Return to Module Overview Page (https://onlinelearning.berkeley.edu/courses/665040/pages/module-two)

2.7: Strings

Strings in Java are *not* arrays of characters. They are examples of the mystical creatures called "objects" that we briefly encountered last week. They've got a few interesting properties we need to cover now, even though we're putting off talking about objects themselves.

You can create Strings this way:

```
String s = "A lovely String";
```

Strings know how long they are, just like arrays. For Strings, length() is a function:

```
//len gets 15
int len = s.length();
```

Note that a double-quoted character string is a String object too:

```
//len still gets 15
int len = "A lovely String".length();
```

A String variable can refer to no String at all, like this:

```
String no_string = null;
```

"null" is a special literal value, like a "0" pointer in C. It's not the same as an empty String:

```
String emptyString = "";

// i gets "0"
int i = emptyString.length();

String nullString = null;

// runtime error!
int j = nullString.length();
```

Another interesting property of Strings is their support for the "+" operator. You can add anything in Java to a String, and the result is a String! For example:

```
int answer = 42;
String message = "The answer is " + answer;
```

This works if any of the operands of the "+" is a String. Be careful of the following mistake:

```
// oops!
String s = "Three plus four is " + 3 + 4;
```

s ends up containing "Three plus fours is 34", because the + in 3 + 4 concatenates the numbers to the String rather than adding them.

You can compare two Java strings for equality using the equals() function:

```
String s = "something";
if (s.equals("something else"))
   thisDoesntGetExecuted();
else
   thisDoes();
```

Although the syntax for equals() may look odd to you, it should soon seem quite familiar, as all Java functions work the same way. I will say that the equals() function is being called "on" or "through" the String s.

You can compare Strings for alphabetical ordering using compareTo():

```
String a = "Albert Aligator";
if (a.compareTo("Zachary Zebra") < 0)
   thisGetsExecuted();
```

String comes before the second one alphabetically; 0 if they are equal; and greater than one if the first String is alphabetically greater than the second. You can read about both of these functions, and many others, on the API documentation page for java.lang.String.