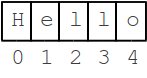
**Java Substring v2**

* Simple substring(start) v1 -- previous video
* str.substring(start, end)
* Chars beginning at **start**
* Up to but not including **end**

There is a more complex version of substring() that takes both start and end index numbers:**substring(int start, int end)** returns a string of the chars beginning at the start index number and running up to but not including the end index.



String str = "Hello";

String a = str.substring(2, 4); // a is "ll" (not "llo")

String b = str.substring(0, 3); // b is "Hel"

String c = str.substring(4, 5); // c is "o" -- the last char

The c example above uses substring(4, 5) to grab the last char. The 5 is one more than the index of the last char. However, this does not go out of bounds because of the substring() "up to but not including" use of the end index. Incidentally, the length of the resulting substring can always be computed by subtracting (end - start) -- try it with the examples above.

CodingBat Practice> [missingChar](http://codingbat.com/prob/p190570)

## String Index Errors: "String Index Out Of Bounds" or "String Index Out Of Range"

* Common mistake: index greater than length
* Index Out of Bounds Error
* If-statement check length first

It is very common to get little errors with the index numbers fed into substring(). The valid index numbers for substring are basically 0, 1, 2, ... str.length(), so code needs to be careful not to pass in numbers outside that range. Note that the last number, str.length(), is one beyond the end of the string. You need this number to fit the "up to but not including" way that substring() works. For the above "Hello" examples, the valid index numbers are always in the range 0..5 since the length of "Hello" is 5.

Often avoding substring() out of bounds errors involves adding logic to check the length of the string. For example, suppose we want to take the first 4 chars of a string, like this...

// Suppose we want the first 4 chars of str

String a = str.substring(0, 4); // WRONG error sometimes

The problem with the above is .. what if the str length is less than 4? In that case, substring(0, 4) refers to non-existent chars and will fail wen run. One possible solution will add if-logic like this:

if (str.length() >= 4) {

a = str.substring(0, 4);

}

else {

// whatever you want to do when length is < 4

}

The point: don't assume that a string is long enough, check the length() before calling substring()

CodingBat Practice> [front3](http://codingbat.com/prob/p136351)

More problems to try

* CodingBat Practice> [right2](http://codingbat.com/prob/p130781)
* CodingBat Practice> [withoutEnd2](http://codingbat.com/prob/p174254)

[CodingBat.com](http://codingbat.com/) code practice.