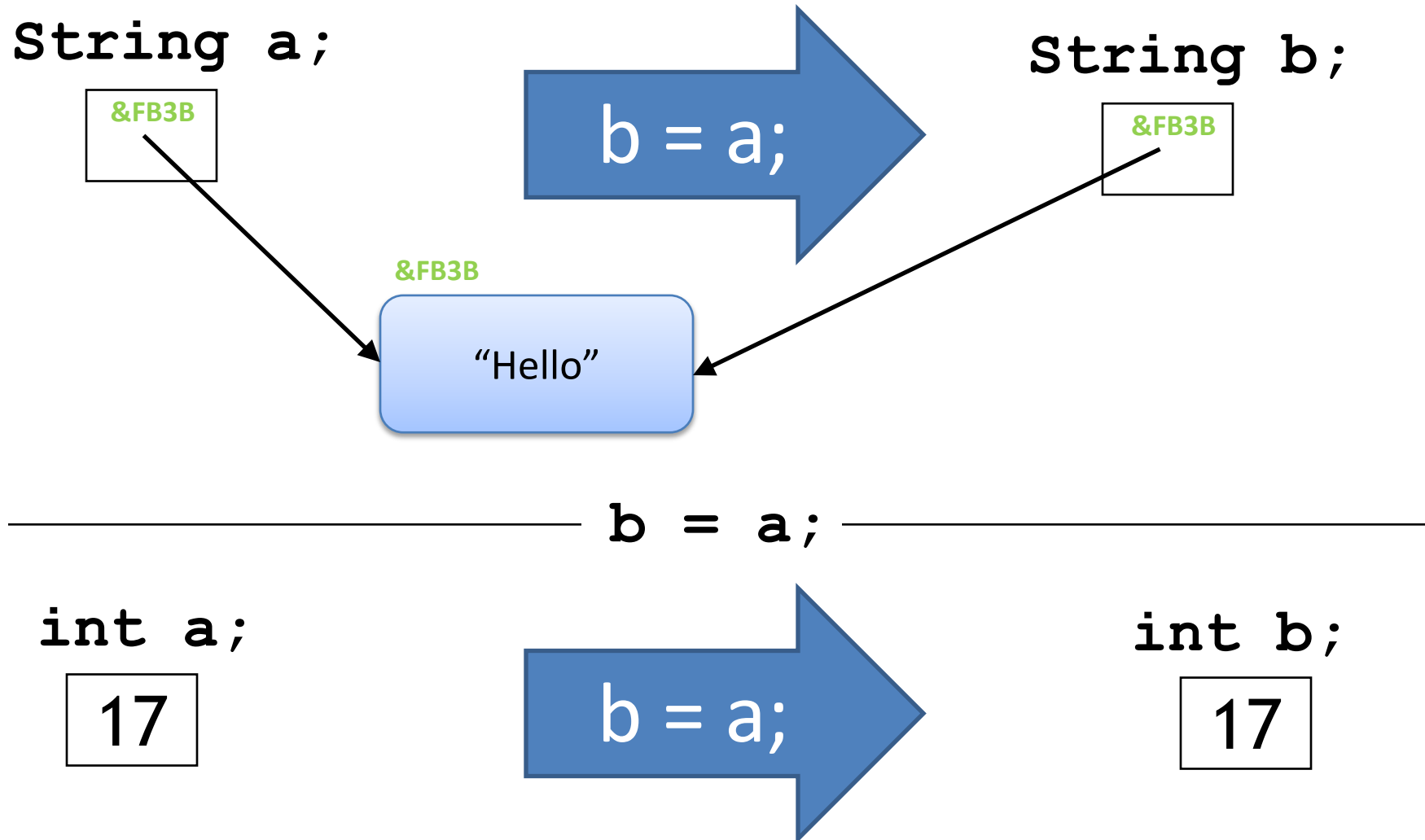
Primitive types vs. **Object** types



`b = a;`

Object types

Primitive types vs. Object types



Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus** **Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
 - **Internal**
 - **External**
 - **Dot notation**
7. 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 variablese.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)**, version 8.

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

- More information on the **String's methods**:

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

- Explore newer versions

Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus** **Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method Calls**
 - **Internal**
 - **External**
 - **Dot notation**
7. Using String methods: some **examples**

Strings - 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 <i>leading and trailing</i> whitespace removed.
String	substring (int beginIndex, int endIndex)	Returns a string that is a substring of this string.
char	charAt (int index)	Returns the char value at the specified index.

Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus** **Object** Types
4. Strings and **Java API**
5. Strings and **methods**
6. **Method calls**
 - ➡ – **Internal**
 - **External**
 - **Dot notation**
7. 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*...

