Intro to Programming II
Strings

Chris Brooks

Department of Computer Science
University of San Francisco
5-0: UML Diagrams

A UML class diagram is another formalism for designing a program.

△ Provides a way of thinking about objects and their interaction.
△ This is different from functional decomposition.

Objects are represented as boxes.

Arrows indicate data flow.

Method input and output are placed inside boxes.
Strings in Java are objects

This means that they have a set of methods they respond to:

- `compareTo()`, `equals()`
- `indexOf()`
- `length()`
- `replace()`
- `startsWith()`, `endsWith()`
- etc
Unlike most other objects, Strings also have special behavior for creating *string literals* and for *concatenation*.

String literals are strings where the value is known at compile time:

- String s1 = “hello world”
- String s2 = “USF”
- String s3 = “I love Java”

We can create a string without calling *new*. 
5-3: Overloaded operators

We can also use the ‘+’ symbol to concatenate strings.

- String s1 = “hello”
- String s2 = “world”
- String s3 = s1 + s2 // s3 = “hello world”

This is a phenomenon called overloading; an operator is redefined to provide different functionality.
Strings are immutable

- This means that once a string is created, it can’t be changed.
- To change it, you need to create a copy.

```java
String s1 = "hello world"

To change “hello” to “goodbye”, we’d need to do:

```java
String s2 = s1.replace("hello", "goodbye");
```

s2 is “goodbye world”, s1 is unchanged
To find the character at a particular location, use `charAt(int index)`

```java
String s1 = "I love Java";
for (int i = 0; i < s1.length(); i++) {
    System.out.println(s1.charAt(i));
}
```
Write a program that:
- Reads in a string from System.in
- Iterates over the string and prints out all the vowels.
To test whether two strings have the same contents, use `equals()` or `equalsIgnoreCase()`

`==` will test for *object equality*

- String `s1 = “foo”;
- String `s2 = “foo”;
- `s1.equals(s2)`, but not `s1 == s2`

You can also use `compareTo()`

- Returns -1 if `s1` comes before `s2`, 0 if they’re equal, and 1 if `s1` comes after `s2`. 
Write a program that will:

- Read a string in from System.in;
- Print out the first word in the string.
We can use the Scanner class to read from a file instead of System.in

```
try {
    Scanner sc = new Scanner(new File("studentlist"));
    while (sc.hasNext()) {
        System.out.println(sc.next());
    }
} catch (FileNotFoundException e) {
    System.out.println("File not found.");
}
```
5-10: String practice

6. Read in the file “studentfile” and print out all names beginning with ‘a’.

6. Read in the file ’studentfile’ and print out all people with first names of longer than 5 letters.

6. Read in the file ’studentfile’ and print out all people whose Last name comes after ’jones’ in the alphabet.