

# Intro to Programming II

## Strings and Files

Chris Brooks

Department of Computer Science  
University of San Francisco

Department of Computer Science — University of San Francisco — p. 1/77

### 6-2: Introduction

- In project 1, you'll be working extensively with strings.
  - If you do the extra credit, you'll also be working with files.
- Therefore, it's useful to take a bit of time to remember how to work with strings.

Department of Computer Science — University of San Francisco — p. 2/77

### 6-3: Strings in Java

- 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

Department of Computer Science — University of San Francisco — p. 3/77

### 6-4: 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.

Department of Computer Science — University of San Francisco — p. 4/77

### 6-5: Strings are immutable

- 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.
- String s1 = "hello world"
- To change "hello" to "goodbye", we'd need to do:
- String s2 = s1.replace("hello", "goodbye");
- s2 is "goodbye world", s1 is unchanged

Department of Computer Science — University of San Francisco — p. 5/77

### 6-6: Iterating over strings

- To find the character at a particular location, use charAt(int index)

```
String s1 = "I love Java"
for (int i = 0; i < s1.length(); i++) {
    System.out.println(s1.charAt(i));
}
```

Department of Computer Science — University of San Francisco — p. 6/77

### 6-7: String practice

- Write a program that:
  - Reads in a string from System.in
  - Iterates over the string and prints out all the vowels.

### 6-8: String equality

- 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.

### 6-9: String practice

- Write a program that will:
  - Read a string in from System.in;
  - Print out the first word in the string.

### 6-10: Using Scanner to read from files

- 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.");
}
```

### 6-11: String practice

- Read in the file "studentfile" and print out all names beginning with 'a'.
- Read in the file 'studentfile' and print out all people with first names of longer than 5 letters.
- Read in the file 'studentfile' and print out all people whose Last name comes after 'jones' in the alphabet.

### 6-12: Building Project 1

- The first class to consider is the Token class.
- Two instance variables:
  - type
  - value
- class variables for each type
- setters and getters
- toString(), plus a unit test.