AP Computer Science A

UNIT 2 TOPIC 7

Strings Methods Part 1

Do Now: complete the Warm Up in Google Classroom

Do Now: Warm Up!

Open up the official documentation for the Java String class:
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html

Find a method for the String class and answer these questions:

- What is the method you chose?
- What does the method do?
- What are its parameters?
- What is its return type?

This documentation is known as an **API** -- "**application programming interface**" -- which is a fancy CS term for "how to use something", in this case, *how to use the Java String class!*

UNIT 2 Using Objects

College Board Standards

Unit 2 Topic 7

ENDURING UNDERSTANDING

VAR-1

To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE

VAR-1.E

For `String` class:

a. Create `String` objects.

b. Call `String` methods.

ESSENTIAL KNOWLEDGE

VAR-1.E.6

Application program interfaces (APIs) and libraries simplify complex programming tasks.

VAR-1.E.7

Documentation for APIs and libraries are essential to understanding the attributes and behaviors of an object of a class.

VAR-1.E.8

Classes in the APIs and libraries are grouped into packages.

VAR-1.E.9

The `String` class is part of the `java.lang` package. Classes in the `java.lang` package are available by default.

VAR-1.E.10

A `String` object has index values from 0 to `length` — 1. Attempting to access indices outside this range will result in an `IndexOutOfBoundsException`.

VAR-1.E.11

A `String` object can be concatenated with an object reference, which implicitly calls the referenced object's `toString` method.

LEARNING OBJECTIVE

VAR-1.E

For `String` class:

a. Create `String` objects.

b. Call `String` methods.

ESSENTIAL KNOWLEDGE

VAR-1.E.12

The following `String` methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:

- `String(String str)` — Constructs a new `String` object that represents the same sequence of characters as `str`
- `int length()` — Returns the number of characters in a `String` object
- `String substring(int from, int to)` — Returns the substring beginning at index `from` and ending at index `to - 1`
- `String substring(int from)` — Returns `substring(from, length())`
- `int indexOf(String str)` — Returns the index of the first occurrence of `str`; returns –1 if not found
- `boolean equals(String other)` — Returns `true` if `this` is equal to `other`; returns `false` otherwise
- `int compareTo(String other)` — Returns a value < 0 if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value > 0 if `this` is greater than `other`

VAR-1.E.13

A string identical to the single element substring at position `index` can be created by calling `substring(index, index + 1)`.

VAR-1.E.2

`String` objects are immutable, meaning that `String` methods do not change the `String` object.

String objects: the nitty gritty

The String data type is considered a **non-primitive** data type.

Non-primitive data types usually have methods we can call to manipulate them.

When we create a String like this...

```
String myStr = "hello";
```

...we are creating a String object, and we can call lots of different methods on the `myStr` object, as detailed by the Java API -- these are the "instance methods"

All Methods	Static Methods	Instance Methods	Concrete Methods	Deprecated Methods

Modifier and Type	Method	Description
char	charAt(int index)	Returns the char value at the specified inde
IntStream	chars()	Returns a stream of int zero-extending the
int	codePointAt(int index)	Returns the character (Unicode code point)

String objects: the nitty gritty

A String holds characters in a sequence.

Each character is at a position, or **index**, **which starts with 0** (**NOT 1**, like in Snap).

An index is a number associated with a position in a String (sort of like an array or list, although a String is not an array or list). Here is an example:

```
String str = "AP CSA is awesome!"
```

character:	A	P		C	S	A		i	s		a	w	e	s	o	m	e	!
index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

String objects: the nitty gritty

A String holds characters in a sequence.

Each character is at a position, or **index**, **which starts with 0** (**NOT 1**, like in Snap).

An index is a number associated with a position in a String (sort of like an array or list, although a String is not an array or list). Here is an example:

```
String str = "AP CSA is awesome!"
```

character:	A	P		C	S	A		i	s		a	w	e	s	o	m	e	!
index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

The first index of a String is index 0 -- always! Please forget what you learned with Snap!
When I snap my fingers, you will forget that "index 1 is the first index"!

There are *many* String methods!

Fortunately, for this course, you only need to master a few of these for the AP exam 😊

Although there will be others we will use because they are useful (even though they won't be on the AP exam)!

