Splitting a String

In Java, ofter you have to take a long string (maybe an entire document) and split it into words or n-grams. Let's look at how to do that.

In English, a sentence is a series of words. A word is a series of characters separated by spaces. For example, in the sentance "We hold these truths to be self-evident", the words are "We", "hold", "these", "truths", "to", "be", and "self-evident".

And the input would look like a string:

```
String text = "We hold these truths to be self-evident";
```

The goal is to split the string into an arraylist of strings, well, "words". And the result would be a list of words:

```
List<String> words = ["We", "hold", "these", "truths", "to", "be", "self-evident"];
```

But how do we do that in Java?

Using a for loop (ugh)

One way to split a string into words is to use a`for` loop and examine each character in the string. When we encounter a space, we know that we have reached the end of a word. We can then extract the word and add it to a list of words.

Here's an example of how you might do this:

```
import java.util.ArrayList;
import java.util.List;

public class WordSplitter {

    public static List<String> splitWords(String text) {
        List<String> words = new ArrayList<>();
        StringBuilder word = new StringBuilder();

        for (int i = 0; i < text.length(); i++) {
            char c = text.charAt(i);
            if (c == ' ') {
                words.add(word.toString());
                word = new StringBuilder();
            } else {
                word.append(c);
            }
        }

        if (word.length() > 0) {
            words.add(word.toString());
        }
        return words;
    }

    public static void main(String[] args) {
        String text = "We hold these truths to be self-evident";
        List<String> words = splitWords(text);
        for (String word : words) {
            System.out.println(word);
        }
    }
}
```

This code defines a class `WordSplitter` with a method `splitWords` that takes a string `text` and returns a list of words. It uses a `for` loop to iterate over each character in the string, building up a word until it encounters a space. When it encounters a space, it adds the word to the list of words and starts building a new word.

Notice, the `if` statement in the `for` loop checks if the character is a space. If it is, the current word is added to the list of words and a new `StringBuilder` is created to start building the next word.

The `if` statement after the `for` loop checks if there is a word left to add to the list of words. If there is, it adds it.

The `main` method creates a string `text` and calls the `splitWords` method to split the string into words. It then prints each word to the console.

This is a simple and straightforward way to split a string into words, but it's not the most efficient. It creates a new `StringBuilder` for each word, which can be slow for long strings.

Using the `split` method

A more efficient way to split a string into words is to use the `split` method provided by the `String` class. The `split` method takes a regular expression as an argument and splits the string into an array of substrings based on the regular expression.

Here's an example of how you might use the `split` method to split a string into words:

```
import java.util.Arrays;

public class WordSplitter {

    public static void main(String[] args) {
        String text = "We hold these truths to be self-evident";
        String[] words = text.split("\\s+");
        System.out.println(Arrays.toString(words));
    }
}
```

Much shorter, eh? The `split` method takes a regular expression as an argument. In this case, the regular expression `\\s+` matches one or more whitespace characters. So the `split` method splits the string `text` into an array of words based on spaces.

(yeah, we will talk about *regular expressions* later. For now, just know that `\\s+` means "one or more whitespace characters")

(Yeah but, what's "whitespace characters" mean? Well, it's a space, a tab, or a newline. Basically, any character that doesn't print anything on the screen. But are important for formatting text and humans reading it.)

The `split` method returns an array of words, which is then printed to the console using `Arrays.toString`.

You could print out tht array of words one by one, but that's not as fun as using `Arrays.toString`. If you used a foreach loop, you'd have to write more code. And who wants to do that? But here is how you could do it:

```
for (String word : words) {
    System.out.println(word);
}
```

Or you could use a for loop:

```
for (int i = 0; i < words.length; i++) {
    System.out.println(words[i]);
}
```

That way is fun because it shows you understand how arrays work. But it's not as fun as using `Arrays.toString`. So, you know, do what you want.

You could even use a `while` loop:

```
int i = 0;
while (i < words.length) {
    System.out.println(words[i]);
    i++;
}
```

Here you really have to remember to increment `i` or you'll be stuck in an infinite loop. But that's fun, right? No, it's not. It's not fun at all. But you are *newbie* so what's an infinite loop between friends?

Conclusion

Splitting a string into words is a common task in Java programming. You can use a `for` loop to iterate over each character in the string and build up words, or you can use the `split` method to split the string based on a regular expression.

The `split` method is more efficient and easier to use, but the `for` loop gives you more control over the splitting process. You can choose which method to use based on your needs and preferences.

And we spent some time looking at different ways to step through an array. You can use a `for` loop, a `foreach` loop, a `while` loop, or a `do-while` loop. Each has its own strengths and weaknesses, and you can choose the one that best fits your needs.

And that's it. You've learned how to split a string into words in Java. Congratulations! You're well on your way to becoming a Java programmer. Keep practicing and learning, and you'll be writing Java code like Kristofer or Dolio in no time. (heh, heh, heh)

Exercises

1. Write a program that takes a string as input and splits it into words. Print each word to the console.
2. Modify the program to split the string into words and count the number of words in the string. Print the number of words to the console.
3. Write a program that takes a string as input and splits it into words. Count the number of occurrences of each word in the string and print the results to the console. Use a `Map` to store the word counts. Pretend the input looks like this: "In Java, we use the keyword for for a number of things like for loops, for each loops, inifinite loops, and for fun."
4. Write a program that takes a string as input and splits it into words. Count the number of occurrences of each word in the string and print the results to the console. Use a `Map` to store the word counts. Pretend the input looks like this: "In Java, we use the keyword for for a number of things like for loops, for each loops, inifinite loops, and for fun." But this time, sort the words by frequency, with the most frequent words first. Yes, go ahead lookup the word `frequency` to be sure what it means. (Big hint, the first words would be "for" which occurs 4 times, and "loops" which occurs 3 times.) (oh just do the rest of the words, too. It's good practice.)