CSC 207
Algorithms and Object-Oriented Design

Strings and StringBuilders
Weiss, sections 2.3 and 15.4 (pages 35–37 and page 584)
Strings

• A string is a sequence of characters.
• String is a reference type.
• The String class is immutable. A string object has an unchanging value.

String greeting = “Hello world!”;

• “Hello world!” is a string literal.
• Compiler creates a String object that contains “Hello world!” as its value.
Strings

- String can be created using **new** keyword and a **constructor**.

```java
char[] helloArray = {'h', 'e', 'l', 'l', 'o', ' '};
String helloString = new String(helloArray);
System.out.println(helloString);
```

- `length()` method counts the number of characters in a String object.

```java
int len = helloString.length(); // len stores 6
```

- `charAt()` returns the $i$-th character is a String object.

```java
char c = helloString.charAt(1); // c stores ‘e’
```
Strings

• `concat()` method concatenates two strings. “My name is `.concat(“John Smith.”);`

• Strings are commonly concatenated using + operator. “My name is “ + “John Smith”;

• The + operator is widely used in `printf` statements.

`String s = “My name is “; System.out.println(s + “John Smith.”);`
• Convert primitive numeric types to strings.
  double d = 858.48;
  String s = Double.toString(d); // “858.48”
  String s = String.valueOf(d); // “858.48”

  ➤ Double is a \textit{wrapper} class that wraps double numbers.

• \texttt{parseXXX()} converts strings to primitive numbers.
  double d = Double.parseDouble(“858.48”);
Strings

• `equals()` method compares two strings.
  `string1.equals(string2);`
• returns true if `string1` and `string2` store the same sequence of characters.

• `compareTo()` compares two strings lexicographically.
  `int res = string1.compareTo(string2);`
• Returns an `integer` that indicates `string1` is greater than (res > 0), equal to (res = 0), or less than (res < 0) `string2`. 
StringBuilder

• StringBuilder objects are like String objects, except that they can be modified.

• Internally, StringBuilder objects are like variable-length array. At any point, length and content can be changed.

• StringBuilder provides simpler code. It concatenates strings faster than String.
StringBuilder

• StringBuilder has capacity() method. It returns the number of allocated character spaces.
• The capacity is usually greater than the length.

StringBuilder sb = new StringBuilder();
sb.append("Greeting");

• Use toString() method to convert StringBuilder to String
+ versus append()

- Strings are immutable.
- Calling + operator in a loop (for concatenating strings) creates new string objects.
- Large space overhead for a loop with many iterations.

```java
String str = "";
for (int i = 0; i < 100; i++) {
    str += i;
}

StringBuilder sb = new StringBuilder();
for (int i = 0; i < 100; i++) {
    sb.append(i);
}
String str = sb.toString();
```