

Strings and Things

ISE 208 (Intermediate Programming)
SUNY at Stony Brook

Strings Are Fundamental!

- The `String` class is probably the most commonly used class in Java programming
- Let's take a quick look at the methods provided by the `String` class...

String Methods

- `String()` — creates a new `String`
 - ex. `String s = new String("Hello!");`
 - Shorthand: `String s = "Hello!";`
- `length()` — returns the total number of characters in the `String`
- `trim()` — returns a new `String` with no leading or trailing whitespace
- `substring(start, end)` — returns a new `String` containing the characters at positions *start* up to (but not including) *end*

String Methods

- `charAt ()` — returns the character at a given index
 - Indices range from 0 to $(\text{length} - 1)$
- `indexOf (str)` — returns the first index at which *str* occurs in the string (or -1 if it isn't there)
- `toLowerCase ()/toUpperCase ()` — return a *new copy* of the string in lower/uppercase
 - The original string is unchanged; Java strings are *immutable*

Objects vs. Primitives

- Remember the difference between primitive (built-in) types and objects:
 - Primitive variables hold an actual value
 - Object variables (reference types) only hold the *address* of an object!
- This causes problems when we try to compare the values of object variables

Comparing Strings

- Java performs *shallow comparisons* by default (using the == operator)
 - Java looks at the value immediately associated with a variable
- This is okay for primitive types
- This means that Java compares the *memory addresses* contained in reference variables, not their contents!

Equality and Strings

- `String` (like many classes) has methods to test equality based on *content*, not location in memory
- `equals()` — returns true if two strings contain the same character sequence
 - Usage: `firstString.equals(secondString)`
 - `equals()` requires both strings to have identical capitalization
 - Use `equalsIgnoreCase()` to avoid this

Comparing Strings

- `compareTo()` — compares two strings for their relative ordering
 - Ordering is lexicographical (alphabetical, by length, and by case)
 - Usage: `firstString.compareTo(secondString)`
 - This method returns an integer value:
 - Positive : `firstString > secondString`
 - 0 : the two strings are identical (including case)
 - Negative : `firstString < secondString`