Special Topic A: String

```java
public class StringExample {
    public static void main(String[] args) {
        String foo = "asdf";
        foo += "jkl";
        System.out.println(foo.substring(1));
    }
}
```
String as an Array of Characters

“Hello world!“

Characters: Hello world!
String Literals

- Double quotes at both ends, almost any character in the middle
  - No double quote in the middle!
  - No newlines

“This is a string literal.”
“Hello world!”
“foo is=”
“"
String Literals

- Double quotes at both ends, almost any character in the middle
  - No double quote in the middle!
  - No newlines

“This is a string literal.”
“Hello world!”
“foo is=”
“”

This is the empty string.
It is a String with length zero.
It has exactly zero characters in it.
Escape Strings

- Use backslash \ in string literals to change what the next character means:

  "line 1\nline 2"
  "123\t456"
  "insert \"double quote\" there"
  "just one backslash \\""
Escape Strings

• Use backslash \ in string literals to change what the next character means:

  "line 1 \n line 2"

  "123 \t 456"

  "insert \" double quote \" there"

  "just one backslash \\
backslash"
Concatenation

- The + operator concatenates two Strings (joins them end to end)

"Hello " + "world!"

"concat" + "enation"

"" + "asdf"
Concatenation

- The + operator concatenates two Strings (joins them end to end)

```
"Hello "+"world!"
"concat"+"enation"
""+"asdf"
```

```
"Hello world!"
"concatenation"
"asdf"
```
Concatenation

- Actually, you can concatenate **anything** to a `String`

int foo = ... ;
“foo is: ”+foo

double avg = ... ;
“class average=”+avg
Key Properties of Strings

• Immutable
  - Never change, make copies instead

• Reference Semantics
  - Do NOT compare with == !!!
  - Use .equals() instead
  - Details later
String Variables

• String variables work pretty much like int

```java
String word1 = "foo";
String word2 = "bar";
String word3 = "foo" + "bar";
String msg = "average=" + avg;
msg += " with more to go";
```
String Command-Line Arguments

- We've seen how to read ints from the command line:

```java
int a = Integer.parseInt(args[0]);
```

- It's even simpler with Strings:

```java
String s = args[0];
```
String Methods

Since String is an object, it has methods!

length()
equals(String)
compareTo(String)
charAt(int)
substring(int, int)
toCharArray()
indexOf(char)

...and many more!

Technically, this actually is equals(Object) but we'll ignore that for this semester.

Google for “java documentation String”
length()

To get the number of characters in foo:

String foo = ... ;
int fooLen = foo.length();
equals()

To see if `foo` and `bar` are identical:

```java
String foo = ... ;
String bar = ... ;
if (foo.equals(bar))
{
    ...
}
```

NOTE:
Returns `boolean`
To compare foo and bar and see which comes first:

String foo = ... ;
String bar = ... ;
if (foo.compareTo(bar) < 0)
{
    ...
}

NOTE:
Returns int:

0 means “equal”
< 0 means “less than”
> 0 means “more than”
charAt()

To read a single character out of `foo`:

```java
String foo = ... ;
char c = foo.charAt(2);
```

**NOTE:**
Returns `char`
substring()

To read a range of characters out of foo:

String foo = ... ;
String sub = foo.substring(2,5);

NOTE:
Returns String

The parameters are:
startIndex (inclusive)
endIndex (exclusive)
substring() (version 2)

To read a range of characters out of foo, all the way to the end:

```java
String foo = ... ;
String sub = foo.substring(2);
```

**NOTE:**
Returns String
To convert `foo` to an array of `char`:

```java
String foo = ...;
char[] arr = foo.toCharArray();
```

**NOTE:**
We'll talk about arrays soon.
indexOf()

To find a certain character in foo:

String foo = ... ;
int idx = foo.indexOf('x');

NOTE:
Returns int

Returns index of first instance.

Returns -1 if not found.