Escapes, Strings, and Concatenation

Escape Sequences

Escape sequences are used as an escape sequence to make special edits or characters inside of strings. They are always opened with a backslash

The most common escape sequences include

\' - print single quote
\" - print double quote
\\ - print backslash
\r - carriage return
\n - create new line
\t - tab
\b - insert a backspace in the text

Maydm

Escape Sequence

No, Escape Sequences are not what programmers initiate when they are bogged down in bugs.

Maydm

Examples of Escape Sequences in Use

```
System.out.println(“This is how \“you\” print double quotes.”);
System.out.println(“Single quotes don’t need special characters
inside double quotes.”);
System.out.println(‘However, inside single quotes the compiler
won\’t compile properly without an escape sequence.’);
System.out.println(“We \t can \t print \t forms and \n do cool \n
things with \r escape sequences\\”);
```

Let’s try playing with sequences in a new java file.
Find escapes.txt in day-2/primitive_types and type out a new program as instructed.

Maydm

Manipulating Strings

Recall that strings store text and can be presented to users if saved in a variable. In this lesson we'll learn many ways strings can be manipulated using Java methods.

Strings can be concatenated, meaning they can be combined together, kind of like adding 2 and 2, but instead of 4, concatenating 2 and 2 would result in 22.

```
String concatenateString = “2” + “2”;
System.out.println(concatenateString); // 22
```

Maydm

Practice Strings

Open Intro-to-Concatenation.txt and follow the instructions to create your own concatenated statements.

Maydm

Uses for Concatenation

Concatenation can be useful for many purposes such as combining first and last names.

```
String fName = “John”;
String lName = “Doe”;
String fullName = fName + “ “ + lName;
System.out.println(fullName); // John Doe
```

Maydm

Practice Name Concatenation

Open *Concatenate-your-name.txt* and practice using Strings and variables to print a concatenated form of your name.

Note that you'll replace "ENTERY_YOUR_NAME" with your name.

This is a common practice in coding instructions. Generally, if something is in all caps with "_" to separate words then you're meant to fill in specific information.

Maydm

More Practice With Name Concatenation

Open *concatenate-user-input-name.txt* and create the newest version of the name concatenator.

Maydm

Make a Form

Let's create a form in a new java file that names some of Chance the Rapper's credits. *Open chances-form.txt to get started*

The end product will look something like this:

Year	Album
2018	I Might Need Security
2016	All We Got
2016	Blessings
2016	Coloring Book
2016	Mixtape

Adding Strings to Strings

We can concatenate endlessly by storing variable results.

```
String password = " ";
password = password + "tom ";
password = password + "purple ";
password = password + "greece ";
System.out.println(password); // tom purple greece
```

This snippet instructs Java to add one word right after the other.
We can also use a special operator += to accomplish this goal.

```
password += "greece";
```

Let's try concatenating with operators and variables with *variable-concatenation.txt*

Maydm

Compare Strings

Java provides methods that make some forms of string manipulation and comparison easy to execute.

One such method is equals(). This method enables us to compare two strings for equality and returns a boolean response.

```
String wordOne = “Something different”;
String wordTwo = “Same”;
String wordThree = “Same”;
System.out.println(“Is “ + wordOne + “the same as “ + wordTwo + “?”);
System.out.println(wordOne.equals(wordTwo));
```

Compare Strings Practice

Open the text file compare-strings.txt to write your own comparison program.

Breaking down the equals method:

```
wordOne.equals(wordTwo);
```

The first element of the method, “wordOne” is the first variable to compare.
“equals” is the name of the method
The last element in the parens is the variable being compared against.

Maydm

How Long was that Again?

A very common thing developers measure is the length of a string. Java makes a string's length easy to measure with the `length()` method.

```
String myName = “Jane Doe”;
Int nameLength = myName.length(); // 8
```

The first element of the method is the first variable that will be measured.,
“length” is the name of the method. The parens invoke the method function.

We can practice by writing a snippet of code that will ask for a name and print the length of a user's first and last name.

Sample of String Length Counter

```
import java.util.Scanner;


public class LengthString {
   public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        String fName = "Jane";
      System.out.print("What is your first name? " );
        fName = keyboard.next();
        System.out.println("The length of your first name is "  +
fName.length());
    }
}
```

Let's practice by writing a snippet of code that will ask for a name and print the length of a user's first and last name. Open length-of-strings.txt for more details.

Maydm

A Case for Change

Did you notice in the string comparison exercise that Java thinks "SAME" and "Same" are not...the *same*?
As humans, we'd think, "yeah, they're the same, just one is in all caps."
But computers are basically stupid and very literal. They only know what we tell them through programs.

Maydm

Changing Case

We can help Java recognize matching words with two methods called toUpperCase() and toLowerCase().

Let's open *case-changing-exercise.txt* to experience these methods in action..

```
String question = “WHY ARE YOU YELLING?”;
String answer = “I’m not yelling”;
System.out.println(question.toLowerCase());
System.out.println(answer.toUpperCase());
```

Maydm

Where Did I Put That String Again?

We can use another Java method called indexOf() to search for a character or string within a string.

Meta right?

indexOf() is a powerful method for searching.
Note that the first position of indexOf() is 0.

Talk about starting with nothing!

Maydm

indexOf() String Positions

In indexOf() and other indexing types, the first position is the zeroth position. Let's look at how a computer reads the place value of a String.

S	T	R	I	N	G
0	1	2	3	4	5

Maydm

Count from Zero

Here's another example:

```
String zero = “zero”;
int position = zero.indexOf(“z”);
int location = zero.indexOf(“o”);
Int spot = zero.indexOf(“more than one”);
System.out.println(position); // 0
System.out.println(location); // 3
System.out.println(spot); // -1
```

Where did -1 come from? “more than one” is not in String zero so Java returns -1 meaning, I can't find it.

Maydm

indexOf() Snippet

Let's create a new Java snippet by using the instructions in *string-index.txt.*

Experiment with the indexOf() method until you feel comfortable using it. Are there any surprising ways you can get a return of -1?

Maydm

lastIndexOf()

lastIndexOf() does what it sounds like, searches for a specific value of a string from the end of a string.

The first value in a string is still 0. The index of strings do not change.

Like before, if the lastIndexOf() method cannot find the specific value then it will return -1.

Let's practice by following the directions in *last-string-index.txt*

Maydm

More about indexOf()

In our example, we have two iterations of “Maydm” and “Made” with that precise capitalization. We can find the first iteration of each word with indexOf() and the last iteration with lastIndexOf().
What if we had an occurrence more than two times? How could we find those occurrences?

By adding a second parameter inside the parenthesis…

```
position.indexOf(“Maydm”, 15); // Searches forward from character 15
position.lastIndexOf(“Maydm”, 15) // Searches backward from char 15
```

Let’s practice by using the starter code in “MoreIndexParams.java”

Strings

In this module, you have learned all about Java Strings.

How to write escape sequences, concatenate strings together, create forms, compare strings, measure the length of a string, change case, and how to index the location of characters or words in long strings.

Maydm

Move on to Arrays

Please open the Java Arrays Module

Maydm