*Calls
Invokes*

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

...to this method that
exists in the same sketch.


Internal method calls

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

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)

Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus** **Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
 - **Internal**
 -  – **External**
 - **Dot notation**
7. Using String methods: some **examples**

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:

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

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

(objects {e.g.String} are usually defined in their own separate files)

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();
```

Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus** **Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
 - **Internal**
 - **External**
 -  – **Dot notation**
7. Using String methods: some **examples**

Dot Notation

- Java code can call methods of other objects using dot notation.

- The syntax is:

object.methodname (parameter-list)

- It consists of:

- An **object**
- A dot
- A method name
- The parameter(s) for the method

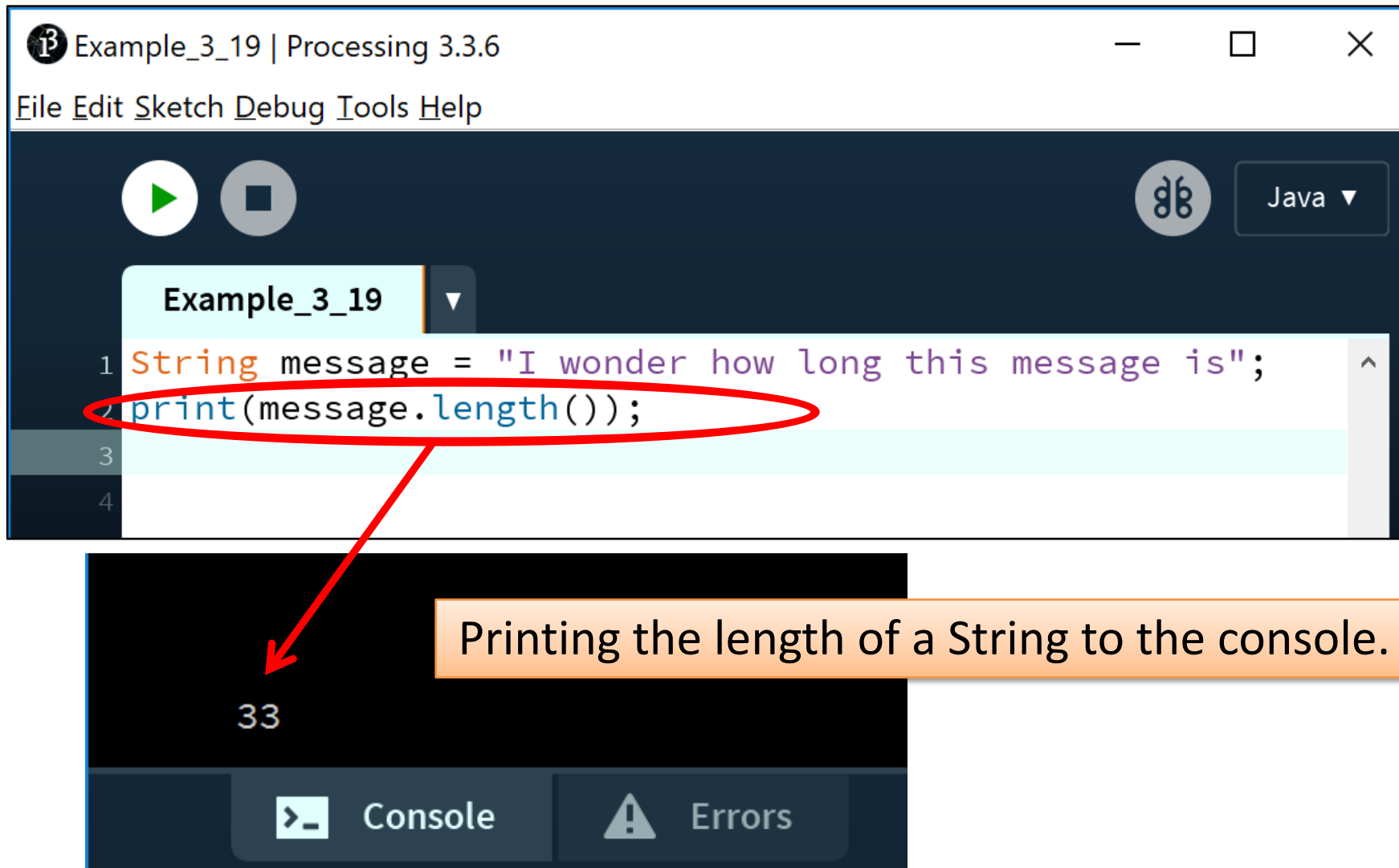


name.length();

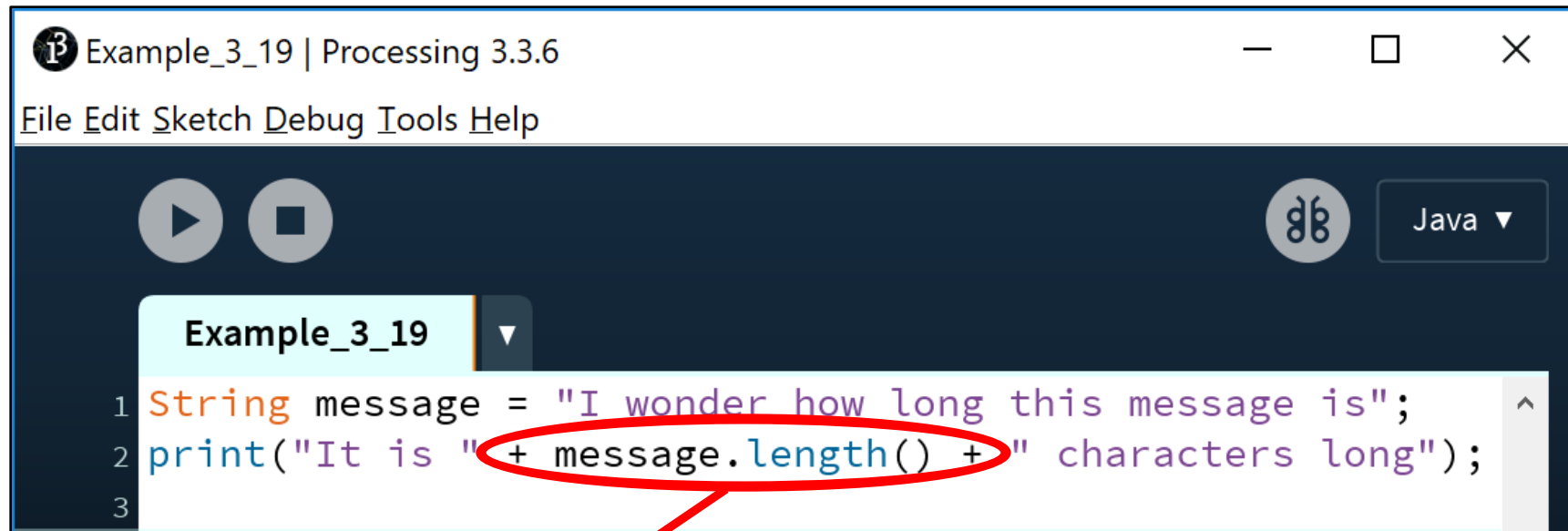
Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus** **Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
 - **Internal**
 - **External**
 - **Dot notation**
7. Using String methods: some **examples**

Example 3.19, Version 1



Example 3.19, Version 2

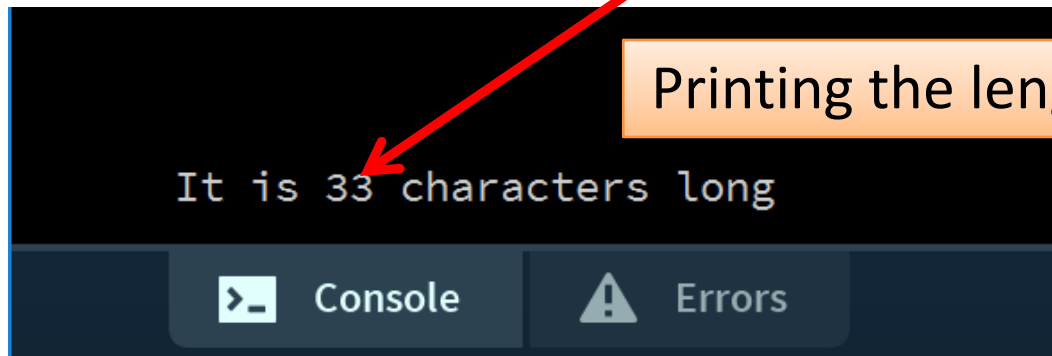


Example_3_19 | Processing 3.3.6

File Edit Sketch Debug Tools Help

Example_3_19

