Download

Day 6: Strings

Download

Agenda

- Strings
- Immutability
- Length
- Concatenation
- Equals
- Compare To
- Basic String Methods
- Project 1
- Lab 3

Download

Strings

Download

Review

To create a string, our data type is a `String`. We then give a variable name. We can then set it to a value surrounded by double-quotes.

```
String <variable name> = "";
```

Immutability

Strings are immutable, this means we cannot change a string by removing a value or changing a value.

However, we can add to a string by concatenation.

```
String s = "hello";

s = s + " world";

System.out.println(s); // hello world
```

Immutability

While this is okay, the problem with this is that it creates a new string in memory. If we're creating small strings, this isn't so bad. But when we keep adding to it, we keep creating new strings in memory. This is bad.

```
String s = "hello";

s = s + " world"; // Different than
the first s.

System.out.println(s); // hello
world
```

Length

With strings, we can get the length of the string using the `.length()` method.

```
String s = "Hello World";

System.out.println(s.length());

//outputs 11
```

Length use

This is especially important if we want to loop/iterate through a string.

We can use the `charAt` method to get the at each index/location.

```
String s = "Hello World";

for(int i = 0; i < s.length(); i++)
{

  System.out.println(s.charAt(i));

}

// Prints out each character
```

Concatenation

We've seen this already before, but adding any value to a string will "concatenate" the value. This simply extends the string and creates a new and longer string.

```
String a = "Hello";

a = a + " World";

System.out.println(a);

// Outputs "Hello World"
```

String Equality

You may have noticed before that the following code gives us a false value.

This is because, with Java Strings, we cannot compare those two values directly using the == operator.

```
System.out.print("Continue? ");

String choice = input.nextLine();

if(choice == "n") {

  System.out.println("Continuing");

}
```

String Equality

However, the following code prints true somehow. Why?

```
String a = "Hello";

String b = "Hello";

System.out.println(a == b);

// outputs true
```

Download

The answer lies in memory and how Java optimizes it.

```
String a = "Hello";

// say a is stored at 0x1234abc;

String b = "Hello";

// Java sees the values being the
same and reuses the same memory
location at 0x1234abc
```

But in this case, the "n" is not the same memory location as the string in `choice`, even if it holds the same value.

```
System.out.print("Continue? ");

String choice = input.nextLine();
// say, choice is at 0x1282acd

if(choice == "n") { // say "n" is
at 0x1829bed

  System.out.println("done");

}
```

So how can we check for String equality?

equals

We can use the `.equals` method. This wll check if the values in the Strings are the same.

This method takes in another String and returns `true` if they are identical in **value**, not in memory.

```
System.out.print("Continue? ");

String choice = input.nextLine();

if(choice.equals("n")) {

  System.out.println("done");

} // will print "done" if choice is exactly "n"
```

equalsIgnoreCase

If we don't care about case sensitivity, then we can use the `equalsIgnoreCase` method. This does the same, but ignores the case of the string, checking if they match that way.

```
System.out.print("Continue? ");

String choice = input.nextLine();

if(choice.equalsIgnoreCase("n")) {

  System.out.println("done");

} // prints "done" as long as
choice is "N" or "n"
```

Download

compareTo

If we want to compare two strings, we cannot use the < and > operators. Instead, we must use the `compareTo` method.

This will return a negative number, 0, or a positive number depending on whether the other string is "greater", "lesser", or "equal" to the string. These compare the ASCII values.

```
String a = "hello";

String b = "Hello";

System.out.println(a.compareTo(b));

// gives -32
```

ASCII Table

asciitable.com

Download

compareToIgnoreCase

If we want to compare two strings but ignore case sensitivity, we can use `compareToIgnoreCase`. The return values are the same, but ignore case sensitivity.

```
String a = "hello";

String b = "Hello";

System.out.println(a.compareTo(b));

// gives 0
```

Download

String Functions

- substring
- charAt
- indexOf
- lastIndexOf
- replace
- split
- startsWith
- strip
- toCharArray
- toLowerCase
- toUpperCase
- trim

Java Documentation for Strings

Download

Javadoc

Download

Explore the javadoc page with Strings. These are all of the methods we can call on a String.

Download

Project 1

Download

Password Validator and Generator

Download

Task

- You will create a simple Password Validator and Generator
- The user will be given two options after logging in. They can validate a password or generate a new password.
- You will store a username and password in the program, by updating the variables in the code to save a login credential.

Download

Password Validation

For a password to be "valid", it must satisfy the following requirements:

- At least 8 characters long.
- Contain 1 uppercase character.
- Contain 1 lowercase character.
- Contain 1 digit.

Ask the user to input the password they want to validate.

Download

Checking for uppercase

We can simply check if the character is greater than the character of 'A' and less than the character of 'Z'

```
char c = 'H';

c >= 'A' && c <= 'Z'; // evals to true
```

Checking for uppercase

This works because characters actually use ASCII to represent the data.

asciitable.com

Ultimately, characters still represent numbers.

```
char c = 'H';

c >= 'A' && c <= 'Z'; // evals to
true
```

Download

Generating an uppercase character

In order to generate a random uppercase character, we would simply get a random number from 0 to 26, then shift it by the character of 'A'.

This works as 'A' is the character representation for the decimal value of 65. This means we're basically adding 65 to our number, then casting it to a character.

```
char c = (char) ((Math.random() *
26) + 'A');
```

Download

Generating a password

Ask the user for the length of the password, then create a string with random uppercase letters, lowercase letters, and digits. The password should be at least 8 characters long.

Download

Lab 3