Lab 1

Counting Uppercase and Lowercase Characters

You are going to write a program that takes a string and prints out two messages. One message tells you how many uppercase characters are in the string. The other message tells how many lowercase characters are in the string. The program will ignore all numbers and special characters (punctuation, symbols, etc.).

String Methods

You will need two string methods that were not covered earlier to help with this project:
* `Character.isUpperCase()` - Returns `true` if the character is uppercase, `false` if the character is lowercase
* `Character.isLowerCase()` - Returns `true` if the character is lowercase, `false` if the character is uppercase

Variables

You will need four variables for this project. One variable will count all of the lowercase characters, another to count the uppercase characters, a third to check each character of the string, and lastly a string variable.

```
int loweCount = 0;
int upperCount = 0;
char ch;
String myString = "Roses are Red, Violets are Blue";
```

Iterating Over the String

The next thing to do is iterate over the string and to check each character of the string. A simple `for` loop works best.

```
for (int i = 0; i < myString.length(); i++) {
  ch = myString.charAt(i);
}
```

Checking for Uppercase and Lowercase

It does not matter if you check for an uppercase character first or check for a lowercase character. Let's start with lowercase characters. Ask if the character is lowercase and increment the appropriate counting variable.

```
if (Character.isLowerCase(ch)) {
  lowerCount += 1;
}
```

What you **do not** want to do is use an `else` statement. This will not give you an accurate count. For example, asking if a special character is lowercase will return `false`. However, that does not mean that it is an uppercase character either. Special characters are neither uppercase nor lowercase. So use an `else if` statement and ask if the character is uppercase. If so, increment the uppercase counting variable.

```
else if (Character.isUpperCase(ch)) {
  upperCount += 1;
}
```

Print the Results

The final step is to print the messages with the count values.

```
System.out.println("There are " + lowerCount + " lowercase
        characters.");
System.out.println("There are " + upperCount + " uppercase
        characters.");
```

There should be 4 uppercase characters and 21 lowercase characters.

▼ **Code**

```
int lowerCount = 0;
int upperCount = 0;
char ch;
String myString = "Roses are Red, Violets are Blue";

for (int i = 0; i < myString.length(); i++) {
  ch = myString.charAt(i);
  if (Character.isLowerCase(ch)) {
    lowerCount += 1;
  }
  else if (Character.isUpperCase(ch)) {
    upperCount += 1;
  }
}

System.out.println("There are " + lowerCount + " lowercase
      characters.");
System.out.println("There are " + upperCount + " uppercase
      characters.");
```

Lab 2

Reverse a String

You are going to write a program that takes a string and prints it in reverse order.

Variables

You are going to need two variables. The first is the original string. Since strings are immutable, we will need a second variable to represent the reversed string. Make the reversed string an empty string.

```
String myString = "The brown dog jumps over the lazy fox";
String reversedString = "";
```

Since the string needs to be reversed, our loop should start at the end of the string and work its way to the first character. We can use a `for` loop that starts at the end of the string and iterate backwards, or we can use a `while` loop to do the same thing. Let's try a `while` loop for this situation. To do this, we need a variable that represents the index. Normally, the index starts at zero, but this loop will go backwards through the string. The starting index needs to be the last character in the string. The `length()` method returns the length of a string. However, the length of a string is always one greater than the last index (because indices start counting at zero). So the starting index should be the length of the string minus one.

```
int index = myString.length() - 1;
```

String Iteration

The `while` loop should run as long as `index` is greater than or equal to 0.

```
while (index >= 0)
```

Reversing a string comes down to taking the a character from the end and putting it at the front of a new string. This will be done by appending the character at `index` to `reversedString`. It is important to remember that

strings are immutable. The line of code below is overwriting `reversedString` with a new string. It is not updating the contents of the string.

```
reversedString += myString.charAt(index);
```

Decrement the `index` variable to avoid an infinite loop.

```
index -= 1;
```

Printing the result

Once the loop has finished running, print `reversedString`.

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

You should see `xof yzal eht revo spmuj god nworb ehT`.

▼ **Code**

```
String myString = "The brown dog jumps over the lazy fox";
String reversedString = "";
int index = myString.length() - 1;

while (index >= 0) {
  reversedString += myString.charAt(index);
  index -= 1;
}

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

Lab 3

Swapping the Case of Characters

You are going to write a program that takes a string and prints a new string where all of the uppercase letters become lowercase, and the lowercase letters become uppercase.

Variables

You are going to need two string variables and one char variable. The first string variable represents the original string, and the second represents the modified string. The char variable will represent each character of the original string later on. For now, the modified string can be empty.

```
String originalString = "THE BROWN DOG JUMPS over the lazy fox";
String modifiedString = "";
char ch;
```

String Iteration

It does not matter if you start at the beginning of the string or the end for iteration. A simple `for` loop is the easiest way to iterate through the `originalString`. Next, set `ch` to take on each character of the string at every index location.

