The `String` Class

Creating and Manipulating Text Data

Alwin Tareen

The String Class

- Java does not have a built-in primitive data type for Strings.
- Instead, the standard Java library has a predefined class called String.

Instantiating(Creating) a String Object

```
String awaken = "Good Morning";
System.out.println(awaken);
```

Strings are immutable

In Java, a String is considered immutable. Once it has been created, it cannot be altered or changed.

String Concatenation

Joining text

- Java allows you to use the + sign to join two Strings together.

```
String first = "choco";
String second = "late";
String candy = first + second; // chocolate
```

- You can also concatenate a String with a numerical value.

```
int total = 58;
System.out.println("The total is: " + total);
```

String Indexes

Assigning numbers to each letter

```
String fruit = "watermelon";
```

- We can assign indexes to each letter of this word in the following manner:

letter	w	a	t	e	r	m	e	l	o	n
index	0	1	2	3	4	5	6	7	8	9

- Notice how the first letter in this word(the `w`) corresponds to index `0`.
- Therefore, the last letter in this word(the `n`) is assigned an index of `9`.

Substrings

In Java, you can extract a section from a larger `String` with the `substring()` method.

Substring with 2 parameters: `substring(m, n)`

Generally, you should regard this method as follows:

- Start with the index of the first letter that you **want** (`m`).
- End with the index of the first letter that you **don't want** (`n`).

letter	w	a	t	e	r	m	e	l	o	n
index	0	1	2	3	4	5	6	7	8	9

```
String fruit = "watermelon";
String duration = fruit.substring(2, 6); // term
```

Substrings

One step beyond: `substring(m, n)`

- Let's say I wanted to extract the `String` `"berry"` from `"strawberry"`.
- `Java` will allow you to consider the index that is one step beyond the end of the `String`.

letter	s	t	r	a	w	b	e	r	r	y	
index	0	1	2	3	4	5	6	7	8	9	10

- The following `Java` statement is legal:

```
String flavor = "strawberry";
String piece = flavor.substring(5, 10); // berry
```

Substrings

Substring with 1 parameter: `substring(m)`

- This method begins with the letter corresponding to the index `m`.
- It then extracts all of the letters up to and including the end of the `String`.
- This version behaves as a kind of shortcut.

letter	p	e	p	p	e	r	m	i	n	t
index	0	1	2	3	4	5	6	7	8	9

```
String seasoning = "peppermint";
String herb = seasoning.substring(6); // mint
```

Determining the length of a String

- The `length()` method indicates how many characters there are in a `String`.

```
String fruit = "watermelon";
int num = fruit.length();
System.out.println("Number of letters = " + num);
```

- A common use of the `length()` method is to use it with a `for` loop to iterate through each of the letters in the `String`.

```
for (int i = 0; i < fruit.length(); i++)
{
    System.out.println(fruit.substring(i, i+1));
}
```

Searching within a String

The indexOf(str) method

- This method allows you to search for an individual character or a substring within another String.
- If the search is successful, then the method returns the index of the substring.
- If the substring is not found within the String, then the method returns -1.

```
String lunch = "cheeseburger";
int position = lunch.indexOf("burg"); // 6
int location = lunch.indexOf("e"); // 2
int section = lunch.indexOf("raw"); // -1
```

Equality of String Objects

The equals() method

- This method allows you to check if two Strings are equal.
- Note that you cannot use the == operator to compare Strings, because Strings are not primitive data types.

```
String drink = "water";
String beverage = "water";
boolean result = drink.equals(beverage); // true

String soda = "sprite";
String pop = "pepsi";
boolean outcome = soda.equals(pop); // false
```

Comparing String Objects

The compareTo(str) method

- This method compares each String's relative position in the ASCII chart of text symbols.

Digits

Value	Symbol
48	0
49	1
50	2
51	3
52	4
53	5

Uppercase

Value	Symbol
65	A
66	B
67	C
68	D
69	E
70	F

Lowercase

Value	Symbol
97	a
98	b
99	c
100	d
101	e
102	f

Comparing String Objects

- Upon examining the ASCII table, we can see that the following relation is true:

```
digits < uppercase letters < lowercase letters
```

- Consider the following statement:

```
boolean result = phrase.compareTo(sentence);
```

- If phrase alphabetically **precedes** sentence:
  $\rightarrow$ result contains a negative int.
- If phrase alphabetically **follows** sentence:
  $\rightarrow$ result contains a positive int.
- If phrase is alphabetically **equal** to sentence:
  $\rightarrow$ result contains zero.

The `String` Class: End of Notes