

Strings

Strings and their methods

Produced Ms. Mairead Meagher
by: Dr. Siobhán Drohan



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

<code>char letter = 'n';</code>	<code>//Assign 'n' to the letter variable</code>
<code>char letter = 'N';</code>	<code>//Assign 'N' to the letter variable</code>

// INVALID USE

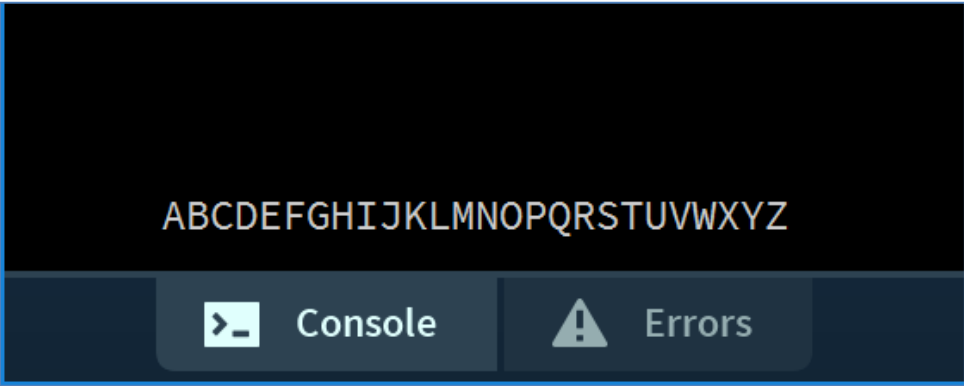
<code>char letter = n;</code>	<code>//ERROR – no single quotes around n. (Unless // n is itself a char variable)</code>
<code>char letter = "n";</code>	<code>//ERROR – double quotes around n.</code>
<code>char letter = "not";</code>	<code>//ERROR – char can only hold one character.</code>

Primitive Types: char

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

Example 6.12 – Alphabet

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

A screenshot of a code editor's console window. The main area is black with the text 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' in a light blue font. At the bottom, there are two tabs: 'Console' with a prompt icon and 'Errors' with a warning triangle icon.

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.

