Character Strings

- `class String`
- `class StringBuffer`
  (Java 1.4 and before: thread-safe)
- `class StringBuilder`
  (Java 5: thread-unsafe)

(Cf. array of characters)
Based on 2-byte Unicode characters

---

String literals and concatenation

- String `s = "abc";`
- `System.out.println( s + "def" );`
  - “+” stands for string concatenation.
  - Built-in operator overload.

- Left associativity of “+”
  - `(s + 3 + 5)` evaluates to “abc35”.
  - `(s + (3 + 5))` evaluates to “abc8”.
  - `(5 + 3 + s)` evaluates to “8abc”.

---

String conversions

- `class Object` supports `toString()`-method to convert a class instance into a printable string form: `classname@hashcode`
- `toString()` can be overridden.
  ```java
  public String toString() { ... }
  ```
- `println/print` methods in `System.out`, the “+”-expressions, and the “+=”-statements invoke `toString()` implicitly, to coercer an instance to a string.

---

class `String` vs
class `StringBuffer/StringBuilder`

- immutable
  - contents of `String` instance unchangeable
  - cannot insert or append characters
- fixed length
  - `s.length()`
    - number of characters in the string
- mutable
  - contents of `StringBuffer` instance modifiable
- growable
  - grows automatically when characters added
  - `sb.length()`
  - `sb.capacity()`
    - total allocated capacity
Assignment

String s = "abc";
s += 2+3+5;
s.equals("abc10") == true
s.equals("abc235") == false
(Recall that '+' is left-associative.)

<table>
<thead>
<tr>
<th>TYPE</th>
<th>TO STRING</th>
<th>FROM STRING</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>String.valueOf(boolean)</td>
<td>new Boolean(String).booleanValue()</td>
</tr>
<tr>
<td>int</td>
<td>String.valueOf(int)</td>
<td>Integer.parseInt(String)</td>
</tr>
<tr>
<td>long</td>
<td>String.valueOf(long)</td>
<td>Long.parseLong(String)</td>
</tr>
<tr>
<td>float</td>
<td>String.valueOf(float)</td>
<td>new Float(String).floatValue()</td>
</tr>
<tr>
<td>double</td>
<td>String.valueOf(double)</td>
<td>new Double(String).doubleValue()</td>
</tr>
</tbody>
</table>

Comparison

- "abc".equals("abc") is true.
- "abc".equalsIgnoreCase("ABC") is true.
- "abc" == new String("abc") is false.
- s == s is true.
- s1.compareTo(s2)
  - negative: s1 lexicographically precedes s2
  - zero: s1 is equal s2
  - positive: s1 lexicographically follows s2

Useful Methods

- String t = "abca";
- String s = new String(t);
  - s.charAt(2) is ‘c’.
  - s.indexOf('a') is 0.
  - s.indexOf("bc") is 1.
  - s.indexOf('a',2) is 3.
  - s.lastIndexOf('a') is 3.
  - regionMatches, startsWith, endsWith, etc.
- Pure ASCII applications can be inefficient because Java uses 16-bit Unicode characters.