Intro to Strings

Lecture 5
CGS 3416 Fall 2015

September 18, 2015

Strings in Java

- In Java, a string is an object. It is not a primitive type.
- The String class is used to create and store immutable strings.
  - Immutable objects are objects that don't change once created.
  - Kinda like "final" primitive types.
- Class StringBuilder creates objects that store flexible and changeable strings.
  - We'll learn this later on in the course.

The String class

- Part of java.lang package
- 13 constructors and close to 50 methods
- String class API from java.oracle.com – full listing of String class features
- Once you build a String object, it is fixed – it cannot be changed.
  - This is easier than it sounds. The only methods that can alter or set the instance variables are the constructors. All other methods that seem to change a string do so by returning a brand new String object
  - You can assign a String reference variable to a new string, discarding the old one

A common way to construct a String

One constructor allows the use of a string literal as the parameter.
Example string constructions:

```
String greeting = new String("Hello, World!");
String name = new String("Marvin Dipwart");
String subject = new String("Math");

// also, a shorthand notation for building strings

String sentence = "The quick brown fox sat around for
                  a while";

// this is not quite equivalent to using the
//constructor above, but you still get a string
//variable (which is what we care about right now)
```

Empty Strings

The constructor with no parameters allows the building of an empty string:

```
String s = new String();
// s refers to an empty string object
```

Note that if you only declare a String variable, but you do not assign it to anything, it is not yet attached to any string:

```
String s1; // s1 does not refer to any string yet
```

The equals() method

equals() – for comparing two strings (i.e. their contents), returns true or false

```
if (str1.equals(str2))
     System.out.print("The strings are the same");
```

equalsIgnoreCase() - just like equals(), except that the case of the letters doesn't matter in making a match. For instance, "Apple" would be equal to "apple" with this method.

Don't try to compare strings by using $==$, $<$, $>$, etc. These would only compare the String reference variables, not the String objects themselves.

The compareTo() method

compareTo() – also for comparing two strings, good for sorting.

```
if (str1.compareTo(str2) <0)
     System.out.print("str1 comes before str2 in
             lexicographic ordering");
else if (str1.compareTo(str2) == 0)
     System.out.print("str1 is the same as str2");
else if (str1.compareTo(str2) >0)
     System.out.print("str2 comes before str1 in
             lexicographic ordering");
```