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;
```

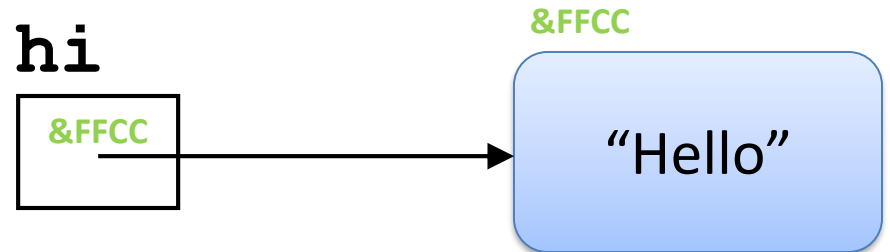
Directly stored
in memory...

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

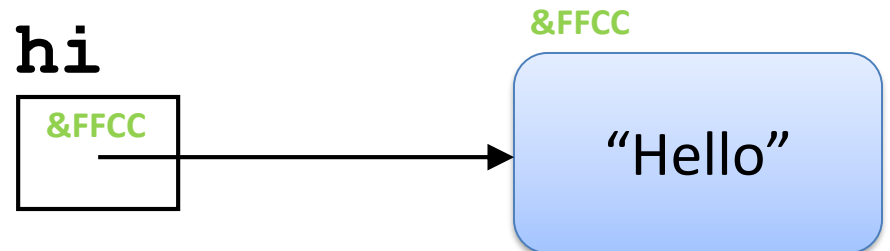
With object types, the variable holds the **memory address** of where the object is located
– not the values inside the object.

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



Primitive types

b = a;

int a;



17

Primitive types vs. Object types

Primitive types

b = a;

int a;

17

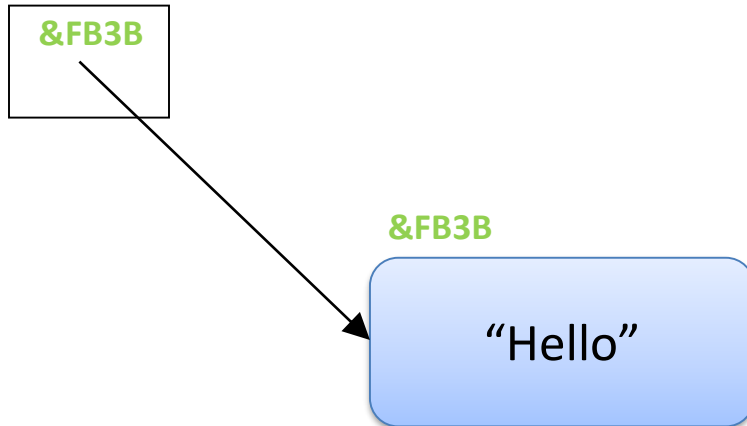
b = a;

int b;

17

Primitive types vs. Object types

String a;



b = a;

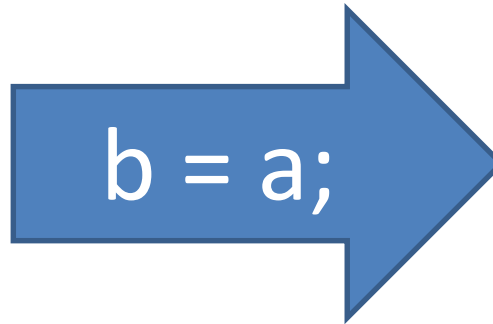
Object types

Primitive types vs. Object types

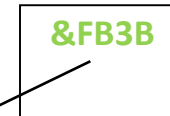
String a;



`&FB3B`



String b;



b = a;

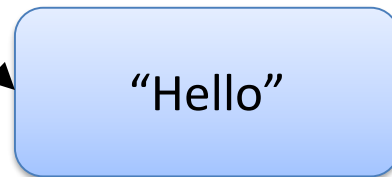
Object types

Primitive types vs. Object types

String a;

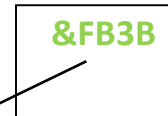


`&FB3B`



b = a;

String b;



b = a;

int a;



b = a;

int b;



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 variables
e.g.
 - every String object has a built-in method that can capitalise its letters.

Java's API

Java's **A**pplication **P**rogramming Interface (**API**), version 10:

<https://docs.oracle.com/javase/10/docs/api/overview-summary.html>

The screenshot shows a web browser window displaying the 'Overview (Java SE 10 & JDK 10)' page. The browser's address bar shows the URL 'https://docs.oracle.com/javase/10/docs/api/overview-summary.html'. The page has a navigation bar with tabs for 'OVERVIEW', 'MODULE', 'PACKAGE', 'CLASS', 'USE', 'TREE', 'DEPRECATED', 'INDEX', and 'HELP'. Below the navigation bar, there are links for 'PREV', 'NEXT', 'FRAMES', 'NO FRAMES', and 'ALL CLASSES', along with a search bar. The main content area is titled 'Java® Platform, Standard Edition & Java Development Kit Version 10 API Specification'. It states that the document is divided into three sections: Java SE, JDK, and JavaFX. Each section provides a brief description of the APIs. At the bottom, there is a table with columns for 'Module' and 'Description'. The first row in the table is for the 'java.activation' module, which defines the JavaBeans Activation Framework (JAF) API.

Module	Description
java.activation	Defines the JavaBeans Activation Framework (JAF) API.

Java's API and Strings



The screenshot shows the Oracle Java SE 10 & JDK 10 API documentation page for the `String` class. The page is titled "String (Java SE 10 & JDK 10)" and is part of the "JDK 10 Documentation". The search bar is located in the top right corner, and a red arrow points to it from a text box that says "To view the String Class, enter **String** in the Search box."

Module `java.base`
Package `java.lang`
Class String

`java.lang.Object`
`java.lang.String`

All Implemented Interfaces:
`Serializable`, `CharSequence`, `Comparable<String>`

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because `String` objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

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.

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

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();
```

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 6.13, Version 1

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

33



Console

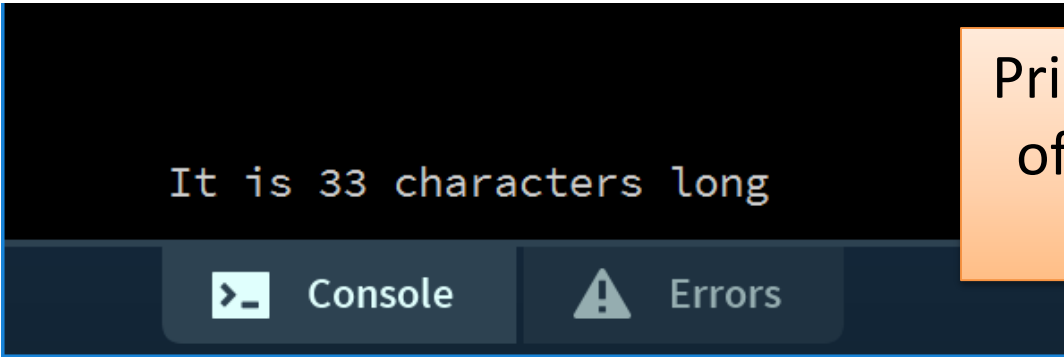


Errors

Printing the length
of a String to the
console.

Example 6.13, Version 2

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



The screenshot shows a dark-themed console window. The text "It is 33 characters long" is displayed in a light gray font. Below the text, there is a dark blue bar with two tabs: "Console" (with a terminal icon) and "Errors" (with a warning triangle icon).

It is 33 characters long

Printing the length
of a String to the
console.

Example 6.14

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

```
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



Console



Errors

Example 6.15

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

```
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
```



Console



Errors

Example 6.16

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

```
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
```



Console



Errors

Example 6.16

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

```
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
```



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.