```
1 String message = "I wonder how long this message is";  
2 print("It is " + message.length() + " characters long");  
3
```



Printing the length of a String to the console.

Example 3.20

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

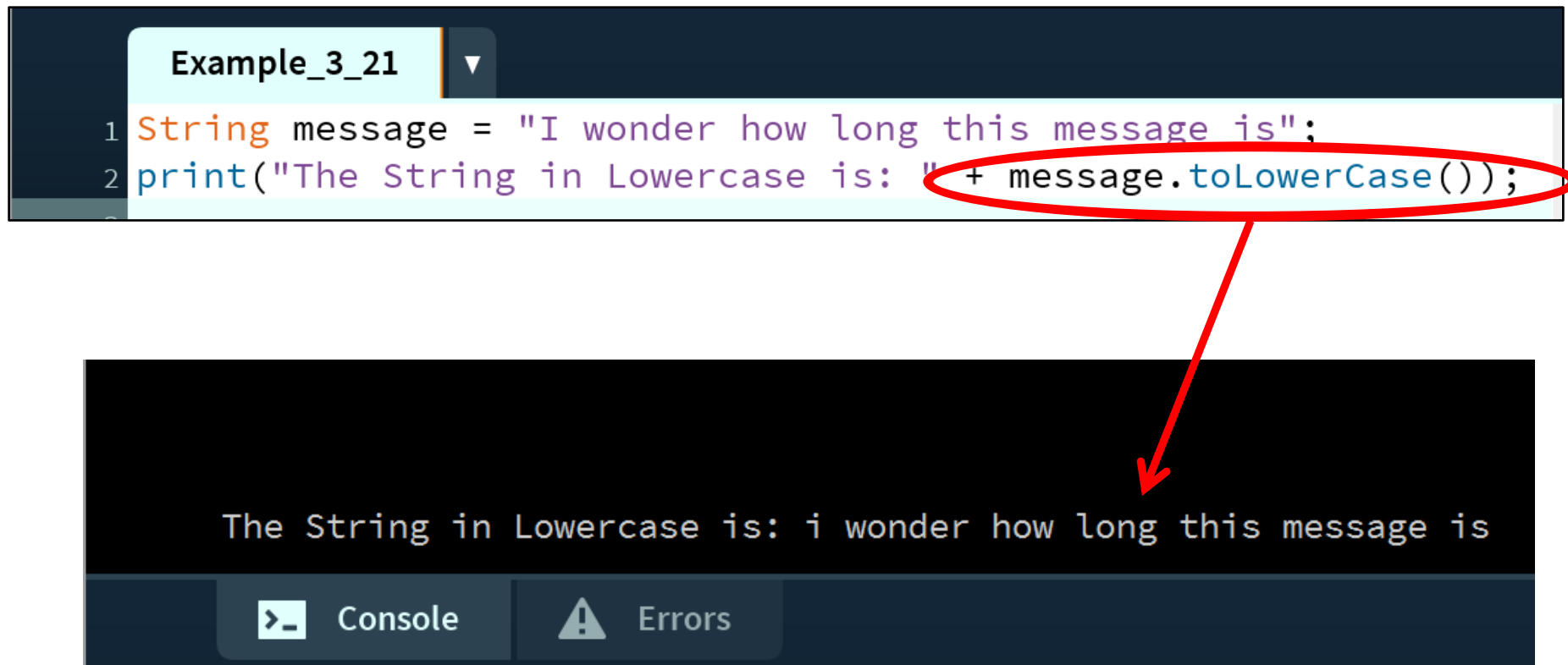
The screenshot shows the Processing IDE interface. The title bar reads "Example_3_20 | Processing 3.3.6". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". Below the menu bar are buttons for running (a green play button) and stopping (a grey square button). On the right, there is a "Java" language selector. The code editor shows a file named "Example_3_20" with the following code:

```
1 String message = "I wonder how long this message is";  
2 print("The String in Uppercase is: " + message.toUpperCase());  
3
```

The `toUpperCase()` method call in line 2 is circled in red. A red arrow points from this circle to the console output. The console at the bottom shows the output: "The String in Uppercase is: I WONDER HOW LONG THIS MESSAGE IS". The console has tabs for "Console" and "Errors".

Example 3.21

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



```
Example_3_21
1 String message = "I wonder how long this message is";
2 print("The String in Lowercase is: " + message.toLowerCase());
```

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

> Console ! Errors

Example 3.22

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

```
Example_3_22 ▼
3
4 String trimmedMessage = message.trim();
5 int trimmedLengthOfMsg = trimmedMessage.length();
6
7 println("The original message " + message
8         + " is " + originalLengthOfMsg + " characters long");
9
10 println("The trimmed message " + trimmedMessage
11         + " is " + trimmedLengthOfMsg + " characters long");
12
```

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



Console



Errors

SUMMARY - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus** **Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
 - **Internal**
 - **External**
 - **Dot notation**
7. Using String methods: some **examples**

Questions?



References

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