Image Credit: Julius Schorzman

# Java Lesson 7

Concept Review

# Strings

- A String is a collection of characters that is stored within the computer's memory.
- For example, say that you wanted to store a mailing address in the computer's memory, such as:
  - *"2323 West 14th Street, Tempe, AZ 85281"*
- How would we store this data in memory? This is where the String comes in to the rescue!
  - String address = *"2323 West 14th Street, Tempe, AZ 85281"*;

# Strings

- It is important to note that a String is an ***object*** (of the String class) and not a primitive data type, such as an integer or double.
- Why is this important to know? Well, consider the following coding example. What do you think will be the result?

```
public class CompareStrings
{
   public static void main(String[] args)
   {
      String aName = new String("Carmen");
      String anotherName = new String("Carmen");

      if (aName == anotherName)
         System.out.println("Yes, the names are the same!");
      else
         System.out.println("Sorry, but the names are NOT the same!");
   }
}
```

# Strings

```
Sorry, but the names are NOT the same!
```

- Much to one's surprise, Java does not consider the two Strings to be equal! Here's why:

- Java is not actually evaluating the contents of the Strings for equivalency, but instead, their memory addresses instead! This is because a String is an object of the String class and not a primitive data type.

```
String aName = new String("Carmen");
String anotherName = new String("Carmen");

if (aName == anotherName)
   System.out.println("Yes, the names are the same!");
else
   System.out.println("Sorry, but the names are NOT the same!");
```

# Strings

- To fix this issue, Java includes an *equals()* method with all Strings that allow you to compare them for equality.
- By using the *equals()* method, we can fix this issue:

```
String aName = new String("Carmen");
String anotherName = new String("Carmen");

if (aName.equals(anotherName))
   System.out.println("Yes, the names are the same!");
else
   System.out.println("Sorry, but the names are NOT the same!");
```

```
Yes, the names are the same!
```

# Strings

- Some additional methods that you might find helpful when working with Strings:

- equalsIgnoreCase() compares the contents of two Strings for equality, while ignoring case differences.

- toUpperCase() converts the characters in the String to uppercase.

- toLowerCase() converts the characters in the String to lowercase.

- length() returns the length of the String.

- charAt(x) returns the character at position x within the String.

- replace() allows you to replace all occurrences of a specified character within the String.

# The Character Class

- The Character class provides some powerful methods that you may find useful, such as:
  - toUpperCase() converts all characters in the String to their upper case equivalents.
  - toLowerCase() converts all characters in the String to the lowercase equivalents.
  - isDigit() determines whether a character is a digit.
  - isLetter() determines whether a character is a alphabetic letter.
  - isLetterOrDigit() returns true if the character is a letter or numeric digit.
  - isWhitespace() returns true if the character is considered whitespace, such as a space, tab, newline, etc.