```
for (int i = 0; i < originalString.length(); i++) {
  ch = originalString.charAt(i);
}
```

String Methods

You are going to use the `Character.isUpperCase()` and `Character.isLowerCase()` methods to test if a character is uppercase or lowercase. In addition, you will be using the `toUpperCase()` and `toLowerCase()` methods to convert characters to their new cases.

Conditional

For consistency, We will test if a character is lowercase first. However, you may choose to test for uppercase first. It does not matter as long as the conversion is correct.

```
if (Character.isLowerCase(ch))
```

If this is true, then append the uppercase version of the character to the variable `modifiedString`.

```
modifiedString += Character.toUpperCase(ch);
```

If the conditional is false, then append the lowercase version of the character to `modifiedString`.

```
else {
  modifiedString += Character.toLowerCase(ch);
}
```

You do not need to worry about special characters. Converting them to uppercase or lowercase has no effect.

Printing the Results

Once the loop has finished, print both the original string and the modified string.

```
System.out.println("The original string is: " + originalString);
System.out.println("The modified string is: " + modifiedString);
```

You should see the following output:

```
The original string is: THE BROWN DOG JUMPS over the lazy fox
The modified string is: the brown dog jumps OVER THE LAZY FOX
```

▼ **Code**

```
String originalString = "THE BROWN DOG JUMPS over the lazy
      fox";
String modifiedString = "";
char ch;

for (int i = 0; i < originalString.length(); i++) {
  ch = originalString.charAt(i);
  if (Character.isLowerCase(ch)) {
    modifiedString += Character.toUpperCase(ch);
  }
  else {
    modifiedString += Character.toLowerCase(ch);
  }
}

System.out.println("The original string is: " +
      originalString);
System.out.println("The modified string is: " +
      modifiedString);
```

Lab 4

Count the Vowels

You are going to write a program that counts the number of vowels that appear in a string. For the purpose of this exercise, vowels are upper and lowercase `a, e, i, o, u`.

Variables

For this project, you will need three variables. One will be the string. Another will be a char to represent each character of the string. The final variable will be a count of all the vowels.

```
String myString = "The Brown Dog Jumps Over The Lazy Fox";
char ch;
int count = 0;
```

String Iteration

Use a `for` loop to iterate through the string. Then set `ch` to check every character in the string.

```
for (int i = 0; i < myString.length(); i++) {
  ch = myString.charAt(i);
}
```

Checking for a Vowel

Use a conditional to see if the characters in the string are equal to any vowels. Make sure to account for both uppercase and lowercase vowels.

```
if (ch == 'a' || ch == 'e' || ch == 'i' ||
    ch == 'o' || ch == 'u' || ch == 'A' ||
    ch == 'E' || ch == 'I' || ch == 'O' ||
    ch == 'U') {
```

Note that characters are wrapped in single quotes `' '`, not double quotes in Java.

Incrementing the Counter

If `ch` equals any of the vowels, increment the `count` variable.

```
count += 1;
```

Printing the Result

The string may contain no vowels, one vowel, or more than one vowels. Thus, you'll need conditionals to output the appropriate responses.

```
if (count == 0) {
  System.out.println("There are no vowels in the string.");
}
else if (count == 1) {
  System.out.println("There is 1 vowel in the string.");
}
else {
  System.out.println("There are " + count + " vowels in the
        string.");
}
```

You should see that there are 9 vowels in the string.

▼ **Code**

```
String myString = "The Brown Dog Jumps Over The Lazy Fox";
char ch;
int count = 0;

for (int i = 0; i < myString.length(); i++) {
  ch = myString.charAt(i);
  if (ch == 'a' || ch == 'e' || ch == 'i' ||
      ch == 'o' || ch == 'u' || ch == 'A' ||
      ch == 'E' || ch == 'I' || ch == 'O' ||
      ch == 'U') {
    count += 1;
  }
}

if (count == 0) {
  System.out.println("There are no vowels in the string.");
}
else if (count == 1) {
  System.out.println("There is 1 vowel in the string.");
}
else {
  System.out.println("There are " + count + " vowels in the
        string.");
}
```

Lab Challenge: Vowel Replacement

Replacing Vowels with a *

You are going to write a program that takes a string called `myString` and returns the string but with a `*` in the place of vowels. Assume that vowels are upper and lowercase `a, e, i, o, u`. For example, if `myString = "Hello"`, then your program will print `"H*ll*"`.

Some of the code has already been filled out for you. Your task is to complete the program so that it produces some of the sample output below. If you accidentally change anything from the original code, you can copy and paste the code back into the text editor.

```
public class LabChallenge {
  public static void main(String args[]) {

    String myString = args[0];
    char ch;

    //add code below this line



    //add code above this line
  }
}
```

Click to compile your code

Test your code with a few different values

▼ **Expected Output**

`Hll`

▼ **Expected Output**

`ppl`

▼ **Expected Output**

```
W trm ln!
```

Requirements

- You **should not** make any changes to the code that already exists. If you accidentally delete any existing code, you can copy and paste the entire program from above.
- You can use **any** number of string methods and conditionals to produce the desired output.