CSC 102 String WorkSheet

1. **Declare and assign a string variable, `firstName`, to your first name.**
   String firstName = "Victor";

2. **Write a Java statement that finds out how many characters are in `firstName`.**
   int length = firstName.length();

3. **Write a Java statement that displays the third character is.**
   char thirdChar = firstName.charAt(2);

4. **Declare and assign a string variable, `lastName`, to your last name.**
   String lastName = "Ejiasi";

5. **Declare and assign a string variable, `fullName`, that combines `firstName` and `lastName`.**
   ```
   String fullName = firstName + " " + lastName;
   ```

6. **Write Java statements that will read in your first and last name to a variable called `thisName` and then check if they are the same `thisName` to `fullName`. Make sure you do not consider capitalization when checking if the variables are the same.**
   String thisName = "Victor Ejiasi";
   boolean sameName = thisName.equalsIgnoreCase(fullName);

7. **Tell me what the result of this test was. Was it what you expected?**
   The result of the test was true since I entered my name for the firstName and lastName variable when this name was checked since I used my name there as well it was in fact the same as I expected.

8. **Write a Java statement that will return an `int` when *comparing* `Lumberjacks!` to `m`.**
   String lumberjacks = "Lumberjacks!";
   int compare = lumberjacks.compareTo("m");
   System.out.println("Compare Lumberjacks! to m: " + compare);

9. **Write the Java statement that will read in 1 word.**
   String word = "word";

10. **Write the Java statement that will read in a group of words.**
    String groupOfWords = "group of words";

11. **What is the problem with the nextLine method?**
    It reads in the entire line, including the newline character, so it will not work if you want to read in a single word.

12. **State the method that will *compare* two Strings without considering the capitalization.**
    equalsIgnoreCase

13. Write the Java statement the will see if `CSC102` ends with `0`.

```
String csc102 = "CSC102";
boolean endsWithZero = csc102.endsWith("0");
System.out.println("Does CSC102 end with 0? " + endsWithZero);
```

14. Write the Java statement that will find the first occurance of `w` in the String `word`.

```
int firstOccurance = word.indexOf("w");
System.out.println("First occurance of w in word: " + firstOccurance);
```

15. Write the Java statement that will fine the last occurance of `w` in the String `knowledgeable`.

```
String knowledgeable = "knowledgeable";
int lastOccurance = knowledgeable.lastIndexOf("w");
System.out.println("Last occurance of w in knowledgeable: " + lastOccurance);
```

Formatting Console Output – `printf`

1. **Write the Java statement that formats so that the output looks like: The total is $1600.00**

```
double total = 1600.00;
System.out.printf("The total is $%.2f%n", total);
```

2. **Write the Java statement that formats the String, `formatMe`, with 6 spaces infront of it.**

```
String formatMe = "Hello";
System.out.printf("%6s%n", formatMe);
```

3. **Is the number positive or negative when aligning to the left?**

   When aligning to the left, positive numbers are preceded by a space, and negative numbers are preceded by a minus sign (-).

4. **Is the number positive or negative when aligning to the right?**

   When aligning to the right, positive numbers are preceded by a space, and negative numbers are preceded by a minus sign (-).

5. **What is the format specifier for a whole number?**

   The format specifier for a whole number (integer) is %d.

6. **What is the format specifier for a decimal number?**

   The format specifier for a decimal number (floating-point number) is %f.

7. **What is the format specifier for a character?**

   The format specifier for a character is %c.

8. **What is the format specifier for a String?**

   The format specifier for a String is %s.