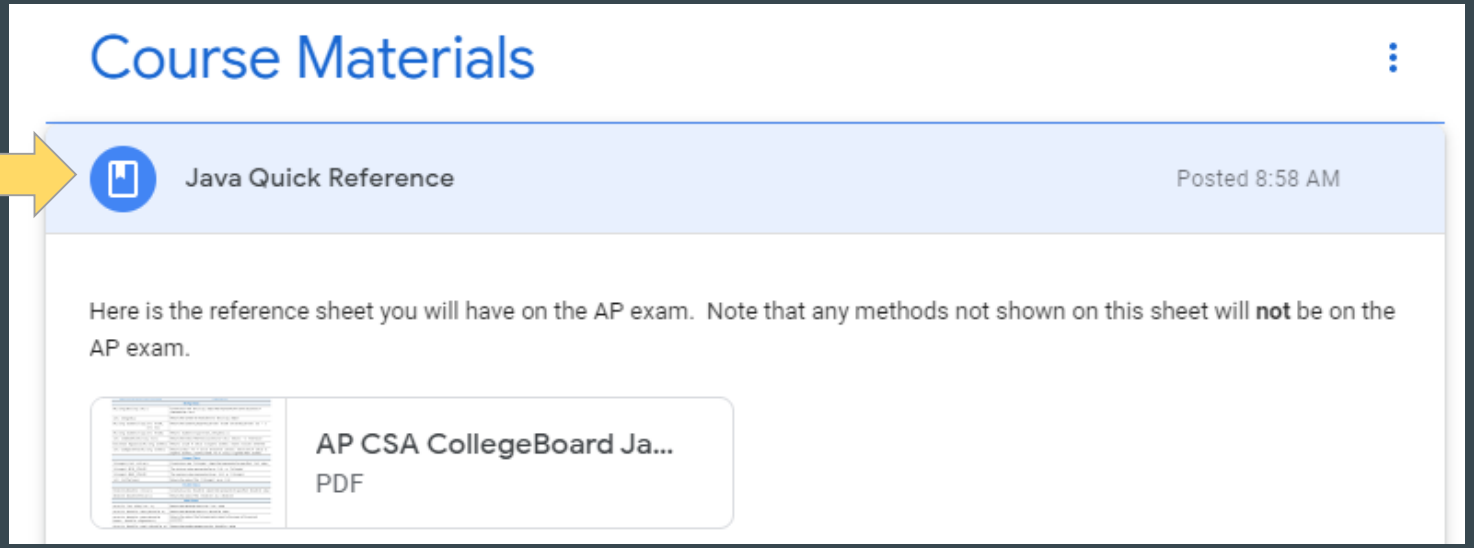
Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods | Deprecated Methods

Modifier and Type	Method and Description
char	**charAt**(int index) Returns the char value at the specified index.
int	**codePointAt**(int index) Returns the character (Unicode code point) at the specified index.
int	**codePointBefore**(int index) Returns the character (Unicode code point) before the specified index.
int	**codePointCount**(int beginIndex, int endIndex) Returns the number of Unicode code points in the specified text range of this String.
int	**compareTo**(String anotherString) Compares two strings lexicographically.
int	**compareToIgnoreCase**(String str) Compares two strings lexicographically, ignoring case differences.
String	**concat**(String str) Concatenates the specified string to the end of this string.
boolean	**contains**(CharSequence s) Returns true if and only if this string contains the specified sequence of char values.
boolean	**contentEquals**(CharSequence cs) Compares this string to the specified CharSequence.
boolean	**contentEquals**(StringBuffer sb) Compares this string to the specified StringBuffer.
boolean	**endsWith**(String suffix) Tests if this string ends with the specified suffix.
boolean	**equals**(Object anObject) Compares this string to the specified object.
boolean	**equalsIgnoreCase**(String anotherString) Compares this String to another String, ignoring case considerations.
byte[]	**getBytes**() Encodes this String into a sequence of bytes using the platform's default charset, storing the result into a new byte a
byte[]	**getBytes**(Charset charset) Encodes this String into a sequence of bytes using the given **charset**, storing the result into a new byte array.
void	**getBytes**(int srcBegin, int srcEnd, byte[] dst, int dstBegin) **Deprecated.** This method does not properly convert characters into bytes. As of JDK 1.1, the preferred way to do this is via the **get**
byte[]	**getBytes**(String charsetName) Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array.
void	**getChars**(int srcBegin, int srcEnd, char[] dst, int dstBegin) Copies characters from this string into the destination character array.
int	**hashCode**() Returns a hash code for this string.

String Methods to Know are on the CB "Java Quick Reference"

- The College Board has selected a *handful* (subset) of `String` methods that you need to know for the AP Exam (and will learn how to use today)
- You don't have to memorize them; they are on the College Board's "**Java Quick Reference**" which you will use starting today, **and will have on the day of the AP Exam**



Or just google "CSA Java Quick Reference" and download the PDF

String Methods to Know

You saw this one in our last lab, it's the String constructor and can be used to create a string object:

```
String str = new String("hello!");
```

Java Quick Reference

Accessible methods from the Java library that may be included in the exam

Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value `<0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `>0` if `this` is greater than `other`

These are the SIX String methods to know well -- these are on the Java Quick Reference sheet, which you will have on exam day

How to read this reference sheet

Example:

`int length()`	Returns the number of characters in a `String` object

Return type

How to read this reference sheet

Example:

`int length()`	Returns the number of characters in a `String` object

Method name

How to read this reference sheet

Example:

`int length()`	Returns the number of characters in a `String` object

Parameters, if any (the `length` method has none)

How to read this reference sheet

Example:

`int length()`	Returns the number of characters in a `String` object

Description of what the method does and what it returns (if anything)

Reference Sheet vs. Java API

Reference Sheet:

`int length()`	Returns the number of characters in a `String` object

Java API:

`int`	`length()`	Returns the length of this string.

