Set 2: Basic String Operations

Skill 2.1: Concatenate String type variables
Skill 2.2: Use the *length* method to find the number of characters in a String
Skill 2.3: Retrieve a portion of a String using the *substring* method
Skill 2.4: Convert between lower and upper case
Skill 2.5: Use escape sequences to print special characters

Skill 2.1: Concatenate String type variables

Skill 2.1 Concepts

Concatenation means to combine or attach things together. In Java this is done with the plus (+) sign.

Consider the following example. Below, the variables initialized to "Hello" and "good buddy" are concatenated and assigned to a new variable c. The program prints Hellogood buddy to the screen.

Code

```java
String mm = "Hello";
String nx = "good buddy";
String c = mm + nx;
System.out.println(c);
```

Output

```
Hellogood buddy
```

Notice, in the example above that there is no space between Hello and good. A space between the words "Hello" and "good" could have been achieved by concatenating a space between these words as shown below,

Code

```java
String mm = "Hello";
String nx = "good buddy";
String c = mm + " " + nx;
System.out.println(c);
```

Output

```
Hello good buddy
```

The example below also illustrates another way a space could have been achieved,

Code

```
String mm = "Hello";

String nx = " good buddy";

String c = mm + nx;

System.out.println(c);
```

Output

```
Hello good buddy
```

It is possible to concatenate a String with a numeric variable as follows. This is a useful technique for converting an int variable type to a String variable type.

Code

```
int x = 17;

String s = "Was haben wir gemacht?"; //German for "What have we done"

String combo = s + " " + x;

System.out.println(combo);
```

Output

```
Was haben wir gemacht? 17
```

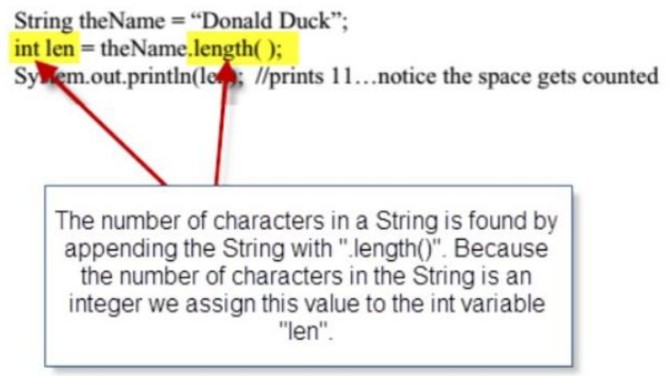
[Skill 2.1 Exercise 1]

Skill 2.2: Use the *length* method to find the number of characters in a String

Skill 2.2 Concepts

In the previous lesson we learned about a *String* type variable. While, *String* type variables are nothing more than a serious of characters, they are also considered objects in Java (as oppossed to primitives). The String object class in Java provides a library of methods which are useful for manipulating *String* type variables.

To access these methods, we can use the "dot" notation. One method in the String library that is useful is the lenth() method. This method can be used to find the number of characters in a String. Consider the following example,



```
String theName = "Donald Duck";
int len = theName.length( );
System.out.println(len);   //prints 11…notice the space gets counted
```

The number of characters in a String is found by appending the String with ".length()". Because the number of characters in the String is an integer we assign this value to the int variable "len".

[Skill 2.2 Exercise 1]

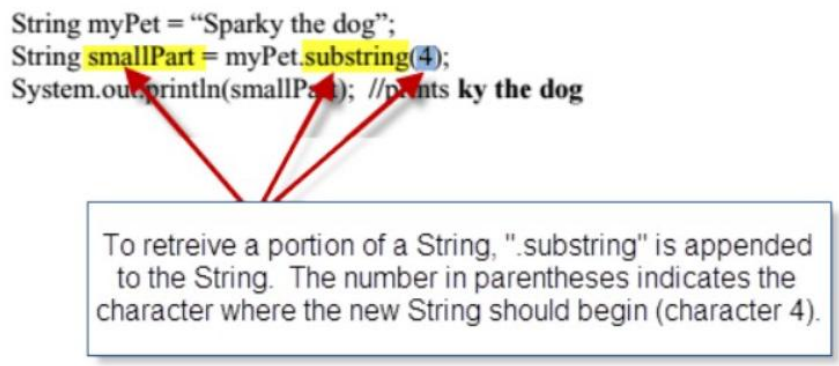
Skill 2.3: Retrieve a portion of a String using the *substring* method

Skill 2.3 Concepts

The *substring* method can be used to indicate the portion of the String we want to retrieve. In computer science counting begins at 0. In the String "Sparky the dog", the character assignments are as follows,

