

Working with Strings in Java



String methods



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/)
by Christine Alvarado, Mia Minnes, and Leo Porter, 2015.

By the end of this video you will be able to...

- Work with the String class's built-in methods to manipulate Strings

Strings can do lots of things!

<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>

length

toCharArray

indexOf

charAt

split

$$\text{Flesch score} = 206.858 - 1.015 \left(\frac{\# \text{ words}}{\# \text{ sentences}} \right) - 84.6 \left(\frac{\# \text{ syllables}}{\# \text{ words}} \right)$$

Counting the number of syllables will involve looking at the characters in a String.

```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i) == letter)
        {
            return true;
        }
    }
    return false;
}
```

**Loop over the indexes
of character array in
the string**

```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i) == letter)
        {
            return true;
        }
    }
    return false;
}
```

**length() returns the
number of characters
in the String**

```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i) == letter)
        {
            return true;
        }
    }
    return false;
}
```

**Get each letter and
compare it to the
char in question**

```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i) == letter)
        {
            return true;
        }
    }
    return false;
}
```

**charAt(i) returns the
char at index i in the
String**


```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i) == letter)
        {
            return true;
        }
    }
    return false;
}
```

**If no letters match,
return false.**

```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i) == letter)
        {
            return true;
        }
    }
    return false;
}
```

```
public static boolean hasLetter(String word, char letter)
{
    for (char c : word.toCharArray())
    {
        if (c == letter)
        {
            return true;
        }
    }
    return false;
}
```

toCharArray returns
the chars in a String,
as a char[]

**Same method, using
a for-each loop**

```
public static boolean hasLetter(String word, char letter)
{
    for (char c : word.toCharArray())
    {
        if (c == letter)
        {
            return true;
        }
    }
    return false;
}
```

```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i).equals(letter))
        {
            return true;
        }
    }
    return false;
}
```

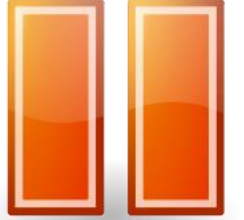
**<IVQ Placeholder
(NOT PI)>**

**Is it OK to replace the
== with .equals?**

```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i) == letter)
        {
            return true;
        }
        else {
            return false;
        }
    }
    return false;
}
```

**<IVQ Placeholder
(NOT PI)>
Does this method
work the same way?
(SUPPORT VIDEO ON
THIS)**

```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i) == letter)
        {
            return true;
        }
    }
    return false;
}
```



Change this method so that it returns the index where it first finds letter (or -1 if it doesn't find it)?

```
public static boolean hasLetter(String word, char letter)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word.charAt(i) == letter)
        {
            return truei;
        }
    }
    return false-1;
}
```

**built-in String method
indexOf(String str)
does exactly this, but with
a String to match.**

String method
`indexOf(String str)`

index position 8 in the character array



```
String text = "Can you hear me? Hello, hello?"  
int index = text.indexOf("he"); // index is 8
```

String method indexOf(String str)

index position 17 in the character array



```
String text = "Can you hear me? Hello, hello?"  
int index = text.indexOf("he");    // index is 8  
index = text.indexOf("He");        // index is 17
```

For dealing with case, check out String methods:
equalsIgnoreCase, toLowerCase, toUpperCase

String method indexOf(String str)

```
String text = "Can you hear me? Hello, hello?"  
int index = text.indexOf("he");    // index is 8  
index = text.indexOf("He");        // index is 17  
index = text.indexOf("Help");      // index is -1
```

$$\text{Flesch score} = 206.858 - 1.015 \left(\frac{\# \text{ words}}{\# \text{ sentences}} \right) - 84.6 \left(\frac{\# \text{ syllables}}{\# \text{ words}} \right)$$

Counting the number of words will involve splitting apart the String.

String method split(String pattern)

```
String text = "Can you hear me? Hello, hello?"  
String[] words = text.split(" ");
```



String method split(String pattern)

What if we add an extra space
here

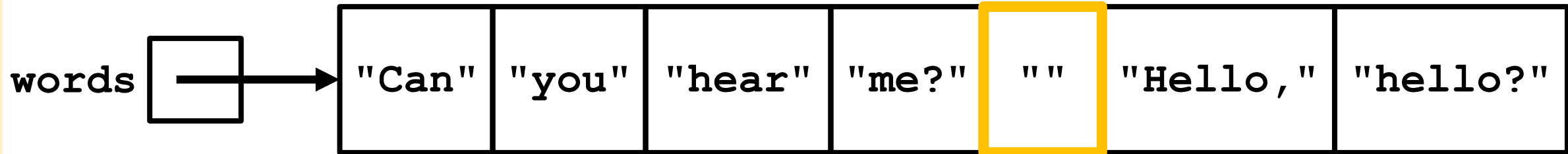


```
String text = "Can you hear me? Hello, hello?"  
String[] words = text.split(" ");
```

String method split(String pattern)

What if we add an extra space
here

```
String text = "Can you hear me? Hello, hello?"  
String[] words = text.split(" ");
```



String method split(String pattern)

```
String text = "Can you hear me? Hello, hello?"  
String[] words = text.split(" ");
```

split(String regex)

Splits this string around matches of the given **regular expression.**