**Java String indexOf Parsing**

* str.indexOf(String target) -- searches left-to-right for target
* Returns index where found, or -1 if not found
* Use to find the first (leftmost) instance of target
* str.lastIndexOf(String target) -- searches right-to-left instead

The indexOf(String target) method searches left-to-right inside the given string for a "target" string. The indexOf() method returns the index number where the target string is first found or -1 if the target is not found. Like equals(), the indexOf() method is case-sensitive, so uppercase and lowercase chars are considered to be different.

So basically, if you were going to write a for-loop to iterate over a string and look for a string, indexOf() can just do it for you. Note that indexOf() works best if you want to find the **first**instance of the target. If you want to find all the instances, see the next section.

String str = "Here there everywhere";

int a = str.indexOf("there"); // a is 5

int b = str.indexOf("er"); // b is 1

int c = str.indexOf("eR"); // c is -1, "eR" is not found

**String lastIndexOf()**

There is also a lastIndexOf(String target) variant that searches for the target from right to left, instead of the default left-to-right.

CodingBat Practice> [doubleX](http://codingbat.com/prob/p186759)

**String indexOf() lastIndexOf() With fromIndex**

There are versions of indexOf() lastIndexOf() that take a fromIndex to specify where the search should begin.

**int indexOf(String target, int fromIndex)** -- searches left-to-right for the target as usual, but starts the search at the given fromIndex. The fromIndex does not actually need to be valid. If it is negative, the search happens from the start of the string. If the fromIndex is greater than the string length, then -1 will be returned.

**int lastIndexof(String target, int fromIndex)** -- does a right-to-left search, beginning at the given fromIndex.

Using these, it is possible to write a loop to find **all** the instances of the target in a string, not just the first. However, it's kind of complicated (example below).

**indexOf Trap**

Note that using indexOf() this way to find all the instances of a target string is actually **more complicated** than just doing the search with the standard for-loop. I think people are kind of in love with indexOf() as they think they are getting something for nothing, but it's actually kind of a mess. You be the judge!

Here is an example that calls indexOf() in a loop to find all the "OOP" in string -- note how the "start" variable is used in the loop to keep resetting the start point of the search:

// Example using indexOf() to find all the instances of "OOP" in a string

void findOOP(String str) {

int start = 0;

while (true) {

int found = str.indexOf("OOP", start);

if (found != -1) {

// Found one -- do whatever here

}

if (found == -1) break;

start = found + 2; // move start up for next iteration

}

}

**String Parsing**

"Parsing" is the process of taking a string from a file or the network or the user, and processing it to extract the information we want. A common strategy is to use a few calls to indexOf() to figure out the index numbers of something interesting inside the string. Then use substring() to pull out the desired part of the string. After processing the data, we can use + to reassemble the parts to make a new string.

Suppose we have a string that contains some text with a pair parenthesis somewhere inside of it, like this: "It was hot (darn hot!) I'm telling you". Suppose we want to fix the string so that the part in parenthesis is in all upper case. We can use two calls to indexOf() to find the '(' and ')', and substring to extract the text. Example code:

String string = "It was hot (so hot!) I'm telling you.";

int left = string.indexOf("(");

int right = string.indexOf(")");

// pull out the text inside the parens

String sub = string.substring(left+1, right); // sub is "so hot!"

sub = sub.toUpperCase(); // sub is "SO HOT!"

// Put together a new string

String result =

string.substring(0, left+1) + // It was hot (

sub + // SO HOT!

string.substring(right); // ) I'm telling you.

// result is "It was hot (SO HOT!) I'm telling you."

**String Code Examples**

Here are a few string code examples of loops and the string methods: length(), +, substring(), charAt(), equals(), and indexOf().

**Example 1**

Given a string, returns a string made of repetitions of that string. This code demonstrates using + to assemble a larger string in a loop:

public String repeat(String string, int count) {

String result = "";

for (int i=0; i<count; i++) {

result = result + string;

}

return result;

}

**Example 2**

Given a string, returns how many '!' chars it contains. This code demonstrates using a for loop, calling length(), substring()/charAt(), and equals() to look at all the chars in a string:

public String countExclaim(String string) {

int count = 0;

for (int i=0; i<string.length(); i++) {

if (string.substring(i, i+1).equals("!")) count++;

// if (string.charAt(i) == '!') count++; // or with charAt()

}

return count;

}

**Example 3**

Given a string where the string "OOP" appears at least two times, find the first and last OOP in the whole string. Return the text from between the two OOP. Demonstrates using indexOf() and substring():

public String oopPair(String string) {

int start = string.indexOf("OOP");

int end = string.lastIndexOf("OOP");

return string.substring(start+1, end);

}

**Example 4**

Suppose you have a string like this:"Once there was a woman name:angelina: and a man name:tony: and their friend name:jane: and ...". Inside of a long text there are little "name:" sections. Write code to find and print all the names. Demonstrates a more complex use of indexOf() and substring() in a loop to parse a string:

public void printNames(String string) {

int i = 0;

while (true) {

int found = string.indexOf("name:", i);

if (found == -1) break;

int start = found + 5; // start of actual name

int end = string.indexOf(":", start);

System.out.println(string.substring(start, end));

i = end + 1; // advance i to start the next iteration

}

}

[CodingBat.com](http://codingbat.com/) code practice. Copyright 2012 Nick Parlante.