Letter	S	p	a	r	k	y		t	h	e		d	o	g
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13

If we wanted to print all the characters starting at position 4 ("k"), we could use substring(4). This is illustrated below,

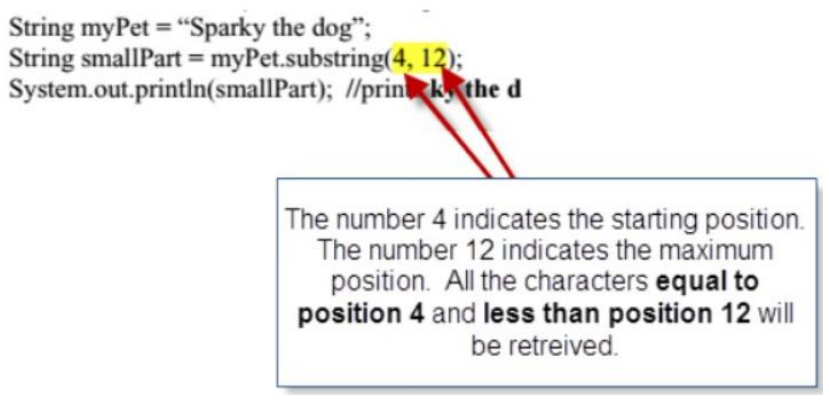


```
String myPet = "Sparky the dog";
String smallPart = myPet.substring(4);
System.out.println(smallPart);  //prints ky the dog
```

To retreive a portion of a String, ".substring" is appended to the String. The number in parentheses indicates the character where the new String should begin (character 4).

Another application of the substring() method involves retrieving the characters from the middle of a String.

If we want to print all the characters starting at k (position 4) and ending at d (position 11), we could use substring(4, 12). This is illustrated below,



```
String myPet = "Sparky the dog";
String smallPart = myPet.substring(4, 12);
System.out.println(smallPart);  //prints ky the d
```

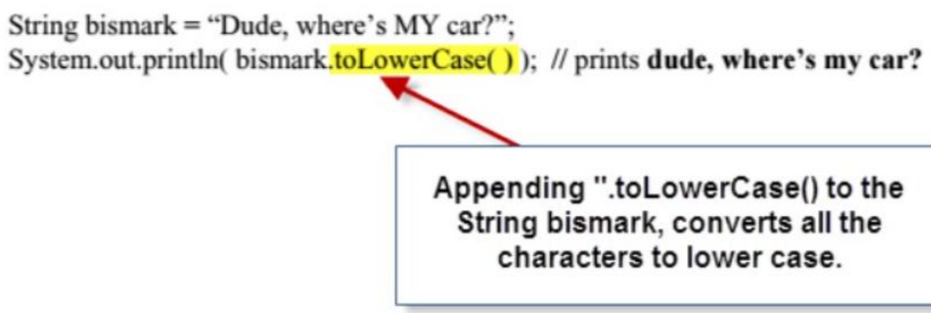
The number 4 indicates the starting position. The number 12 indicates the maximum position. All the characters **equal to position 4** and **less than position 12** will be retreived.

Skill 2.3 Exercise 1

Skill 2.4: Convert between lower and upper case

Skill 2.4 Concepts

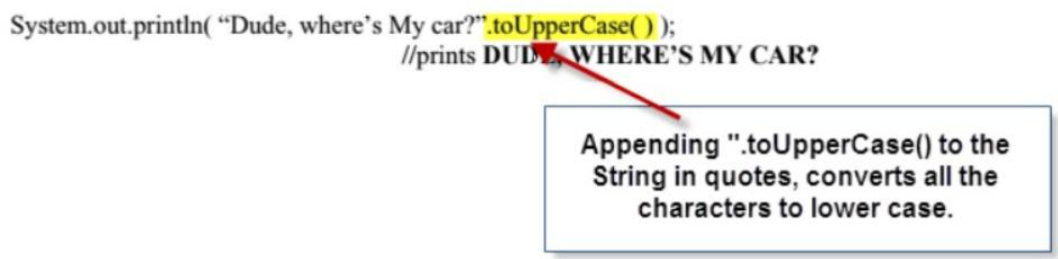
The method *toLowerCase* converts all characters to lower case (small letters)



```
String bismark = "Dude, where's MY car?";
System.out.println( bismark.toLowerCase( ) );  // prints dude, where's my car?
```

Appending ".toLowerCase() to the String bismark, converts all the characters to lower case.

