

Finding a Gene in DNA

▶	What is a String	2 min
▶	Understanding Strings	3 min
▶	Developing an Algorithm	5 min
▶	Positions in Strings	8 min
▶	Translating into Code	11 min
▶	Java Math	8 min
📄	Programming Exercise: Finding a Gene and Web Links	10 min
★	Practice Quiz: Finding a Gene in DNA	6 questions

Finding All Genes in DNA

Debugging Code

Using the StorageResource Class


Review

Before Proceeding, Some New Concepts

- How do we find ATG in a String?
 - How do we represent its position?
- How do we get all the letters in a range?

The first of them is how we can represent the position of something in a String.

Positions in Strings



Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript

English

0:03

To move forward implementing a gene finding program, we will explore a couple of important String topics in this video. [The first of them is how we can represent the position of something in a String.](#)

0:14

To answer this question we return to the recurring concept that everything is a number.

0:20

That is, we will give each position in the String a numeric index. Notice that these numbers start at 0 in the first position, not 1. This may seem a bit odd since we usually start counting from one, not zero. However, many programming languages start numbering sequences of things, such as Strings, from zero because it makes some tasks easier as we shall see later.

0:45

These numbers which describe the positions in the String are called indices or indexes. Either word is okay as an okay plural of index.

0:55

For example, if I wanted to talk about this e, I might say, the letter at index 3 is e.

1:03

We have now answered the first question. We can represent the position with a number, and we can talk about the index in the String where we find the ATG.

1:13

Now it is time to answer the second question. How could we get all the letters in a particular range,

1:19

which we could now be a bit more precise, and say between two particular indices?

1:26

One option would be for you write your own algorithm to do this. However, you can also use a built-in method of the String class.

1:34

Whenever there is a built-in method to accomplish a particular task, it is better to use it than write your own. Not only does it save you work but the built-in method has already been heavily tested by expert programmers. So you can be very confident that it works correctly.

1:50

For this particular task, you want to use the built-in substring method. However, before we show you how to do that, let's take this example String and make it an actual Java string assigned to a variable.

2:04

Here we have a variable declared with the name s of type String and an equal sign to make an assignment statement. And a semicolon at the end of the line to end the statement. However this isn't quite correct yet.

2:20

We also need to put the String literal in quotation marks as shown here. If we did not put these and just wrote the word. Java would think dukeprogramming was the name of a variable and give us an error that it is undefined. By putting the text in quotation marks, Java knows that we want a String with that literal text.

2:41

So now we have a valid statement which makes the variable s be the String dukeprogramming.

2:49

Next, you can see an example of using the substring method. Here, we declare another variable of type String called x, and assign it to the result of s.substring (4,7). What do these numbers mean, and what does this method call do?

3:08

The first number specifies the index in s from which we want to start making our substring. The letter at this index will be included in the resulting String.

3:19

The second number specifies the index in s where we want to stop making our substring. The letter in this index will be excluded from the resulting String. The method will stop right before it gets to that letter.

3:33

This may seem odd, why would you want to specify the index after where you want to stop? There are a variety of reasons why this method and many others are designed this way. But one nice reason is that the length of the resulting String will be the difference between the two numbers. 7 minus 4 is 3, so we will get a three-letter String as our answer.

3:54

In particular, you will end up with this three-letter String, made up of the letters from the indices 4, 5, and 6 of s.

4:04

So x will be the String pro.

4:09

While you are learning about this built in method of String, let's take a moment to talk about a few other useful methods, and what they do. You just saw substring and learned how it gives you the letters in a particular range of indices. Another useful method is .length, which tells you how many characters are in the String. The String s has length 15. Notice that for a String of length 15 the valid indices are 0 through 14. If you try to access an index outside of this range your program will have an error with a String out of bounds exception.

4:48

Another really useful method is .indexOf. You pass this method another String and it tries to find the first occurrence of that String within the String you called the method on. For example, here we have asked the indexOf method to find the String program within the String s.

5:08

You can see that the first occurrence of program is right here, since it starts at index 4. That is the value that the method call would return.

5:18

Here's another call to indexOf, s.indexOf("g"). Can you figure out what this would return?

5:27

This will result in 7 since the first occurrence of the String g starts at index 7.

5:35

If you call .indexOf on a String that is not found, such as .indexOf("f"), it returns -1. You can also give .indexOf a second parameter specifying the index to start searching from.

5:52

Here we have passed 8 as a second parameter. So the .indexOf method will ignore characters in the indices 0 to 7, as it searches for g. It will then find this g, which is the first one when you start looking from index 8, so then this method call evaluates to 14. Another useful method is .startsWith, which tells you if a String starts with another String. Here you can see that s starts with duke, so this method call evaluates to true.

6:27

Likewise, .endsWith checks to see if a String ends with another String. s does not end with king, so this method call evaluates to false.

6:38

Wow, that was a lot of methods, and a bunch of information. How would you know all of this without us telling you about every method in the String class? And should you be memorizing all of these details? Of course not! Programming is not about memorization.

6:54

Although, as you program a lot, methods that you use commonly will naturally become familiar.

6:59

Instead, you should learn how to make use of the language documentation, which describes all of the built-in classes and their methods. If you search in the internet for java String, your first result will probably be something from docs.oracle.com. Oracle is the company that makes Java and docs.oracle.com is the website where they host the language documentation. If you click on this link you will end up with a page which tell you all about the String class. If you scroll down a bit, you will find a rather long list of all the built-in methods in String. Here are a few of them, including a few that you have just learned about, .indexOf and .length. These entries give a brief description of the methods. And if you click on one of the method names, you will get a more detailed description of what that method does. There is also a documentation page on the course site at dukelearntoprogram.com. That page has simplified documentation of some important methods for quick reference. Great! Now you not only know about indices and Strings and some useful built in methods but also how to learn about other methods when you need them.

Downloads

Lecture Video	mp4
Subtitles (English)	WebVTT
Transcript (English)	txt

Would you like to [help us translate](#) the transcript and subtitles into additional languages?