# Object-Oriented Development (CIS1056-N) Supplemental Documentation on Strings

## Introduction

So far, we have only dealt with Java’s primitive types. The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are **immutable**; their values cannot be changed after they are created. String operations such as concatenate create a new String rather than modifying the existing one. String buffers support mutable strings (not covered in this document).

Strings can be constructed by:

* Using the new operator.
* Directly assigning a string literal to a String Object Reference variable.

For example:

String firstName = new String("James"); // Explicit construction

String lastName = "Mead";             // Implicit construction

The class String includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase.

The Java language provides special support for the string concatenation operator ( + ), and for conversion of other objects to strings.

## Key String Methods

The following methods are based on the Oracle String class documentation available at: <https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>

### charAt()

Returns the character at the specified index (first char is at index position zero).

* **Syntax:** char charAt(int index)
* **Example:** String sample = "peter piper picked a peck";

char firstE = sample.charAt(1); // returns 'e'

### concat()

This creates another String by joining the current String to the String given as the argument.

* **Syntax:** String concat(String str)
* **Example:** String first = "Java is ";

String second = "great!";

String third = first.concat(second);

Neither the calling String, first, nor the argument are changed. A similar result is obtained using the + operator:

String third = first + second;

### equals()

Compares this string to another string object and returns true if both contain the same sequence of characters.

* **Syntax:**   boolean equals(String anotherString)
* **Example:** String example = "BOB";

boolean b = example.equals("BOB");

System.out.println(b);

### equalsIgnoreCase()

Compares this string to another string ignoring case of letters.

* **Syntax:**   boolean equals(String anotherString)
* **Example:** String example = "BOB";

boolean b = example.equalsIgnoreCase("bob"); System.out.println(b);

### indexOf()

This method returns the position (index) of the specified char argument. The index is counted from zero at the first character. If the character in the argument is not found, the method returns -1.

* **Syntax:** int indexOf(char character)
* **Example:** String fullName = "James Fairbairn";

int a = fullName.indexOf('F');  // returns 6

int b = fullName.indexOf('z');  // returns -1

There is another similar method: int lastIndexOf(char character)

### indexOf() With 2 Parameters

A second version (overloaded) of the method has 2 parameters. This version searches the calling string starting from the position of the second argument.

* **Syntax:** int indexOf(char character, int fromIndex)
* **Example:** String fullName = "James Fairbairn";

System.out.println(fullName.indexOf('a', 3));

### length()

Returns the length of this string. Spaces count too!

* **Syntax:** int length()
* **Example:** int result;

String name = "James Fairbairn";

result = name.length();

// Simpler use would be

System.out.println("Length: " + name.length());

### toLowerCase()

This returns a new String turning any uppercase characters in the calling String to lowercase. All other characters remain the same. The state of the calling String object is not modified – you must assign the return value.

* **Syntax:** String toLowerCase(char character)
* **Example:** String capitalName = "BOB";

String smallerName = capitalName.toLowerCase();

System.out.println(smallerName); // bob

System.out.println(capitalName); // BOB (is unchanged)

### toUpperCase()

This returns a new String turning any uppercase characters in the calling String to lowercase. All other characters remain the same. The state of the calling String object is not modified – you must assign the return value.

* **Syntax:** String toUpperCase(char character)
* **Example:** String smallerName = "bob";

String capitalName = capitalName.toUpperCase();

System.out.println(smallerName); // bob (is unchanged)

System.out.println(capitalName); // BOB

### startsWith() and endsWith()

Checks (tests) if this string starts with the specified prefix or ends with specified suffix. Both take a String argument and return a Boolean (true for a match).

* **Syntax:** boolean startsWith(String prefix)

boolean endsWith(String suffix)

* **Example:** String fullName = "James Fairbairn";

boolean a = fullName.startsWith("Jam");

boolean b = fullName.endsWith("bairn");

### substring()

This extracts a chunk (substring) from a String. Returns the substring made from the character number given as the first argument, up to, but excluding the character as the second argument. Note the first character in a String is 0.

* **Syntax:** String substring(int beginIndex, int endIndex)
* **Example:** String fullName = "Steven James Mead";

String middleName = fullName.substring(7, 12);

## Paper Exercise Expected Outputs

Paper exercise so **no programming** (initially). What output would you expect from the following?

### Expected Output One

String redHerring = "Baa, baa, black sheep";

int a = redHerring.length();

### Expected Output Two

String festival = "Christmas";

System.out.println(festival.indexOf('s'));

System.out.println(festival.lastIndexOf('s'));

### Expected Output Three

String fastfood = "pepperoni and mushroom pizza";

System.out.println(fastfood.substring(0, 6));

System.out.println(fastfood.substring(14,18));

### Expected Output Four

String first = "Java is great!";

String second = "Java is ";

String third = "great!";

second = second.concat(third);

boolean result = first.equals(second);

### Name the Method

Consider a String with a first, middle and last name e.g. “Kristin Scott Thomas”.

How can you use String methods to extract the middle name from the String?

## Document History

Revision 0 (9-Oct-22): This is the initial version of the 2022/23 strings document.