STRINGS AND REFERENCE VARIABLES

CS302 – Introduction to Programming
University of Wisconsin – Madison
Lecture 4

By Matthew Bernstein – matthewb@cs.wisc.edu

The Char Data Type

- A **char** is a primitive data type that holds a single character
- Char variables actually store integers [0 - 65,535] where each integer in this range corresponds to a unique character specified by the 16-bit Unicode Transformation Format (UTF-16)
- Example:

```
char someChar = ‘a’;
char anotherChar = ‘?’;
char someChar = 3425; // Valid but you would never do this
```

Strings

- In general we deal a lot with sequences of character (our whole language is composed of sequences of characters)
- We need a good way of representing a sequence of characters when writing programs
- The solution is a data type called a **String**
- (A string is an **object**, not a primitive data type like ints and doubles)

What Are String Objects?

- Strings are Java **objects** that simply wrap a sequence of **char** variables

- Using String objects, you don't need to deal with the underlying implementation of this sequence
- Strings also allow you to use various methods such as finding substrings or returning a char variable that is at a specific index of the String
- You don't have to memorize all of String's methods, but when manipulating Strings for various purposes these methods come in handy

What can you do with them?

- Append them together
- Find substrings
- Find characters at specific index of String
- Etc.

Concatenation

- We can **concatenate** two strings together to form a new string
- Example:

```
String firstStr = “Hello ”;
String secondStr = “world.”;
String finalStr = firstStr + secondStr;
```

Finding Characters

- You can find a character at a specific index of the string using String's charAt(int index) method
- The index is zero-based
- Example:

```
String str = "Sample String";
str.charAt(1);  // This will return the character "a"
```

Finding Substrings

- You can extract a substring by using string's substring(int start, int pastEnd) method
- Example:

```
String greeting = "Hello, World!";
String sub = greeting.substring(0, 5); // sub is "Hello"
```

Programming Exercise

- Write a program that takes the string “Hello John!” and returns the String “Hello Sarah!”.
- I give you the strings “Hello John!” and “Sarah”
- You can only use String methods (You cannot simply assign a string variable to a string **literal**)

System.out.format()

- System.out.format("format-string" [, arg1, arg2, …])
- This method provides a different strategy for writing output to the console.
- You provide a template for the output along with values you want inserted into the template
- You specify where you want these values inserted with **specifiers**
- Useful Specifiers:
  - "%s" -> insert string
  - "%f" -> insert floating-point number
  - "%d" -> insert integer

Example Using format()

```
final String ROCK = “rock”;
final String PAPER = “paper”;
final String SCISSORS = “scissors”;

String roundResult = “You threw %s, computer threw %s”;

…

System.out.format(roundResult, ROCK, PAPER);
```

Reference Variables vs. Primitive Variables

- A variable that holds an object (such as a String or a Scanner) is called a **reference variable**
- Reference variables are fundamentally different from **primitive variables** (recall primitive variables store data types like ints, doubles, and chars)

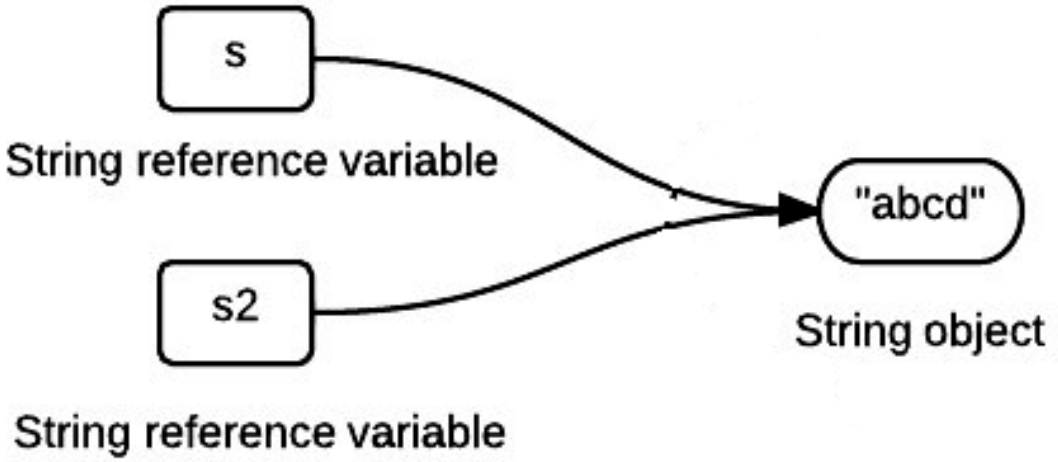
- How are they different?
- A reference variable stores the “memory address” of the object, but not the object itself
- A primitive variable stores an actual value, not a reference to that value

Memory Diagram

- The following code:

```
String s = “abcd”;
String s2 = s;
```

- Produces the following in your computer’s memory:



Cool Link

- TED talk by Ramesh Raskar: Imaging at a trillion frames per second
- http://www.ted.com/talks/ramesh_raskar_a_camera_that_takes_one_trillion_frames_per_second.html