Pretty darn similar!

How to read this reference sheet

`int length()`	Returns the number of characters in a `String` object

So, based on this, how do we use the `length` method?

How to read this reference sheet

`int length()`	Returns the number of characters in a `String` object

So, based on this, how do we use the `length` method?

```
String str = "Hello!";
int len = str.length();    // method called on the object
System.out.println(len);
```

What will get printed?

How to read this reference sheet

`int length()`	Returns the number of characters in a `String` object

So, based on this, how do we use the `length` method?

```
String str = "Hello!";
int len = str.length();
System.out.println(len);
```

What will get printed? 6

Each character, including punctuation symbols like !, are considered part of the string's length

How to read this reference sheet

`int length()`	Returns the number of characters in a `String` object

So, based on this, how do we use the `length` method?

```
String str = "Hello!";
int len = str.length();
System.out.println(len);
```

What will get printed? 6

Each character, including punctuation symbols like !, are considered part of the string's length

Strings and Indexes

Every character in a String has an **index**. Index is the *location* of the character.

<u>Index values ALWAYS start at 0</u>

Example: “Hello!”

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

You will see many methods that work the index values. Pay careful attention and **always take time to write out the String with index values *before* answering questions!**

Review: How to read a method description

`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found

What is the return value?

Does it have any parameters?

Review: How to read a method description

`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found

What is the return value? `int`

Does it have any parameters?

Review: How to read a method description

`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found

What is the return value? `int`

Does it have any parameters? yes, one -- a `String`

How to read this reference sheet

`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found

So, based on this, how do we use the `indexOf` method?

```
String str = "Hello!";
int idx = str.indexOf("ell");
System.out.println(idx);
```

What will get printed?

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

How to read this reference sheet

`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found

So, based on this, how do we use the `indexOf` method?

```
String str = "Hello!";
int idx = str.indexOf("ell");
System.out.println(idx);
```

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

What will get printed? **1**

The string "ell" occurs in "Hello!" starting at index 1

How to read this reference sheet

`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found

So, based on this, how do we use the `indexOf` method?

```
String str = "Hello!";
int idx = str.indexOf("all");
System.out.println(idx);
```

What will get printed?

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

How to read this reference sheet

`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found

So, based on this, how do we use the `indexOf` method?

```
String str = "Hello!";
int idx = str.indexOf("all");
System.out.println(idx);
```

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

What will get printed? **-1**

The string "all" does NOT occur in "Hello!", the -1 result means "not found"

Substring method

There are **two substring** methods (which means the substring method is an **overloaded** method on the String class!)

`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`

Gives you all characters from index "from" through index "to - 1"

`String substring(int from)`	Returns `substring(from, length())`

Gives you all characters from index "from" through the end of the String

Substring method

```
String s = "Hello!";
String q1 = s.substring(1, 5);
System.out.println(q1);
String q2 = s.substring(3);
System.out.println(q2);
```

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

Substring method

```
String s = "Hello!";
String q1 = s.substring(1, 5);// index 1 through 4 (5-1)
System.out.println(q1);
String q2 = s.substring(3);
System.out.println(q2);
```

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

Substring method

```
String s = "Hello!";
String q1 = s.substring(1, 5);// index 1 through 4
System.out.println(q1);  // prints "ello"
String q2 = s.substring(3);
System.out.println(q2);
```

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

Substring method

```
String s = "Hello!";
String q1 = s.substring(1, 5);// index 1 through 4
System.out.println(q1);  // prints "ello"
String q2 = s.substring(3);  // index 3 through the end
System.out.println(q2);
```

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

Substring method

```
String s = "Hello!";
String q1 = s.substring(1, 5);// index 1 through 4
System.out.println(q1);  // prints "ello"
String q2 = s.substring(3);  // index 3 through the end
System.out.println(q2); // prints "lo!"
```

Char	H	e	l	l	o	!
Index	0	1	2	3	4	5

Agenda

- U2T7 Lab 1: String Methods (due tomorrow)

Finish early? Check out some of the other classes in the Java API! System, Scanner, DecimalFormat, or maybe some others you know about!