In the for loop, if **k+=1** is changed to **k+=2**, then what is the result of the following call?

```
System.out.println(reverse("computer"));
```

- optr
- retupmo
- cmue
- rtpo
- eumc

Correct

Continue

8:22 / 9:28

Counting Loops

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

Downloads

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

Would you like to help us translate the transcript and subtitles into additional languages?

Interactive Transcript

Search Transcript | English

0:03
Hi, as you get ready to solve problems that are slightly more complex, you'll learn some new programming constructs to help. In this video you'll learn about counting loops. You've used different kinds of loops in solving problems before. You used a while(true) loop in finding codons in DNA or tags in a web page. And you have used a for each loop within iterable many times. As an reading lines in a file or processing pixels in an image. You've also used indexes to access and reference parts of a string. You know that indexes start with zero in Java and with other many languages. So, when you use .indexOf you know it returns zero when a match is found at the first character in a string. Both .indexOf and .substring use indexes to access the elements or characters of a string.

0:56
We'll look at code to access individual characters in a string. We'll do this in the context of looking at the reverse of a string. Where the reverse of CGATTA is ATTAGC and the reverse of TIP, T-I-P is PIT, P-I-T.

1:15
Scientists studying genomes often have to look at a string backwards to analyze the DNA.

1:24
It can also be a source of fun or word play to look at phrases that are palindromes. The same read backwards as forward. Here's a sentence in Russian that you can see reads backwards and forwards, if you understand the cyrillic alphabet. It means, the boar pressed the eggplant.

1:42
In Spanish, we have a palindromic sentence that means, there jogs the tortilla.

1:49
While in French, the sentence [FOREIGN] means. And, how is the cow?

1:55
Finally, we see an English sentence. Draw, O Caesar, erase a coward. Which is particularly appropriate as we implement a Caesar cipher in this module. We'll use a new loop, the counting loop, to reverse a string.

2:10
Indexing a string in a loop can be done in several ways, but we'll look at a very common approach. We must understand a loop with three parts. Each part is separated from the other parts by a semi-colon. As you can see in the code on the slide. The first part of the loop is called the initialization. Here the variable k is assigned a value 0. And this happens once before the loop guard and the loop body are executed.

2:37
The loop guard is evaluated each time, before the loop block or body maybe executed. When the loop guard is true, the body is executed. And when it's false the loop executes. Sometimes the loop guard is called the loop test. The increment here happens after all the statements in the loop body are executed. After the increment the loop guard is evaluated again to see if the loop continues or exits. We'll look at this more closely. To understand the for loop, we'll compare it to the while loop that you've seen before, though not with this precise style of counting in a loop. The for loop does not provide more power than a while loop. Or allow you to solve different problems. The for loop is simply syntactic sugar. Or a dressing up of a while loop before loop puts all the parts in one place. And many programmers think this makes the loop easier to write and to read.

3:35
As we've discussed, the initialization happens once before the loop guard is tested. You can see in the comparison here how initialization happens before the while loop, too.

3:46
The loop guard is evaluated to see if the loop body will execute. When the loop guard is false, the loop is over. Both the while and the for loop exit when this loop guard or test is false.

4:00
When the guard is true, the loop body executes, and as the last statement in the body, we see the increment statement. Which will execute.

4:09
We'll trace through the execution of a for loop in a particular example of a reverse function to better understand the loop. And to trace this, we'll look at the call reverse pit. This means the value of parameter s is the string pit.

4:26
The local variable r, or ret, will accumulate the reversed string. It's initialized to the empty string before the loop. As we trace through the code, the green arrow indicates the statement that will execute next.

4:39
The loop index, or control variable k, is initialized to 0. Remember that the loop initialization only happens once in a for loop. And the variable k is only accessible within the loop, but not after the loop. Its scope is limited.

4:54
When the loop guard is checked, the value of k, which is 0, is less than s dot length, which is three since pit has three characters. So loop guard evaluates to true. Loop guards are always boolean expressions. The loop body now executes. And the string method charAt acts as a character at a specific index. We should point out here that some people say charAt, the way I do, and some people say charAt either one is fine. Since k is 0, the expression s.careAt(0) evaluates to p.

5:31
We've shown the character 'p' with single quotes which is used in Java to indicate the primitive type care. The value of red is shown as the empty string in double quotes. Because these double quotes in Java indicate a string literal. Concatenating the character p to the empty string yields a new string p. The variable ret will be assigned the value p. The string variable changes and it's no longer pointing to the empty string as it used to. Remember that strings in Java are immutable. We can create new ones, but we can't change a string.

6:10
The increment executes after the loop body. This changes k, so that it has the value 1. After the increment statement, we're ready to trace the next iteration of the loop. The local variable ret has the value p. The loop control variable k has the value 1 and we're ready to continue the trace.

6:30
K has the value 1. The length of s, the string pit, is three. And so the loop body executes, since the guard evaluates to true.

6:39
Remember that the .carat method accesses the kth character. Here that's the character i, whose index is 1. The character i is prepended via string concatenation to ret, which is p, and this creates a new string, ip. The assignment statement changes ret, so that references this new string. And now the loop increment will execute, the value of k is 2, and the loop will continue to execute.

7:09
We'll now trace through the last iteration of the loop. The local variable ret has the value ip. The loop variable k has the value 2 and as we can see here the loop guard will be evaluated. Since 2 is less than 3 the guard is true and the loop body executes. S.charAt t evaluates to t, as you can see here. The character t is prepended to the string ip to form the string tip.

7:42
Variable ret now references tip and the loop continues. The increment statement executes which changes k to have the value 3. Now the loop guard will be evaluated again.

7:58
As the loop guard is evaluated here, you can see that the value of k is 3, and the length of s is 3 as well.

8:06
Since 3 is not less than 3, the loop guard is false. Control in the program continues to the statement after the loop. The value of ret is tip, the reverse of the string parameter s, so tip will be returned.

8:23
It's good to know that you'll see others write code and you should understand that. Many programmers use i as a loop control or index variable. Some programmers think the letter i is hard to distinguish from the number 1. But i is more common than k in reading other code.

8:42
Many programmers use the post-increment operator i++ instead of i plus gets 1. We won't explain the nuances of i++ here, but it's a very common idiom in using loops. And it's fine to use i++ by itself, in the loop increment. Sometimes it's useful to define the loop index variable before the loop, rather than inside the parentheses of the loop. This allows the value of i to be referenced or accessed after the loop is over. When the variable is defined within the parenthesis of the loop, that loop control variable can only be referenced within the loop body, but not after the loop.

9:23
Have fun programming and programming and programming as you loop and loop and loop.