|  |
| --- |
| **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** |
| 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** |
| 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](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **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,

**A screenshot of a computer code

AI-generated content may be incorrect.**

**[Skill 2.2 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **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,

**A screen shot of a computer

AI-generated content may be incorrect.**

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,

**A screenshot of a computer

AI-generated content may be incorrect.**

**[Skill 2.3 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **Skill 2.4: Convert between lower and upper case** |

**Skill 2.4 Concepts**

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

A screenshot of a computer

AI-generated content may be incorrect.

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

A close-up of a computer screen

AI-generated content may be incorrect.

**[Skill 2.4 Exercises 1 & 2](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **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,

**A close-up of a computer screen

AI-generated content may be incorrect.**

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

A red line with black text

AI-generated content may be incorrect.

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,

A red arrow pointing to a file

AI-generated content may be incorrect.

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

A red arrow pointing to a white box

AI-generated content may be incorrect.

**[Skills 2.5 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**