Strings

Learning Objectives

- To be able to manipulate `String` values

Strings

- **Strings** are “objects” of the String class
- Strings hold sequences of characters (a, b, c, $, etc)
- Write `String variable_name;` to declare a string object
- A string like other objects can be initialized to a `null reference`
- `A null reference` means that the variable does not refer to a space in memory

    - `String variable_name = null;` creates a null string object

String initialization

- There are two ways to initialize a string
- `String variable_name = new String(string_literal);`
    - Example: String name = new String(“Lisa”);

- `String variable_name = string_literal;`
    - Example: String name = “Lisa”;

String operations

- **Concatenation**
- Use the **“+”** or **“+=”** operators to concatenate (combine) two Strings

```
String a = “Serena”;
String b = “ Williams”;
String c = a + b;
System.out.println(c); // prints Serena Williams
```

String operations

- Using **“+”** or **“+=”** operators to append a primitive type value to a String will automatically convert that value to String

```
String a = “Serena”;
String b = “ Williams”;
String c = a + b + 100;
System.out.println(c); // prints Serena Williams100
```

String methods

- **`int length()`** method returns the number of characters in the string, including spaces and special characters like punctuation

```
String a = “Serena”;
a.length(); // returns 6
```

- **`String substring(int from, int to)`** method returns a new string with the characters in the current string starting with the character at the `from` index and ending at the character *before* the `to` index (if the to index is specified, and if not specified it will contain the rest of the string)

```
String a = “Serena”;
           0 1 2 3 4 5
a.substring(0, 3); // returns Ser
                      0 1 2
```

String methods

- **`int indexOf(String str)`** method searches for the string str in the current string and returns the index of the beginning of str in the current string or -1 if it isn't found

```
String a = "Serena";
```

0 1 2 3 4 5

```
a.indexOf("er");  // return 1
a.indexOf("ena"); // return 3
a.indexOf("zer"); // return -1
```

Comparing Strings

- Strings (and objects) **cannot** be compared using operators like **==** and **<** or **>**
- The method **`compareTo`** compares two strings character by character.
    - If they are **equal**, it returns **0**
    - If the **first string** is alphabetically ordered **before** the **second string** it returns a **negative number**
    - If the **first string** is alphabetically ordered **after** the **second string**, it returns a **positive number**

Comparing Strings

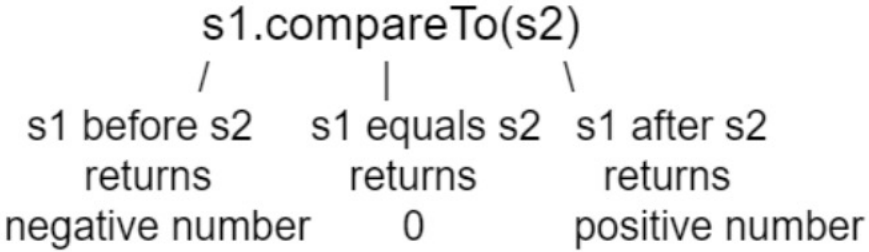

Figure 2: compareTo returns a negative or positive value or 0 based on alphabetical order

```
String a = “Serena”;

String b = “Williams”;
```

S comes before **W** in the alphabet

```
a.compareTo(b); // return -4 negative number
b.compareTo(a); // return 4 positive number
```

String equality

- The `equals` method compares the two strings character by character and returns true or false

```
String a = “Serena”;
String b = “Williams”;
a.equals(b); // return false
a.equals(a); // return true
```

- `compareTo` and `equals` are case-sensitive.