Object-Oriented Programming Fundamentals

Lecture/Workshop (Week 2) Reference – The String Class

A summary of some useful methods of the String class. Explanations extracted from the API documentation at:

http://docs.oracle.com/javase/8/docs/api/index.html

---

charAt

`public char charAt(int index)`

Returns the `char` value at the specified index. An index ranges from `0` to `length() - 1`. The first `char` value of the sequence is at index `0`, the next at index `1`, and so on, as for array indexing.

---

compareTo

`public int compareTo(String anotherString)`

Compares two strings lexicographically. The comparison is based on the Unicode value of each character in the strings. The character sequence represented by this `String` object is compared lexicographically to the character sequence represented by the argument string. The result is a negative integer if this `String` object lexicographically precedes the argument string. The result is a positive integer if this `String` object lexicographically follows the argument string. The result is zero if the strings are equal; `compareTo` returns `0` exactly when the `equals(Object)` method would return `true`.

This is the definition of lexicographic ordering. If two strings are different, then either they have different characters at some index that is a valid index for both strings, or their lengths are different, or both. If they have different characters at one or more index positions, let *k* be the smallest such index; then the string whose character at position *k* has the smaller value, as determined by using the < operator, lexicographically precedes the other string. In this case, `compareTo` returns the difference of the two character values at position `k` in the two string -- that is, the value:

```
this.charAt(k)-anotherString.charAt(k)
```

If there is no index position at which they differ, then the shorter string lexicographically precedes the longer string. In this case, `compareTo` returns the difference of the lengths of the strings -- that is, the value:

```
this.length()-anotherString.length()
```

---

compareToIgnoreCase

`public int compareToIgnoreCase(String str)`

Compares two strings lexicographically, ignoring case differences.

---

equals

`public boolean equals(Object anObject)`

Compares this string to the specified object. The result is `true` if and only if the argument is not `null` and is a `String` object that represents the same sequence of characters as this object.

---

equalsIgnoreCase

`public boolean **equalsIgnoreCase**(String anotherString)`

Compares this `String` to another `String`, ignoring case considerations. Two strings are considered equal ignoring case if they are of the same length and corresponding characters in the two strings are equal ignoring case.

Two characters `c1` and `c2` are considered the same ignoring case if at least one of the following is true:

- The two characters are the same (as compared by the == operator)
- Applying the method `Character.toUpperCase(char)` to each character produces the same result
- Applying the method `Character.toLowerCase(char)` to each character produces the same result

---

indexOf

`public int **indexOf**(int ch)`

Returns the index within this string of the first occurrence of the specified character. If a character with value `ch` occurs in the character sequence represented by this `String` object, then the index of the first such occurrence is returned. … if no such character occurs in this string, then `-1` is returned.

---

indexOf

`public int **indexOf**(int ch, int fromIndex)`

Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.

---

indexOf

`public int **indexOf**(String str)`

Returns the index within this string of the first occurrence of the specified substring. … if it does not occur as a substring, `-1` is returned.

---

indexOf

`public int **indexOf**(String str, int fromIndex)`

Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

---

isEmpty

`public boolean **isEmpty**()`

Returns `true` if, and only if, `length()` is `0`.

---

lastIndexOf

`public int **lastIndexOf**(int ch)`

Returns the index within this string of the last occurrence of the specified character. … if no such character occurs in this string, then `-1` is returned. The `String` is searched backwards starting at the last character.

---

lastIndexOf

`public int **lastIndexOf**(int ch, int fromIndex)`

Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.

---

lastIndexOf

`public int **lastIndexOf**(String str)`

Returns the index within this string of the rightmost occurrence of the specified substring. If it does not occur as a substring, `-1` is returned.

---

lastIndexOf

`public int lastIndexOf(String str, int fromIndex)`

Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.

---

length

`public int length()`

Returns the length of this string.

---

replace

`public String replace(char oldChar, char newChar)`

Returns a new string resulting from replacing all occurrences of `oldChar` in this string with `newChar`.

Examples:

```
"mesquite in your cellar".replace('e', 'o')
        returns "mosquito in your collar"
"the war of baronets".replace('r', 'y')
        returns "the way of bayonets"
"sparring with a purple porpoise".replace('p', 't')
        returns "starring with a turtle tortoise"
"JonL".replace('q', 'x') returns "JonL" (no change)
```

---

substring

`public String substring(int beginIndex)`

Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

Examples:

```
"unhappy".substring(2) returns "happy"
"Harbison".substring(3) returns "bison"
"emptiness".substring(9) returns "" (an empty string)
```

---

substring

`public String substring(int beginIndex, int endIndex)`

Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex-beginIndex`.

Examples:

```
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
```

---

toLowerCase

`public String toLowerCase()`

Converts all of the characters in this `String` to lower case using the rules of the default locale.

---

toUpperCase

`public String toUpperCase()`

Converts all of the characters in this `String` to upper case using the rules of the default locale.

---

trim

`public String trim()`

Returns a copy of the string, with leading and trailing whitespace omitted.

There are many more methods in the String class, look them up in the API