Strings (Teaching)

lillie

24 January 2025

Review

1. Strings, and how they're stored
2. Zero Indexing
3. String Concatenation
4. Type Coercion
5. String Equivalence (`==`, `.equals`)

Methods

1. `str.length()`
   - The length (amount of characters) of the given string `str`
2. `str.charAt(strIndex)`
   - `IndexOutOfBoundsException` if `strIndex` $\notin$ `[0, str.length)`
   - otherwise the character located at the index `strIndex`
3. `str.indexOf(strAlt)`
   - `-1` if `strAlt` $\notin$ `str`
   - otherwise the index of `strAlt`
4. `str.indexOf(strSub, strIndex)`
   - `-1` if `strAlt` $\notin$ `strAfter` where `strAfter` is `str` starting at index `strIndex`
   - otherwise the index of `strAlt` after the index `strIndex`
5. `str.contains(strAlt)`
   - `true` if `strAlt` is a substring of `str`
   - otherwise `false`
6. `str.compareTo(strAlt)`
   - $\begin{cases} 0 & \text{if } \texttt{str} = \texttt{strAlt} \\ n > 0 & \text{if } \texttt{str} > \texttt{strAlt} \\ n < 0 & \text{if } \texttt{str} < \texttt{strAlt} \end{cases}$
7. `str.substring(startIndex)`
   - `IndexOutOfBoundsException` if `startIndex` $\notin$ `[0, str.length)`
   - otherwise `str` starting at `startIndex`
8. `str.substring(startIndex, endIndex)`
   - `IndexOutOfBoundsException` if `startIndex` $\notin$ `[0, str.length)` or `endIndex` $\notin$ `(0, str.length]`
   - otherwise `str` starting at `startIndex` (inclusive) and ending at `endIndex` (exclusive)

9. `str.substring(0)`
    - This is equivalent to `str`
10. `str.substring(0, endIndex)`
    - `IndexOutOfBoundsException` if `endIndex` $\notin$ `(0, str.length]`
    - otherwise `str` ending at `endIndex` (exclusive)
11. String Pool Caching
    - Duplicate String Literals are optimized in a way where they point to the same String in an area called the String Constant Pool
    - Concatenated String Literals are optimized to be a singular String
    - Introduce how String instances (`new String("cargo")`) are different (Heap)

Examples

1. `str.length`
    - `"".length()` $= 0$
    - `"aaa".length()` $= 3$
2. `str.charAt`
    - `"".charAt(0)` = `IndexOutOfBoundsException`
    - `"car".charAt(0)` = `'c'`
    - `"car".charAt(1)` = `'a'`
    - `"car".charAt(1)` = `'r'`
3. `str.indexOf`
    - `"cargo".indexOf("car")` $= 0$
    - `"cargo".indexOf("go")` $= 3$
    - `"cargo".indexOf("core")` $= -1$
    - `"cargo".indexOf("m")` $= -1$
4. `str.contains`
    - `"cargo".contains("core")` = `false`
    - `"cargo".contains("car")` = `true`
    - `"cargo".contains("go")` = `true`
5. `str.compareTo`
    - `"cargo".compareTo("cargo")` $= 0$
    - `"carga".compareTo("cargo")` < 0
    - `"cargo".compareTo("carga")` > 0
6. `str.substring`
    - `"cargo".substring(0)` = `"cargo"`
    - `"cargo".substring(1)` = `"argo"`
    - `"cargo".substring(1)` = `IndexOutOfBoundsException`
    - `"cargo".substring(0, 5)` = `"cargo"`
    - `"cargo".substring(0, 3)` = `"car"`
    - `"cargo".substring(3, 5)` = `"go"`
7. String Pool Caching
    - When `==` is `true`
    - Inline concatenation optimization
    - When `==` is `false`

This document is for a mini-lecture on Strings for Teaching (SP25)