The method *toUpperCase* converts all characters to upper case (capital letters)



```
System.out.println( "Dude, where's My car?".toUpperCase( ) );
                                        //prints DUDE, WHERE'S MY CAR?
```

Appending ".toUpperCase() to the String in quotes, converts all the characters to lower case.

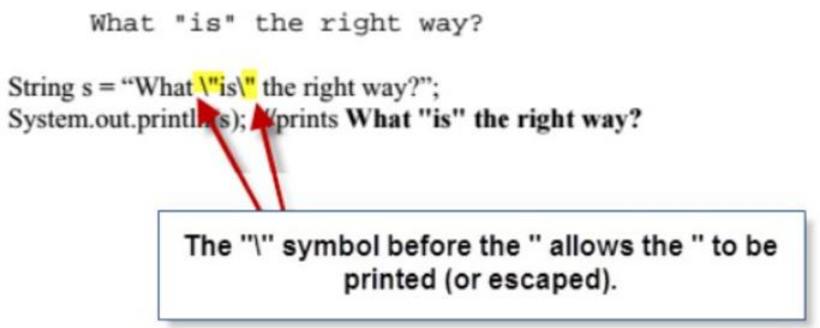
[Skill 2.4 Exercises 1 & 2]

Skill 2.5: Use escape sequences to print special characters

Skill 2.5 Concepts

To force a quote character (") to printout, or to be part of a String, use the escape sequence, "\". Note escape sequences always start with a "\" character.

Consider the following example,

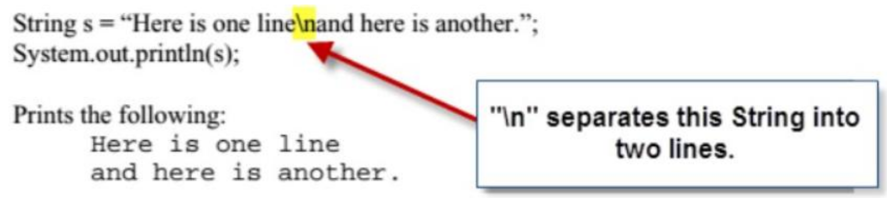


```
What "is" the right way?
```

```
String s = “What \"is\" the right way?”;
System.out.println(s); //prints What "is" the right way?
```

The "\" symbol before the " allows the " to be printed (or escaped).

Another escape sequence, \n, will create a new line (also called a line break) as shown below



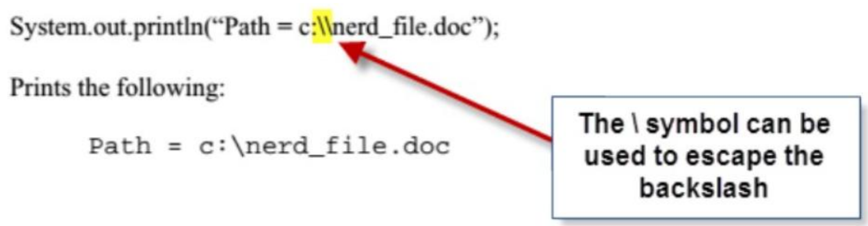
```
String s = “Here is one line\nand here is another.”;
System.out.println(s);
```

Prints the following:

```
Here is one line
and here is another.
```

"\n" separates this String into two lines.

The escape sequence, \\, will allow us to print a backslash within our String. Otherwise, if we try to insert just a single \ it will be interpreted as the beginning of an escape sequence,



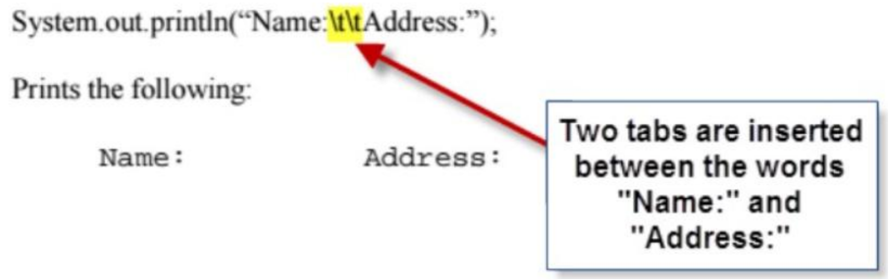
```
System.out.println(“Path = c:\\nerd_file.doc”);
```

Prints the following:

```
Path = c:\nerd_file.doc
```

The \ symbol can be used to escape the backslash

The escape sequence, \t, will allow us to tab over. The following code tabs twice.



```
System.out.println(“Name:\t\tAddress:”);
```

Prints the following:

```
Name:           Address:
```

Two tabs are inserted between the words "Name:" and "Address:"

Skills 2.5 Exercise 1