Java API spec for class String

Here is the Java API webpage for class String. Each object contains a list, or string, of characters. Any string literal in your, like "abc" is maintained in a object of class String.

This description tells us two important things. First, String objects are immutable; they cannot be changed.

Second, a String is implemented as an array of values of type char. You don't yet know about arrays in Java, but they are much like arrays in other programming languages. So, here's how we might represent the array containing the String "bcdefgh". Knowing that a String is implemented in an array can help you estimate how fast various operations are. For example, to find the index of a character may take time proportional to the length of the string since each character may have to be looked at to find c in the array.

When talking about a string s, we often use the notation s[k] to denote the character at index k. This notation is useful, even if it is not Java. In the same way, s[h..k] denotes the substring of s that contains the characters s[h], s[h+1], …, s[k]; s[h..] denotes the substring of s starting at h and going to the end; and s[h..h-1] denotes the empty string starting at position h.

At this point, take 5 minutes to look through the Method Summaries and Details of the following String methods, because you will be using them in writing methods that manipulate strings. Some of them are overloaded.

Instance methods

length	charAt	substring		
indexOf lastIndexOf	indexOf lastIndexOf			
startsWith	startsWith	contains	compareTo	equals
trim	replace			
toLowerCase()	toUpperCase()			

Static methods:

valueOf

There are two important points to remember.

1. s.substring(h, k) yields a new string that contains elements s[h..k-1]; it does *not* contain s[k].
2. indexOf methods return -1 if the character or string begin searched for does not there.

We now develop a function to show you how we tend to develop code, thinking in terms of English, not Java, and translating into Java when we know what is to be done. The silly function we develop is designed to use many of the functions of class String

```
/**String s contains a list of at least one name.
   Adjacent names separated by one or more space characters.
   If there is only one name, return it; otherwise,
    return the first name but with its first character placed at the end and with all chars in upper case. */
public static char getName(String s)
```

For example, f(" Abe") is "Abe" and f("Abe Lincoln ") is "EAB".

To start, let's remove the spaces at the beginning and end of s. Is there a function that can help us do this? Yes!

trim(). But be careful, the call trim(s) does not *change* s but produces a new String with surrounding spaces removed, so we have to assign the result to s.

Now we have to determine whether there is only one name. That's the case if there are no space characters, or blanks, in s. If there are two or more names, there is a blank after the first name. So let's look for the first blank. We use function indexOf and store the result in a new local variable k.

Now, if k is –1, s contains no blank, so we can return s.

Otherwise, we have to construct a string consisting of the last character of the first name, which is just before the blank character s[k], catenated with everything that comes before, s[0..k-2], but all in upper case. Now let's translate this into java. First comes s[k-1] —we use function charAt. Next comes the substring s[0..k-2] —we use function substring, but note that the second parameter is k–1 and not k–2 because of the way substring is defined. Now, we have to make all those characters into upper case —we use instance function toUpperCase. We are done.

Note that the code works if the first name is a single character. That's because s[0..-1] and s.substring(0, 0) are the empty string "".

```
/**String s contains a list of at least one name.
    Adjacent names separated by one or more space characters.
    If there is only one name, return it; otherwise,
     return the first name but with its last character placed at the beginning and all chars in upper case. */
public static char getName(String s)
    // Remove spaces at beginning and end
    s= trim();

    // Store in k the index of the first blank in s (-1 if none)
    int k= s.indexOf(' ');
    if (k == -1) return s.charAt(0);

    // return s[k–1] + s[0..k–2], all in upper case
    return (s.charAt(k–1)  + s.substring(0, k–1)).toUpperCase();

}
```