CS180 Lab 03

Strings, but for real this time

Week 3
Jan 21. - Jan 27.

String Methods

There are many methods of the `String` class available that can help when manipulating `Strings`.

- Remember that all methods are separated from the Object or Class by a period (`.`) and are terminated by a set of parentheses which may or may not contain parameters (and as always, a semicolon).

`s1.length()`

Returns the length of the String.

- Return type: `int`
- Parameters: none

Ex:

```
String s1 = “Hello”;
int a = s1.length(); // a = 5
```

`s1.concat(String s2)`

Returns a new `String`, where the contents of `s2` are appended to `s1`.

- Return type: `String`
- Parameters: `String s2`, a second string

Ex:

```
String s1 = “Hello”;
String s2 = “World”;
String s3 = s1.concat(s2); // s3 = “HelloWorld”
```

s1 + s2

Same as `s1.concat(String s2)`.

Ex:

```
String s1 = “Hello”;
String s2 = “World”;
String s3 = s1 + s2; // s3 = “HelloWorld”
```

`s1.indexOf(String s2)` | `s1.indexOf(char c)`

Returns the location (index) of the first occurance of the `String` or `char`.

- Return type: `int`
- Parameters: `String s2`, a second string `^ char c`, a character

Ex:

```
String s1 = “Hello”;
int a = s1.indexOf(‘l’); // a = 2
```

`s1.substring(int a, int b)`

Returns a `String` that contains only the characters from index a to index b. If b is not given, `s1.length()` is inferred and used.

- Return type: `String`
- Parameters: `int a`, the start index || `int b`, the end index

Ex:

```
String s1 = “Boilermaker”;
String s2 = s1.substring(1, 6); // s2 = “oiler”
```

`s1.replace(char old, char new)`

Returns a `String` that resembles `s1`, except every occurance of `char old` is replace with `char new`.

- Return type: `String`
- Parameters: `char old`, the `char` to be replaced `&& char new`, the `char` that replaces `old`

Ex:

```
String s1 = “Purdue”;
String s2 = s1.replace(‘P’, ‘B’); // s2 = “Burdue”
```

`s1.compareTo(String s2)`

Returns an `int` a such that if a < 0, `s1` comes before `s2`, if a > 0, `s1` comes after `s2`, and if a == 0, `s1` and `s2` are the same `String`.

- Return type: `String`
- Parameters: `String s2`, a second string

Ex:

```
String s1 = “Purdue”;
String s2 - “purdue”;
int a = s1.compareTo(s2); // a < 0
```

`s1.compareToIgnoreCase(String s2)`

Same as s1.compareTo(String s2) except case is ignored.

- Return type: `String`
- Parameters: `String s2`, a second string

Ex:

```
String s1 = “Purdue”;
String s2 - “purdue”;
int a = s1.compareToIgnoreCase(s2); // a = 0
```

`s1.trim()`

Returns a `String` that resembles `s1`, except all the whitespace at the beginning and end is “trimmed” out.

- Return type: `String`
- Parameters: none

Ex:

```
String s1 = “   IU SUCKS   ”;
String s2 = s1.trim(); // s2 = “IU SUCKS”
```

Need more help?

See the Java API for descriptions of all methods mentioned and more.
This is a great resource, as it is available to you during live coding exams.

Pro Tip: Use the Java 9, 10, or 11 API, as there is now a search bar!
https://docs.oracle.com/javase/9/docs/api/