Strings

Chapter 14

Strings

- *String*: A sequence of characters surrounded by double quotes (also called a *string literal*)
  - `"This is a string literal."`
  - `"123 Main Street!"`
  - `"X"`
  - `""`
- `String` is a class
- String variables are objects of the `String` class

Creating Strings

- Using the constructor
  - String s = new String(“hello”);
- Shortcut!
  - String t = “goodbye”;

Escape Sequences

- What if we wanted to print the quote character?
  - `System.out.println("I said "Hello" to you.")`
  - Invalid Syntax!
- An *escape sequence* is a series of characters that represent a special character.
- Escape sequences begin with a backslash (\)
  - `System.out.println("I said \"Hello\" to you.")`

`\t`	tab
`\n`	newline
`\"`	double quote
`\'`	single quote
`\\`	backslash

String Methods

- String objects are special objects called *immutable*.
  - They cannot be changed
- To change a `String` reference, create a new `String` and point your reference at that new `String`

length

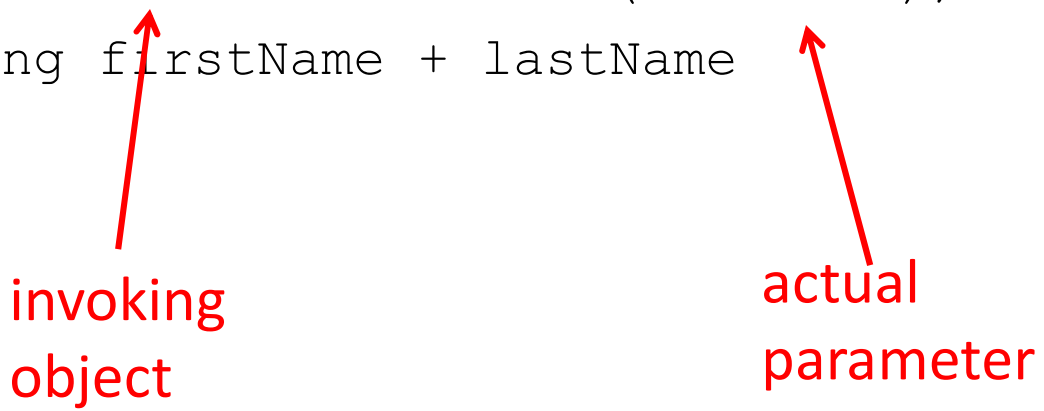
- public int length()
- Returns the number of characters in the String

```
String s = “Hello there!”;
int n = s.length();
// n holds 12
```

Concatenation

- public String concat(String str)
  - Returns a new string which is the *invoking object* concatenated with the *actual parameter*
  - Does **not** change the invoking object

```
String firstName = "Jessica";
String lastName = "Masters";
String fullName = firstName.concat(lastName);
// same as saying firstName + lastName
```



Concatenation (cont.)

- You can assign the new String back to the original variable

```
String s1 = “Hello”;
String s2 = “ There”;
s1 = s1.concat(s2);
```

String Concatenation

- The *string concatenation operator* (+) can also be used to append one string to the end of another
  - Can also append a number to a String
- The result is a new `String`

```
s1 = s1+ s2
// s1 = s1.concat(s2);
// same as s1 += s2
```

```
String s1 = "Peanut butter" + " and jelly";
// s1 refers to a String
// "Peanut butter and jelly"

String s2 = "The answer is " + 25
//s2 refers to a String "The answer is 25"
```

String Concatenation (cont.)

- A string cannot be broken across two lines in a program, so they must be concatenated.

```
System.out.println("This is my really long
  string that stretches way out over this
  line and onto the next." );
// This is invalid! What type of error?

System.out.println("This is a really long"
  + "string that stretches way out over "
  + "two lines but since I used the "
  + "concatenation operator it works!" );
```

String Concatenation (cont.)

- The + operator is used for both string concatenation and for arithmetic addition.
  - If **both** operands are numeric, the + adds them.
  - If **either** or **both** of the operands are strings, the + performs string concatenation.
- The + operator is evaluated left to right, but parentheses can be used to force the order.

String Concatenation (cont.)

- What will print?

```
System.out.println(3+4);
System.out.println(“3” + “4”);
System.out.println(3 + 4 + “5”);
System.out.println(“3” + “4” + 5 + “6”);
System.out.println(“3” + (4 + 5))
System.out.println(3 + 4 + “5” + 6);
```

Replacing Characters

- public String replace (char oldChar, char newChar)

```
String s1 = “Hello”;
String s2 = s1.replace(‘e’,’a’);
// s1 still holds “Hello”
// s2 holds “Hallo”
```

Case

- public String toLowerCase()
- public String toUpperCase()

```
String s1 = “Hello”;
String s2 = s1.toUpperCase();
// s1 still holds “Hello”
// s2 holds “HELLO”
```

Substrings

- Each character in a string has an index
  - Indices begin at zero
- public String substring(int offset, int endIndex)
  - Returned substring will start at `offset`
  - Returned substring will end at (`endIndex` – 1)
  - Length of returned substring is (`endIndex` – `offset`)

Substrings (cont.)

```
String greeting = “Good night”;
//  G   o   o   d       n   i   g  h  t
//  0   1   2   3   4   5   6   7  8  9

String shorter = greeting.substring(3, 7);
// holds “d ni”
// length = 7-3 = 4

shorter = greeting.substring(0,1);
// holds “G”
// length = 1-0 = 1
```

The StringBuilder Class

- The `StringBuilder` class allows you to create modifiable strings.
- `capacity` specifies how many characters can be stored.
  - Once the `capacity` is reached, it expands.
- To create a `StringBuilder` object, you can specify a default `String` or an initial capacity.
- You can add onto a `StringBuilder` object with the `append` or `insert` method.

Summing Up

- String are immutable!
  - Invoking a method returns a new String- it does not change the invoking object.
- Strings are indexed starting at position 0.
- Oracle has some great tutorial pages on String and StringBuilder: http://docs.oracle.com/javase/tutorial/java/data/strings.html